Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина «Методы численного анализа»

**ОТЧЕТ**

к лабораторной работе №8

на тему:

**«ЧИСЛЕННОЕ ДИФФЕРЕНЦИРОВАНИЕ**

**И ИНТЕГРИРОВАНИЕ ФУНКЦИЙ»**

БГУИР 1-40-04-01

|  |
| --- |
| Выполнил студент группы 253504  Дмитрук Богдан Ярославович |
| 29.11.2023 |
| (дата, подпись студента) |
| Проверил  Анисимов Владимир Яковлевич |
|  |
| (дата, подпись преподавателя) |

Минск 2023

**СОДЕРЖАНИЕ**

[1. Цели выполнения задания 3](#_Toc147528122)

[2. Краткие теоретические сведения 4](#_Toc147528123)

[3. Задание 11](#_Toc147528124)

[4. Программная реализация 12](#_Toc147528125)

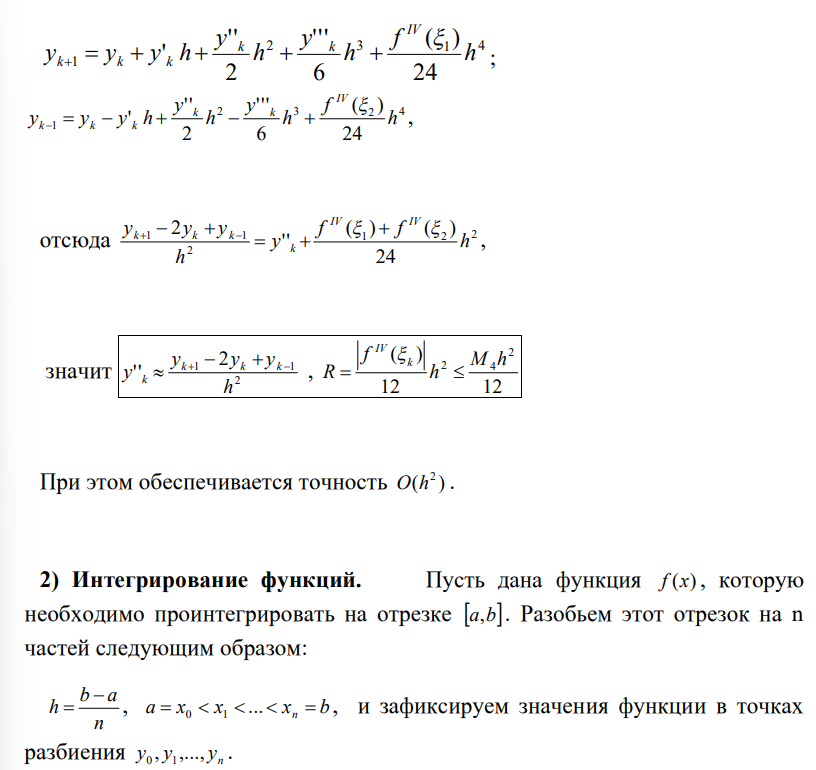
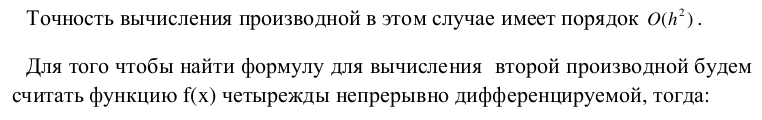
[5. Полученные результаты 16](#_Toc147528126)

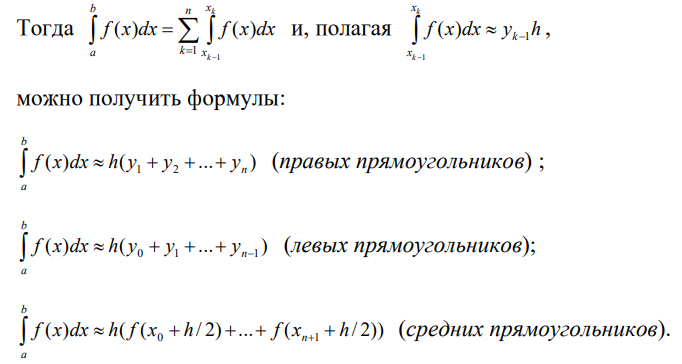
[6. Выводы 18](#_Toc147528127)

