Университет ИТМО
МФ КТиУ, Ф ПИиКТ

Лабораторная работа №2
Дисциплина «Вычислительная математика»

Численное решение нелинейных уравнений и систем

Выполнил
Галлямов Камиль Рустемович

Преподаватель:
Машина Екатерина
Алексеевна

г. Санкт-Петербург
2024 г.

Решение нелинейного уравнения

Вид уравнения: $\mathbf{x^3+2.84x^2-5.606x-14.766=0}$

1. Графическое отделение корней:

2. Интервалы изоляции корней:

Для левого корня: [-3.2; -3]

Для центрального корня: [-2.2; -2]

Для правого корня: [2.2; 2.4]

3. Уточнение корней с точностью eps = 10^{-2}:

Левый корень – 5 (метод простой итерации)

Центральный корень – 3 (метод Ньютона)

Правый корень – 1 (метод половинного деления)

Уточнение правого корня (метод половинного деления):

№ шага	a	b	x	f(a)	f(b)	f(x)	|a-b|
1	2.2	2.4	2.3	-2.706	1.962	-0.469	0.2
2	2.3	2.4	2.35	-0.469	1.962	0.722	0.1
3	2.3	2.35	2.325	-0.469	0.722	0.120	0.05
4	2.3	2.325	2.3125	-0.469	0.120	-0.176	0.025
5	2.3125	2.325	2.31875	-0.176	0.120	-0.028	0.0125
6	2.31875	2.325	**2.321875**	-0.028	0.120	0.046	0.00625<eps

Уточнение центрального корня (метод Ньютона):

$f(x) = x^3+2.84x^2-5.606x-14.766$
$f'(x) = 3x^2+5.68x-5.606$
$f''(x) = 6x + 5.68$
Производные сохраняют знак на интервале изоляции, поэтому метод Ньютона эффективен.
Начальное приближение: $x_0 = -2$
$f(-2) = -0.194$
$f''(-2) = -6.32$
Знаки функции и второй производной совпадают, поэтому это подходящее начальное приближение.

№ итерации	x_k	$f(x_k)$	$f'(x_k)$	x_{k+1}	$\| x_{k+1} - x_k\|$
0	-2	-0.194	-4.966	-2 - 0.194 / 4.966 = -2.039	0.039
1	**-2.039**	-0.005	-4.715	-2.039 - 0.005 / 4.715 = -2.040	0.001<eps
2	-2.04	-0.0005	-4.708	-2.04 - 0.0005 / 4.708 = -2.0401	

Уточнение левого корня (метод простой итерации):

$x^3+2.84x^2-5.606x-14.766 = 0$
$x = (x^3+2.84x^2-14.766) / 5.606$
$fi(x) = (x^3+2.84x^2-14.766) / 5.606$
$fi'(x) = (3x^2 + 5.68x) / 5.606$
На отрезке [-3.2; -3] условие сходимости не выполняется.

Введём ненулевой параметр h.
$(x^3+2.84x^2-5.606x-14.766) * h = 0$
$x = (x^3+2.84x^2-5.606x-14.766) * h + x$
$fi(x) = (x^3+2.84x^2-5.606x-14.766) * h + x$

fi’(x) = $(3x^2 + 5.68x – 5.606) * h + 1$

Возьмём h = -0.1. |fi’(x)| <= q <1

На отрезке [-3.2; -3] условие сходимости выполняется.

fi(x) = $(x^3+2.84x^2-5.606x-14.766) * -(0.1) + x$

№ итерации	x_k	x_{k+1}	f'(x_{k+1})	| x_{k+1} - x_k|
0	-3.2	-3.149	-0.177	0.051
1	-3.149	-3.131	-0.066	0.018
2	-3.131	**-3.124**	-0.025	0.007 <eps
3	-3.124	-3.122		
4	-3.122	-3.121		
5	-3.121	-3.120		
6	-3.120	-3.1199		

