Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
Высшего образования
Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа 2 по вычислительной математике

Численное решение нелинейных уравнений и систем

Вариант №13

Группа: P3216
Выполнил:
Сиразетдинов А.Н.
Проверил:
Малышева Т. А.

Г. Санкт-Петербург
2024

Оглавление

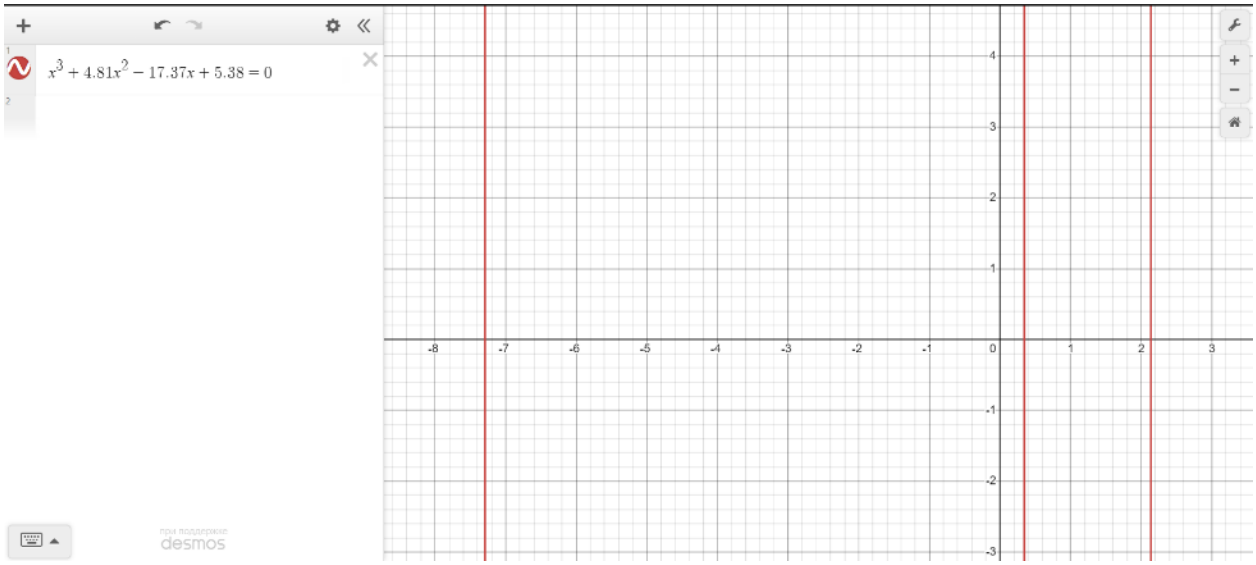
Цель работы

Изучить численные методы решения нелинейных уравнений и их систем, найти корни заданного нелинейного уравнения/системы нелинейных уравнений, выполнить программную реализацию методов.

Вычислительная реализация задачи

Решение нелинейного уравнения

$$x^3 + 4{,}81x^2 - 17{,}37x + 5{,}38$$



Интервалы изоляции корней:

1) [-8;-7]
2) [0;1]
3) [2;3]

Для уточнения значения корней воспользуемся:

1) Метод хорд
2) Метод Ньютона
3) Метод простой итерации

Уточняем крайний левый корень:

Метод хорд							
N шага	a	b	x	f(a)	f(b)	f(x)	$\lvert x_{k+1} - x_k \rvert$
1	-8,000	-7,000	-7,247	-59,820	19,660	3,246	
2	-8,000	-7,247	-7,286	-59,820	3,246	0,490	-0,04
3	-8,000	-7,286	-7,292	-59,820	0,490	0,073	-0,01

Уточняем средний корень:

$$f'(x) = 3x^2 + 9{,}62x - 17{,}37$$

Метод ньютона					
N шага	X_k	$f(x_k)$	$f'(x_k)$	X_{k+1}	$\lvert x_{k+1} - x_k \rvert$
1	0,000	5,380	-17,370	0,310	0,31
2	0,310	0,491	-14,103	0,345	0,03
3	0,345	0,007	-13,699	0,345	0,00

