Министерство образования Республики Беларусь
Учреждение образования
“БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ”

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: Методы численного анализа

ОТЧЁТ
к лабораторной работе 8
на тему

Численное дифференцирование и интегрирование функций

Выполнил: студент группы 053502
Герчик Артём Вадимович

Проверил: Анисимов Владимир Яковлевич

Минск 2022

Оглавление

Цели выполнения задания

Изучить методы численного вычисления производных и методы численного интегрирования. Сравнить методы по трудоёмкости, точности. Выполнить тестовое задание по численному дифференцированию и интегрированию.

Краткие теоретические сведения

1) Численное дифференцирование. Для получения формул вычисления производной разобьем отрезок *[a, b]* на n частей следующим образом:

$$h = \frac{b-a}{n}, \quad a = x_0 < x_1 < \ldots < x_n = b.$$

Тогда $y_k = f(x_k)$, $y_k' = f'(x_k)$, и по формуле Тейлора (считая функцию дважды непрерывно дифференцируемой) получаем

$$y_{k+1} = y_k + y_k'h + f''(\xi)h^2\frac{1}{2},$$

или

$$y'_k = \frac{y_{k+1} - y_k}{h} - \frac{f''(\xi)h}{2} \ ;$$

где ξ – некоторая точка на $[x_k, x_{k+1}]$.

Таким образом получаем формулу для приближенного вычисления производной : $\boxed{y'_k \approx \frac{y_{k+1} - y_k}{h}, \text{ с погрешностью } R \le \frac{M_2 h}{2}}$,

где $M_2 = \max\limits_{a \le x \le b} |f''(x)|$.

Таким образом, обеспечивается точность $O(h)$.

Далее воспользуемся следующей теоремой.

Теорема(о среднем).

Пусть функция f(x) непрерывна на отрезке [a, b] и $x_1, \ldots, x_n \in [a, b]$. Тогда ∃ точка $\xi \in [a, b]$ такая, что $\frac{f(x_1) + \ldots + f(x_n)}{n} == f(\xi)$.

Считая функцию трижды непрерывно дифференцируемой, получим:

$$y_{k+1} = y_k + y'_k h + \frac{y''_k}{2} h^2 + \frac{f'''(\xi_1)}{6} h^3;$$

$$y_{k-1} = y_k - y'_k h + \frac{y''_k}{2} h^2 - \frac{f'''(\xi_2)}{6} h^3.$$

Отсюда можем определить производную как

$y'_k = \dfrac{y_{k+1} - y_{k-1}}{2h} + \dfrac{f'''(\xi_1) + f'''(\xi_2)}{12} h^2$ и, применяя теорему о среднем, получаем:

$$y'_k = \frac{y_{k+1} - y_{k-1}}{2h} + \frac{f'''(\xi_k)}{6} h^2.$$

Т.е. имеет место формула для приближенного вычисления производной:

$\boxed{y'_k \approx \dfrac{y_{k+1} - y_{k-1}}{2h}, \ R \le \dfrac{M_3 h^2}{6}}$, где $M_3 = \max\limits_{a \le x \le b} |f'''(x)|$.

Точность вычисления производной в этом случае имеет порядок $O(h^2)$.

Для того чтобы найти формулу для вычисления второй производной будем считать функцию f(x) четырежды непрерывно дифференцируемой, тогда:

$$y_{k+1} = y_k + y'_k h + \frac{y''_k}{2} h^2 + \frac{y'''_k}{6} h^3 + \frac{f^{IV}(\xi_1)}{24} h^4;$$

$$y_{k-1} = y_k - y'_k h + \frac{y''_k}{2} h^2 - \frac{y'''_k}{6} h^3 + \frac{f^{IV}(\xi_2)}{24} h^4,$$

отсюда $\dfrac{y_{k+1} - 2y_k + y_{k-1}}{h^2} = y''_k + \dfrac{f^{IV}(\xi_1) + f^{IV}(\xi_2)}{24} h^2$,

значит $\boxed{y''_k \approx \dfrac{y_{k+1} - 2y_k + y_{k-1}}{h^2}, \ R = \dfrac{|f^{IV}(\xi_k)|}{12} h^2 \le \dfrac{M_4 h^2}{12}}$

При этом обеспечивается точность $O(h^2)$.