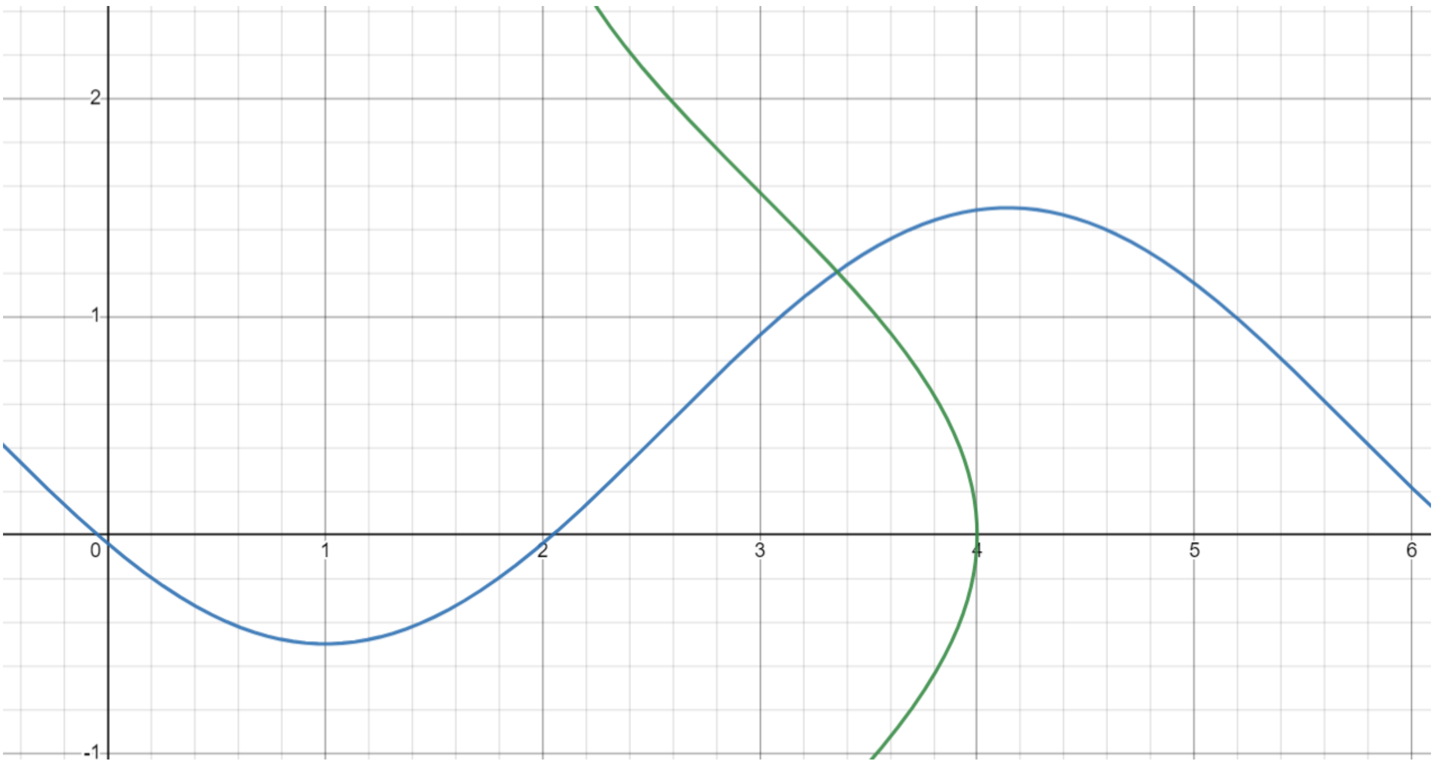
Решение системы нелинейных уравнений

Вид системы:

cos (x – 1) + y = 0.5

x – cos (y) = 3

1. Графическое отделение корней:



Интервал изоляции x: [3.2; 3.6]

Интервал изоляции y: [1; 1.4]

2. Решение системы с точностью eps = 10^{-2}:

y = - cos (x – 1) + 0.5

x = cos (y) + 3

$$\begin{cases} x = \cos(y) + 3 \\ y = -\cos(x-1) + 0.5 \end{cases}$$

Начальные приближения: x = 3.4, y = 1.2

fi_x = $\mathbf{\cos(y) + 3}$

fi_y = $\mathbf{-\cos(x-1) + 0.5}$

$(fi_x)'_x = 0$ $(fi_x)'_y = -\sin(y)$ $|(fi_x)'_x| + |(fi_x)'_y| = |\sin(y)|$

$(fi_y)'_x = \sin(x - 1)$ $(fi_y)'_y = 0$ $|(fi_y)'_x| + |(fi_y)'_y| = |\sin(x - 1)|$

1 шаг.

$x^{(1)} = fi_x(x^{(0)}, y^{(0)}) = \cos(1.2) + 3 = 3.362$ $|x^{(1)} - x^{(0)}| = 0.038$

$y^{(1)} = fi_y(x^{(0)}, y^{(0)}) = -\cos(2.4) + 0.5 = 1.237$ $|y^{(1)} - y^{(0)}| = 0.037$

2 шаг.

