Федеральное государственное автономное образовательное учреждение высшего образования
Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки:

Системное и Прикладное Программное Обеспечение

(09.03.04 Программная инженерия)

Дисциплина «Основы программной инженерии»

Отчет

По лабораторной работе №2

Численное решение нелинейных уравнений и систем

Вариант №10

Студент

Карташев Владимир Сергеевич,
группа P3215

Преподаватель / Практик

Малышева Татьяна Алексеевна

г. Санкт-Петербург, 2024 г.

Методы вычислительной реализации

Крайний правый корень - Метод Ньютона
Крайний левый корень - Метод половинного деления
Центральный корень - Метод простой итерации

Система нелинейных уравнений - Метод простой итерации

Нелинейное уравнение:
$x^3 - 3,125x^2 - 3,5x + 2,458$
Система нелинейных уравнение:
$$\begin{cases} \sin(x + 0,5) - y = 1 \\ \cos(y - 2) + x = 0 \end{cases}$$

Методы программной реализации

Решение нелинейных уравнений:
Метод хорд, метод Ньютона, Метод простой итерации
Решение систем нелинейных уравнений:
Метод Ньютона

Цель лабораторной работы

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

Рабочие формулы

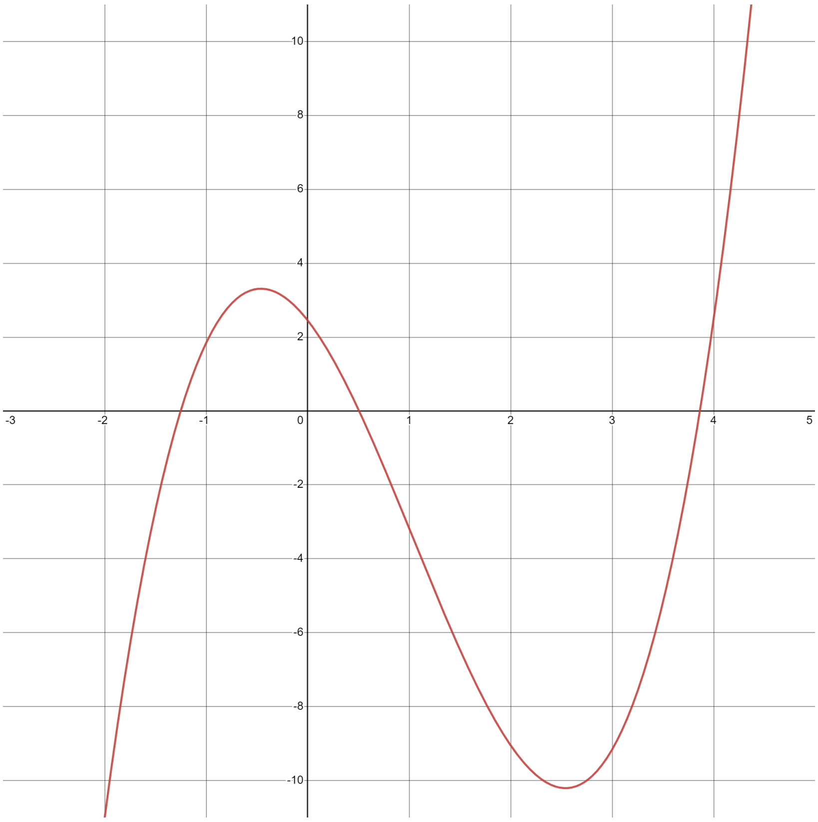
Метод Ньютона: $x_i = x_{i-1} - \frac{f(x_{i-1})}{f'(x_{i-1})}$

Метод половинного деления: $x_i = \frac{a_i + b_i}{2}$

Метод простой итерации: $x_{i+1} = \varphi(x_i)$

Метод хорд: $x_i = \frac{a_i f(b_i) - b_i f(a_i)}{f(b_i) - f(a_i)}$

Решение нелинейного уравнения



Точность: $\varepsilon = 10^{-2} = 0,01$

Уравнение: $x^3 - 3,125x^2 - 3,5x + 2,458 = 0$

Уточнение правого корня уравнения методом Ньютона $x \in (3; 4)$

№ итерации	x_k	$f(x_k)$	$f'(x_k)$	x_{k+1}	$\lvert x_{k+1} - x_k\rvert$
1	4.000	2.458	19.500	3.874	0.126
2	3.874	0.138	17.309	3.866	0.008