2) Интегрирование функций. Пусть дана функция $f(x)$, которую необходимо проинтегрировать на отрезке $[a,b]$. Разобьем этот отрезок на n частей следующим образом:

$h = \frac{b-a}{n}$, $a = x_0 < x_1 < \ldots < x_n = b$, и зафиксируем значения функции в точках разбиения $y_0, y_1, \ldots, y_n$.

Тогда $\int_a^b f(x)dx = \sum_{k=1}^{n} \int_{x_{k-1}}^{x_k} f(x)dx$ и, полагая $\int_{x_{k-1}}^{x_k} f(x)dx \approx y_{k-1}h$,

можно получить формулы:

$\int_a^b f(x)dx \approx h(y_1 + y_2 + \ldots + y_n)$ (*правых прямоугольников*) ;

$\int_a^b f(x)dx \approx h(y_0 + y_1 + \ldots + y_{n-1})$ (*левых прямоугольников*);

$\int_a^b f(x)dx \approx h(f(x_0 + h/2) + \ldots + f(x_{n+1} + h/2))$ (*средних прямоугольников*).

Проанализируем точность наиболее точной из них формулы средних прямоугольников.

Пусть $\Phi(x) = \int_{x_{k-1}+h/2}^{x_k} f(x)dx$ => $\int_{x_{k-1}}^{x_k} f(x)dx = \Phi(x_k) - \Phi(x_{k-1})$. (*)

Считая исходную функцию дважды непрерывно дифференцируемой, получаем

$$\Phi(x_k) = \Phi(x_{k-1} + h/2) + \Phi'(x_{k-1} + h/2)\frac{h}{2} + \Phi''(x_{k-1} + h/2)\frac{h^2}{8} + \frac{\Phi'''(\xi_1)}{6}\frac{h^3}{8} =$$
$$= f(x_{k-1} + h/2)\frac{h}{2} + f'(x_{k-1} + h/2)\frac{h^2}{8} + f'''(\xi_1)\frac{h^3}{48}$$

Значит $\Phi(x_{k-1}) = -f(x_{k-1} + h/2)\frac{h}{2} + f'(x_{k-1} + h/2)\frac{h^2}{8} - f'''(\xi_2)\frac{h^3}{48}$

Полученные значения подставим в (*) и приведем подобные:

$$\int_{x_{k-1}}^{x_k} f(x)dx = \Phi(x_k) - \Phi(x_{k-1}) = f(x_{k-1} + h/2)h + \frac{f''(\xi_1) + f(\xi_2)}{48} h^3;$$

Таким образом

$$\int_a^b f(x)dx = \sum_{k=1}^{n} f(x_{k-1} + h/2)h + \sum_{k=1}^{n} f''(\xi_k)\frac{h^3}{24} = h\sum_{k=1}^{n} f(x_{k-1} + h/2) + \frac{f''(\xi)nh^3}{24}, \ R \le \frac{M_2 nh^3}{24};$$

То есть оценка точности для данного метода $O(h^2)$.

Используя формулы правых и левых прямоугольников (взяв их среднее арифметическое) получим *формулу трапеций*:

$$\int_a^b f(x)dx \approx h(\frac{y_0 + y_n}{2} + y_1 + ... + y_{n-1}), R \le \frac{M_2 nh^2}{12}.$$

Можно показать, что ее точность тоже $O(h^2)$.

Формула Симпсона.

