In [1]:

```
%matplotlib inline
#Импротируем необходимые функции
import numpy as np # работа с массивами и линейной алгеброй
import matplotlib.pyplot as plt # для отрисовки графиков
import scipy.linalg as la # функции линейной алгебры
import math as math
```

In [2]:

```
# Формируем матрицы и находим решение через стандартную функцию. (потом сравним с резул
ьтатом моего решения)
n = 3
A = np.random.rand(n, n)
b = np.random.rand(n)
x_table = np.linalg.solve(A,b)
#print(A)
```

In [3]:

```
# Напишем функцию, которая будет разлагать матрицу A на матрицы L(нижнетреугольная, на
 диагонали единички) и U(верхнетреугольная),
#и выдаст вектор перестановок
def Gauss_dissolution(A, b, n):
    L = np.zeros((n,n))
    U = A
    #print(U,'\n')
    P = np.ones(n)
    for i in range (n - 1): # Идем по строкам с самой первой
        #print(i, ' \n')
        max_el = math.fabs( U[i, i] ) # Берем первый элемент как максимальный
        #print(max_el, ' \n')
        max_i = i # номер максимального элемента
        for m in range(i, n): #найдем главный элемент первого столбца
            if math.fabs(U[m, i]) > max_el: # находим максимальный элемент в столбце
                max_el = U[m,i]
                max_i = m
        if max_el != 0:
            if max_i == i: #наш диагональный элемент максимален -- ничего не меняем
                P[i] = i
            elif max_i > i: # меняем местами
                for x in range (n):
                    temp = b[i]
                    b[i] = b[max_i]
                    b[max_i] = temp
                    #print ('max_i > i')
                    temp = L[i, x]
                    L[i, x] = L[max_i, x]
                    L[max_i, x] = temp
                    temp = U[i, x]
                    U[i, x] = U[max_i, x]
                    U[max_i, x] = temp
                    #print(U, '\n')
                P[i] = max_i # зафиксировали изменение в матрице перестановок
            L[i,i] = 1
            for j in range(i + 1, n): #начинаем проход по подматричке --это строки подм
атрички(формируем U)
                L[j, i] = U[j, i] / U[i, i]
                for k in range(i, n):
                    U[j, k] = U[j,k] - L[j, i] * U[i, k]
    L[n-1, n-1] = 1
    P[n - 1] = n - 1
    return (L, U, P, b)
```

In [4]:

```
# Проверим, как работает моя функция. (я забыла функцию,что разделяет матрицы l и U)
A1 = A
LU, piv = la.lu_factor(A1)
#print(piv)
#print(LU)
#print(b)
my_L, my_U, my_P, my_b = Gauss_dissolution(A, b, n)
#print(my_L)
#print(my_U)
#print(my_P)
#print(my_b)
```

In [5]:

```
#решим систему уравнений: Ly = b
y = np.zeros(n)
for i in range(n):
    y[i] = my_b[i]
    for j in range(i-1, -1, -1):
        y[i] = y[i] - y[j]*my_L[i, j]
#print(y)

#проверим решение
x = np.linalg.solve(my_L,my_b)
#print(x)
```

In [6]:

```
#решим систему уравнений: Ux = y
x = np.zeros(n)
for i in range(n-1, -1, -1):
    x[i] = y[i]
    for j in range(i+1, n):
        x[i] = x[i] - x[j]*my_U[i, j]
    x[i] = x[i] / my_U[i,i]
print(x)

#проверим решение
x1 = np.linalg.solve(my_U,y)
print(x1)
print(x_table) #то, что мы получили встроенной функцией
```

```
[-0.04276509  0.51975798 -0.28884701]
[-0.04276509  0.51975798 -0.28884701]
[-0.04276509  0.51975798 -0.28884701]
```

In [7]:

```
print(la.norm(x - x_table))
```

```
1.3092278833360675e-16
```