МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО

ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

«Национальный исследовательский ядерный университет

«МИФИ» (НИЯУ МИФИ)

Институт финансовых технологий и экономической безопасности
Кафедра финансового мониторинга

Лабораторная работа №1:

По курсу “Численные методы”

Работу выполнил: студент группы С21-762: Л.К. Хоай

Проверил: Саманчук В.Н.

Москва 2023

Постановка задачи

Найти простой корень многочлена методом простой итерации:

$1{,}78\cdot x^5+3{,}2\cdot x^4-5\cdot x^3-9{,}7\cdot x^2+x-21=0$

Методика решения

Для решения поставленной задачи была написана программа на языке Python, в которой реализован метод простой итерации для решения нелинейного уравнения.

Теоретическая справка

Метод простой итерации — один из простейших численных методов решения уравнений. Метод основан на принципе сжимающего отображения, который применительно к численным методам в общем виде также может называться методом простой итерации или методом последовательных приближений.

Идея метода простой итерации состоит в том, чтобы уравнение $f(x)=0$ привести к эквивалентному уравнению

$$x=\varphi(x),$$

так, чтобы отображение $\varphi(x)$ было сжимающим. Если это удаётся, то последовательность итераций $x_{i+1}=\varphi(x_i)$ сходится. Такое преобразование можно делать разными способами. В частности, сохраняет корни уравнение вида

$$\varphi(x)=x-\lambda(x)f(x),$$

$\lambda(x)\neq 0$ на исследуемом отрезке. Если в качестве $\lambda(x)$ выбрать константу того же знака, что и производная в окрестности корня, то мы получаем простейший метод итерации.

Выберем не которое нулевое приближение x_0 и вычислим дальнейшее приближения:

$$x_{n+1}=\varphi(x_n),\qquad n=0,\ 1,\ 2,\ \ldots \tag{23}$$

Очевидно, если x_n стремится к некоторому пределу $\bar{x}$, то этот предел есть корень исходного уравнения.

Исследуем условия сходимости. Если $\varphi(x)$ имеет непрерывную производную, тогда

$$x_{n+1}-\bar{x}=\varphi(x_n)-\varphi(\bar{x})=(x_n-\bar{x})\,\varphi'(\xi), \tag{24}$$

где точка ξ лежит между точками x_n и $\bar{x}$. Поэтому если всюду $|\varphi'(x)|\leqslant q<1$, то отрезки $|x_n-\bar{x}|$ убывают не медленней членов геометрической прогрессии со знаменателем $q<1$ и последовательность x_n сходится при любом нулевом приближении. Если $|\varphi'(\bar{x})|>1$, то в силу непрерывности $|\varphi'(x)|$ больше единицы и в некоторой окрестности корня; в этом случае итерации не могут сходиться. Если $|\varphi'(\bar{x})|<1$, но вдали от корня $|\varphi'(x)|>1$, то итерации сходятся, если нулевое приближение выбрано достаточно близко к корню; при произвольном нулевом приближении сходимости может не быть.

Решение задачи

```python
from math import fabs
def f(x):
    return 1.78*(x**5)+3.2*(x**4)-5*(x**3)-9.7*(x**2)+x-21
x0 = float(input("x0="))
e = float(input("e="))
print('-------------')
print('x=', x0, 'f(x)=', f(x0))
x1 = x0-f(x0)/100
print('x=', x1, 'f(x)=', f(x1))
while fabs(x1-x0) >= e:
    x0 = x1
    x1 = x0-f(x0)/100
    print('x=', x1, 'f(x)=', f(x1))
```

Результат работы

```
x0=0
e=0.000001
-------------
x= 0.0 f(x)= -21.0
x= 0.21 f(x)= -21.257124638022
x= 0.42257124638022 f(x)= -22.560789621906892
x= 0.6481791425992889 f(x)= -25.020258364264166
x= 0.8983817262419305 f(x)= -28.429645128759848
x= 1.182678177529529 f(x)= -31.2769672522108
x= 1.495447850051637 f(x)= -28.601862498806447
x= 1.7814664750397016 f(x)= -14.103125601112422
x= 1.9224977310508258 f(x)= 0.0033689366434686008
x= 1.922464041684391 f(x)= -0.00070029301870278
x= 1.922471044614578 f(x)= 0.00014553858422416965
x= 1.9224695892287358 f(x)= -3.0247878168410125e-05
x= 1.9224698917075174 f(x)= 6.286484595108277e-06

Process finished with exit code 0
```

Заключение

В работе требовалось решить нелинейное уравнение методом простой итерации. Для решения задачи была написана программа на языке программирования Python.