Дана функция $f(x)$, которую необходимо проинтегрировать на отрезке $[a,b]$. Разобьем этот отрезок на 2n частей следующим образом:

$$h = \frac{b-a}{2n}, \quad a = x_0 < x_1 < ... < x_n = b. \ \int_a^b f(x)dx = \sum_{k=0}^{n-1} \int_{x_{2k}}^{x_{2k+2}} f(x)dx$$

Будем аппроксимировать элементарную трапецию некоторой параболической трапецией (например $y = ax^2 + bx + c$) через точки вида

A(x_{2k}, y_{2k}), B(x_{2k+1}, y_{2k+1}), C(x_{2k+2}, y_{2k+2});

Составим систему: $$\begin{cases} y_{2k} = ax_{2k}^2 + bx_{2k} + c \\ y_{2k+1} = ax_{2k+1}^2 + bx_{2k+1} + c \quad (**) \\ y_{2k+2} = ax_{2k+2}^2 + bx_{2k+2} + c \end{cases}$$

Посчитаем определитель: $\Delta = \begin{vmatrix} x_{2k}^2 & x_{2k} & 1 \\ x_{2k+1}^2 & x_{2k+1} & 1 \\ x_{2k+2}^2 & x_{2k+2} & 1 \end{vmatrix} \neq 0$, значит данная система имеет решение a,b,c и причем единственное.

Решая ее мы получим так называемую малую формулу Симпсона:

$\int_{x_{2k}}^{x_{2k+2}} (ax^2 + bx + c)dx = ... = \frac{h}{3}(y_{2k} + 4y_{2k+1} + y_{2k+2})$, где a, b и c берутся из системы (**).

По сдвоенному элементарному промежутку запишем

$$\int_a^b f(x)dx = \sum_{k=0}^{n-1} \int_{x_{2k}}^{x_{2k+2}} f(x)dx = \sum_{k=0}^{n-1} \frac{h}{3}(y_{2k} + 4y_{2k+1} + y_{2k+2}) =$$

$$= \frac{h}{3}(y_0 + 2(y_2 + ... + y_{2n-2}) + 4(y_1 + ... + y_{2n-1}) + y_{2n})$$

Таким образом, получается формула Симпсона:

$$\int_a^b f(x)dx \approx \frac{h}{3}[(y_0 + y_{2n}) + 2(y_2 + ... + y_{2n-2}) + 4(y_1 + ... + y_{2n-1})],\ R \leq \frac{(b-a)^4 h^4 M_4}{180};$$

Данная формула обеспечивает точность вычислений $O(h^4)$.

Анализируя рассмотренные методы численного интегрирования мы можем сделать вывод, что расчет по формуле Симпсона является наиболее точным.

