Федеральное государственное автономное

образовательное учреждение высшего образования

КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Направление: Фундаментальная информатика и

Информационные технологии

Отчета по Учебной практике

Вычислительные методы

Студент 3 курса

Коробейников

Кирилл

09-732 группа

Табулирование трансцендентных функций.

Постановка задачи.

Вариант 10.

Одна из специальных функций математической физики - интегральный синус, определяется следующим образом

$= \quad \overline{}$

Цель задания – изучить и сравнить различные способы приближения данной функции.

Для решения этой задачи мы будем использовать язык программирования Python версии 3.6

1 задание.

Задание:

1. Про табулировать $\mathrm{Si}(x)$ на отрезке с шагом h с точностью , основываясь на ряде Тейлора, предварительно вычислив его

$$= \frac{+}{+ \quad +},$$

где = = = = и получить, таким образом, таблицу

Замечание: h = 0.4

x_0	x_1		x_2	…	x_n
f_0	f_1		f_2	…	f_n

= = + =

Перед решением этой задачи необходимо ввести краткое приложения по определиям для решения этой задачи

Решение:

Степенной ряд с одной переменной — это формальное алгебраическое выражение вида:

$$C_0 + C_1x + C_2x^2 + \cdots + C_nx_n + \cdots = \sum_{n=0}^{\infty} C_nx_n$$

Ряд Те́йлора — разложение функции в бесконечную сумму степенных функций. Они применяются при аппроксимации функции многочленами. Для функции f(x) он выглядит таким образом:

$$f(x) = f(a) + \frac{f'(a)(x-a)}{1!} + \frac{f''(a)(x-a)_2}{2!} + \cdots + \frac{f^{(n)}(a)(x-a)^n}{n!} + \cdots$$

Терема Абеля. Если степенной ряд $\sum_{n=0}^{\infty} C_n x^n$ сходится при некотором значении $x = x_0$, то он сходится абсолютно при всех значениях, для которых $|x| < |x_0|$. Если этот ряд расходится при некотором значении $x = x_1$, то он расходится при всех значениях x, для которых $|x| > |x_1|$.

Реализация функции Тейлора

```
import numpy as np

import math



def q_calc(x,n):

    answ = ((-1)*(x**2)*(2*n+1))/((2*n+2)*(2*n+3)*(2*n+3))

    return answ


def SI():

    xm = np.arange(0,2.1,0.4)

    e = 10**(-4)

    answer = list()

    j=0

    for x in xm:

        a0 = x

        Sum = a0

        i = 0

        while (True):
```

```
            qn = q_calc(x,i)

            ai = a0*qn

            Sum += ai

            a0 =ai

            i += 1

            if (abs(qn) < e):

                break

        Sum = Sum

        answer.append([x,Sum])

        j+=1

    print(answer)


poluch_znach = SI()
```

Результат работы программы :([0.0,0.3964614647513729,0.7720957854819964,1.1080471990137188,1.3891804858704384, 1.6054129768026946], dtype=float)

2 задание.

Задание:

По полученной таблице значений построить интерполяционный полином Ньютона, приближающий Si(x)

$$L_n(x) = f(x_0) + (x - x_0) f(x_0, x_1) + (x - x_0)(x - x_1) f(x_0, x_1, x_2) + \ldots$$
$$+ (x - x_0)(x - x_1) \ldots (x - x_n) f(x_0, x_1, \ldots, x_n)$$

и вычислить погрешность интерполирования

$$= \qquad\qquad = \left| \qquad\qquad \right|$$

1. Проделать описанные выше действия, взяв в качестве узлов интерполяции равномерно распределенные узлы {xi}, i=0,…,n и корни полинома Чебышева, вычисляемые по формуле:

Для решения этой задачи необходимо воспользоваться результатами программы 1

Для начала рассмотрим построение интерполяционного полинома Ньютона на равномерно распределенных узлах.

