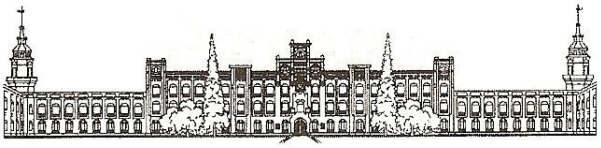


Міністерство освіти і науки України Національний технічний університет України
«Київський політехнічний інститут»

Лабораторна робота №2

з дисципліни «Алгоритми та методи обчислень»

Виконав студент групи: КВ-22
ПІБ: Крутогуз Максим Ігорович
Варіант 12
Перевірив:

Київ 2024

Варіант	Рівняння	Метод уточнення коренів рівняння
12	$(x^2-\sin x)/(\cos x-2)+1=0$	I, X

Рис. 1: Завдання

1 Завдання варіанту

Завдання зображені на рис. 1

2 Метод ітерацій

$$\frac{x^2-\sin x}{\cos x-2}+1=0$$

$$\frac{x^2-\sin x}{\cos x-2}=-1$$

$$x^2-\sin x=-\cos x+2$$

$$x^2=-\cos x+2+\sin x$$

$$x=\pm\sqrt{-\cos x+2+\sin x}$$

$$f'=\frac{\sin x+\cos x}{2\sqrt{-\cos x+2+\sin x}}$$

$$f'<\frac{2}{2\sqrt{2}}=\frac{1}{\sqrt{2}}<\frac{10}{11}<1$$

$$|x_k+1-x_k|\leq\frac{1-\frac{10}{11}}{\frac{10}{11}}\epsilon=\frac{1}{10}\cdot\epsilon$$

3 Метод хорд

$$x_n+1=x_n-\frac{x_n}{f(x_n)-f(c)}(x_n-c), n=0,1,2,3,...$$

4 Код програми

```
class IterationAlgorithm {
 static iterationCount: number

 public static findRoot(error: number, initialValue: number = 0.5) : number {
     this.iterationCount = 0
     let x = initialValue
     let i = 0
     let x_prev;
     do {
         this.iterationCount++
         if (i == 1000) {
             break
```

```
            }

            x_prev = x
            x = Math.sqrt(-Math.cos(x) + 2 + Math.sin(x))

            i++
        } while (10 * Math.abs(x  - x_prev) > error)
        return x
    }

    public static printRoot(error: number, initialValue: number = 0.5) : void {
        console.log(this.findRoot(error, initialValue))
    }
}

class ChordAlgorithm {
    static iterationCount: number

    public static f(x: number) : number {
        return (x**2 - Math.sin(x)) / (Math.cos(x) - 2) + 1
    }

    public static findRoot(error: number, x0: number = 0, x1: number = 1)
    : number | null {
        this.iterationCount = 0

        let xPrev = x0
        let xCur = x1

        let i = 0
        while (true) {
            this.iterationCount++
            if (i == 1000) {
                return null
            }

            const fCur = this.f(xCur)
            const fPrev = this.f(xPrev)

            const xNext = xCur - (fCur*(xCur-xPrev)) / (fCur - fPrev)

            if (Math.abs(xNext - xCur) < error) {
                return xNext
            }

            xPrev = xCur
            xCur = xNext
            i++
        }
    }

    public static printRoot(error: number, x0: number = 0, x1: number = 1) : void {
        console.log(this.findRoot(error, x0, x1))
    }
}

// IterationAlgorithm.printRoot(10**-6, -1)
// ChordAlgorithm.printRoot(10**-6, 0, 1)

```

```
const realRoot = 1.7862432638652274839

const table1 = [0,0,0,0].map((value, index) => {
    const epsilon = 10**(-3*(index + 1))
    const root = IterationAlgorithm.findRoot(epsilon, -1)
    return {epsilon: epsilon, calculation: root, error: Math.abs(root - realRoot)}
})

const table2 = [0,0,0,0].map((value, index) => {
    const epsilon = 10**(-3*(index + 1))
    const root = ChordAlgorithm.findRoot(epsilon, 0, 1)
    if (!root) {
        return {epsilon: epsilon, calculaiton: root, error: null}
    }
    return {epsilon: epsilon, calculation: root, error: Math.abs(root - realRoot)}
})

const table3 = [0, 0, 0, 0, 0, 0, 0, 0, 0].map((value, index) => {
    const epsilon = 10**(-3*(index + 1))
    const startTime1 = performance.now();
    for (let i = 0; i < 5000; i++) {
        const root = IterationAlgorithm.findRoot(epsilon, -1)
    }
    const endTime1 = performance.now();

    const elapsedTime1 = endTime1 - startTime1;

    const startTime2 = performance.now();
    for (let i = 0; i < 5000; i++) {
        const root = ChordAlgorithm.findRoot(epsilon, 0, 1)
    }
    const endTime2 = performance.now();

    const elapsedTime2 = endTime2 - startTime2;

    return {epsilon: epsilon, iterationSpeed1: elapsedTime1 + 'ms',
    iterationSpeed2: elapsedTime2 + 'ms',
    iterationCount1: IterationAlgorithm.iterationCount,
    iterationCount2: ChordAlgorithm.iterationCount}
})

console.table(table1, ['epsilon', 'calculation', 'error'])
console.table(table2, ['epsilon', 'calculation', 'error'])
console.table(table3, ['epsilon', 'iterationSpeed1', 'iterationSpeed2',
'iterationCount1', 'iterationCount2'])
```

5 Результати

Результати зображеня на рис. 2

6 Висновки

На власному досвіді було пересвідчено, що є певні інструменти, за допомогою яких можна знайти наближені корені заданого рівняння. Для цього існують різні способи. Наприклад, в цій роботі, було використано метод ітерацій та метод хорд, кожний яких пропонує власні підходи для знаходження коренів. Метод хорд вимагає меншої кількості ітерацій.

