Белорусский государственный университет
Факультет прикладной математики и информатики

Лабораторная работа №2

Решение системы нелинейных уравнений

Вариант №4

Выполнил:
Студент 2 курса 7 группы ФПМИ
Лубенько Алексей Анатольевич

Преподаватель:
Будник Анатолий Михайлович

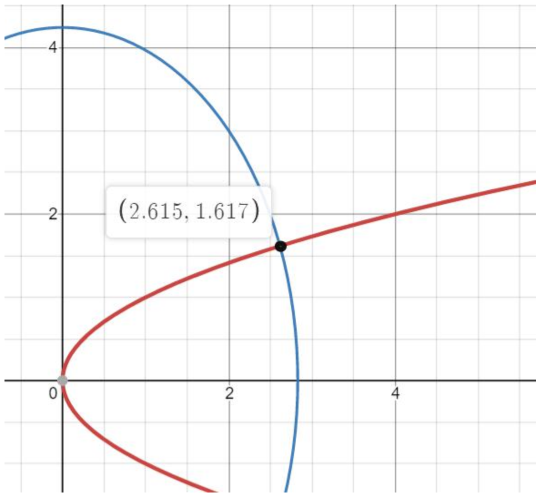
Минск, 2022

Алгоритм решения

Задана система:

$$\begin{cases} \dfrac{x^2}{4} + \dfrac{y^2}{9} = 2 \\ x = y^2 \end{cases}$$

Отделим корень графически:



В качестве начального приближения возьмем $z_0 := (x_0, y_0) = (2.6, 1.6)$

Возьмем $\delta = 0.1$

Тогда $S_\delta = [2.5, 2.7] \times [1.5, 1.7]$

Метод Гаусса-Зейделя с реализацией по методу Ньютона

Запишем нашу систему в виде

$$\begin{cases} \dfrac{x^2}{4} + \dfrac{y^2}{9} - 2 = 0 \\ x - y^2 = 0 \end{cases}$$

Сам алгоритм может быть записан в виде:

$$\begin{cases} \dfrac{(x_{k+1})^2}{4} + \dfrac{(y_k)^2}{9} - 2 = 0 \\ x_{k+1} - (y_{k+1})^2 = 0 \end{cases} \quad k = 0,1.. \; y_0 \, , x_0$$

где

$$x_{k+1}^{s+1} = x_{k+1}^{s} - \left(\frac{\dfrac{(x_{k+1}^s)^2}{4} + \dfrac{(y_k)^2}{9} - 2}{\dfrac{x_{k+1}^s}{2}} \right) \quad s = 0,1.. \; x_{k+1}^0 = x_k$$

$$y_{k+1}^{s+1} = y_{k+1}^{s} - \left(\frac{x_{k+1} - (y_{k+1}^s)^2}{-2y_{k+1}^s} \right) \quad s = 0,1.. \, y_{k+1}^0 = y_k$$

Критерий остановы: $\|z_{k+1} - z_k\| < 10^{-7}$, где норма — кубическая.

Критерий остановы внутренних процессов соответственно:

$|x_{k+1}^s - x_{k+1}^{s-1}| < 10^{-7}$ и $|y_{k+1}^s - y_{k+1}^{s-1}| < 10^{-7}$

Погрешность вычислим по формуле:

$$\max\left(\frac{(x_{k+1})^2}{4} + \frac{(y_{k+1})^2}{9} - 2, (x_{k+1}) - (y_{k+1})^2\right)$$

Листинг программы

```
import numpy.linalg as LA
import numpy as np
x = 2.6
y = 1.6
eps = 10 ** -7
def f1(x, y):
    return x ** 2 / 4 + y ** 2 / 9 - 2
def f2(x, y):
    return x - y ** 2
def J(x, y):
    return np.array([
        [x / 2, 2 * y / 9],
        [1, -2 * y]
    ])
def F(x, y):
    return np.array([
        [x ** 2 / 4 + y ** 2 / 9 - 2],
        [x - y ** 2]
    ])
table = [[x, y, '---']]
k = 0
while max(LA.norm(f1(x, y)), LA.norm(f2(x, y))) > eps:
    prev_x= x
    x= prev_x- f1(x, y) / J(x, y)[0, 0]
    while abs(x- prev_x) > eps:
        prev_x= x
        x= prev_x- f1(x, y) / J(x, y)[0, 0]

    prev_y= y
    y= prev_y- f2(x, y) / J(x, y)[1, 1]
    while abs(y- prev_y) > eps:
        prev_y= y
        y= prev_y- f2(y, y) / J(x, y)[1, 1]
    table.append([x, y,
                  max(LA.norm(f1(x, y)), LA.norm(f2(x, y)))
                  ])
    k += 1
print(f'Решение: {x, y}')
print(f'Количество итераций на внешнем цикле: {k}')
print(f'Погрешность: {LA.norm(F(x, y))}')
```

