Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №7

Чисельні методи

Розв’язування системи нелінійних рівнянь

Варіант 17

Виконала
студентка групи ІТ-91:

Луцай К. А.

Перевірила:

ас. Тимофєєва Ю. С.

Київ 2021

Мета: навчитися розв’язувати систему нелінійних рівнянь за допомогою мови програмування Python, використати методи Ньютона та простих ітерацій.

Теоретичні відомості:

Систему нелінійних рівнянь із n невідомими можна записати у вигляді

$$\begin{cases} f_1(x_1, x_2, \ldots, x_n) = 0 \\ f_2(x_1, x_2, \ldots, x_n) = 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ f_n(x_1, x_2, \ldots, x_n) = 0 \end{cases}$$

Або

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{pmatrix}, \qquad \mathbf{f} = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \ldots \\ f_n(\mathbf{x}) \end{pmatrix}, \qquad \mathbf{0} = \begin{pmatrix} 0 \\ 0 \\ \ldots \\ 0 \end{pmatrix}.$$

Метод ньютона

Якщо визначено початкове наближення x , ітераційний процес знаходження розв’язку системи методом Ньютона можна подати у вигляді

$$\begin{cases} x_1^{(k+1)} = x_1^{(k)} + \Delta x_1^{(k)} \\ x_2^{(k+1)} = x_2^{(k)} + \Delta x_2^{(k)} \\ \ldots\ldots\ldots\ldots\ldots \\ x_n^{(k+1)} = x_n^{(k)} + \Delta x_n^{(k)} \end{cases} \qquad k = 0, 1, 2, \ldots$$

У векторно-матричній формі розрахункові формули мають вигляд

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{\Delta x}^{(k)} \qquad k = 0, 1, 2, \ldots$$

де вектор приростів $\mathbf{\Delta x}^{(k)} = \begin{pmatrix} \Delta x_1^{(k)} \\ \Delta x_2^{(k)} \\ \ldots \\ \Delta x_n^{(k)} \end{pmatrix}$ отримується з розв’язку рівняння

$$\mathbf{f}(\mathbf{x}^{(k)}) + \mathbf{J}(\mathbf{x}^{(k)})\mathbf{\Delta x}^{(k)} = \mathbf{0}$$

Тут $\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} \cdots & \frac{\partial f_2(\mathbf{x})}{\partial x_n} \\ \cdots & \cdots & \cdots \\ \frac{\partial f_n(\mathbf{x})}{\partial x_1} & \frac{\partial f_n(\mathbf{x})}{\partial x_2} \cdots & \frac{\partial f_n(\mathbf{x})}{\partial x_n} \end{bmatrix}$ – матриця Якобі перших похідних векторів-функції $\mathbf{f}(\mathbf{x})$.

Ітераційний процес знаходження розв'язку можна записати у вигляді

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{J}^{-1}(\mathbf{x}^{(k)})\mathbf{f}(\mathbf{x}^{(k)}) \qquad k = 0,1,2,\ldots$$

Метод простих ітерацій

Коли використовується метод простої ітерації, система рівнянь приводиться до еквівалентної системи спеціального вигляду

$$\begin{cases} x_1 = \varphi_1(x_1, x_2, \ldots, x_n) \\ x_2 = \varphi_2(x_1, x_2, \ldots, x_n) \\ \cdots\cdots\cdots\cdots\cdots \\ x_n = \varphi_n(x_1, x_2, \ldots, x_n) \end{cases}$$

Якщо обрано деяке початкове наближення x, наступні наближення в методі простої ітерації знаходять за формулами:

$$\begin{cases} x_1^{(k+1)} = \varphi_1(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}) \\ x_2^{(k+1)} = \varphi_2(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}) \\ \cdots\cdots\cdots\cdots\cdots \\ x_n^{(k+1)} = \varphi_n(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}) \end{cases} \qquad k = 0,1,2,\ldots$$

або у векторній формі

$$\mathbf{x}^{(k+1)} = \boldsymbol{\varphi}(\mathbf{x}^{(k)}), \; k = 0,1,2,\ldots$$

Розв'язок задачі в аналітичній формі:

$$ax_1 - \cos x_2 = 0$$

$$ax_2 - e^{x_1} = 0$$

a = 3

Метод простих ітерацій

x = cos(y)/3

y = (e^x)/3

X0 = [0, 0]

X1 = [1/3, 1/3]

X2 = [0.3145, 0.465]

X3 = [0.298, 0.457]

X4 = [0.299, 0.449]