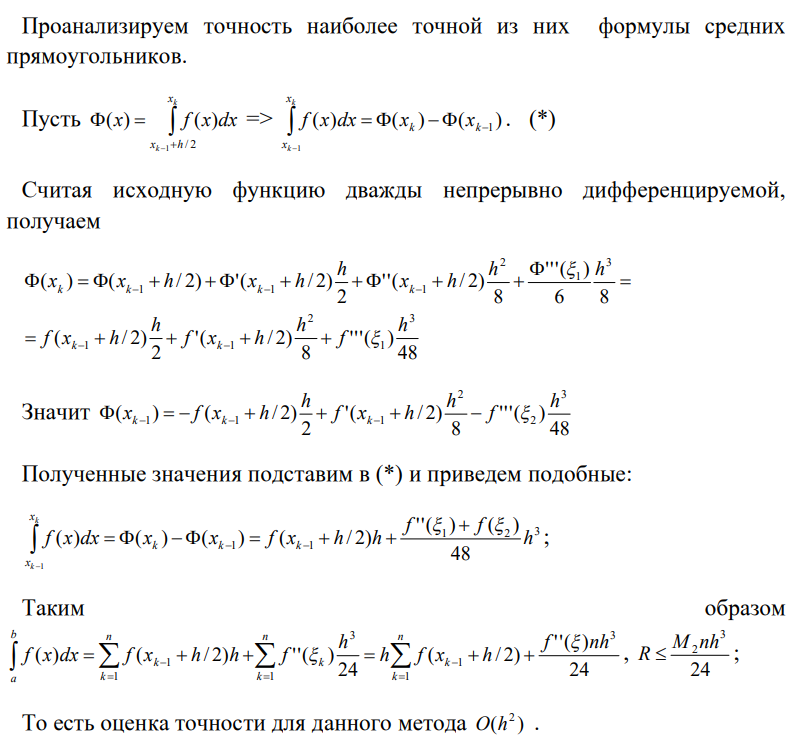
1. ЦЕЛИ ВЫПОЛНЕНИЯ ЗАДАНИЯ
2. Изучить методы численного вычисления производных и методы численного интегрирования.
3. Сравнить методы по трудоёмкости и точности.
4. Составить программный продукт, реализующий выполнение тестового задания в соответствие с вариантом.

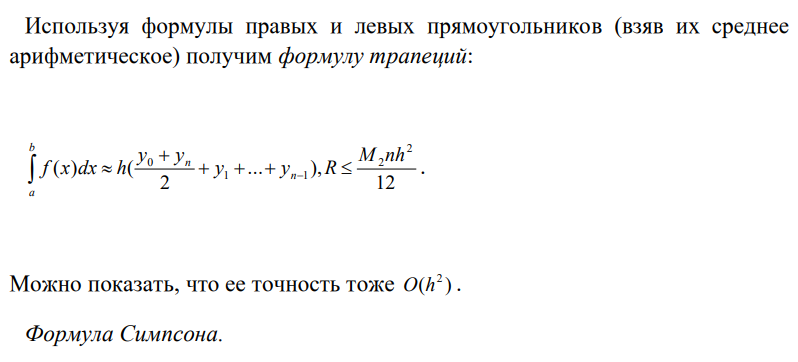
1. **КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

