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]:

```
# Проверим, положительно ли определена матрица, проверив ее собственные значения
def is_pos_def(x):
    return np.all(np.linalg.eigvals(x) > 0)
```

In [3]:

```
# Напишем функцию, которая будет разлагать матрицу A на матрицы C(нижнетреугольная) и
 C.T(верхнетреугольная)
def Cholesky_dissolution(A, n):
    C = np.zeros((n,n))
    for j in range (n): # Идем по столбцам с самого первого
        sum = 0 #сумма квадратов элементов строки до диагонального
        for k in range (j):
            sum = sum + C[j, k] ** 2
        C[j, j] = math.sqrt(A[j, j] - sum) #диагональный элемент
        for i in range (j+1, n): #спускаемся по столбцу вниз
            sum = 0
            for k in range (j):
                sum = sum + C[i, k] * C[j, k]
            C[i, j] = 1 / C[j, j] * (A[i, j] - sum)
    #print(C)

    # Проверим, как работает функция разложения.
    A1 = A
    C1 = np.linalg.cholesky(A1)
    #print(C1)

    return (C)
```

In [4]:

```
#решим систему уравнений: Cy = b
def Solve_Cy(C, b, n):
    y = np.zeros(n)
    for i in range(n):
        y[i] = b[i]
        for j in range(i-1, -1, -1):
            y[i] = y[i] - y[j] * C[i, j]
        y[i] = y[i] / C[i, i]
    #print(y)

    #проверим решение
    x = np.linalg.solve(C, b)
    #print(x)

    return(y)
```

In [5]:

```
#решим систему уравнений: C.T*x = y
def Solve_CTx(C, y, n):
    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]*C.T[i, j]
        x[i] = x[i] / C.T[i,i]
    #print(x)

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

    return(x)
```

In [6]:

```
def Cholesky_method(A, b, n):
    C = Cholesky_dissolution(A, n)
    y = Solve_Cy(C, b, n)
    x = Solve_CTx(C, y, n)
    return(C, x)
```

In [24]:

```
# Формируем матрицы и находим решение через стандартную функцию. (потом сравним с резул
ьтатом моего решения)
n = 3
A = np.random.rand(n, n)
A = (A + A.T)/2 #чтобы получилась симметричная матрица
while (is_pos_def(A) != 1): #проверка на положительность
    A = np.random.rand(n, n)
    A = (A + A.T)/2
    #print(is_pos_def(A))
b = np.random.rand(n)
x_table = np.linalg.solve(A,b)
#print(A)
```

```
False
False
False
False
False
False
False
False
False
False
False
False
False
False
False
False
False
False
False
True
```

In [25]:

```
C, x =  Cholesky_method(A, b, n)
print(C)
C1 = np.linalg.cholesky(A)
print(C1)
```

```
[[0.95331769 0.         0.        ]
 [0.87763378 0.26378923 0.        ]
 [0.34413964 0.6852275  0.58652095]]
[[0.95331769 0.         0.        ]
 [0.87763378 0.26378923 0.        ]
 [0.34413964 0.6852275  0.58652095]]
```

In [26]:

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

```
1.7162554193316897e-14
```