$x^{(2)} = fi_x(x^{(1)}, y^{(1)}) = \cos(1.237) + 3 = 3.328$ $|x^{(2)} - x^{(1)}| = 0.034$

$y^{(2)} = fi_y(x^{(1)}, y^{(1)}) = -\cos(2.362) + 0.5 = 1.211$ $|y^{(2)} - y^{(1)}| = 0.026$

3 шаг.

$x^{(3)} = fi_x(x^{(2)}, y^{(2)}) = \cos(1.211) + 3 = 3.352$ $|x^{(2)} - x^{(1)}| = 0.024$

$y^{(3)} = fi_y(x^{(2)}, y^{(2)}) = -\cos(2.328) + 0.5 = 1.187$ $|y^{(2)} - y^{(1)}| = 0.024$

4 шаг.

$x^{(4)} = fi_x(x^{(3)}, y^{(3)}) = \cos(1.187) + 3 = 3.374$ $|x^{(2)} - x^{(1)}| = 0.022$

$y^{(4)} = fi_y(x^{(3)}, y^{(3)}) = -\cos(2.352) + 0.5 = 1.204$ $|y^{(2)} - y^{(1)}| = 0.017$

5 шаг.

$x^{(5)} = fi_x(x^{(4)}, y^{(4)}) = \cos(1.204) + 3 = 3.359$ $|x^{(5)} - x^{(4)}| = 0.015$

$y^{(5)} = fi_y(x^{(4)}, y^{(4)}) = -\cos(2.374) + 0.5 = 1.220$ $|y^{(5)} - y^{(4)}| = 0.016$

6 шаг.

$x^{(6)} = fi_x(x^{(5)}, y^{(5)}) = \cos(1.220) + 3 = 3.344$ $|x^{(6)} - x^{(5)}| = 0.015$

$y^{(6)} = fi_y(x^{(5)}, y^{(5)}) = -\cos(2.359) + 0.5 = 1.209$ $|y^{(6)} - y^{(5)}| = 0.011$

7 шаг.

$x^{(7)} = fi_x(x^{(6)}, y^{(6)}) = \cos(1.209) + 3 = \mathbf{3.354}$ $|x^{(6)} - x^{(5)}| = 0.01 <= eps$

$y^{(7)} = fi_y(x^{(6)}, y^{(6)}) = -\cos(2.344) + 0.5 = \mathbf{1.198}$ $|y^{(6)} - y^{(5)}| = 0.009 <= eps$

Программная реализация нелинейных уравнений и систем

Методы: 1 (метод половинного деления), 4 (метод секущих), 5 (метод простой итерации).

```
import matplotlib.pyplot as plt
from math import sin, e


def bisection_method(func, a, b, eps):
    while abs(a - b) > eps:
        x = (a + b) / 2
        if func(a) * func(x) > 0:
            a = x
        else:
            b = x
    return (a + b) / 2


def secant_method(func, x0, eps):
    x1 = x0 - func(x0) / get_derivative_at_point(func, x0)
    while abs(x1 - x0) >  eps:
        x2 = x1 - (x1 - x0) * f(x1) / (f(x1) - f(x0))
        x0, x1 = x1, x2
    return x1


def simple_iteration_method(func, x0, a, b, eps):
    max_func_ = 0
    x = a
    while x < b:
        max_func_ = max(max_func_, abs(get_derivative_at_point(func, x)))
        x += eps
    if get_derivative_at_point(func, a) > 0:
        h = -1 / max_func_
    else:
        h = 1 / max_func_
    fi = lambda x: x + h * func(x)
    fi_ = lambda x: 1 + h * get_derivative_at_point(func, x)

    x = fi(x0)
    while abs(x - x0) > eps:
        x, x0 = fi(x), x
    return x


def verify(func, a, b, eps=0.00001):
    # true - if there is only one root, false - else
    if func(a) * func(b) < 0 and get_derivative_at_point(func, a) *
get_derivative_at_point(func, b) > 0:
        x = a
        while x < b:
            if get_derivative_at_point(func, a) * get_derivative_at_point(func, x) <= 0:
                return False
            x += eps
        return True
    return False


def get_derivative_at_point(func, x0, dx=0.000001):
    return (func(x0 + dx) - func(x0)) / dx


def draw_plot(func, a, b, root, eps):
    xs = []
```