X5 = [0.3, 0.45]

Метод Ньютона

J = [[3, sin(y)],[e^x, 3]]

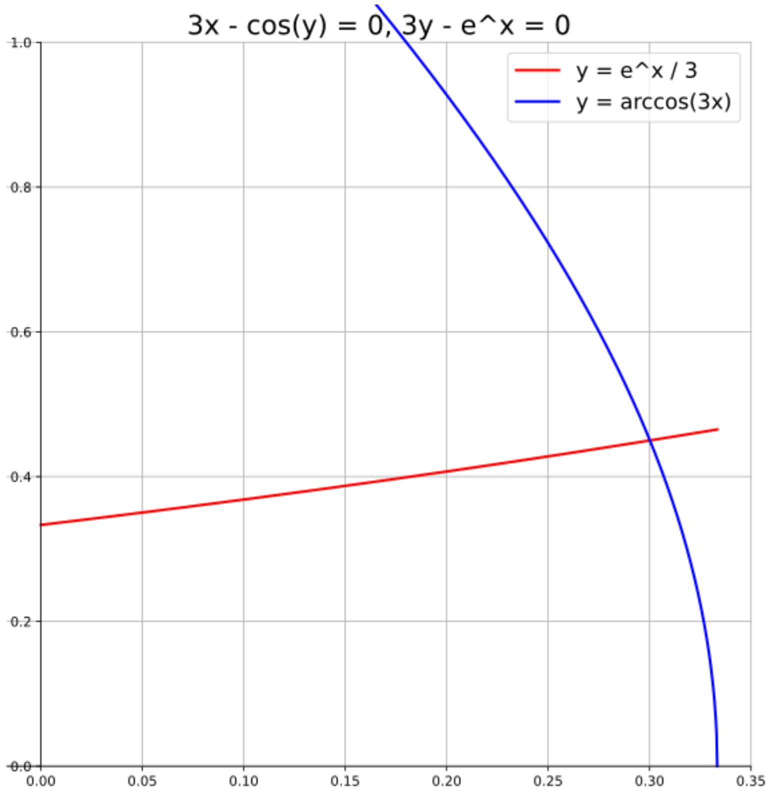
X0 = [0, 0]

X1 = [1/3, 0.222]

X2 = [0.306, 0.478]

X3 = [0.299, 0.456]

X4 = [0.3, 0.45]



Лістинги програм розв’язування задачі:

Початкові задані за умовою значення подані у вигляді функцій, що приймають вектор значень x. Для метода Ньютона також записано Якобіан

```
import math
import numpy as np

def FUNC_newt(X):
    return np.array([3*X[0] - math.cos(X[1]), 3*X[1] - math.exp(X[0])])

def JACOB(X):
    return np.array([[3, math.sin(X[1])],
                     [math.exp(X[0]), 3]])

def FUNC_iter(X):
        return np.array([math.cos(X[1])/3, math.exp(X[0])/3])
```

В якості аргументів методи приймають вектор функцій та Якобіан у випадку Ньютона

```
def newton(func, jacob):
    X = np.array([0, 0], dtype=float)
    for it in range(10):
        J = jacob(X)
        Y = func(X)
        X = X - np.linalg.solve(J, Y)
    return X

def simpiter(func):
    X = np.array([0, 0], dtype=float)
    for it in range(10):
        X = func(X)
    return X
```

Виклик методів та графічне представлення:

```
print("3x - cos(y) = 0")
print("3y - e^x = 0")
print("Метод Ньютона:", newton(FUNC_newt, JACOB))
print("Метод простих ітерацій:", simpiter(FUNC_iter))

x = np.linspace(0, 1/3, 1000)
y1 = np.exp(x)/3
y2 = np.arccos(3*x)
fig = plt.figure(figsize=(10, 10))
ax = plt.subplot(ylim=(0, 1))
ax.plot(x, y1, color="red", label="y = e^x / 3", lw=2)
ax.plot(x, y2, color="blue", label="y = arccos(3x)", lw=2)
ax.legend(fontsize=16)
ax.grid(True)
plt.title("3x - cos(y) = 0, 3y - e^x = 0", fontsize=20)
plt.show()
```

Результати виконання програми:

```
3x - cos(y) = 0
3y - e^x = 0
Метод Ньютона: [0.30014631 0.45001877]
Метод простих ітерацій: [0.30014658 0.45001905]
```

Висновки: було написано програму для розв'язання системи нелінійних рівнянь на мові Python, було реалізовано методи Ньютона та простих ітерацій.