2021

Лабораторная работа по курсу

«Методы Численного Анализа»

Решение систем нелинейных уравнений

Вариант 8

Выполнил: Прудникович

Владислав Вячеславович

2 курс 6 группа

Преподаватель: Будник Анатолий Михайлович

**Постановка задачи**

Найти решение системы нелинейных уравнений:

Применяя следующие методы:

1. Метод простой итерации
2. Метод Ньютона с постоянной производной

Необходимо:

А. Отделить хотя бы один (если их несколько) корень системы нелинейных уравнений.

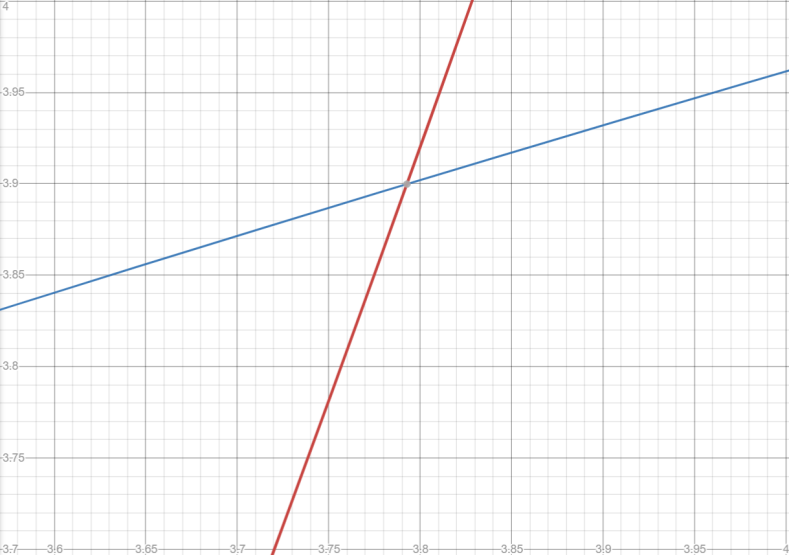
Б. Выбрать начальное приближение , найти двумя вышеуказанными методами решение с точностью . Критерием остановки итерационного процесса выбрать следующий:

В. Сравнить методы по скорости сходимости и точности (Оценить количество итераций и величину невязки каждого метода)

Пункт А. Отделение корня.

Так как не существует универсального численного метода отделения корня при решении систем нелинейных уравнений, мы воспользуемся графическим методом.

Для этого построим графики уравнений заданной системы и найдём окрестность их пересечения.



Рассматриваемый корень располагается на компакте:

[3.77 3.8] [ 3.88 y 3.91].

Пункт Б0. Выбор начального приближения.

В качестве начальных приближений и возьмём середины соответствующих сторон найденной области.

= 3.785, = 3.895. Тогда

= 0.21213 = .

Пусть

(x,y) = ,

(x,y) = .6.

Тогда решение следующей системы:

Эквивалентно решению заданной системы нелинейных уравнений.

Пункт Б1. Решение СНУ методом простой итерации

Приведем и к каноническому виду:

x = (x,y) =

y = (x,y) =

//[Заметим, что функции и определены и непрерывны в области Ω(, ) и рассмотрим значения частных прозводных на ней.

= < 0.265 Ω.

= < 0.133 Ω.

Наши отображения являются сжимающими с коэффициентами сжатия q1 = 0.265 , q2 =0.133.

Проверим выполнение следующих условий:

1. q1)\*)) и (1-q2)\*))

Следовательно, метод простой итерации сходится для данной системы отображений и .]//

Запустим алгоритм МПИ, используя найденные ранее :

, , i = 0, 1, …

Остановим его, когда выполнится условие , где

- точка с координатами и .

