­­­Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Лабораторная работа №3 по дисциплине  
«Вычислительная математика»

"Численное интегрирование"

Выполнил: Сафронов Егор Михайлович

Группа: P3213

Преподаватель: Малышева Татьяна Алексеевна

г. Санкт-Петербург

2022

Цель работы:

Найти приближенное значение определенного интеграла с требуемой точностью различными численными методами.

Задание лабораторной работы:

1. № варианта определяется как номер в списке группы согласно ИСУ
2. Вычислить интеграл точно.
3. Вычислить интеграл по формуле Ньютона – Котеса при .
4. Вычислить интеграл по формулам средних прямоугольников, трапеций и Симпсона при .
5. Сравнить результаты с точным значением интеграла.
6. Определить относительную погрешность вычислений.
7. Реализовать в программе методы по выбору пользователя:

* Метод прямоугольников (3 модификации: левые, правые, средние)
* Метод Симпсона

1. Предусмотреть вывод результатов: значение интеграла, число разбиения интервала интегрирования для достижения требуемой точности.

Вычислительная реализация через Ньютона-Лейбница:

Расчет методом Ньютона-Котеса:

Использованные веса

Формула для расчета , где

y0 = 2.000000 x1 = 2.333333 x2 = 2.666667 x3 = 3.000000

x0 = 2.000000 y1 = 12.000000 y2 = 25.777778 y3 = 44.000000

x4 = 3.333333 x5 = 3.666667 x6 = 4.000000

y4 = 67.333333 y5 = 96.444444 y6 = 132.000000

res = 103.333333

Расчет методом средних прямоугольников:

где каждое – это значение, рассчитанное как

y0 = 6.569444 -- значение f(x0 + h / 2)

x0 = 2.000000

h = 0.333333

y1 = 6.569444 y2 = 18.375000 y3 = 34.291667 y4 = 54.986111 y5 = 81.125000

x1 = 2.333333 x2 = 2.666667 x3 = 3.000000 x4 = 3.333333 x5 = 3.666667

y6 = 113.375000 -- вычисленное значение вне пределов интегрирования, в подсчете не используется

x6 = 4.000000

type: center

square res: 105.097222

square n: 6

Расчет методом трапеций:

где каждое – это значение, рассчитанное как

y0 = 2.000000

x0 = 2.000000

h = 0.333333

x1 = 2.333333 x2 = 2.666667 x3 = 3.000000 x4 = 3.333333 x5 = 3.666667

y1 = 12.000000 y2 = 25.777778 y3 = 44.000000 y4 = 67.333333 y5 = 96.444444

x6 = 4.000000

y6 = 132.000000

trapezia res: 104.185185

trapezia n: 6

Расчет методом Симпсона:

y0 = 2.000000 – начальный y

x0 = 2.000000

h = 0.333333

x1 = 2.333333 x2 = 2.666667 x3 = 3.000000 x4 = 3.333333 x5 = 3.666667

y1 = 12.000000 y2 = 25.777778 y3 = 44.000000 y4 = 67.333333 y5 = 96.444444

x6 = 4.000000

y6 = 132.000000 -- последний y

sumEven = 93.111111 – сумма нечетных

sumOdd = 152.444444 – сумма четных

simpson res: 103.333333

simpson n: 6

Сравнение способов:

newton-cotes res: 103.333333 – относительная погрешность <0.0000001%

simpson res: 103.333333 – относительная погрешность <0.0000001%

trapezia res: 104.185185 – относительная погрешность 0.824%

square res: 105.097222 – относительная погрешность 1.706%

**true res: 103.3(3)**

Самой точный результат дали методы Симпсона и Ньютона-Котеса, что логично. Затем по точности идет метод трапеций, а затем метод средних прямоугольников.

Примеры работы программы:

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generated

Text

Description automatically generated

Реализация расчета Метода Симпсона:

func entryPointSimpson(innerN int) {  
 xI1 = a  
 h = (b - a) / float64(innerN)  
 xI = xI1 + h  
 yI0 = f(a)  
 yI = f(a)  
 counter++

for i := 0; i < innerN; i++ {  
 yI = f(xI)  
 xI1 = xI  
 xI = xI1 + h  
 if counter != innerN {  
 if counter%2 == 0 {  
 sumEven += yI  
 } else {  
 sumOdd += yI  
 }  
 }  
 counter++  
 }  
 yI = f(b)   
 result = h / 3 \* (yI0 + 4\*sumOdd + 2\*sumEven + yI)  
}

Реализация расчета Метода прямоугольников:

func entryPointSquare(calcType string, innerN int) {  
 result = 0  
 if calcType == "left" {  
 xI1 = a  
 h = (b - a) / float64(innerN)  
 xI = xI1 + h  
 yI1 = f(a)  
 result += h \* yI1  
  
 for i := 0; i < innerN; i++ {  
 xI1 = xI  
 yI1 = f(xI)  
 xI = xI1 + h  
 result += h \* yI1  
 }  
 } else if calcType == "right" {  
 xI1 = a  
 h = (b - a) / float64(innerN)  
 xI = xI1 + h  
 yI = f(xI)  
 result += h \* yI  
  
 for i := 0; i < innerN; i++ {  
 xI1 = xI  
 xI = xI1 + h  
 yI = f(xI)  
 result += h \* yI  
 }  
 } else if calcType == "center" {  
 h = (b - a) / float64(innerN)  
 xI1 = a  
 xI = xI1 + h  
 yI = f((xI1 + xI) / 2)  
 result += h \* yI

for i := 0; i < innerN; i++ {  
 result += h \* yI  
  
 xI1 = xI  
 xI = xI1 + h  
 yI = f((xI1 + xI) / 2)  
 }  
 }  
}

Выводы по работе:

Реализованы методы Симпсона и прямоугольников. А также посчитан интеграл методами Ньютона-Котеса и трапеций. Методы Нютона-Котеса и Симпсона для подобных функций вне конкуренции так как раньше всех достигает необходимой точности.