Ответ: x = 1.9224698917075174

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО

ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

«Национальный исследовательский ядерный университет

«МИФИ» (НИЯУ МИФИ)

Институт финансовых технологий и экономической безопасности
Кафедра финансового мониторинга

Лабораторная работа №2:

По курсу “Численные методы”

Работу выполнил: студент группы С21-762: Л.К. Хоай

Проверил: Саманчук В.Н.

Москва 2023

Постановка задачи

Решить систему нелинейных уравнений методом Ньютона:

$$\begin{cases} x \cdot z^2 + x^2 \cdot y - 5 = -1{,}25 \\ -1{,}5 \cdot x \cdot y \cdot z + 2 \cdot y^3 + 3{,}7 \cdot x^2 = 10{,}825 \\ x \cdot z^3 + 7 \cdot z^2 \cdot y^3 = -16 \end{cases}$$

Методика решения

Для решения поставленной задачи была написана программа на языке Python, в которой реализован метод Ньютона для решения системы нелинейных уравнений.

Теоретическая справка

Рассмотрим систему нелинейных уравнений

$$F(x) = 0,\ F(x),\ x \in \mathbb{R}^n, \tag{1.1}$$

и предположим, что существует вектор $\bar{x} \in D \subset R^n$, являющийся решением системы (1.1). Будем считать, что $F(x) = (f_1(x), f_2(x), \ldots, f_n(x))^T$, причём $f_i(\cdot) \in \mathbb{C}^1(D)\ \forall i$.

Разложим $F(x)$ в окрестности точки $\bar{x}$: $F(x) = F(x^0) + F'(x^0)(x - x^0) + o(\|x - x^0\|)$. Здесь

$$F'(x) = \frac{\partial F(x)}{\partial x} = \begin{bmatrix} \dfrac{\partial f_1(x)}{\partial x_1}, & \dfrac{\partial f_1(x)}{\partial x_2}, & \ldots & \dfrac{\partial f_1(x)}{\partial x_n} \\ \dfrac{\partial f_2(x)}{\partial x_1}, & \dfrac{\partial f_2(x)}{\partial x_2}, & \ldots & \dfrac{\partial f_2(x)}{\partial x_n} \\ \ldots & \ldots & \ldots & \ldots \\ \dfrac{\partial f_n(x)}{\partial x_1}, & \dfrac{\partial f_n(x)}{\partial x_2}, & \ldots & \dfrac{\partial f_n(x)}{\partial x_n} \end{bmatrix}$$

называется матрицей Якоби, а её определитель – якобианом системы (1.1). Исходное уравнение заменим следующим: $F(x^0) + F'(x^0)(x - x^0) = 0$. Считая матрицу Якоби $F'(x^0)$ неособой, разрешим это уравнение относительно x: $\hat{x} = x^0 - [F'(x)]^{-1}F(x^0)$. И вообще положим

$$x^{k+1} = x^k - [F'(x^k)]^{-1}F(x^k). \tag{1.2}$$

При сделанных относительно $F(\cdot)$ предположениях имеет место сходимость последовательности $\{x^k\}$ к решению системы со скоростью геометрической прогрессии при условии, что начальное приближение x^0 выбрано из достаточно малой окрестности решения $\bar{x}$.

При дополнительном предположении $F(\cdot) \in \mathbb{C}^2[a, b]$ имеет место квадратичная сходимость метода, т.е.

