МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА (МИИТ)»

РУТ (МИИТ)

Кафедра ЦТУТП

**ЛАБОРАТОРНАЯ РАБОТА №1**

ПО ДИСЦИПЛИНЕ «ЧИСЛЕННЫЕ МЕТОДЫ»

НА ТЕМУ «ИНТЕРПОЛИРОВАНИЕ»

ВАРИАНТ №6

Выполнил: студент группы

УПМ-311 Масеёнок Е.Н.

Проверил: Посвянский В.П.

МОСКВА 2022

**Постановка задачи**

Составить таблицу значений аргумента и заданной функции на отрезке с шагом . Вывести на печать полученную таблицу . Выделить в таблице узлы интерполирования . Произвести перенумерацию узлов

Используя схему Эйткена и узлы интерполирования равные значениям аргумента в таблице с номерами , , составить таблицу значений соответствующих многочленов порядка не выше первого, третьего, шестого на отрезке с шагом . Протабулировать заданную функцию на том же отрезке с шагом *h1*. Вывести на печать полученные таблицы значений.

Начертить и напечатать графики функции и интерполяционных многочленов, используя полученные значения в таблицах. На каждом рисунке привести график одного интерполяционного многочлена и график заданной функции.

**Код программы**

import math

import matplotlib

import numpy as np

from prettytable import PrettyTable

import matplotlib.pyplot as plt

def renumber():

global x\_ren\_L1, y\_ren\_L1, x\_ren\_L3, y\_ren\_L3, x\_ren\_L6, y\_ren\_L6

x\_ren\_L1 = []

y\_ren\_L1 = []

x\_ren\_L3 = []

y\_ren\_L3 = []

x\_ren\_L6 = []

y\_ren\_L6 = []

for i in range(2):

x\_ren\_L1.append(x[i+k])

y\_ren\_L1.append(y[i+k])

for i in range(4):

x\_ren\_L3.append(x[i+k])

y\_ren\_L3.append(y[i+k])

for i in range(7):

x\_ren\_L6.append(x[i+k])

y\_ren\_L6.append(y[i+k])

print("\nТаблица перенумерованных узлов (k=2)")

print("x(k):[", end="")

for i in range(len(x\_ren\_L1)):

print('{:.2f}'.format(x\_ren\_L1[i]), end="")

if i != len(x\_ren\_L1)-1:

print(', ', end="")

print(']')

print("y(k):[", end="")

for i in range(len(y\_ren\_L1)):

print('{:.2f}'.format(y\_ren\_L1[i]), end="")

if i != len(y\_ren\_L1)-1:

print(', ', end="")

print(']')

print()

print("x(k):[", end="")

for i in range(len(x\_ren\_L3)):

print('{:.2f}'.format(x\_ren\_L3[i]), end="")

if i != len(x\_ren\_L3)-1:

print(', ', end="")

print(']')

print("y(k):[", end="")

for i in range(len(y\_ren\_L3)):

print('{:.2f}'.format(y\_ren\_L3[i]), end="")

if i != len(y\_ren\_L3)-1:

print(', ', end="")

print(']')

print()

print("x(k):[", end="")

for i in range(len(x\_ren\_L6)):

print('{:.2f}'.format(x\_ren\_L6[i]), end="")

if i != len(x\_ren\_L6)-1:

print(', ', end="")

print(']')

print("y(k):[", end="")

for i in range(len(y\_ren\_L6)):

print('{:.2f}'.format(y\_ren\_L6[i]), end="")

if i != len(y\_ren\_L6)-1:

print(', ', end="")

print(']')

print()

return 0

def Aitkens\_scheme():

x\_new = np.around(list(np.linspace(1, 3, 51)), 2)

y\_new = []

for i in range(len(x\_new)):

y\_new.append(np.around((math.sqrt(x\_new[i]\*x\_new[i])\*(1/math.exp(x\_new[i]))), 3))

l1 = []

l3 = []

l6 = []

count = 0

for q in range(len(x\_new)):

l = y\_ren\_L1.copy()

for i in range(1, len(x\_ren\_L1)):

for j in range(i, len(x\_ren\_L1)):