Наиболее точным вариантом будет выбор октаэдрической нормы. В нашем случае она запишется следующим образом:

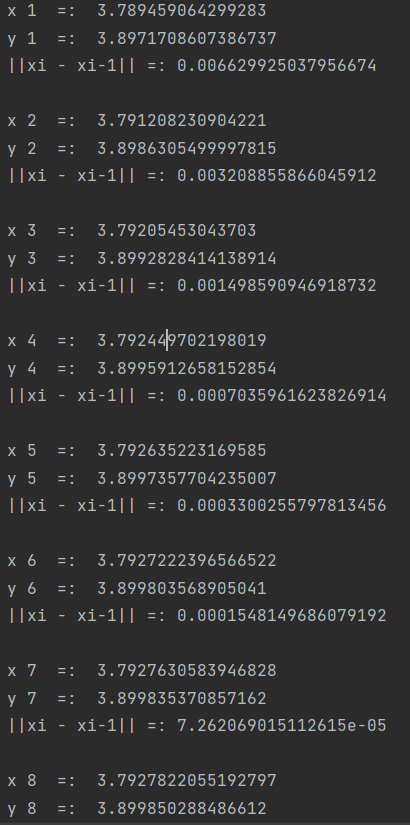
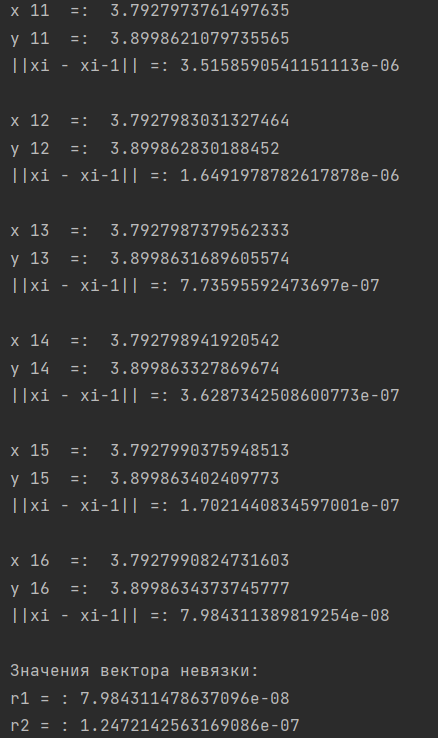
+| - | .

Значение вектора невязки найдём, подставив полученные и в исходные уравнения.

**Код программы:**

import numpy as np  
  
def f1(x0,y0):  
 r1 = y-0.5\*x\*x+x-0.5  
 return r1  
def f2(x0,y0):  
 r2 = 2\*x+y-1/6\*y\*\*3-1.6  
 return r2  
def q1(x0, y0):  
 x0 = np.sqrt(2\*x0+2\*y0-1)  
 return x0  
def q2(x0, y0):  
 y0 = pow(12\*x0+6\*y0-9.6, 1/3)  
 return y0  
norm = 1  
eps = 1e-7  
x0 = 3.785  
y0 = 3.895  
count = 1  
while (norm > eps):  
 x = q1(x0, y0)  
 print("x", count, " =: ", x)  
 y = q2(x0, y0)  
 print("y", count, " =: ", y)  
 norm = np.abs(x-x0) + np.abs(y-y0)  
 print("||xi - xi-1|| =:", norm, "\n")  
 x0 = x  
 y0 = y  
 count += 1  
print("Значения вектора невязки:")  
print("r1 = :", f1(x, y))  
print("r2 = :", f2(x, y))

**Результат работы программы:**

Вектор решений:

Вектор невязки:

= 7.984311478637096e-08, = 1.247214256316e-07

Пункт Б2. Решение СНУ методом Ньютона

с постоянной производной (матрицей Якоби)

Метод Ньютона позволяет решить систему вида

В нашем случае

(x,y) = ,

(x,y) = .6.

Для заданной системы можем построить матрицу Якоби:

J(x,y) = =

= +1, = 1

= 2, = 1 - 0.5

Так как мы используем метод Ньютона с постоянной матрицей Якоби, то итерационный алгоритм нахождения решения будет выглядеть следующим образом:

= 3.785, = 3.895.

+ , где =.

Вычислим начальную матрицу Якоби и найдём обратную к ней.