(index)	epsilon	calculation	error
0	0.001	1.7862323340463035	0.00001092981892392153
1	0.000001	1.786243241111405	2.275382238181578e-8
2	1e-9	1.7862432638551093	1.0118128557223827e-11
3	1e-12	1.7862432638652064	2.1094237467877974e-14

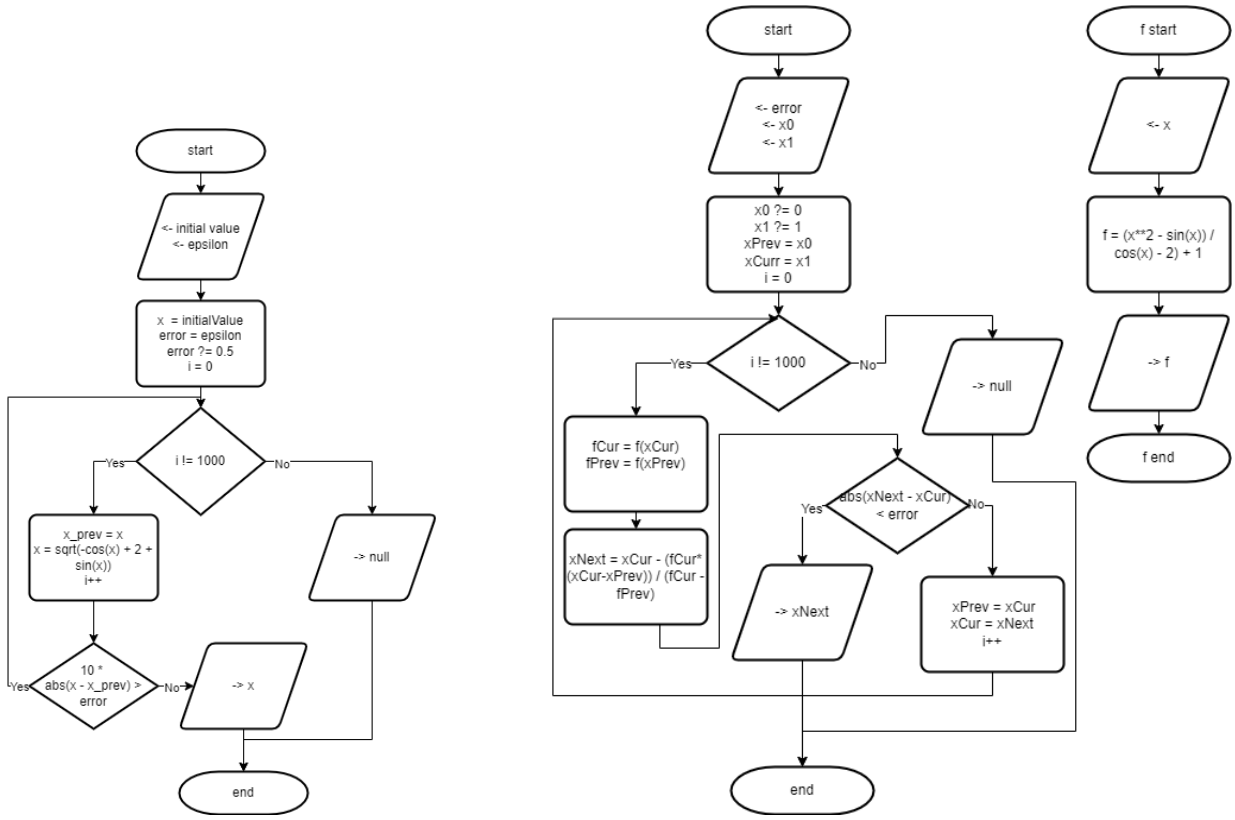
(index)	epsilon	calculation	error
0	0.001	1.7862432644312252	5.659976931582378e-10
1	0.000001	1.7862432638652272	2.220446049250313e-16
2	1e-9	1.7862432638652272	2.220446049250313e-16
3	1e-12	1.7862432638652277	2.220446049250313e-16

(index)	epsilon	iterationSpeed1	iterationSpeed2	iterationCount1	iterationCount2
0	0.001	'5.2009000005200505ms'	'4.261300000362098ms'	9	7
1	0.000001	'1.826100000180304ms'	'1.7033999999985099ms'	13	8
2	1e-9	'2.6597999995574355ms'	'1.275700000114739ms'	18	8
3	1e-12	'3.0131999999284744ms'	'1.5861999997869134ms'	22	9
4	1e-15	'3.3846000004559755ms'	'1.8612000001594424ms'	27	9
5	1e-18	'3.090099999681115ms'	'1.971400000154972ms'	27	11
6	1.0000000000000001e-21	'4.261100000701845ms'	'2.4503000006079674ms'	27	11
7	1.0000000000000001e-24	'3.2561000008136034ms'	'1.9877000004053116ms'	27	11
8	1e-27	'3.136899999342859ms'	'2.140800000168383ms'	27	11

Рис. 2: Результат виконання програми

7 Git репозіторій

Посилання на git репозіторій



(а) Метод ітерацій (б) Метод хорд

Рис. 3: Алгоритми розв'язання рівнянь із одним невідомим на блок-схемах