Сама программа

```
def _poly_newton_coefficient(x,y):
    """
    x: list or np array contanining x data points
    y: list or np array contanining y data points
    """

    m = len(x)

    x = np.copy(x)
    a = np.copy(y)
    for k in range(1,m):
        a[k:m] = (a[k:m] - a[k-1])/(x[k:m] - x[k-1])
```

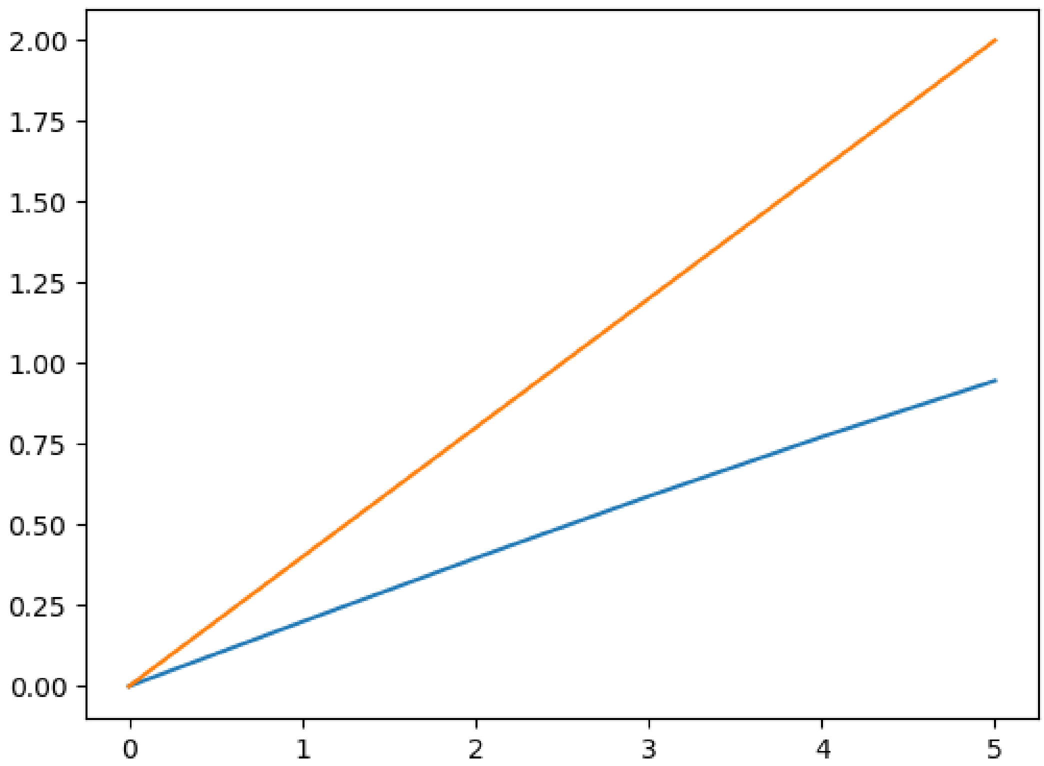
```
    return a


def newton_polynomial(x_data, y_data, x):
    """
    x_data: data points at x
    y_data: data points at y
    x: evaluation point(s)
    """
    a = _poly_newton_coefficient(x_data, y_data)
    n = len(x_data) - 1 # Degree of polynomial
    p = a[n]
    for k in range(1,n+1):
        p = a[n-k] + (x -x_data[n-k])*p
    return p


print(newton_polynomial(x,y,xm))
```

Выходным значением является список значений интерполяционного полинома в точках, заданных списком X_list.

Результат выполнения программы

[[0.0, 0.0], [0.4, 0.3964614647513729], [0.8, 0.7720957854819964], [1.2000000000000002, 1.1080471990137188], [1.6, 1.3891804858704384], [2.0, 1.6054129768026946]]

[0.   0.19954673 0.39646146 0.58813209 0.77209579 0.94608062]