Уточнение левого корня уравнения методом половинного деления
$x \in (-2; -1)$

№ шага	a	b	x	$f(a)$	$f(x)$	$\lvert a - b\rvert$
1	-2.000	-1.000	-1.500	-11.042	1.833	1.000
2	-1.500	-1.000	-1.250	1.833	-0.003	0.500

Уточнение центрального корня уравнения методом простой итерации
$x \in (0; 1)$

№ итерации	x_k	x_{k+1}	$\varphi(x_{k+1})$	$f(x_{k+1})$	$\lvert x_{k+1} - x_k \rvert$
1	0	0,702	0,361	-1,193	0,702
2	0,702	0,361	0,599	0,834	0,341
3	0,361	0,599	0,443	-0,545	0,238
4	0,599	0,443	0,552	0,381	0,156
5	0,443	0,552	0,478	-0,258	0,109
6	0,552	0,478	0,529	0,18	0,074
7	0,478	0,529	0,495	-0,12	0,051
8	0,529	0,495	0,518	0,081	0,034
9	0,495	0,518	0,502	-0,055	0,023
10	0,518	0,502	0,513	0,04	0,016
11	0,502	0,513	0,506	-0,025	0,011
12	0,513	0,506	0,511	0,016	0,007

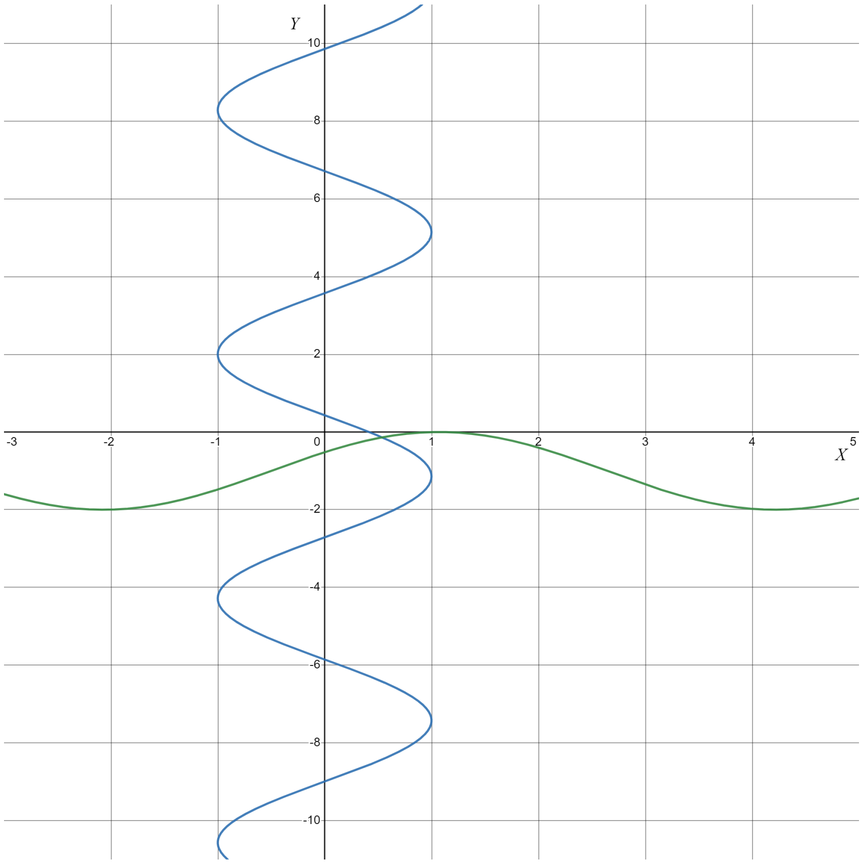
Ответ:

$x_1 = -1.25$ - левый

$x_2 = -1.25$ - центральный

$x_3 = 3.8659$ - правый

Решение системы нелинейных уравнений



Точность: $\varepsilon = 10^{-2} = 0,01$

Система: $\begin{cases} \sin(x + 0,5) - y = 1 \\ \cos(y - 2) + x = 0 \end{cases}$

$$\begin{cases} \sin(x+0,5) - y = 1 \\ \cos(y-2) + x = 0 \end{cases} \Rightarrow \begin{cases} x = -\cos(y-2) \\ y = \sin(x+0,5) - 1 \end{cases}$$

