hw3-task1

这里使用了`sympy`进行矩阵的一些运算，个别细节会和作业过程不太一样，但最后的结果应该是没问题的。

```
dec = 6 # 设置每一步计算保留小数点后位数（精度，可以自己调整）
import numpy as np
np.set_printoptions(formatter={'float': ('{: 0.' + str(dec) + 'f}').format})
import matplotlib.pyplot as plt
import matplotlib as mpl
mpl.rcParams['text.usetex'] = True
import sympy as sp
```

1 习题3.1

1.1 第一问

```
mat_aug = sp.Matrix([
    [2, 1, 1, 4],
    [1, 3, 2, 6],
    [1, 2, 2, 5]
])
mat_aug
```

$$\begin{bmatrix} 2 & 1 & 1 & 4 \\ 1 & 3 & 2 & 6 \\ 1 & 2 & 2 & 5 \end{bmatrix}$$

```
res = mat_aug.LUdecomposition()
print("L: ")
res[0]
```

```
L:
```

$$\begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{2} & \frac{3}{5} & 1 \end{bmatrix}$$

```
print("U:")
res[1]
```

```
U:
```

$$\begin{bmatrix}2 & 1 & 1 & 4\\0 & \frac{5}{2} & \frac{3}{2} & 4\\0 & 0 & \frac{3}{5} & \frac{3}{5}\end{bmatrix}$$

```python
x = mat_aug[0:3, 0:3].QRsolve(mat_aug[0:3, -1])
print("result: x is")
x
```

```
result: x is
```

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

```python
print("检验: ")
res[0] @ res[1]
```

```
检验:
```

$$\begin{bmatrix}2 & 1 & 1 & 4\\1 & 3 & 2 & 6\\1 & 2 & 2 & 5\end{bmatrix}$$

1.2 第二问

```python
mat_aug = sp.Matrix([
    [3, 2, -7, -4],
    [8, 2, -3, -5],
    [4, 6, -1, 13]
])
mat_aug
```

$$\begin{bmatrix}3 & 2 & -7 & -4\\8 & 2 & -3 & -5\\4 & 6 & -1 & 13\end{bmatrix}$$

```python
res = mat_aug.LUdecomposition()
print("L: ")
res[0]
```

```
L:
```

$$\begin{bmatrix} 1 & 0 & 0 \\ \frac{8}{3} & 1 & 0 \\ \frac{4}{3} & -1 & 1 \end{bmatrix}$$

```python
print("U:")
res[1]
```

```
U:
```

$$\begin{bmatrix} 3 & 2 & -7 & -4 \\ 0 & -\frac{10}{3} & \frac{47}{3} & \frac{17}{3} \\ 0 & 0 & 24 & 24 \end{bmatrix}$$

```python
x = mat_aug[0:3, 0:3].QRsolve(mat_aug[0:3, -1])
print("result: x is")
x
```

```
result: x is
```

$$\begin{bmatrix} -1 \\ 3 \\ 1 \end{bmatrix}$$

```python
print("检验: ")
res[0] @ res[1]
```

```
检验:
```

$$\begin{bmatrix} 3 & 2 & -7 & -4 \\ 8 & 2 & -3 & -5 \\ 4 & 6 & -1 & 13 \end{bmatrix}$$

2 习题3.2

2.1 第一问

```python
mat_aug = sp.Matrix([
    [3, 4, -3, 10],
    [3, -2, 4, 11],
    [2, -1, -1, 4]
])
mat_aug
```

$$\begin{bmatrix} 3 & 4 & -3 & 10 \\ 3 & -2 & 4 & 11 \\ 2 & -1 & -1 & 4 \end{bmatrix}$$

```python
res = mat_aug.LUdecomposition()
print("L:")
res[0]
```

```
L:
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ \frac{2}{3} & \frac{11}{18} & 1 \end{bmatrix}$$

```python
print("U:")
res[1]
```

```
U:
```

$$\begin{bmatrix} 3 & 4 & -3 & 10 \\ 0 & -6 & 7 & 1 \\ 0 & 0 & -\frac{59}{18} & -\frac{59}{18} \end{bmatrix}$$