Уточняем крайний правый корень:

$$f'(x) = 3x^2 + 9{,}62x - 17{,}37$$

$$f'(2) = 13{,}87$$

$$f'(3) = 38{,}49$$

$$\lambda = -\frac{1}{38{,}49} = -0{,}026$$

$$x = x - 0{,}026(x^3 + 4{,}81x^2 - 17{,}37x + 5{,}38)$$

$$\varphi(x) = -0{,}026x^3 - 0{,}125x^2 + 1{,}452x - 0{,}14$$

Условие сходимости:

$$\varphi'(x) = -0{,}078x^2 - 0{,}25x + 1{,}452$$

$$\varphi(2) = 0{,}64$$

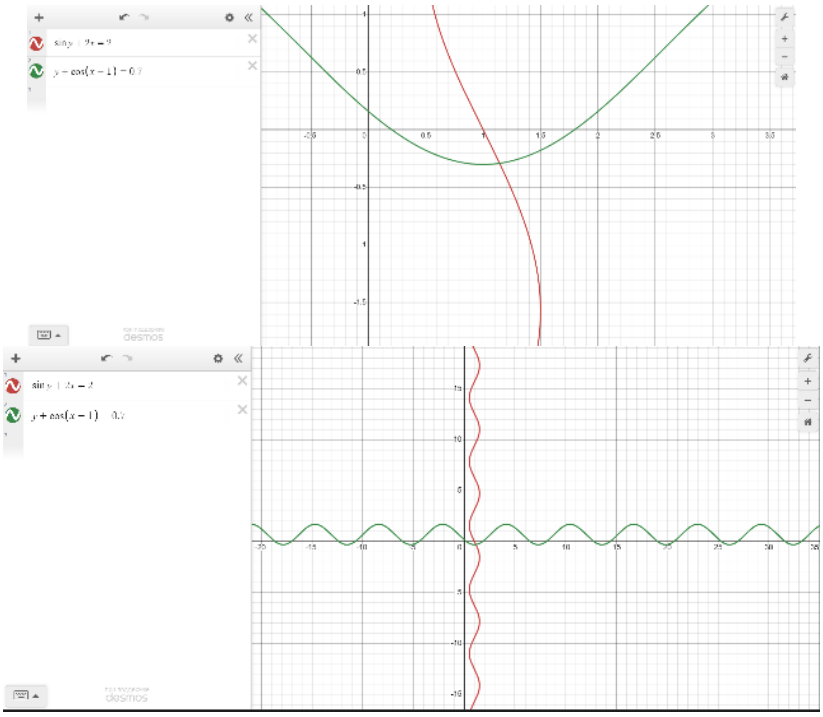
$$\varphi(3) = 0$$

Условие сходимости выполняется

Метод простой итерации				
N шага	x	x_{k+1}	$f(x_k+1)$	$\lvert x_{k+1} - x_k \rvert$
1	2,000	2,056	-1,309	0,06
2	2,056	2,091	-0,768	0,03
3	2,091	2,112	-0,432	0,02
4	2,112	2,124	-0,231	0,01

Решение системы нелинейных уравнений

$$\begin{cases} \sin y + 2x = 2 \\ y + \cos(x - 1) = 0{,}7 \end{cases}$$



Решение уравнений находится в области

$$\begin{cases} 1 < x < 1{,}5 \\ -0{,}5 < y < 0 \end{cases}$$

Выразим $\varphi(x)$

$$\begin{cases} x = 1 - \frac{1}{2}siny \\ y = 0{,}7 - cos(x-1) \end{cases}$$

Проверяем условие сходимости

$\frac{\partial\varphi_1}{x} = 0 \qquad \frac{\partial\varphi_1}{y} = -\frac{1}{2}cosy$

$\frac{\partial\varphi_2}{x} = sin(x-1) \qquad \frac{\partial\varphi_2}{y} = 0$

$|-\frac{1}{2}cosy| < 0{,}5$

$|sin(x-1)| < 0{,}5$

$\max(0{,}5;\ 0{,}5) = 0{,}5 < 1$ => *Процесс сходящийся*

Начальное приближение: $x = 1, y = -0{,}5$

Метод итераций						
N шага	x_k	y_k	x_k+1	y+k+1	|x_k+1 - x_k]	|y_k+1 - y_k]
1	1,000	-0,500	1,240	-0,300	0,24	0,20
2	1,240	-0,300	1,148	-0,271	0,09	0,03
3	1,148	-0,271	1,134	-0,289	0,01	0,02
4	1,134	-0,289	1,143	-0,291	0,01	0,00