Программная реализация

```
import math

import numpy as np

np.random.seed(42)

# MAIN TASK #

L, R, DerXdot = 0, 2, 1


def f(x): return np.arctan(np.sqrt(x))


def F(x): return x * np.arctan(np.sqrt(x)) - np.sqrt(x) +
np.arctan(np.sqrt(x))


def fd1(x): return 1 / (2 * (x ** (1 / 2)) * (1 + x))


def fd2(x): return -1 / (4 * x ** (3 / 2) * (1 + x)) -
1 / (2 * (x ** (1 / 2)) * (1 + x) ** 2)
```

```
def fd3(x): return 3 / (8 * x ** (5 / 2) * (1 + x)) + 1 /
(2 * x ** (3 / 2) * (1 + x) ** 2) + 1 / (
        (x ** (1 / 2)) * (1 + x) ** 3)


def fd4(x): return -15 / (16 * x ** (7 / 2) * (1 + x)) -
9 / (8 * x ** (5 / 2) * (1 + x) ** 2) - 3 / (
        2 * x ** (3 / 2) * (1 + x) ** 3) - 3 / ((x **
(1 / 2)) * (1 + x) ** 4)


M2deLR, M4deLR = 0, 0

# MAIN TASK #

# # First Example #
#
#
# L, R, DerXdot = -1, 1, 0
#
#
# def f(x): return np.exp(x)
#
#
# def F(x): return np.exp(x)
```

```
#
#
# def fd1(x): return np.exp(x)
#
#
# def fd2(x): return np.exp(x)
#
#
# def fd3(x): return np.exp(x)
#
#
# def fd4(x): return np.exp(x)
#
#
# M2deLR, M4deLR = 0, 0
#
# # First Example #


# # Second Example #
#
#
# L, R, DerXdot = -1, 1, 1 / 2
#
#
# def f(x): return np.sqrt(1 - x ** 2)
```

```
#

#

# def F(x): return (1 / 2) * x * np.sqrt(1 - x ** 2) + (1
/ 2) * np.arcsin(x)

#

#

# def fd1(x): return -x / np.sqrt(1 - x ** 2)

#

#

# def fd2(x): return -x ** 2 / (-x ** 2 + 1) ** (3 / 2) -
1 / np.sqrt(-x ** 2 + 1)

#

#

# def fd3(x): return -3 * x ** 3 / (-x ** 2 + 1) ** (5 /
2) - 3 * x / (-x ** 2 + 1) ** (3 / 2)

#

#

# def fd4(x): return -15 * x ** 4 / (-x ** 2 + 1) ** (7 /
2) - 18 * x ** 2 / (-x ** 2 + 1) ** (5 / 2) - 3 / (
#             -x ** 2 + 1) ** (3 / 2)

#

#

# M2deLR, M4deLR = 0, 0

#

# # Second Example #
```

```
IntEps = 0.000001
IntFormatString = "{:.7f}"

DerEps = 0.01
DerFormatString = "{:.3f}"


def diff_first(f, x, d):
    return (f(x + d) - f(x - d)) / (2 * d)


def diff_first_via_estimation(f, x):
    M2 = abs(fd2(x))
    df = 2 * DerEps / M2
    M3 = abs(fd3(x))
    ds = (6 * DerEps / M3) ** (1 / 2)
    return diff_first(f, x, min(df, ds))


def diff_first_via_ten_in_minus_5(f, x):
    d = 10.0 ** -5
    return diff_first(f, x, d)


def derivative_second(f, x, d):
```

```
    return (f(x + d) - 2 * f(x) + f(x - d)) / (d ** 2)


def derivative_second_via_estimation(f, x):
    M4 = abs(fd4(x))
    d = (12 * DerEps / M4) ** (1 / 2)
    return derivative_second(f, x, d)


def derivative_second_via_ten_in_minus_4(f, x):
    d = 10.0 ** -4
    return derivative_second(f, x, d)


print("Численное дифференцирование и интегрирование
функций \n")

print()

print("Первая производная = " +
DerFormatString.format(fd1(DerXdot)))
print("Приближенное    = " +
DerFormatString.format(diff_first_via_estimation(f,
DerXdot)))
print("Округленное   = " +
DerFormatString.format(diff_first_via_ten_in_minus_5(f,
DerXdot)))
```

```
print()


print("Вторая Производная = " + 
DerFormatString.format(fd2(DerXdot)))
print("Приближенное     = " + 
DerFormatString.format(derivative_second_via_estimation(f
, DerXdot)))
print("Округленное    = " + 
DerFormatString.format(derivative_second_via_ten_in_minus
_4(f, DerXdot)))
print()



def integral_via_middle_rectangles(f, L, R, N):
    h = (R - L) / N
    x = L + h / 2
    s = 0.0
    while x < R:
        s += f(x) * h
        x += h
    return s



def integral_via_trapezoids(f, L, R, N):
    h = (R - L) / N
    x = L + h / 2
    s = 0.0
```

```
    while x < R:
        s += ((f(x - h / 2) + f(x + h / 2)) / 2) * h
        x += h
    return s


def integral_via_simpson(f, L, R, N):
    h = (R - L) / N
    x = L + h / 2
    s = 0.0
    while x < R:
        fa = f(x - h / 2)
        fm = f(x)
        fb = f(x + h / 2)
        s += (fa + 4 * fm + fb) * h / 6
        x += h
    return s


def integral_via_random_segments(method, f, L, R):
    LeftCoeff, RightCoeff = 1 / 3, 1 / 2
    h_prev = R - L
    ans_prev = method(f, L, R, 1)
    while True:
        h_new = h_prev * (LeftCoeff + (RightCoeff -
LeftCoeff) * 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) < IntEps:
            print("\nN =", N)
            return ans_new
        ans_prev = ans_new
        h_prev = h_new


def integral_via_middle_rectangles_via_estimation(f, L, R):
    if M2deLR > 0.0:
        M2 = M2deLR
        h = (24 * IntEps / (R - L) / M2) ** (1 / 2)
        N = np.ceil((R - L) / h)
        return integral_via_middle_rectangles(f, L, R, N)
    else:
        return integral_via_random_segments(integral_via_middle_rectangles, f, L, R)


def integral_via_trapezoids_via_estimation(f, L, R):
    if M2deLR > 0.0:
        M2 = M2deLR
```

```
        h = (12 * IntEps / (R - L) / M2) ** (1 / 2)
        N = np.ceil((R - L) / h)
        return integral_via_trapezoids(f, L, R, N)
    else:
        return
integral_via_random_segments(integral_via_trapezoids, f,
L, R)


def integral_via_simpson_via_estimation(f, L, R):
    if M4deLR > 0.0:
        M4 = M4deLR
        h = (180 * IntEps / (R - L) / M4) ** (1 / 4)
        N = np.ceil((R - L) / h)
        return integral_via_simpson(f, L, R, N)
    else:
        return
integral_via_random_segments(integral_via_simpson, f, L,
R)


print()
intprecised = F(R) - F(L)
print("Интеграл =            " +
IntFormatString.format(intprecised))
```

```
def delta(intappr): return np.ceil(abs(intappr -
intprecised) * (1 / (IntEps / 10))) * (IntEps / 10)



intappr =
integral_via_middle_rectangles_via_estimation(f, L, R)
print(("Метод Средних Прямоугольников = " +
IntFormatString + " | дельта = " +
IntFormatString).format(intappr,

delta(intappr)))
intappr = integral_via_trapezoids_via_estimation(f, L, R)
print(("Метод Трапеций = " + IntFormatString + " | дельта
= " + IntFormatString).format(intappr, delta(intappr)))
intappr = integral_via_simpson_via_estimation(f, L, R)
print(("Метод Симпсона = " + IntFormatString + " | дельта
= " + IntFormatString).format(intappr, delta(intappr)))
print()
```

