#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。