Программная реализация задачи

Метод половинного деления

```
class HalfDivision(Method):
    name = "половинного деления"

    def solve(self) -> None:
        table = PrettyTable(["N", "a", "b", "x", "f(a)", "f(b)", "f(x)", "|a - b|"])
        f = self.equation.function
        a = self.left
        b = self.right
        x = (a + b) / 2
        iter_count = 1
        while numpy.abs(a - b) > self.accuracy or f(x) > self.accuracy:
            table.add_row(list(map(lambda i: round(i, self.symbols_after_dot),
                                   [iter_count, a, b, x, f(a), f(b), f(x), numpy.abs(a - b)])))
            if f(a) * f(x) < 0:
                b = x
            else:
                a = x
            x = (a + b) / 2
            iter_count += 1
        table.add_row(list(map(lambda i: round(i, self.symbols_after_dot),
                               [iter_count, a, b, x, f(a), f(b), f(x), numpy.abs(a - b)])))
        print(table)
        print(f"Найденный корень: {round(x, self.symbols_after_dot)}")
        print(f"Значение функции: {f(x)}")
        print(f"Количество итераций: {iter_count + 1}")
```

Метод простых итераций

```
def solve(self) -> None:
    table = PrettyTable(["N", "x", "x_next", "f(x_next)", "|x_next - x|"])
    f = self.equation.function
    f_der_left = scipy.misc.derivative(f, self.left, dx=1e-6)
    f_der_right = scipy.misc.derivative(f, self.right, dx=1e-6)
    lambda_var = 1 / max(f_der_left, f_der_right)
    if f_der_right > 0 and f_der_left > 0:
        lambda_var = - 1 * lambda_var
    phi = lambda x: x + lambda_var * f(x)
    phi_der = lambda x: scipy.misc.derivative(phi, x, dx=1e-6)
    if phi_der(self.left) >= 1 or phi_der(self.right) >= 1:
        print("Условие сходимости не выполнено(", file=stderr)
        return
    print(f"{phi_der(self.left)=} \n{phi_der(self.right)=}")

    x_prev = self.left
    x = phi(x_prev)
    iter_count = 1
    while numpy.abs(f(x)) > self.accuracy and iter_count < self.max_iter_count and x < self.right:
        table.add_row(list(map(lambda i: round(i, self.symbols_after_dot),
                               [iter_count, x_prev, x, f(x), numpy.abs(x_prev - x)])))
        x_prev = x
        x = phi(x_prev)
        iter_count += 1
    table.add_row(list(map(lambda i: round(i, self.symbols_after_dot),
                           [iter_count, x_prev, x, f(x), numpy.abs(x_prev - x)])))
    print(table)
    print(f"Найденный корень: {round(x, self.symbols_after_dot)}")
    print(f"Значение функции: {f(x)}")
    print(f"Количество итераций: {iter_count}")
```

Метод секущих

```
def solve(self) -> None:
    table = PrettyTable(["N", "x_prev", "x", "x_next", "f(x_next)", "|x_next - x|"])
    f = self.equation.function
    if f(self.left) * scipy.misc.derivative(f, self.left, dx=dx, n=2) > 0:
        x_prev = self.left
        x = x_prev + self.accuracy
    else:
        x_prev = self.right
        x = x_prev - self.accuracy
    x_next = x - (x - x_prev) / (f(x) - f(x_prev)) * f(x)
    iter_count = 1
    while numpy.abs(x_next - x) > self.accuracy and x < self.right:
        table.add_row(list(map(lambda i: round(i, self.symbols_after_dot),
                               [iter_count, x_prev, x, x_next, f(x_next), numpy.abs(x_next - x)])))
        x_prev = x
        x = x_next
        x_next = x - (x - x_prev) / (f(x) - f(x_prev)) * f(x)
        iter_count += 1
    table.add_row(list(map(lambda i: round(i, self.symbols_after_dot),
                           [iter_count, x_prev, x, x_next, f(x_next), numpy.abs(x_next - x)])))
    print(table)
    print(f"Найденный корень: {round(x, self.symbols_after_dot)}")
    print(f"Значение функции: {f(x)}")
    print(f"Количество итераций: {iter_count}")
```