$x_0 = 1;\ y_0 = 1$

№ итерации	x	y	$\lvert x_{i-1} - x_i \rvert$	$\lvert y_{i-1} - y_i \rvert$	макс. разность
0	1	1	-	-	0
1	-0,54	-0,003	1,54	1,003	1,54
2	0,419	-1,04	0,959	1,037	1,037
3	0,995	-0,205	0,576	0,835	0,835
4	0,593	-0,003	0,402	0,202	0,402
5	0,419	-0,112	0,174	0,109	0,174
6	0,515	-0,205	0,096	0,093	0,096
7	0,593	-0,151	0,078	0,054	0,078
8	0,548	-0,112	0,045	0,039	0,045
9	0,515	-0,134	0,033	0,022	0,033
10	0,534	-0,151	0,019	0,017	0,019
11	0,548	-0,141	0,014	0,01	0,014
12	0,54	-0,134	0,008	0,007	0,008

Листинг программы:

Функции для подсчета значения функции и производной по массиву коэффициентов $[a, b, c, ..., y, z, n]$ уравнения $n + zx + yx^2 + ... + ax^m$

```
func FindFunctrion(x float64, coefficients []float64) float64 {
   var y float64
   for i, val := range coefficients {
      if i == 0 {
         y += val
      } else {
         y += val * math.Pow(x, float64(i))
      }
   }

   return y
}

func FindTheDerivative(coefficients []float64) []float64 {
   var derivative []float64
   for i, val := range coefficients {
      if i == 0 {
         continue
      }
      derivative = append(derivative, val*float64(i))
   }
   return derivative
}
```

Метод Хорд:

```
func ChordMethod(e, a, b float64, coefficients, derivative1, derivative2 []float64) {
   var (
      x             float64
      prev_x        float64
      f_a           float64
      f_b           float64
      f_x           float64
      differenceAbs float64
   )

   iterationsLimit := 1000

   fmt.Println("[ # ] [ a ]     [ b ]     [ x ]     [ F(a) ]      [ F(b) ]      [ F(x) ]
[ |x_i+1 - x_i| ]")

   for i := 0; i <= iterationsLimit; i++ {
      // Подсчет
      x = ((a*tools.FindFunctrion(b, coefficients) - b*tools.FindFunctrion(a,
coefficients)) / (tools.FindFunctrion(b, coefficients) - tools.FindFunctrion(a,
coefficients)))
      f_a = tools.FindFunctrion(a, coefficients)
      f_b = tools.FindFunctrion(b, coefficients)
      f_x = tools.FindFunctrion(x, coefficients)

      if i == 0 {
         differenceAbs = math.Abs(math.Abs(a) - math.Abs(x))
      } else {
         differenceAbs = math.Abs(math.Abs(x) - math.Abs(prev_x))
      }

      fmt.Printf("%d %5f    %5f    %5f    %5f       %5f       %5f       %5f\n", i, a, b,
x, f_a, f_b, f_x, differenceAbs)

      // Выход, если:
      if differenceAbs <= e {
```

```
            fmt.Println("Причина:", "|x_i+1 - x_i| <= e")
            //os.Exit(0)
        }
        if math.Abs(a-b) <= e {
            fmt.Println("Причина:", "|a - b| <= e")
            //os.Exit(0)
        }
        if math.Abs(f_x) <= e {
            fmt.Println("Причина:", "|f(x)| <= e")
            //os.Exit(0)
        }
        if differenceAbs <= e || math.Abs(a-b) <= e || math.Abs(f_x) <= e {
            fmt.Println("Ответ:", x)
            fmt.Println("Значение функции:", f_x)
            os.Exit(1)
        }

        if tools.FindFunctrion(x, coefficients)*tools.FindFunctrion(b, coefficients) < 0 {
            a = x
        } else if tools.FindFunctrion(x, coefficients)*tools.FindFunctrion(a, coefficients)
< 0 {
            b = x
        }

        prev_x = x
    }

    fmt.Printf("Лимит в %d итераций исчерпан\n", iterationsLimit)
    fmt.Println("Остановка")
    os.Exit(1)
}
```