Полученные результаты

Тестовый пример 1

Функция	*Точка*	*Интервал*
$y = e^x$	$x = 0$	$[-1; 1]$

Численное дифференцирование и интегрирование функций

Первая производная = 1.000
Приближенное = 1.000
Округленное = 1.000

Вторая Производная = 1.000
Приближенное = 1.010
Округленное = 1.000

Интеграл = 2.3504024

N = 1231
Метод Средних Прямоугольников = 2.3504021 | дельта = 0.0000003

N = 3465
Метод Трапеций = 2.3504025 | дельта = 0.0000001

N = 35
Метод Симпсона = 2.3504024 | дельта = 0.0000001

Тестовый пример 2

Функция	Точка	Интервал
$y = \sqrt{1 - x^2}$	$x = 0.5$	$[-1; 1]$

Численное дифференцирование и интегрирование функций

Первая производная = -0.577
Приближенное = -0.577
Округленное = -0.577

Вторая Производная = -1.540
Приближенное = -1.550
Округленное = -1.540

Интеграл = 1.5707963

N = 18688
Метод Средних Прямоугольников = 1.5707965 | дельта = 0.0000002

N = 59133
Метод Трапеций = 1.5707962 | дельта = 0.0000001

N = 10892
Метод Симпсона = 1.5707961 | дельта = 0.0000002

ЗАДАНИЕ
Вариант 7
Задание. В каждом варианте найти численное значение первой и второй производной в точке, являющейся серединой заданного интервала, с точностью до 0,01. Вычислить с заданной точностью интегралы по формулам прямоугольников, трапеций, Симпсона. Сравнить методы по точности.

7.	$arctg\sqrt{x}$	[0 ; 2]	Симпсона	0,000001	**1,451735**5

Численное дифференцирование и интегрирование функций

Первая производная = 0.250
Приближенное = 0.250
Округленное = 0.250

Вторая Производная = -0.250
Приближенное = -0.261
Округленное = -0.250

Интеграл = 1.4517363

N = 6293
Метод Средних Прямоугольников = 1.4517366 | дельта = 0.0000004

N = 23919
Метод Трапеций = 1.4517361 | дельта = 0.0000002

N = 9955
Метод Симпсона = 1.4517362 | дельта = 0.0000001

Выводы

Таким образом, в ходе выполнения лабораторной работы были изучены 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.