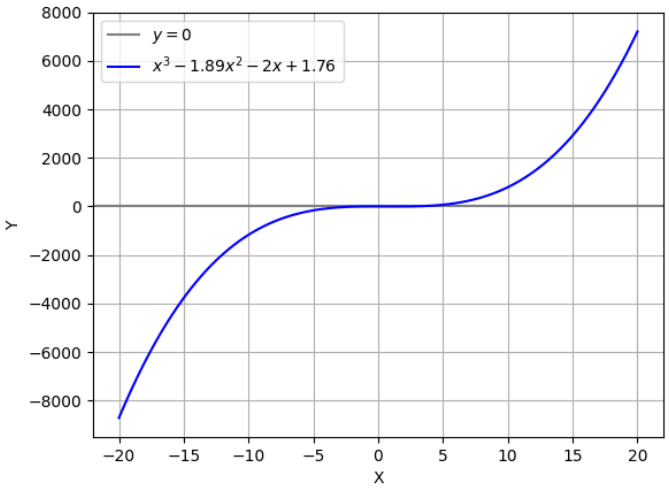
Метод Ньютона для систем нелинейных уравнений

```
def solve(self) -> None:
    x_0 = 0  # Just to make first loop go
    y_0 = 0
    prev_x = x_0
    prev_y = y_0
    dx = self.x_start
    dy = self.y_start
    iter_count = 1
    table = PrettyTable(["N", "x", "y", "dx", "dy", "f1", "f2"])

    while (np.abs(dx) > self.accuracy or np.abs(dy) > self.accuracy) and iter_count <= self.max_iter_count:
        prev_x = x_0
        prev_y = y_0
        x_0 += dx
        y_0 += dy
        df1_dx, df1_dy, df2_dx, df2_dy = self.count_Jakobian(x_0, y_0)
        if df1_dx * df2_dy - df1_dy * df2_dx == 0:
            print("Последовательность не сходится к корню!")
            return
        left_side = np.array([[df1_dx, df1_dy], [df2_dx, df2_dy]]).astype('float64')
        right_side = (-1 * np.array([self.equation.f1(x_0, y_0), self.equation.f2(x_0, y_0)])).astype('float64')
        dx, dy = np.linalg.solve(left_side, right_side)
        table.add_row(list(map(lambda i: round(i, self.symbols_after_dot),
                               [iter_count, x_0, y_0, dx, dy, self.equation.f1(x_0, y_0), self.equation.f2(x_0, y_0)])))
        iter_count += 1
    if iter_count >= self.max_iter_count:
        print(table)
        print(f"{self.max_iter_count} итераций не достаточно чтобы прийти к результату!", file=sys.stderr)
        return
    print(table)
    x_0 += dx
    y_0 += dy
    print(f"Вектор неизвестных: ({x_0} ; {y_0})")
    print(f"Количество итераций: {iter_count}")
    print(f"Вектор погрешностей: ({np.abs(x_0 - prev_x)} ; {np.abs(y_0 - prev_y)})")
    print(f"Проверка: ")
    print("f1 = ", self.equation.f1(x_0, y_0))
    print("f1 = ", self.equation.f2(x_0, y_0))
```

