**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение высшего образования  
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ имени Н.Э.БАУМАНА  
(национальный исследовательский университет)»**

Факультет: Информатика и системы управления

Кафедра: Теоретическая информатика и компьютерные технологии

**Лабораторная работа № 6**

“Решение систем нелинейных уравнений методом Ньютона”

по дисциплине «Численные методы»

Вариант 17

Работу выполнил

студент группы ИУ9-62Б

Сербин Денис

Москва, 2022

# **1. Цель работы**

Целью работы является решение системы нелинейных уравнений методом Ньютона с начальным приближением, выбранным как графическое решение системы.

**2. Постановка задачи**

Дано:

Система уравнений:

, где x = , f =

Начальное приближение:

Найти:

1. Графическое решение системы уравнений
2. Решение системы уравнений методом Нтютона

**3. Индивидуальный вариант**

**4. Теоретические сведения**

Метод Ньютона – один из самых распространенных методов решения системы нелинейных уравнений.

Приближение в методе Ньютона строится по рекуррентной зависимости , где - столбцы (k+1) и k приближения к решению.

- матрица Якоби производной от f по

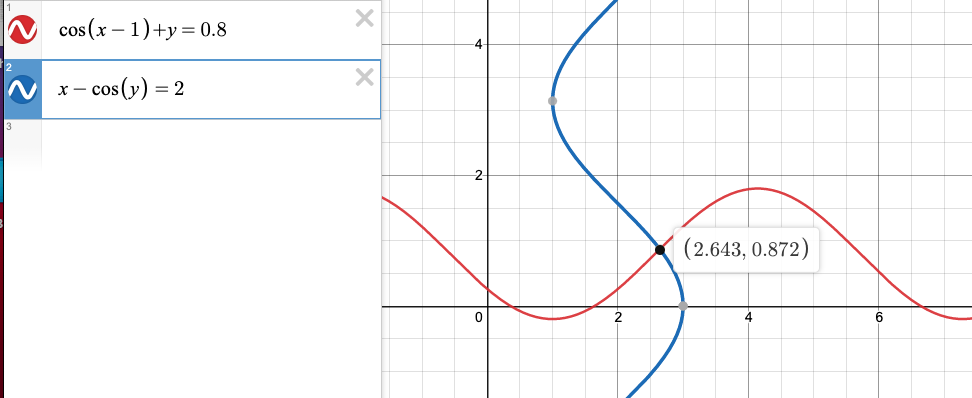
Приближение методом Ньютона считается в 2 этапа: сначала решаем систему уравнений: , где y -решение СЛАУ, после (k+1) приближение это сумма k приближения и решения y: .

Для решения системы нелинейных уравнений с точностью сравниваем c погрешностью приближения: <

Метод Ньютона сходится если , f = дважды неприрывно дифференцируема по всем переменным и начальное приближение находится достаточно близко к точному решению системы.

**5. Практическая реализация**

Решим графически систему уравнений:



Возьмем за начальное приближение и погрешность

Реализация метода Ньютона на Python:

**import** math  
  
x\_0 = [3, 1.0]  
  
  
**def** df1\_x(x):  
 **return** -math.sin(x-1)  
  
  
**def** df1\_y(y):  
 **return** 1  
  
  
**def** df2\_x(x):  
 **return** 1  
  
  
**def** df2\_y(y):  
 **return** math.sin(y)  
  
  
**def** func1(x, y):  
 **return** math.cos(x-1)+y-0.8  
  
  
**def** func2(x,y):  
 **return** x-math.cos(y)-2  
  
  
**def** get\_matrix(x,y):  
 **return** [  
 [df1\_x(x), df1\_y(y)],  
 [df2\_x(x), df2\_y(y)]  
 ]  
  
  
**def** gauss(d, m):  
 **for** k **in** range(0, len(d)):  
 max = k  
 **for** i **in** range(k+1, len(d)):  
 **if** abs(m[i][k]) > abs(m[max][k]):  
 max = i  
  
 temp = m[k]  
 m[k] = m[max]  
 m[max] = temp  
  
 t = d[k]  
 d[k] = d[max]  
 d[max] = t  
  
 **for** i **in** range(k+1, len(d)):  
 fact = m[i][k]/m[k][k]  
 d[i] -= fact\*d[k]  
 **for** j **in** range(k, len(d)):  
 m[i][j] -= fact\*m[k][j]  
  
 out = []  
 **for** i **in** range(0,len(d)):  
 out.append(0.0)  
  
 **for** i **in** range(len(d)-1, -1, -1):  
 sum = 0.0  
 **for** j **in** range(i+1, len(d)):  
 sum += m[i][j]\*out[j]  
 out[i] = (d[i] - sum) / m[i][i]  
  
 **return** out  
  
  
**def** my\_max(x, y):  
 **return** x **if** x > y **else** y  
  
  
**if** \_\_name\_\_ == '\_\_main\_\_':  
 x\_matr = get\_matrix(x\_0[0], x\_0[1])  
 f\_vector = [-func1(x\_0[0], x\_0[1]), -func2(x\_0[0], x\_0[1])]  
 y = gauss(f\_vector, x\_matr)  
 x\_k = [  
 x\_0[0] + y[0],  
 x\_0[1] + y[1]  
 ]  
 print(x\_k[0], x\_k[1])  
 diff = my\_max(abs(x\_k[0] - x\_0[0]), abs(x\_k[1] - x\_0[1]))  
 **while** diff > 0.001:  
 x\_k\_1 = x\_k  
 x\_matr = get\_matrix(x\_k[0], x\_k[1])  
 f\_vector = [-func1(x\_k[0], x\_k[1]), -func2(x\_k[0], x\_k[1])]  
 y = gauss(f\_vector, x\_matr)  
 x\_k = [  
 x\_k[0] + y[0],  
 x\_k[1] + y[1]  
 ]  
 print(x\_k[0], x\_k[1])  
 diff = my\_max(abs(x\_k[0] - x\_k\_1[0]), abs(x\_k[1] - x\_k\_1[1]))

**6. Результаты**

Для тестирования был взят индивидуальный вариант: c начальным приближением и погрешностью

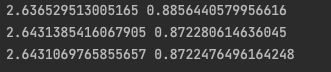


Рис.1 Выходные результаты программы

Погрешность была преодолена за 3 итерации программы.

**7. Вывод**

В ходе выполнения лабораторной работы были выполнено графическое решение системы нелинейных уравнений, а так же приближение с помощью метода Ньютона, и написана реализация на языке программирования Python.

Данный метод вычисляет результат с высокой точностью и за малое число приближений при малой точности начального приближения.