, = =

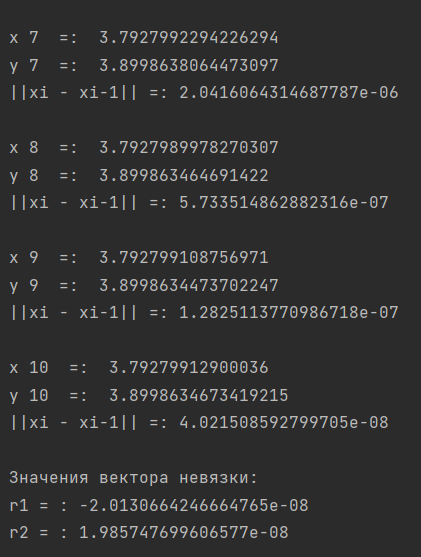
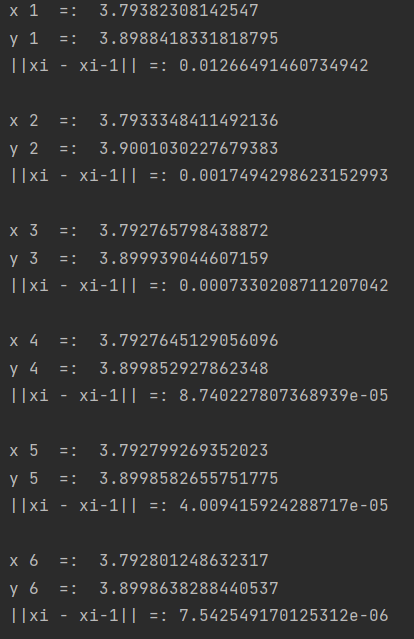
=

Остановимся, когда выполнится: +| - | .

**Код программы:**

import numpy as np  
  
def f1(x0,y0):  
 r1 = y0-0.5\*x0\*x0+x0-0.5  
 return r1  
def f2(x0,y0):  
 r2 = 2\*x0+y0-1/6\*y0\*y0\*y0-1.6  
 return r2  
def mist1(x0,y0):  
 Ji = ([-0.40304067, -0.122402219],  
 [-0.0612011093, -0.170445089])  
 mist1 = -1\*(Ji[0][0]\*f1(x0, y0)+Ji[0][1]\*f2(x0, y0))  
 return mist1  
def mist2(x0,y0):  
 Ji = ([-0.40304067, -0.122402219],  
 [-0.0612011093, -0.170445089])  
 mist2 = -1 \* (Ji[1][0] \* f1(x0, y0) + Ji[1][1] \* f2(x0, y0))  
 return mist2  
norm = 1  
eps = 1e-7  
x0 = 3.785  
y0 = 3.895  
count = 1  
while (norm > eps):  
 mist11 = mist1(x0, y0)  
 mist12 = mist2(x0, y0)  
 x = x0+mist11  
 y = y0+mist12  
 print("x", count, " =: ", x)  
 print("y", count, " =: ", y)  
 norm = np.abs(x-x0) + np.abs(y-y0)  
 print("||xi - xi-1|| =:", norm, "\n")  
 x0 = x  
 y0 = y  
 count += 1  
 print("Значения вектора невязки:")  
 print("r1 = :", f1(x, y))  
 print("r2 = :", f2(x, y))

**Результат работы программы:**



Вектор решений:

Вектор невязки:

= -2.0130664246664765e-08,

= 1.985747699606577e-08.

Пункт В. Анализ и сравнение методов.

Метод простой итерации для решения СНУ в нашем случае сработал за 16 итераций, в то время как для метода Ньютона с постоянной производной потребовалось 10. Это говорит о более высокой скорости сходимости метода Ньютона. Даже при использовании только одной матрицы Якоби в методе Ньютона мы достигаем лучшей скорости сходимости, чем в МПИ.

Для сравнения точности методов рассмотрим вектор невязки для каждого из них. Заметим, что в критерии останова в обоих методах использовалась октаэдрическая норма.

Вектор невязки для МПИ:

= 7.984311478637096e-08,

= 1.247214256316e-07.

Вектор невязки метода Ньютона с постоянной производной :

= -2.0130664246664765e-08,

= 1.985747699606577e-08.

Исходя из полученных данных можно сказать, что порядок точности решения у обоих методов совпадает. Метод Ньютона с постоянной матрицей Якоби обладает нелинейной скоростью сходимости, однако для успешной работы алгоритма нужно решить СЛАУ или найти обратную матрицу Якоби, что может быть затруднительно при большом количестве координат х.