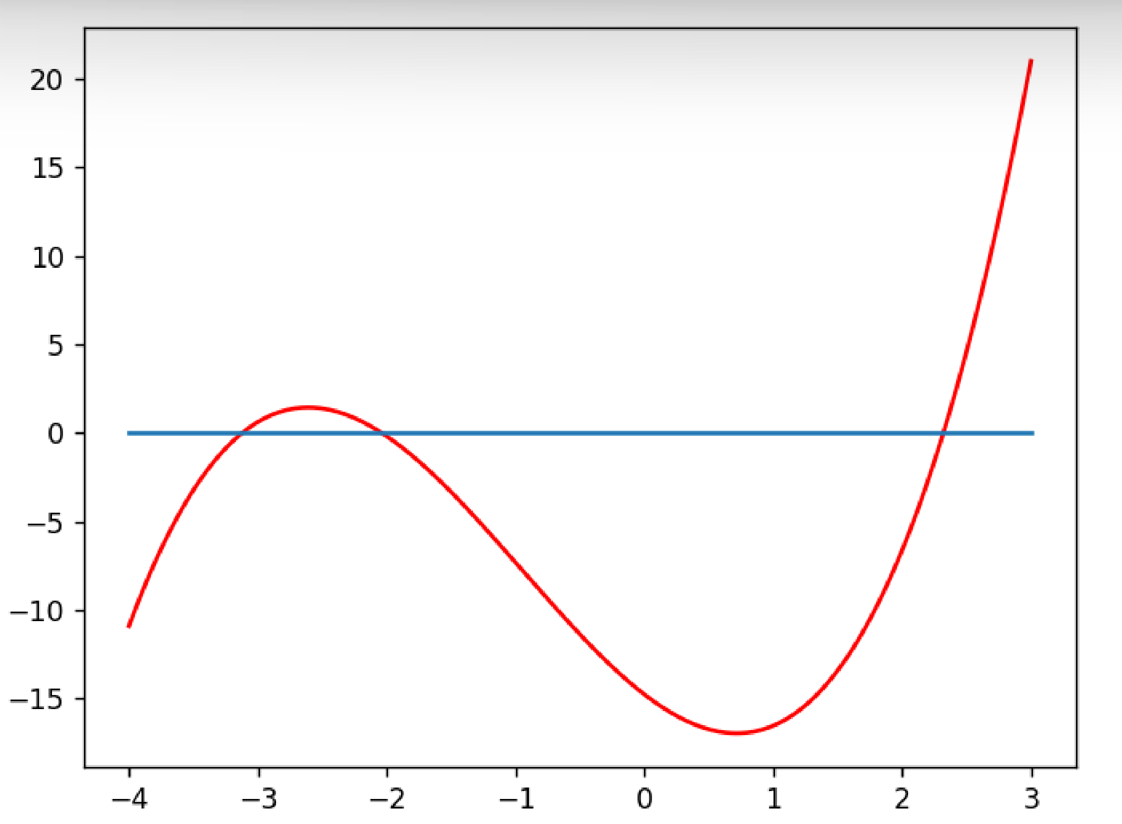
```
    ys = []
    x = a
    while x < b:
        xs.append(x)
        ys.append(func(x))
        x += eps
    plt.xlabel("X")
    plt.ylabel("Y")
    plt.plot(xs, ys, 'g')
    #  plt.plot([root], [func(root)], 'r')
    plt.annotate('x', xy=(root, func(root)))
    plt.plot([a, b], [0, 0], 'b')
    plt.show()


print('variants of functions:')
print('1. x ** 3 + 2.84 * x ** 2 - 5.606 * x - 14.766')
print('2. x ** 3 - x + 4')
print('3. sin(x ** 2) + x + 2')
print('4. e ** sin(x) + x ** 7 - 3')
print('input the number of function (1 or 2 or 3 or 4): ')
case_number = input()
while case_number not in {'1', '2', '3', '4'}:
    print('input the number of function (1 or 2 or 3 or 4):')
    case_number = input()

case_number = int(case_number)
eps = 0.0001
if case_number == 1:
    f = lambda x: x ** 3 + 2.84 * x ** 2 - 5.606 * x - 14.766
    isolation_intervals = [(-3.2, -3), (-2.2, -2), (2.2, 2.4)]
    a, b = isolation_intervals[2]
elif case_number == 2:
    f = lambda x: x ** 3 - x + 4
    a, b = -2, -1
elif case_number == 3:
    f = lambda x: sin(x ** 2) + x + 2
    a, b = -2, -1.6
else:
    f = lambda x: e ** sin(x) + x ** 7 - 3
    a, b = 0.6, 1
fl = input('input "Y" if you want to set [a; b]: ')
if fl == 'Y':
    while 1:
        print('input a and b with endline:')
        try:
            a, b = float(input()), float(input())
        except Exception:
            continue
        break
x0 = (a + b) / 2
print("Проверка:", verify(f, a, b))
if not verify(f, a, b):
    exit(0)
print("Метод деления пополам:", bisection_method(f, a, b, eps))
print("Метод секущих:", secant_method(f, x0, eps))
print("Метод простой итерации:", simple_iteration_method(f, x0, a, b, eps))
root = bisection_method(f, a, b, eps)
draw_plot(f, -4, 3, root, 0.001)
```

