```
#exercise 5.2
#Gauss Elimination
#15.11.4
#author: chuanlu

import numpy as np
from copy import deepcopy

def find_max(A):
    '''
    输入一个向量 A，找到其最大值的下标，并返回
    '''
    index = 0;
    for i in range(len(A)):
        if abs(A[i]) > abs(A[index]):
            index = i
    return index

def _column_pivoting_LU_decomposition(A, b):
    '''
    输入一个矩阵 A，对其进行列选主元的 LU 分解，
    将分解的结果放在 A 中输出，其中 A 的上三角部分为 U，A 的
    严格下三角部分为 L 的严格下三角部分，L 的主对角线全为 1
    同时，置换阵 P = P(n-1) * P(n-2) *...* P(1)直接作用于
    常数向量 b 上，返回新的常数向量 b
    '''
    n = len(A)
    for k in range(n - 1):
        #选择主元，并交换第 K 行和主元所在的第 p 行
        p = find_max(A[k:, k])
        if p == 0:
            #若主元就在第 K 行，则不进行置换
            pass
        else:
            p = p + k
            #WTF!!!!numpy 怎么可以不支持运算符重载！！！！
            temp = deepcopy(A[k, :])
            A[k,:] = A[p,:]
            A[p,:] = temp
            #将第 K 个置换阵 P(k)作用在 b 上，也即交换 b 的第 k 行和第 p 行
            temp = deepcopy(b[k])
            b[k] = b[p]
            b[p] = temp
        if A[k, k] != 0:
```

```
                A[k+1:, k] /= A.item((k, k))
                A[k+1:, k+1:] -= A[k+1:, k] * A[k, k+1:]
            else:
                #为奇异阵
                break
    return A, b

def foward_substitution(A, b):
    '''
    输入一个经过 LU 分解的矩阵 A，及其对应的经过/未经变换的
    常数向量 b，使用前代法解出此下三角方程的解，并将其
    置于 b 中输出
    '''
    n = len(A)
    #L 的主对角线元素用向量 diag 储存
    diag = [1.0 for i in range(n)]
    diag = np.array(diag, dtype = float)
    for i in range(n - 1):
        b[i, 0] /= diag[i]
        b[i+1:, 0] -= b.item(i) * A[i+1:, i]
    b[n - 1, 0] /= diag.item(n - 1)
    return b

def backword_substitution(A, y):
    '''
    输入一个经过 LU 分解的矩阵 A，及其对应的常数向量 y,
    使用回代法解出此上三角方程的解，并将其置于 y 中输出
    '''
    n = len(A)
    for j in range(n - 1, 0, -1):
        y[j] /= A.item((j, j))
        y[: j, 0] -= y.item(j) * A[:j, j]
    y[0] /= A.item((0, 0))
    return y

def column_pivoting_gauss_elimination(A, b):
    '''
    输入一个矩阵 A 和一个常数向量 b，采用列选主元的高斯消去法，
    计算这个方程的数值解
    PA = LU; b = Pb
    Ly = b
    Ux = y
    solve x
    '''
```

```
    #首先，对 A 和 b 进行 LU 分解，以及行置换变换
    A, b = _column_pivoting_LU_decomposition(A, b)
    #对置换后的 A 和 b 使用前代法，解出 y
    y = foward_substitution(A, b)
    x = backword_substitution(A, y)
    return x

def _q1():
    A = np.matrix([[3.01, 6.03, 1.99], [1.27, 4.16, -1.23], [0.987, -4.81, 9.34]])
    b = np.matrix([[1.0], [1.0], [1.0]])
    x = column_pivoting_gauss_elimination(A, b)
    det = np.linalg.det(A)
    cond = np.linalg.cond(A)
    print(x)
    print("det:", det)
    print("cond:", cond)

def _q2():
    A = np.matrix([[3.00, 6.03, 1.99], [1.27, 4.16, -1.23], [0.990, -4.81, 9.34]])
    b = np.matrix([[1.0], [1.0], [1.0]])
    x = column_pivoting_gauss_elimination(A, b)
    det = np.linalg.det(A)
    cond = np.linalg.cond(A)
    print(x)
    print("det:", det)
    print("cond:", cond)

def main():
    _q1()
    _q2()

if __name__ == '__main__':
    main()
```

运行结果如下：

```
[[ 1592.59962484]
 [ -631.9113762 ]
 [ -493.61772476]]
det: 33.9049105537
cond: 24.7540221356


[[ 119.52733813]
 [ -47.14260443]
 [ -36.84025611]]
det: 34.3432579419
cond: 24.4361123302
[Finished in 0.3s]
```

当 cond 值较大时，对原系数矩阵的微小扰动将会导致计算结果的巨大变化，这意味着此问题是病态的。