Метод Ньютона:

```
func NewtonMethod(e, a, b float64, coefficients, derivative1, derivative2 []float64) {
    var (
        xi            float64
        f_xi          float64
        d1_xi         float64
        x_i_plus_1    float64
        differenceAbs float64
    )

    // Условие сходимости
    if (tools.FindFunctrion(a, derivative1)*tools.FindFunctrion(b, derivative1) > 0) &&
        (tools.FindFunctrion(a, derivative2)*tools.FindFunctrion(b, derivative2) > 0) {
    } else {
        fmt.Println("Не удалось установить начальное приближение!")
        os.Exit(1)
    }

    if tools.FindFunctrion(a, coefficients)*tools.FindFunctrion(a, derivative2) > 0 {
        xi = a
    } else if tools.FindFunctrion(b, coefficients)*tools.FindFunctrion(b, derivative2) > 0
{
        xi = b
    } else {
        fmt.Println("Не удалось установить начальное приближение!")
        os.Exit(1)
    }

    iterationsLimit := 1000

    fmt.Println("[ # ] [ x_i ]       [ f'(x_i) ]       [ f(x_i) ]    [ x_i+1 ]     [|x_i+1
- x_i|]")

    for i := 0; i <= iterationsLimit; i++ {
```

```
        f_xi = tools.FindFunctrion(xi, coefficients)
        d1_xi = tools.FindFunctrion(xi, derivative1)
        if d1_xi == 0 {
            fmt.Println("Стоп")
            fmt.Println("Причина:", "f'(xi) == 0")
            fmt.Println("Ответ:", "не найден из-за невыполнения условия сходимости!")
            os.Exit(0)
        }
        x_i_plus_1 = xi - (f_xi / d1_xi)
        differenceAbs = math.Abs(x_i_plus_1 - xi)
        fmt.Printf("%d %5f    %5f        %5f        %5f        %5f\n", i, xi, f_xi, d1_xi,
x_i_plus_1, differenceAbs)

        if differenceAbs <= e {
            fmt.Println("Стоп")
            fmt.Println("Причина:", "|x_i+1 - x_i| <= e")
            fmt.Println("Значение функции:", f_xi)
            fmt.Println("Ответ:", x_i_plus_1)
            os.Exit(0)
        } else if math.Abs(f_xi/d1_xi) <= e {
            fmt.Println("Стоп")
            fmt.Println("Причина:", "|f(x_i) / f'(x_i)| <= e")
            fmt.Println("Значение функции:", f_xi)
            fmt.Println("Ответ:", x_i_plus_1)
            os.Exit(0)
        } else if math.Abs(f_xi) <= e {
            fmt.Println("Стоп")
            fmt.Println("Причина:", "|f(x_i)| <= e")
            fmt.Println("Значение функции:", f_xi)
            fmt.Println("Ответ:", x_i_plus_1)
            os.Exit(0)
        }

        xi = x_i_plus_1
    }

    fmt.Printf("Лимит в %d итераций исчерпан\n", iterationsLimit)
    fmt.Println("Остановка")
    os.Exit(1)
}
```

Метод простой итерации:

```
func phiFunction(x, lambda float64, coefficients []float64) float64 {
    return x + lambda*tools.FindFunctrion(x, coefficients)
}

func phiDerivativeFunction(x, lambda float64, coefficients []float64) float64 {
    return 1 + lambda*tools.FindFunctrion(x, coefficients)
}

func SimpleIterationMethod(e, a, b float64, coefficients, derivative1, derivative2
[]float64) {
    // Начальное приближение lambda
    lambda := initialLambda(a, b, derivative1)

    fmt.Println("Лямбда:", lambda)

    var (
        xi            float64
        x_i_plus_1    float64
        phi_x_plus_1  float64
        f_x_plus_1    float64
        differenceAbs float64
    )

    xi = a
```

```
   iterationsLimit := 1000

   fmt.Println("phi'(a)", phiDerivativeFunction(a, lambda, derivative1))
   fmt.Println("phi'(b)", phiDerivativeFunction(b, lambda, derivative1))

   fmt.Println("Index\tx_i\t\tx_i+1\t\t\tphi(x_i+1)\t\t\tf(x_i+1)\t\t\t|x_i+1 - x_i|")

   for i := 1; i <= iterationsLimit; i++ {
      x_i_plus_1 = phiFunction(xi, lambda, coefficients)
      phi_x_plus_1 = phiFunction(x_i_plus_1, lambda, coefficients)
      f_x_plus_1 = tools.FindFunctrion(x_i_plus_1, coefficients)
      differenceAbs = math.Abs(x_i_plus_1 - xi)

      fmt.Printf("%d\t%5f\t%5f\t\t%5f\t\t\t%5f\t\t\t%5ft\n", i, xi, x_i_plus_1,
phi_x_plus_1, f_x_plus_1, differenceAbs)

      if i == iterationsLimit {
         fmt.Println("Количство итераций превышено")
      }

      if math.Abs(f_x_plus_1) <= e {
         fmt.Println("Стоп")
         fmt.Println("Причина:", "|f(x_i+1)| <= e")
         fmt.Println("Ответ:", x_i_plus_1)
         fmt.Println("Значение функции:", f_x_plus_1)
         fmt.Println("Ответ найден за:", i+1, "итераций")
         os.Exit(0)
      }

      xi = x_i_plus_1
   }
}

func initialLambda(a, b float64, derivative1 []float64) float64 {
   // Выбор начального приближения lambda
   maxDerivative := math.Max(math.Abs(tools.FindFunctrion(a, derivative1)),
math.Abs(tools.FindFunctrion(b, derivative1)))
   if tools.FindFunctrion(a, derivative1) > 0 && tools.FindFunctrion(b, derivative1) > 0 {
      return -1 / maxDerivative
   } else {
      return 1 / maxDerivative
   }
}
```