Результат работы программы:

```
Погрешность в узлах интерполяции: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

3 задание.

На той же сетке узлов $\{x_i\}_{i=0}^{n}$ построить таблицу приближенных значений $J_0(x)$, используя составную квадратурную формулу трапеций

$$\int_c^d \varphi(t)\,dt = \sum_{i=1}^{N} \int_{z_{i-1}}^{z_i} \varphi(t)\,dt \approx \sum_{i=1}^{N} S_i(\varphi),$$

где $S_i(\varphi) = h_N \dfrac{\varphi(z_{i-1}) + \varphi(z_i)}{2}$, а z_i – точки разбиения отрезка интегрирования на N частей, $z_i = c + i \cdot h_N$, $h_N = \dfrac{c-d}{N}$.

Интеграл вычислить с точностью $\varepsilon = 10^{-4}$. Точность вычисления интеграла определяется сравнением результатов при различном числе разбиения отрезка интегрирования. Именно, точность ε считается достигнутой, если

$$\left| S^{N}(\varphi) - S^{(2N)}(\varphi) \right| \le \varepsilon,$$

$$S^{N}(\varphi) = \sum_{i=1}^{N} S_i(\varphi).$$

```
def tr_integral(f,xmin,xmax,n):
    dx=(xmax-xmin)/n
    area=0
    x=xmin
    for i in range(n):
        area+=dx*(f[i]+(f[i]+dx))/2
        x+=dx
    return area
```

Результат работы программы:

tr_integral = 2.090399303973407

Выводы: все формулы вычисляют интеграл с примерно одинаковой достаточной точностью. Так как рассматриваемая нами функция гладкая, можно использовать наиболее простые формулы.

ЛИСТИНГ ПРОГРАММЫ

```
import numpy as np

import math

import matplotlib.pyplot as plt



def q_calc(x,n):

    answ = ((-1)*(x**2)*(2*n+1))/((2*n+2)*(2*n+3)*(2*n+3))

    return answ
```

```
def SI():
    xm = np.arange(0,2.1,0.4)
    e = 10**(-4)
    answer = list()
    j=0
    for x in xm:
        a0 = x
        Sum = a0
        i = 0
        while (True):
            qn = q_calc(x,i)
            ai = a0*qn
            Sum += ai
            a0 =ai
            i += 1
            if (abs(qn) < e):
                break
        Sum = Sum
        answer.append([x,Sum])
```

```
        j+=1

    print(answer)



poluch_znach = SI()



#print(poluch_znach)

# берем и считаем Ln(x) и потом вычисляем значения других функций

#Необходимо задать значение x

y_list =
[0.0,0.3964614647513729,0.7720957854819964,1.1080471990137188,1.3891804858704384,1.605412
9768026946]

x = np.array(np.arange(0,2.1,0.4), dtype=float)

y =
np.array([0.0,0.3964614647513729,0.7720957854819964,1.1080471990137188,1.3891804858704384,
1.6054129768026946], dtype=float)

xm = np.arange(0,2.1,0.4)/2



def _poly_newton_coefficient(x,y):

    """

    x: list or np array contanining x data points

    y: list or np array contanining y data points

    """
```

```
    m = len(x)

    x = np.copy(x)
    a = np.copy(y)
    for k in range(1,m):
        a[k:m] = (a[k:m] - a[k-1])/(x[k:m] - x[k-1])

    return a

def newton_polynomial(x_data, y_data, x):
    """
    x_data: data points at x
    y_data: data points at y
    x: evaluation point(s)
    """
    a = _poly_newton_coefficient(x_data, y_data)
    n = len(x_data) - 1 # Degree of polynomial
    p = a[n]
    for k in range(1,n+1):
```

```
        p = a[n-k] + (x -x_data[n-k])*p

    return p




newton_res = list(newton_polynomial(x,y,xm))

print(newton_polynomial(x,y,xm))

print("----------")

plt.plot(newton_res)

plt.plot(x)





#print(math.log(0.4))

def tr_integral(f,xmin,xmax,n):

    dx=(xmax-xmin)/n

    area=0

    x=xmin

    for i in range(n):

        area+=dx*(f[i]+(f[i]+dx))/2

        x+=dx
```

```
    return area



print("tr_integral = {}".format(tr_integral(y_list,0,2,6)))
```

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Глазырина Л.Л., Карчевский М.М. Введение в численные методы, учебное пособие, Казань, Казан. ун-т. 2012, 122 с.
2. Самарский А. А., Гулин А.В. Численные методы. М.: Наука. 1989.