matrix\_help = np.matrix(

[[l[i-1], (x\_ren\_L1[i-1] - x\_new[q])], [l[j], (x\_ren\_L1[j] - x\_new[q])]])

l[j] = np.around(

(np.linalg.det(matrix\_help)/(x\_ren\_L1[j] - x\_ren\_L1[i-1])), 3)

l1.append(np.around((l[j]), 3))

for q in range(len(x\_new)):

l = y\_ren\_L3.copy()

for i in range(1, len(x\_ren\_L3)):

for j in range(i, len(x\_ren\_L3)):

count = count+1

matrix\_help = np.matrix(

[[l[j-1], (x\_ren\_L3[count-1] - x\_new[q])], [l[j], (x\_ren\_L3[j] - x\_new[q])]])

l[j-1] = np.around(

(np.linalg.det(matrix\_help)/(x\_ren\_L3[j] - x\_ren\_L3[count-1])), 3)

count = 0

l\_help = l.copy()

for t in range(i, len(x\_ren\_L3)):

l[t] = l\_help[t-1]

l3.append(np.around((l[j]), 3))

for q in range(len(x\_new)):

l = y\_ren\_L6.copy()

for i in range(1, len(x\_ren\_L6)):

for j in range(i, len(x\_ren\_L6)):

count = count+1

matrix\_help = np.matrix(

[[l[j-1], (x\_ren\_L6[count-1] - x\_new[q])], [l[j], (x\_ren\_L6[j] - x\_new[q])]])

l[j-1] = np.around(

(np.linalg.det(matrix\_help)/(x\_ren\_L6[j] - x\_ren\_L6[count-1])), 3)

count = 0

l\_help = l.copy()

for t in range(i, len(x\_ren\_L6)):

l[t] = l\_help[t-1]

l6.append(np.around((l[j]), 3))

table = PrettyTable()

table.field\_names = ['x', 'y(x)', 'L1(x)', 'L3(x)', 'L6(x)']

for i in range(len(x\_new)):

table.add\_row([x\_new[i], y\_new[i], l1[i], l3[i], l6[i]])

print(table)

return 0

k = 2

print("\nТаблица узлов интерполирования")

x = np.around(list(np.linspace(1, 3, 13)), 2)

print("x(k):[", end="")

for i in range(len(x)):

print('{:.2f}'.format(x[i]), end="")

if i != len(x)-1:

print(', ', end="")

print(']')

y = []

for i in range(len(x)):

y.append(np.around((math.sqrt(x[i]\*x[i])\*(1/math.exp(x[i]))), 2))

print("y(k):[", end="")

for i in range(len(y)):

print('{:.2f}'.format(y[i]), end="")

if i != len(y)-1:

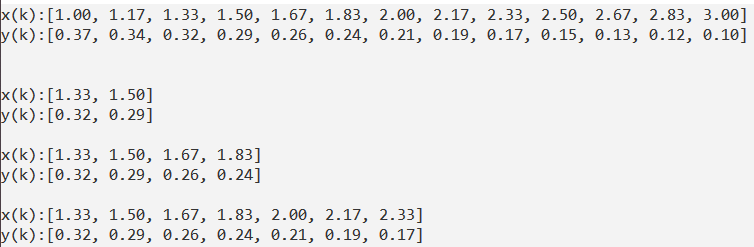
print(', ', end="")

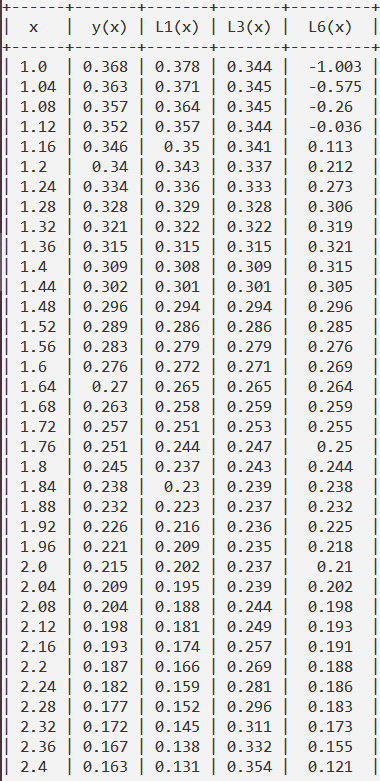
print(']')