Метод Ньютона (для системы) нелинейных уравнений:

```
func newtonMethod(f func(float64, float64) (float64, float64), jacobian func(float64,
float64) (float64, float64, float64, float64), x0, y0 float64) (float64, float64, error) {
   x, y := x0, y0
   fmt.Println("\n")
   for i := 0; i < maxIter; i++ {
      fx, fy := f(x, y)
      jx, jy, jz, jw := jacobian(x, y)

      detJ := jx*jw - jy*jz
      if math.Abs(detJ) < tolerance {
         return 0, 0, fmt.Errorf("Якобиан имеет единичную матрицу")
      }

      deltaX := (jw*fx - jy*fy) / detJ
      deltaY := (-jz*fx + jx*fy) / detJ

      fmt.Println("deltaX:", deltaX)

      x -= deltaX
      y -= deltaY
```

```go
        if math.Abs(deltaX) < tolerance && math.Abs(deltaY) < tolerance {
            fmt.Println("Количество итераций:", i+1)
            return x, y, nil
        }
    }
    return 0, 0, fmt.Errorf("Достигнут лимит итераций")
}
```

Результаты выполнения программы при различных исходных данных:

Ввод:

```
0.01
2.5 -3 -10.3
2 10
```

Вывод:

Метод Хорд:

Считанная функция:
2.5*x^2 + -3*x^1 + -10.3
График создан

Корени в интервале: [-2;-1]
Корени в интервале: [2;3]

a = 2
b = 10
Производная 1-го порядка: [-3 5]
Производная 2-го порядка: [5]

[#]	[a]	[b]	[x]	[F(a)]	[F(b)]	[F(x)]	[|x_i+1 - x_i|]
0	2.000000	10.000000	2.233333	-6.300000	209.700000	-4.530556	0.233333
1	2.233333	10.000000	2.397583	-4.530556	209.700000	-3.121738	0.164250
2	2.397583	10.000000	2.509098	-3.121738	209.700000	-2.088364	0.111515
3	2.509098	10.000000	2.582963	-2.088364	209.700000	-1.369647	0.073865
4	2.582963	10.000000	2.631092	-1.369647	209.700000	-0.886659	0.048130
5	2.631092	10.000000	2.662119	-0.886659	209.700000	-0.569167	0.031026
6	2.662119	10.000000	2.681981	-0.569167	209.700000	-0.363386	0.019863
7	2.681981	10.000000	2.694641	-0.363386	209.700000	-0.231203	0.012659
8	2.694641	10.000000	2.702686	-0.231203	209.700000	-0.146778	0.008046

Причина: |x_i+1 - x_i| <= e
Ответ: 2.702686106472415
Значение функции: -0.14677784411968986

Метод Ньютона:

Считанная функция:
2.5*x^2 + -3*x^1 + -10.3
График создан

Корни в интервале: [-2;-1]
Корни в интервале: [2;3]

a = 2
b = 10
Производная 1-го порядка: [-3 5]
Производная 2-го порядка: [5]

