Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
Московский государственный технический университет имени Н.Э. Баумана

Лабораторная работа №6
по курсу «Численные методы»
«Решение систем нелинейных уравнений методом Ньютона»
Вариант 9

Выполнил:
студент группы ИУ9-62Б
Егоров Алексей

Москва, 2022

1. Цель

Изучение метода Ньютона для решения систем нелинейных уравнений.

2. Постановка задачи

Дано: Система нелинейных уравнений:

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

Найти: $\overline{x} = (x_1, ..., x_n)$ - решение системы нелинейных уравнений

Индивидуальный вариант:

$$\begin{cases} cos(x + 0.5) - y = 2 \\ siny - 2x = 1 \end{cases}$$

3. Теоретические сведения

Представим систему нелинейных уравнений в векторном виде $f(\overline{x}) = 0$, где $\overline{x} = (x_1, ..., x_n)$ - вектор неизвестных, $f = (f_1, f_2, ..., f_n)$ - вектор-функция.

Выбрав начальное приближение $x^0 = (x_1^0, x_2^0, ..., x_n^0)$ к решению систему, следующие приближения в методе Ньютона строим по рекурентной зависимости:

$$x^{k+1} = x^k - (f'(x^k))^{-1} f(x^k), k = 0, 1, 2...$$

где $(f'(x^k))^{-1}$ - матрица, обратная матрице Якоби.

Таким образом решение для системы уравнений из индивидульного варианта получается следующим образом:

$$\begin{cases} f_1(x, y) = cos(x + 0.5) - y - 2 \\ f_2(x, y) = siny - 2x - 1 \end{cases}$$

$$f = \begin{pmatrix} f_1(x, y) \\ f_2(x, y) \end{pmatrix}$$

Матрица Якоби:

$$f'(x,y) = \begin{pmatrix} \frac{\delta f_1}{\delta x} & \frac{\delta f_1}{\delta y} \\ \frac{\delta f_2}{\delta x} & \frac{\delta f_2}{\delta y} \end{pmatrix} = \begin{pmatrix} -sin(x+0.5) & -1 \\ -2 & cosy \end{pmatrix}$$

Методом Гаусса решим СЛАУ:

$$\begin{pmatrix} -sin(x^0+0.5) & -1 \\ -2 & cosy^0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = - \begin{pmatrix} cos(x^0+0.5) - y^0 - 2 \\ siny^0 - 2x^0 - 1 \end{pmatrix}$$

где (x^0, y^0) - начальное приближение

Тогда $(k+1)$-е приближение есть сумма k-го приближения и решения СЛАУ. Будем улучшать приближение пока $max(|x^k - x^{k+1}|, |y^k - y^{k+1}|) > \epsilon$, где ϵ-требуемая точность

4. Практическая реализация:

Листинг 1. Решение системы нелинейных уравнений методом Ньютона

```go
package main

import (
	"fmt"
	"math"
)

type Point struct {
	x, y float64
}

var (
	intersec = Point{0, 0}
)

type function func(float64) float64
```

```go
func gauss(matrix [][]float64, ds []float64, SIZE int) (xs []float64) {
        xs = make([]float64, SIZE)
        for i := 0; i < SIZE; i++ {
                for j := i + 1; j < SIZE; j++ {
                        var k float64 = matrix[j][i] / matrix[i][i]
                        for t := i; t < SIZE; t++ {
                                matrix[j][t] -= k * matrix[i][t]
                        }
                        ds[j] -= k * ds[i]
                }
        }
        for i := SIZE - 1; i >= 0; i-- {
                var k float64 = 0
                for j := i + 1; j < SIZE; j++ {
                        k += matrix[i][j] * xs[j]
                }
                xs[i] = (ds[i] - k) / matrix[i][i]
        }
        return xs
}

var (
        f1 = func(p Point) float64 {
                return math.Cos(p.x+0.5) - p.y - 2
        }
        f2 = func(p Point) float64 {
                return math.Sin(p.y) - 2*p.x - 1
        }
        m2 = [][]function{
                {
                        func(x float64) float64 { return -math.Sin(x + 0.5) }
```

```
				func(y float64) float64 { return -1 },
			},
			{
				func(x float64) float64 { return -2 },
				func(y float64) float64 { return math.Cos(y) },
			},
		}
)

func main() {
		n := 2

		realIntersec := Point{-0.9450111644002, -1.0973941404642}
		cnt := 0
		eps := math.Pow10(-2)

		for {
			cnt += 1
			matrix := [][]float64{
				{m2[0][0](intersec.x), m2[0][1](intersec.y)},
				{m2[1][0](intersec.x), m2[1][1](intersec.y)},
			}
			ds := []float64{-f1(intersec), -f2(intersec)}
			ys := gauss(matrix, ds, n)
			intersec.x += ys[0]
			intersec.y += ys[1]

			if math.Abs(ys[0]) < eps && math.Abs(ys[1]) < eps {
				fmt.Printf("STEP: %d, x=%.16f, y=%.16f, diff_x=%.16f,
					cnt,
					intersec.x,
```

```
                                        intersec.y,
                                        math.Abs(intersec.x-realIntersec.x),
                                        math.Abs(intersec.y-realIntersec.y),
                                )
                                break
                        }

                }
}
```

5. Результат:

С помощью GeoGebra было найдено решение системы:

$(-0.9450111644002, -1.0973941404642)$

При $\epsilon = 10^{-2}$ алгоритм справился за 4 итерации с начальным приближением $(0, 0)$, при этом погрешность оказалась гораздо меньше ϵ:

$STEP : 4, x = -0.9450112121563138, y = -1.0973941573540991, diff_x = 0.0000000477561137$, $diff_y = 0.0000000168898990$

6. Вывод:

В ходе выполнения лабораторной работы был изучен метод Ньютона для решения систем нелинейных уравнений.

В процессе тестирования был сделан вывод о том, что метод Ньютона сходится для системы из индивидуального варианта даже при достаточно удаленном от решения начальном приближении. К тому же погрешность при завершении итерационного вычисления оказалась гораздо ниже ожидаемой: уже на 4 итерации погрешность была меньше 10^{-6} .