```python
x = mat_aug[0:3, 0:3].QRsolve(mat_aug[0:3, -1])
print("result: x is")
x
```

```
result: x is
```

$$\begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix}$$

```python
print("检验: ")
res[0] @ res[1]
```

```
检验:
```

$$\begin{bmatrix} 3 & 4 & -3 & 10 \\ 3 & -2 & 4 & 11 \\ 2 & -1 & -1 & 4 \end{bmatrix}$$

```python
print("det 值: ")
print(mat_aug[0:3, 0:3].det())
```

```
det 值:
59
```

2.2 第二问

```python
mat_aug = sp.Matrix([
    [-18, 3, -1, -1, -15],
    [12, -3, 3, 4, 15],
    [3, 1, -1, 1, 2],
    [1, 1, 1, 1, 6]
])
mat_aug
```

$$\begin{bmatrix} -18 & 3 & -1 & -1 & -15 \\ 12 & -3 & 3 & 4 & 15 \\ 3 & 1 & -1 & 1 & 2 \\ 1 & 1 & 1 & 1 & 6 \end{bmatrix}$$

```python
res = mat_aug.LUdecomposition()
print("L:")
res[0]
```

```
L:
```

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{2}{3} & 1 & 0 & 0 \\ -\frac{1}{6} & -\frac{3}{2} & 1 & 0 \\ -\frac{1}{18} & -\frac{7}{6} & \frac{11}{7} & 1 \end{bmatrix}$$

```python
print("U:")
res[1]
```

```
U:
```

$$\begin{bmatrix} -18 & 3 & -1 & -1 & -15 \\ 0 & -1 & \frac{7}{3} & \frac{10}{3} & 5 \\ 0 & 0 & \frac{7}{3} & \frac{35}{6} & 7 \\ 0 & 0 & 0 & -\frac{13}{3} & 0 \end{bmatrix}$$

```python
x = mat_aug[0:4, 0:4].QRsolve(mat_aug[0:4, -1])
print("result: x is")
x
```

```
result: x is
```

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 0 \end{bmatrix}$$

```python
print("检验：")
res[0] @ res[1]
```

```
检验：
```

$$\begin{bmatrix} -18 & 3 & -1 & -1 & -15 \\ 12 & -3 & 3 & 4 & 15 \\ 3 & 1 & -1 & 1 & 2 \\ 1 & 1 & 1 & 1 & 6 \end{bmatrix}$$

```python
print("det 值: ")
print(mat_aug[0:4, 0:4].det())
```

```
det 值:
-182
```

这里需要注意的是：事实上选主元这个过程会做一次“行变换”，会对最后的 **det** 值产生影响，一定要注意自己在选主元过程中对行变换操纵的次数！

3　习题3.4

```python
mat_aug = sp.Matrix([
    [0.002, 87.13, 87.15],
    [4.453, -7.26, 37.27]
])
mat_aug
```

$$\begin{bmatrix} 0.002 & 87.13 & 87.15 \\ 4.453 & -7.26 & 37.27 \end{bmatrix}$$

```python
res = mat_aug.LUdecomposition()
print("L:")
res[0]
```

```
L:
```

$$\begin{bmatrix} 1 & 0 \\ 2226.5 & 1 \end{bmatrix}$$

```python
print("U:")
res[1]
```

```
U:
```

$$\begin{bmatrix} 0.002 & 87.13 & 87.15 \\ 0 & -194002.205 & -194002.205 \end{bmatrix}$$

```
x = mat_aug[0:2, 0:2].QRsolve(mat_aug[0:2, -1])
print("result: x is")
x
```

```
result: x is
```

$$\begin{bmatrix} 10.0 \\ 1.0 \end{bmatrix}$$

理论上这个矩阵是病态的，`sympy`处理的比较好，可以规避这个问题。一般来讲在矩阵对角占优的情形下，在**规定的有效数字位数**情形下，可以获取比非对角占优更好的数值解，这里就不做展示了。在后续的学习中经常性的会遇到"对角占优的矩阵"。

4 习题3.5