[#]	[x_i]	[f'(x_i)]	[f(x_i)]	[x_i+1]	[|x_i+1 - x_i|]
0	10.000000	209.700000	47.000000	5.538298	4.461702
1	5.538298	49.766965	24.691489	3.522747	2.015551
2	3.522747	10.156118	14.613733	2.827776	0.694971
3	2.827776	1.207461	11.138878	2.719375	0.108401
4	2.719375	0.029377	10.596875	2.716603	0.002772

Стоп
Причина: |x_i+1 - x_i| <= e
Значение функции: 0.02937672938845992
Ответ: 2.7166028642922755

Метод простой итерации:

Считанная функция:
2.5*x^2 + -3*x^1 + -10.3
График создан

Корени в интервале: [-2;-1]
Корени в интервале: [2;3]

a = 2
b = 10
Производная 1-го порядка: [-3 5]
Производная 2-го порядка: [5]

Лямбда: -0.02127659574468085
phi'(a) 0.851063829787234
phi'(b) 0

Index	x_i	x_i+1	phi(x_i+1)	f(x_i+1)	|x_i+1 - x_i|
1	2.000000	2.134043	2.247166	-5.316784	0.134043t
2	2.134043	2.247166	2.341147	-4.417114	0.113123t
3	2.247166	2.341147	2.418190	-3.621020	0.093981t
4	2.341147	2.418190	2.480646	-2.935465	0.077043t
5	2.418190	2.480646	2.530815	-2.357922	0.062457t
6	2.480646	2.530815	2.570813	-1.879884	0.050169t
7	2.530815	2.570813	2.602509	-1.489745	0.039998t
8	2.570813	2.602509	2.627507	-1.174892	0.031697t
9	2.602509	2.627507	2.647146	-0.923039	0.024998t
10	2.627507	2.647146	2.662529	-0.722983	0.019639t
11	2.647146	2.662529	2.674549	-0.564939	0.015383t
12	2.662529	2.674549	2.683924	-0.440620	0.012020t
13	2.674549	2.683924	2.691225	-0.343157	0.009375t
14	2.683924	2.691225	2.696904	-0.266948	0.007301t
15	2.691225	2.696904	2.701319	-0.207479	0.005680t
16	2.696904	2.701319	2.704748	-0.161147	0.004414t
17	2.701319	2.704748	2.707409	-0.125094	0.003429t
18	2.704748	2.707409	2.709474	-0.097067	0.002662t
19	2.707409	2.709474	2.711076	-0.075294	0.002065t
20	2.709474	2.711076	2.712319	-0.058391	0.001602t
21	2.711076	2.712319	2.713282	-0.045274	0.001242t
22	2.712319	2.713282	2.714029	-0.035098	0.000963t
23	2.713282	2.714029	2.714608	-0.027206	0.000747t
24	2.714029	2.714608	2.715056	-0.021086	0.000579t
25	2.714608	2.715056	2.715404	-0.016342	0.000449t
26	2.715056	2.715404	2.715673	-0.012665	0.000348t
27	2.715404	2.715673	2.715882	-0.009815	0.000269t

Стоп
Причина: |f(x_i+1)| <= e
Ответ: 2.7156734577628994
Значение функции: -0.009814550293942403

Система нелинейных уравнений (метод Ньютона):

```
Выберите тип решения уравнения:
1. Решение нелинейных уравнений
2. Решение системы нелинейных уравнений
2
Выберите метод решения системы нелинейных уравнений:
1. Метод Ньютона
1
Выберите решаемое уравнение:
1:
tgxy = x^2
0.5x^2 + 2y^2 = 1
2:
sin(x+y)-1.4x = 0
x^2 + y^2 = 1
1
Введите точность
0.01
Введите начальное прибижение для x
1
Введите начальное прибижение для y
2


deltaX: -0.9741181115719424
deltaX: 4.899147962303645
deltaX: 1.7569666091027636
deltaX: -1.280492689255874
deltaX: -2.773574088202409
deltaX: 0.06208520999392478
deltaX: -0.043574666840793974
deltaX: 0.019882722632935623
deltaX: -0.0033826711945834057
Количество итераций: 9
Решение системы 1: x = -0.6629402769676657, y = -0.6246047278329376
```

Вывод
В ходе выполнения лабораторной работы №2 были изучены и применены численные методы решения нелинейных уравнений и систем нелинейных уравнений. В процессе работы была выполнена как вычислительная, так и программная реализация методов, что позволило закрепить теоретические знания на практике и получить практические навыки программирования и анализа численных методов.