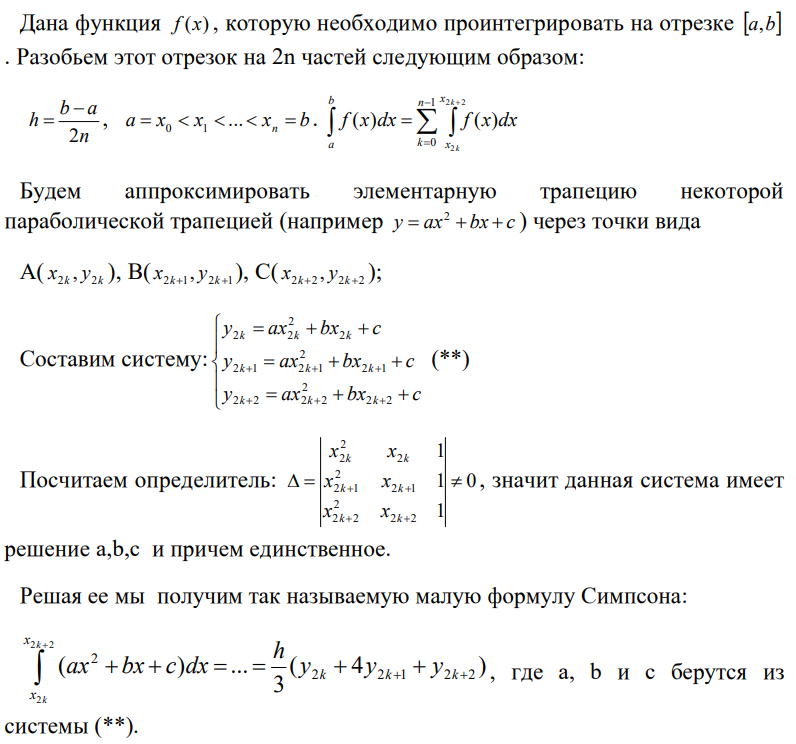
# 

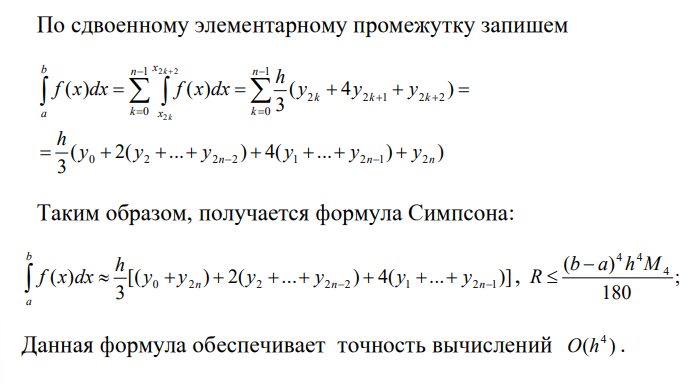


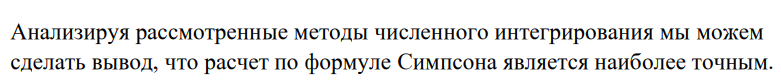




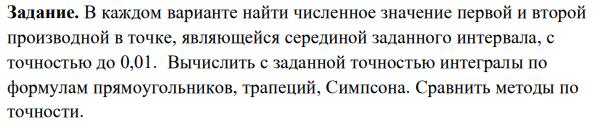


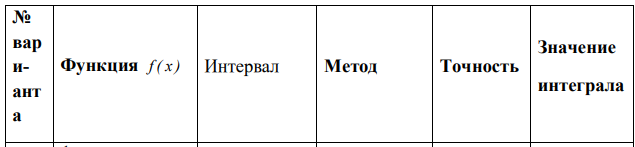






1. **ЗАДАНИЕ**

****





1. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

Функции численного дифференцирования:

def deriviative\_first(f, x, f2, f3):

M2 = abs(f2.subs(z, x))

if M2 == 0:

M3 = abs(f3.subs(z, x))

h = (6 \* epsilon / M3) \*\* (1 / 2)

else:

h2 = 2 \* epsilon / M2

M3 = abs(f3.subs(z, x))

h3 = (6 \* epsilon / M3) \*\* (1 / 2)

h = min(h2, h3)

return (f.subs(z, x + h) - f.subs(z, x - h)) / (2 \* h)

def deriviative\_second(f, x, f4):