```
mat_aug = sp.Matrix([
    [1e-8, 2, 3],
    [-1, 3.712, 4.623],
    [-2, 1.072, 5.643]
])
mat_aug
```

$$\begin{bmatrix} 1.0 \cdot 10^{-8} & 2 & 3 \\ -1 & 3.712 & 4.623 \\ -2 & 1.072 & 5.643 \end{bmatrix}$$

```
print("det 值: ")
mat_aug.det()
```

```
det 值:
```

11.8500001599096

5 习题3.6

```
mat_aug = sp.Matrix([
    [1, 1, -1, 1, 0, 0],
    [2, 1, 0, 0, 1, 0],
    [1, -1, 0, 0, 0, 1]
])
mat_aug
```

$$\begin{bmatrix}1 & 1 & -1 & 1 & 0 & 0\\2 & 1 & 0 & 0 & 1 & 0\\1 & -1 & 0 & 0 & 0 & 1\end{bmatrix}$$

接下来是矩阵计算的详细步骤:

```
mat = mat_aug.copy()
mat
```

$$\begin{bmatrix}1 & 1 & -1 & 1 & 0 & 0\\2 & 1 & 0 & 0 & 1 & 0\\1 & -1 & 0 & 0 & 0 & 1\end{bmatrix}$$

```
mat[1, :] -= mat[0, :]*2
mat
```

$$\begin{bmatrix}1 & 1 & -1 & 1 & 0 & 0\\0 & -1 & 2 & -2 & 1 & 0\\1 & -1 & 0 & 0 & 0 & 1\end{bmatrix}$$

```
mat[2, :] -= mat[0, :]
mat
```

$$\begin{bmatrix}1 & 1 & -1 & 1 & 0 & 0\\0 & -1 & 2 & -2 & 1 & 0\\0 & -2 & 1 & -1 & 0 & 1\end{bmatrix}$$

```
mat[1, :] *= -1
mat[2, :] *= -1
mat
```

$$\begin{bmatrix}1 & 1 & -1 & 1 & 0 & 0\\0 & 1 & -2 & 2 & -1 & 0\\0 & 2 & -1 & 1 & 0 & -1\end{bmatrix}$$

```python
mat[2, :] -= mat[1, :] * 2
mat
```

$$\begin{bmatrix}1 & 1 & -1 & 1 & 0 & 0\\0 & 1 & -2 & 2 & -1 & 0\\0 & 0 & 3 & -3 & 2 & -1\end{bmatrix}$$

```python
mat[2, :] *= sp.Rational(1, 3)
mat
```

$$\begin{bmatrix}1 & 1 & -1 & 1 & 0 & 0\\0 & 1 & -2 & 2 & -1 & 0\\0 & 0 & 1 & -1 & \frac{2}{3} & -\frac{1}{3}\end{bmatrix}$$

```python
mat[1, :] += mat[2, :] * 2
mat[0, :] += mat[2, :]
mat
```

$$\begin{bmatrix}1 & 1 & 0 & 0 & \frac{2}{3} & -\frac{1}{3}\\0 & 1 & 0 & 0 & \frac{1}{3} & -\frac{2}{3}\\0 & 0 & 1 & -1 & \frac{2}{3} & -\frac{1}{3}\end{bmatrix}$$

```python
mat[0, :] -= mat[1, :]
mat
```

$$\begin{bmatrix}1 & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3}\\0 & 1 & 0 & 0 & \frac{1}{3} & -\frac{2}{3}\\0 & 0 & 1 & -1 & \frac{2}{3} & -\frac{1}{3}\end{bmatrix}$$

于是结果呼之欲出，并且检验一下如下:

```python
res = mat[0:3, 3:6]
print("A^-1: ")_
res
```

$$\begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} \\ 0 & \frac{1}{3} & -\frac{2}{3} \\ -1 & \frac{2}{3} & -\frac{1}{3} \end{bmatrix}$$

```
print("检验：")
mat_aug[0:3, 0:3] @ res
```

```
检验:
```

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

当然这里在实际操作中也许用列主元会更好，但这里为了避免行交换带来的一些问题，就不采用列主元的方法了。求得的结果是没有问题的。