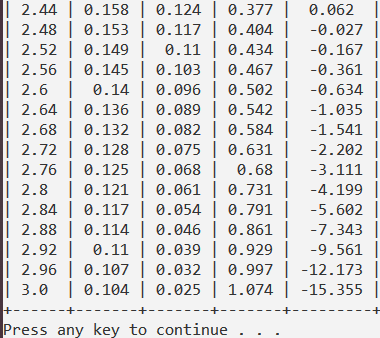
renumber()

Aitkens\_scheme()

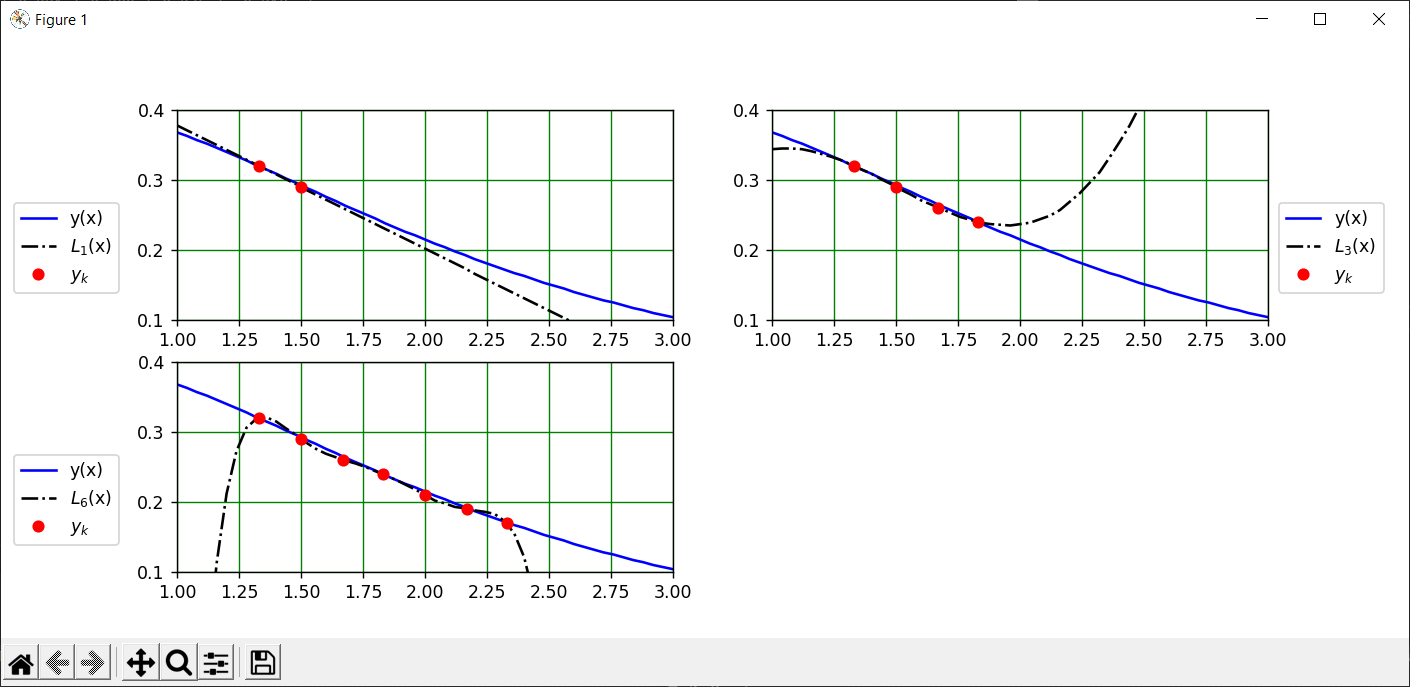
**Результат работы программы**







*Графики функции и интерполяционных многочленов*



**Вывод:**

В данной работе мы увидели, как с помощью схемы Эйткена вычислять значение интерполяционного многочлена Лагранжа, а также рассмотрели графики данного интерполяционного многочлена разной степени.