

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

Лабораторна робота №1

з дисципліни «Алгоритми та методи обчислень»

Виконав студент групи: КВ-22

ПІБ: Крутогуз Максим Ігорович

Перевірив:

Київ 2024

ОБЧИСЛЕННЯ ЗНАЧЕНЬ ФУНКЦІЇ

Загальне завдання:

3

1.2. Завдання для лабораторної роботи

Для заданого варіанта (табл. 1.1) виконати 3 завдання.

1. Побудувати таблицю залежності довжини ряду n , що забезпечує точність функції не меншу за задане значення eps у точці $x = (b + a)/2$, від eps :

eps	n	Абсолютна похибка	Залишковий член
10^{-2}	4	0.005	0.001
...

Значення eps змінюється від 10^{-2} до 10^{-14} з кроком 10^{-3} .

2. Для n (довжина ряду фіксована й дорівнює n), отриманого в п.1 при $eps = 10^{-8}$, у точках $x_i = a + h \cdot i$, $h = (b - a)/10$, $i = 0, \dots, 10$ обчислити абсолютну похибку та залишковий член ряду. Результати подати у вигляді таблиці:

x_i	Абсолютна похибка	Залишковий член
0	0.005	0.001
...

П р и м і т к а. Точним значенням функції вважати результат, що його дає бібліотечна функція Ci , при обчисленні якої отримують всі члени ряду відмінні від нуля.

Індивідуальне завдання:

Варіант	Функція $f(x)$	Інтервал $[a, b]$
12	chx	$[0; 1]$

Код програми:

```

// my task is chx on the interval [0, 1]

interface Task1Data {
  eps: string,
  n: number,
  absoluteError: number,
  reminder: number,
  calculatedValue: number,
  realValue: number
}

interface Task2Data {
  x: number,
  absoluteError: number,
  reminder: number,
  calculatedValue: number,
  realValue: number
}

class Calculation {
  public static readonly a: number = 0
  public static readonly b: number = 1

  public static firstTableGeneration(): void {
    const data: Task1Data[] = []
    const x = (this.b + this.a) / 2
    console.log("x = " + x)
    for (let epsDegree: number = 2; epsDegree <= 14; epsDegree += 3) {
      const eps: number = 10**-epsDegree
      let res: number = 1
      let uk: number = x**2 / 2
      let k = 2
      while (eps < uk*2/3 || ((k = k - 2) && false)) {
        res += uk
        uk *= x**2 / ((k + 1) * (k + 2))
        k += 2
      }
      const task1Data: Task1Data = {
        eps: '10^-' + epsDegree,
        n: k,
        absoluteError: Math.abs(Math.cosh(x) - res),
        reminder: uk,
        calculatedValue: res,
        realValue: Math.cosh(x)
      }
      data.push(task1Data)
    }
    console.table(data)
  }

  public static secondTableGeneration(n: number): void {
    const data: Task2Data[] = []
    const h: number = (this.b - this.a) / 10
    console.log('h = ' + h)
    for (let i = 0; i <= 10; i++) {
      const x: number = this.a + h * i
      let res: number = 1
      let uk: number = x**2 / 2
      for (let k = 2; k <= n; k += 2) {
        res += uk
        uk *= x**2 / ((k + 1) * (k + 2))
      }
      const task2Data: Task2Data = {
        x: x,
        absoluteError: Math.abs(Math.cosh(x) - res),
        reminder: uk,
        calculatedValue: res,
        realValue: Math.cosh(x)
      }
      data.push(task2Data)
    }
    console.table(data)
  }
}

Calculation.firstTableGeneration()

```

Calculation.secondTableGeneration(10)

Налагодження та результати:

x = 0.5


(index)	eps	n	absoluteError	remainder	calculatedValue	realValue
0	'10^-2'	2	0.002625965206380698	0.002604166666666665	1.125	1.1276259652063807
1	'10^-5'	6	9.715082516237317e-8	9.688120039682538e-8	1.1276258680555555	1.1276259652063807
2	'10^-8'	8	2.6962476695757687e-10	2.691144455467372e-10	1.127625964936756	1.1276259652063807
3	'10^-11'	10	5.102585021177219e-13	5.096864498991234e-13	1.1276259652058704	1.1276259652063807
4	'10^-14'	12	6.661338147750939e-16	7.001187498614334e-16	1.12762596520638	1.1276259652063807

h = 0.1

(index)	x	absoluteError	remainder	calculatedValue	realValue
0	0	0	0	1	1
1	0.1	2.220446049250313e-16	2.087675698786812e-21	1.0050041680558033	1.0050041680558035
2	0.2	0	8.551119662230782e-18	1.020066755619076	1.020066755619076
3	0.30000000000000004	1.1102230246251565e-15	1.1094764610389628e-15	1.0453385141288594	1.0453385141288605
4	0.4	3.5083047578154947e-14	3.5025386136497285e-14	1.0810723718384199	1.081072371838455
5	0.5	5.102585021177219e-13	5.096864498991234e-13	1.1276259652058704	1.1276259652063807
6	0.6000000000000001	4.553246668592692e-12	4.544415584415592e-12	1.1854652182377146	1.1854652182422678
7	0.7000000000000001	2.8974156407457485e-11	2.8896118929456636e-11	1.2551690056019689	1.255169005630943
8	0.8	1.4396972503050165e-10	1.4346398161509288e-10	1.337434946160875	1.3374349463048447
9	0.9	5.922542456460178e-10	5.896212799310068e-10	1.4330863848565203	1.4330863854487745
10	1	2.0991941518389012e-9	2.0876756987868096e-9	1.5430806327160496	1.5430806348152437

Малюнок 1: результат програми

FROM THE MAKERS OF WOLFRAM LANGUAGE AND MATHEMATICA



$1 + \frac{(0.5)^2}{2}$

NATURAL LANGUAGE MATH INPUT

Input: $1 + \frac{0.5^2}{2}$

Result: 1.125

Step-by-step solution

Малюнок 2: перевірка результату степеневого ряду для n=2 в точці x=0.5

Висновки:

Було перевірено на практиці теорію наближення чисел за допомогою ряду Тейлора. Для того, аби прорахувати функцію, ми використовуємо наближення функції в деякому недалекому околі точки, якщо ми не можемо зробити потрібно робити деякі перетворення.

Щодо наближення в першій таблиці ми бачимо, що похибка зменшується і з збільшенням членів ряду.

Щодо наближення в другій таблиці ми бачимо, що біля точки 0 менше похибка, навіть з такою самою кількістю членів. Це тому, що ми використовували ряд Маклорена, який має найбільшу точність приближення в точці 0.

Блоксхеми