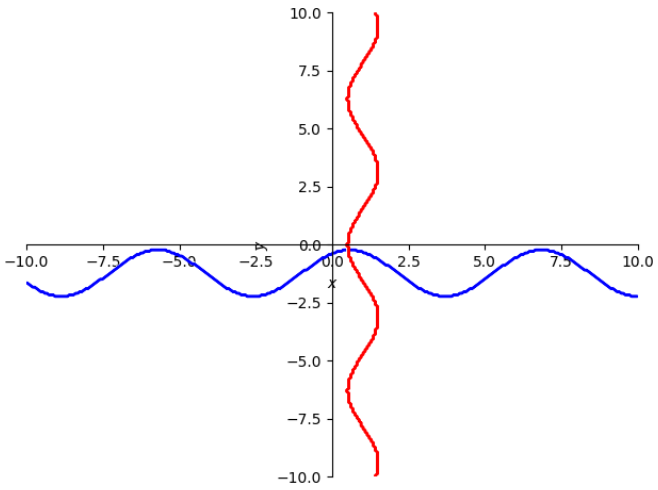
Результат работы программы

```
Выберите уравнение:
0: x^3 + 4.81x^2  - 17.37x + 5.38
1: x^3 - 1.89x^2  - 2x + 1.76
2: \sin{x} + 0.02x^2 - 1
3: \sqrt{x} - 0.05x^3 + 0.5x^2 - 5
1
Вы хотите ввести данные из файла? [y/n] n
Введите левую границу -10
Введите правую границу -1
Введите точность 0.001
Выберите метод:
0: половинного деления
1: секущих
2: простых итераций
0
```




```
+----+--------+--------+--------+----------+-------+----------+---------+
| N  |   a    |   b    |   x    |   f(a)   |  f(b) |   f(x)   | |a - b| |
+----+--------+--------+--------+----------+-------+----------+---------+
| 1  | -10.0  |  -1.0  |  -5.5  | -1167.24 |  0.87 | -210.788 |   9.0   |
| 2  |  -5.5  |  -1.0  | -3.25  | -210.788 |  0.87 | -46.031  |   4.5   |
| 3  | -3.25  |  -1.0  | -2.125 | -46.031  |  0.87 |  -12.12  |   2.25  |
| 4  | -2.125 |  -1.0  | -1.562 |  -12.12  |  0.87 |  -3.544  |  1.125  |
| 5  | -1.562 |  -1.0  | -1.281 |  -3.544  |  0.87 |  -0.883  |  0.562  |
| 6  | -1.281 |  -1.0  | -1.141 |  -0.883  |  0.87 |  0.098   |  0.281  |
| 7  | -1.281 | -1.141 | -1.211 |  -0.883  | 0.098 |  -0.365  |  0.141  |
| 8  | -1.211 | -1.141 | -1.176 |  -0.365  | 0.098 |  -0.127  |   0.07  |
| 9  | -1.176 | -1.141 | -1.158 |  -0.127  | 0.098 |  -0.013  |  0.035  |
| 10 | -1.158 | -1.141 | -1.149 |  -0.013  | 0.098 |  0.043   |  0.018  |
| 11 | -1.158 | -1.149 | -1.154 |  -0.013  | 0.043 |  0.015   |  0.009  |
| 12 | -1.158 | -1.154 | -1.156 |  -0.013  | 0.015 |  0.001   |  0.004  |
| 13 | -1.158 | -1.156 | -1.157 |  -0.013  | 0.001 |  -0.006  |  0.002  |
| 14 | -1.157 | -1.156 | -1.157 |  -0.006  | 0.001 |  -0.002  |  0.001  |
| 15 | -1.157 | -1.156 | -1.156 |  -0.002  | 0.001 |   -0.0   |  0.001  |
+----+--------+--------+--------+----------+-------+----------+---------+
Найденный корень: -1.156
Значение функции: -0.0002692117530675997
Количество итераций: 16
```

```
1:
/ x^2 + y^2 - 4
\ y - 3x^2
2:
/ x^2 * cos(y) + y^2 * sin(x) - 2
\ x^2 + y^2 - 20
0
Вы хотите ввести данные из файла? [y/n] 1
Вы хотите ввести данные из файла? [y/n] n
Введите приближение по X 1
Введите приближение по Y -1
Введите точность 0.001
+---+-------+--------+--------+--------+--------+--------+
| N |   x   |   y    |   dx   |   dy   |   f1   |   f2   |
+---+-------+--------+--------+--------+--------+--------+
| 1 |  1.0  |  -1.0  | -0.689 | 0.996  | 0.709  | 0.540  |
| 2 | 0.311 | -0.004 |  0.19  | -0.181 | -0.230 | -0.379 |
| 3 |  0.5  | -0.185 |  0.01  | -0.017 | -0.018 | -0.016 |
| 4 |  0.51 | -0.202 |  0.0   |  -0.0  |  0.0   |  0.0   |
+---+-------+--------+--------+--------+--------+--------+
Вектор неизвестных: (0.5101501567317369 ; -0.20183841262204685)
Количество итераций: 5
Вектор погрешностей: (0.009796195643457017 ; 0.01697001046695179)
Проверка:
f1 =  -2.77810530135980e-9
f1 =  -8.89813556170793e-10
```



Вывод

В ходе работы были изучены численные методы решения нелинейных уравнений и систем нелинейных уравнений. В результате работы были найдены корни заданных уравнений и систем с использованием различных численных методов, а также были построены графики функций и были написана программа для автоматического нахождения корней в заданной области