Метод простой итерации для систем:
```
import matplotlib.pyplot as plt
from math import cos, sin


def simple_iteration_method(func1, func2, x01, x02, a1, b1, a2, b2, eps):
    x1 = func1(x01)
    x2 = func2(x02)
    while abs(x1 - x01) > eps or abs(x2 - x02) > eps:
        x2, x02 = func1(x1), x2
        x1, x01 = func2(x2), x1
    return x1, x2


def get_derivative_at_point(func, x0, dx=0.000001):
    return (func(x0 + dx) - func(x0)) / dx


def get_derivative_of_x_at_point(func, x0, y0, dx=0.001):
    return (func(x0 + dx, y0) - func(x0, y0)) / dx


def get_derivative_of_y_at_point(func, x0, y0, dy=0.001):
    return (func(x0, y0 + dy) - func(x0, y0)) / dy


def draw_plots(func1, func2, a, b, root, eps):
    xs1, xs2 = [], []
    ys1, ys2 = [], []
    x = a
    while x < b:
        xs1.append(x)
        ys1.append(func1(x))

        xs2.append(func2(x))
        ys2.append(x)
        x += eps

    plt.xlabel("X")
    plt.ylabel("Y")
    plt.annotate('x', xy=(root, func1(root)))
    plt.plot(xs1, ys1, 'r')
    plt.plot(xs2, ys2, 'g')
    plt.plot([a, b], [0, 0], 'b')

    plt.show()


print('variants of systems:')
print('1. sin(x + 1) - y == 0 and 2x + cosy = 2')
print('2. cos (x - 1) + y == 0.5 and x - cos (y) == 3')
print('input the number of system (1 or 2):')
case_number = input()
while case_number not in {'1', '2'}:
    print('input the number of function (1 or 2):')
    case_number = input()

case_number = int(case_number)
if case_number == 1:
    f1 = lambda x: sin(x + 1)
    f2 = lambda y: 1 - cos(y) / 2
    a1, b1 = 0.6, 0.8
    a2, b2 = 0.8, 1
else:
```

```
    f1 = lambda x: -cos(x - 1) + 0.5     # y(x)
    f2 = lambda y: cos(y) + 3            # x(y)
    a1, b1 = 3.2, 3.4
    a2, b2 = 1.1, 1.3

x01 = (a1 + b1) / 2
x02 = (a2 + b2) / 2
x01 = 0.72
x02 = 0.98
eps = 0.00000001
print(simple_iteration_method(f1, f2, x01, x02, a1, b1, a2, b2, eps))
root = simple_iteration_method(f1, f2, x01, x02, a1, b1, a2, b2, eps)[0]
draw_plots(f1, f2, -2, 4, root, 0.1)
```

Тестовые данные

```
Проверка: True
Метод деления пополам: 2.3199218749999995
Метод секущих: 2.319945327355684
Метод простой итерации: 2.3199411198055704
```



Вывод

В ходе выполнения лабораторной работы я познакомился с приближёнными методами для решения нелинейных уравнений и систем нелинейных уравнений.

Методы для уравнений:
Метод половинного деления – простой и хороший метод, напоминает всем известный бинарный поиск. Из минусов – это медленный метод, и при содержании нескольких корней на интервале неизвестно к какому из них приведёт метод.
Метод хорд – простой в реализации, нужно выбирать начальное приближение. Есть вариации с фиксированным концом.
Метод Ньютона – быстрый, минусы: нужно выбирать начальное приближение, нужна дифференцируемость функций, необходимость вычислять производные.
Метод секущих – упрощение метода Ньютона, не требуется вычислять производную, но новое приближение вычисляется на основе двух предыдущих.
Метод простой итерации – простой в реализации, интересный. Из минусов – сходимость в малой окрестности корня, нужно грамотно выбирать начальное приближение.

Методы для систем:
Метод Ньютона – сложный в понимании, важно удачно выбрать начальное приближение.
Метод простой итерации – несложный в понимании, несложно реализовать, мне понравился.