Вывод программы

```
Решение: (2.614921164841173, 1.6170717871638165)
Количество итераций на внешнем цикле: 6
Погрешность: 2.932295628532775e-08
```

Метод секущих

Будем находить новое приближение в виде $z^{k+1} = z^k + \Delta z^k$

где Δz^k будем находить как решение системы $J(z^k)\Delta z^k = -f(z^k)$

$$J(z^k) = \begin{bmatrix} \dfrac{\frac{(x^k)^2}{4} + \frac{(y^k)^2}{9} - \frac{(x^{k-1})^2}{4} - \frac{(y^k)^2}{9}}{x^k - x^{k-1}} & \dfrac{\frac{(x^k)^2}{4} + \frac{(y^k)^2}{9} - \frac{(x^k)^2}{4} - \frac{(y^{k-1})^2}{9}}{y^k - y^{k-1}} \\ \dfrac{x^k - (y^k)^2 - x^{k-1} + (y^k)^2}{x^k - x^{k-1}} & \dfrac{x^k - (y^k)^2 - x^k + (y^{k-1})^2}{y^k - y^{k-1}} \end{bmatrix}$$

$$= \begin{bmatrix} \dfrac{\frac{(x^k)^2}{4} - \frac{(x^{k-1})^2}{4}}{x^k - x^{k-1}} & \dfrac{\frac{(y^k)^2}{9} - \frac{(y^{k-1})^2}{9}}{y^k - y^{k-1}} \\ 1 & \dfrac{-(y^k)^2 + (y^{k-1})^2}{y^k - y^{k-1}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{4}(x^k + x^{k-1}) & \frac{1}{9}(y^k + y^{k-1}) \\ 1 & -y^k - y^{k-1} \end{bmatrix}$$

Критерий остановы: $\|z_{k+1} - z_k\| < 10^{-7}$, где норма — кубическая.

Погрешность вычислим по формуле из предыдущего пункта.

Листинг программы

```
import numpy.linalg as LA
import numpy as np
x1 = 2.6
y1 = 1.6
eps = 10 ** -7
def f1(x, y):
    return x ** 2/4 + y ** 2 / 9 - 2
def f2(x, y):
    return x-y*y
def norm(x,y):
    return abs(max(x,y))
dx=np.array([[0.],[0.]])
f=np.array([[0.],[0.]])
a=np.array([[0.,0.],[0.,0.]])
x0=0
y0=0
x2=0
y2=0
count=0
while norm(f1(x1, y1), f2(x1, y1)) >= eps:
    a[0, 0] = 1/4*(x1+x0)
    a[0, 1] = 1/9*(y1+y0)
    a[1, 0] = 1
    a[1, 1] = -y1-y0
    f[0] = -f1(x1, y1)
    f[1] = -f2(x1, y1)
    dx=LA.solve(a,f)
    x2 = x1 + dx[0]
    y2 = y1 + dx[1]
    x0 = x1
    y0 = y1
    y1 = y2
    x1 = x2
    count+=1
print("Решение: ",(x1,y1))
print("Количество итераций: ", count)
print("Погрешность: ",norm(f1(x1, y1), f2(x1, y1)))
```

Выходные данные:

```
Решение:  (2.61492119, 1.61707179)
Количество итераций:  6
Погрешность:  1.87281302e-11
```

Вывод:

Метод секущих, как и метод Гаусса-Зейделя с реализацией по методу Ньютона, сошелся за 6 итераций. Ответ с большей точностью получен методом секущих (порядка 10^{-11}), т.к. в методе Гаусса-Зейделя внутренние итерации проводились до точности 10^{-7}.