M4 = abs(f4.subs(z, x))

h2 = abs((12 \* epsilon / M4))

h = abs(h2 \*\* (1 / 2))

return (f.subs(z, x + h) - 2 \* f.subs(z, x) + f.subs(z, x - h)) / h2

Функции численного интегрирования:

def int\_middle\_rectangle(f, l, r, n):

h = (r - l) / n

x = l + h / 2

s = 0.0

while x < r:

s += f.subs(z, x) \* h

x += h

return s

def integral\_trapezoid(f, l, r, n):

h = (r - l) / n

x = l + h / 2

s = 0.0

while x < r:

s += ((f.subs(z, x - h / 2) + f.subs(z, x + h / 2)) / 2) \* h

x += h

return s

def integral\_simpson(f, l, r, n):

h = (r - l) / n

x = l + h / 2

s = 0.0

while x < r:

fa = f.subs(z, x - h / 2)

fm = f.subs(z, x)

fb = f.subs(z, x + h / 2)

s += (fa + 4 \* fm + fb) \* h / 6

x += h

return s

def integral\_rand\_segments\_true(method, f, l, r, epsilon=0.000001):

left\_coeff, right\_coeff = 1 / 3, 1 / 2

h\_prev = r - l

ans\_prev = method(f, l, r, 1)

while True:

h\_new = h\_prev \* (left\_coeff + (right\_coeff - left\_coeff) \* np.random.rand())

n = math.floor((r - l) / h\_new)

m = l + h\_new \* n

ans\_new = method(f, l, m, n) + method(f, m, r, 1)

if abs(ans\_new - ans\_prev) < epsilon:

print("\nN :", n)

return ans\_new

ans\_prev = ans\_new

h\_prev = h\_new

def integral\_rand\_segments(method, f, l, r, epsilon=0.000001):

n = 2

ans\_prev = method(f, l, r, 1)

ans\_new = method(f, l, r, n)

while True:

if abs(ans\_new - ans\_prev) < epsilon:

print("\nN :", n)

return ans\_new

ans\_prev = ans\_new

n \*= 2

ans\_new = method(f, l, r, n)

def integral\_middle\_rectangle\_via\_estimation(f, l, r, epsinon=0.000001, m2deLR=0):

if m2deLR > 0.0:

M2 = m2deLR

h = (24 \* epsinon / (r - l) / M2) \*\* (1 / 2)

n = np.ceil((r - l) / h)

return int\_middle\_rectangle(f, l, r, n)

else:

return integral\_rand\_segments(int\_middle\_rectangle, f, l, r)

def integral\_trapezoid\_via\_estimation(f, l, r, epsinon=0.000001, m2deLR=0):

if m2deLR > 0.0:

M2 = m2deLR

h = (12 \* epsinon / (r - l) / M2) \*\* (1 / 2)

n = np.ceil((r - l) / h)

return integral\_trapezoid(f, l, r, n)

else:

return integral\_rand\_segments(integral\_trapezoid, f, l, r)

def integral\_simpson\_via\_estimation(f, l, r, epsinon=0.000001, m4deLR=0):

if m4deLR > 0.0:

M4 = m4deLR

h = (180 \* epsinon / (r - l) / M4) \*\* (1 / 4)

n = np.ceil((r - l) / h)

return integral\_simpson(f, l, r, n)

else:

return integral\_rand\_segments(integral\_simpson, f, l, r)

1. ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ

Вывод программного продукта:

derivative of the first order: 0.967616338994002

derivative with numerical differentiation: 0.967616338994153

delta = 1.50768286744096E-13

derivative of the second order = 0.832809593966394

derivative with numerical differentiation: 0.832810594098064

delta = 0.00000100013166959290

integral: 1.689378830261396

N : 2048

middle rectangle method: 1.68937944240713

delta = 6.12145729661506E-7

N : 16384

trapezoid method: 1.68937866673864

delta = 1.63522756002621E-7

N : 8192

simpson method: 1.68937875914320

delta = 7.11181920021176E-8

1. ВЫВОДЫ

Таким образом, в ходе выполнения лабораторной работы были изучены и сравнены по трудоёмкости и точности методы численного вычисления производных и методы численного интегрирования. Составлена компьютерная программа, на тестовых примерах проверена правильность её работы. Для функции заданного варианта найдено численное значение первой и второй производной в точке, вычислены с заданной точностью интегралы по формулам прямоугольников, трапеций, Симпсона.

Сравнивая полученный результат, можно сказать что формула средних прямоугольников дает самый быстрый, но наименее точный результат, формула трапеций дает результат более точный, но требует больших ресурсов, формула Симпсона же является оптимальной, так как за меньшее чем у формулы трапеций время выдает более точный результат.