$$\|x^{k+1} - \bar{x}\| \le \omega \|x^k - \bar{x}\|^2.$$

Решение задачи

```
import numpy
def f1(x,y,z):
    return x*z*z+x*x*y-3.75
def f2(x,y,z):
    return -1.5*x*y*z+2*(y**3)+3.7*x*x-10.825
def f3(x,y,z):
    return x*(z**3)+7*z*z*(y**3)+16
def f1x(x,y,z):
    return z*z+2*x*y
def f1y(x,y,z):
    return x*x
def f1z(x,y,z):
    return 2*x*z
def f2x(x,y,z):
    return -1.5*y*z+7.4*x
def f2y(x,y,z):
    return -1.5*x*z+6*y*y
def f2z(x,y,z):
    return -1.5*x*y
def f3x(x,y,z):
    return z**3
def f3y(x,y,z):
    return 21*z*z*y*y
def f3z(x,y,z):
    return 14*z*(y**3)

print("Решить систему нелинейных уравнений методом Ньютона:")
print("xz^2+x^2*y-5=1.25\n-1.5xyz+2y^3+3.7x^2=10.825\nxz^3+7z^2*y^3=-16")
x = float(input("x0="))
y = float(input("y0="))
z = float(input("z0="))
e = float(input("e="))
print('x=', x, 'y=', y, 'z=', z, 'f1=', f1(x, y, z),
      'f2=', f2(x, y, z), 'f3=', f3(x, y, z))
A = numpy.matrix([[f1x(x, y, z), f1y(x, y, z), f1z(x, y, z)],
                  [f2x(x, y, z), f2y(x, y, z), f2z(x, y, z)],
                  [f3x(x, y, z), f3y(x, y, z), f3z(x, y, z)]])
B = numpy.array([-f1(x, y, z), -f2(x, y, z), -f3(x, y, z)])
delta = numpy.linalg.solve(A, B)
x = x + delta[0]
y = y + delta[1]
z = z + delta[2]
print('x=', x, 'y=', y, 'z=', z, 'f1=', f1(x, y, z),
      'f2=', f2(x, y, z), 'f3=', f3(x, y, z))

while (delta[0]*delta[0]+delta[1]*delta[1]+delta[2]*delta[2]>e*e):
    A = numpy.matrix([[f1x(x, y, z), f1y(x, y, z), f1z(x, y, z)],
                      [f2x(x, y, z), f2y(x, y, z), f2z(x, y, z)],
                      [f3x(x, y, z), f3y(x, y, z), f3z(x, y, z)]])
    B = numpy.array([-f1(x, y, z), -f2(x, y, z), -f3(x, y, z)])
    delta = numpy.linalg.solve(A, B)
    x = x + delta[0]
    y = y + delta[1]
    z = z + delta[2]
    print('x=', x, 'y=', y, 'z=', z, 'f1=', f1(x, y, z), 'f2=',
          f2(x, y, z), 'f3=', f3(x, y, z))
```

Результат работы

```
Решить систему нелинейных уравнений методом Ньютона:
xz^2+x^2*y-5=1.25
-1.5xyz+2y^3+3.7x^2=10.825
xz^3+7z^2*y^3=-16
x0=1
y0=-1
z0=3
e=0.00000001
x= 1.0 y= -1.0 z= 3.0 f1= 4.25 f2= -4.624999999999999 f3= -20.0
x= 1.5948783255912162 y= -1.2804120038006757 z= 1.6443772874436935 f1= -2.694389412605526 f2= -0.5749214097010746 f3= -16.6414935161773
x= 1.4350215976048453 y= -0.9359163446872011 z= 1.948939607760725 f1= -0.22658368422596764 f2= -0.9189354280491031 f3= 4.825696099372829
x= 1.5011388468394875 y= -0.9961317351198473 z= 1.9984692105825603 f1= 0.0006661747480793956 f2= 0.01832849769328604 f3= 0.3475860903604371
x= 1.4999810119279247 y= -0.9997217372600599 z= 1.9999000798289925 f1= 7.58842031656215e-06 f2= -7.55400396172945e-05 f3= 0.024212571658940973
x= 1.4999990637695761 y= -0.9999809240896513 z= 1.999993003447751 f1= 5.2995408061917715e-09 f2= -3.310423934976825e-07 f3= 0.0016648105474228458
x= 1.4999999357711455 y= -0.9999986591000718 z= 1.999999507873797 f1= 3.899325307088475e-11 f2= -1.5698944366704382e-09 f3= 0.00011704281956426144
x= 1.4999999954885863 y= -0.9999999056588776 z= 1.999999965374014 f1= 1.9628743075372768e-13 f2= -7.759126674500294e-12 f3= 8.234821812891369e-06
x= 1.499999999682608 y= -0.9999999933620407 z= 1.999999997563664 f1= 1.3322676295501878e-15 f2= -4.263256414560601e-14 f3= 5.794128021818779e-07
x= 1.499999999977668 y= -0.9999999995329427 z= 1.9999999998285753 f1= -8.881784197001252e-16 f2= 3.552713678800501e-15 f3= 4.076840021127737e-08

Process finished with exit code 0
```

Заключение

В работе требовалось решить систему нелинейных уравнений методом Ньютона. Для решения задачи была написана программа на языке программирования Python.

Ответ: x = 1.499999999977668
y = -0.9999999995329427
z = 1.9999999998285753