**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

Факультет физико-математических и естественных наук

ОТЧЕТ

по лабораторной работе № 2

дисциплина: Вычислительные методы

Студент:

Группа:

**МОСКВА**

2022 г.

**Цел работу:** чтобыузнать об интерполяционном методе Ларанжа, научиться использовать метод и программировать алгоритм на любом языке программирования.

**Мой вариант:**

****

**Ход работы:**

f(x)=a0+a1(x−x0)+a2(x−x0)(x−x1)+⋯+an(x−x0)(x−x1)…(x−xn)

f(x)=∑i=0naini(x)

1. мы начинаем с импорта необходимых библиотек в наш код и установки точек данных для x, y

import matplotlib.pyplot as mplt

import numpy as np

mplt.style.use('bmh');

# for jupiter

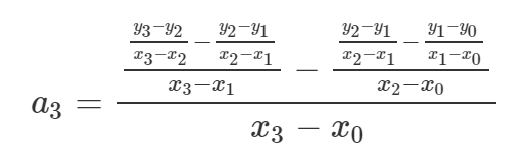
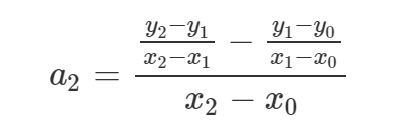
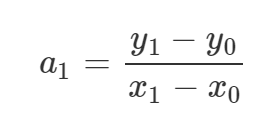
# %matplotlib inline

# x, y data points

x = np.array([0, 0.99, 2]);

y = np.array([-1, 0, 0.434]);

2. затем мы определяем функцию для разделенных таблиц различий (функция печати в цикле for не нужна, используя matplotlib)



*def* divided\_differences\_table(*x*, *y*):

    # divided differences table

    n = len(*y*);

    coeff = np.zeros([n,n]) # returns new array given zeros

    # y column 1st

    coeff[:,0] = *y* # setting y to first column

    # for loop to set divided diff table

    for j in range(1, n):

        for i in range(n-j):

            coeff[i][j] = (coeff[i+1][j-1] - coeff[i][j-1]) / (*x*[i+j] - *x*[i])

        print(*f*"{j}:{coeff[i][j]}");

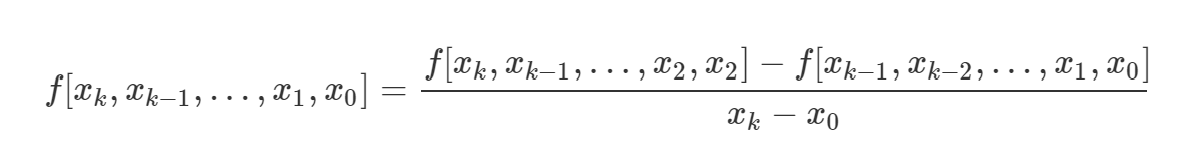
    return coeff

3. определить функцию для вычисления многочлена Ньютона. который принимает 3 переменные

1. это наш коэффициент

2. это данные x

3. это сам x



*def* evaluate\_newton\_poly(*coeff*, *xdata*, *x*):

    # evaluate newton polynomial

    n = len(*xdata*) - 1

    p = *coeff*[n]

    # main loop

    for i in range(1, n+1):

        p = *coeff*[n-i] + (*x* - *xdata*[n-i])\*p

    print(*f*"at {*x*}: evaluated: {p}");

    return p

4. наконец, мы используем наши функции для оценки нашего метода, затем настраиваем matplotlib, чтобы увидеть наш результат интерполяционного полинома ньютона

# get diveded difference coefficient

get\_div\_diff = divided\_differences\_table(x,y)[0,:];

# evaluate new data points

newx = np.arange(x[0], x[-1]+0.1, 0.1) # (start=x[0], stop=x[-1]+0.1 or as we wish, scale=(as we please)pref=0.1);

newy = evaluate\_newton\_poly(get\_div\_diff, x, newx);

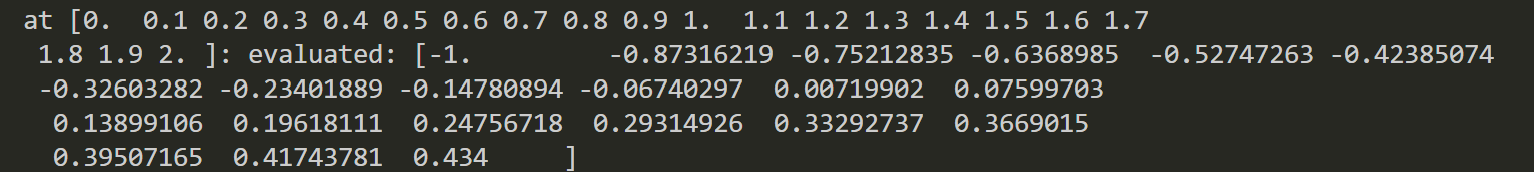
mplt.figure(*figsize*=(8, 6), *dpi*=80) # 80x80

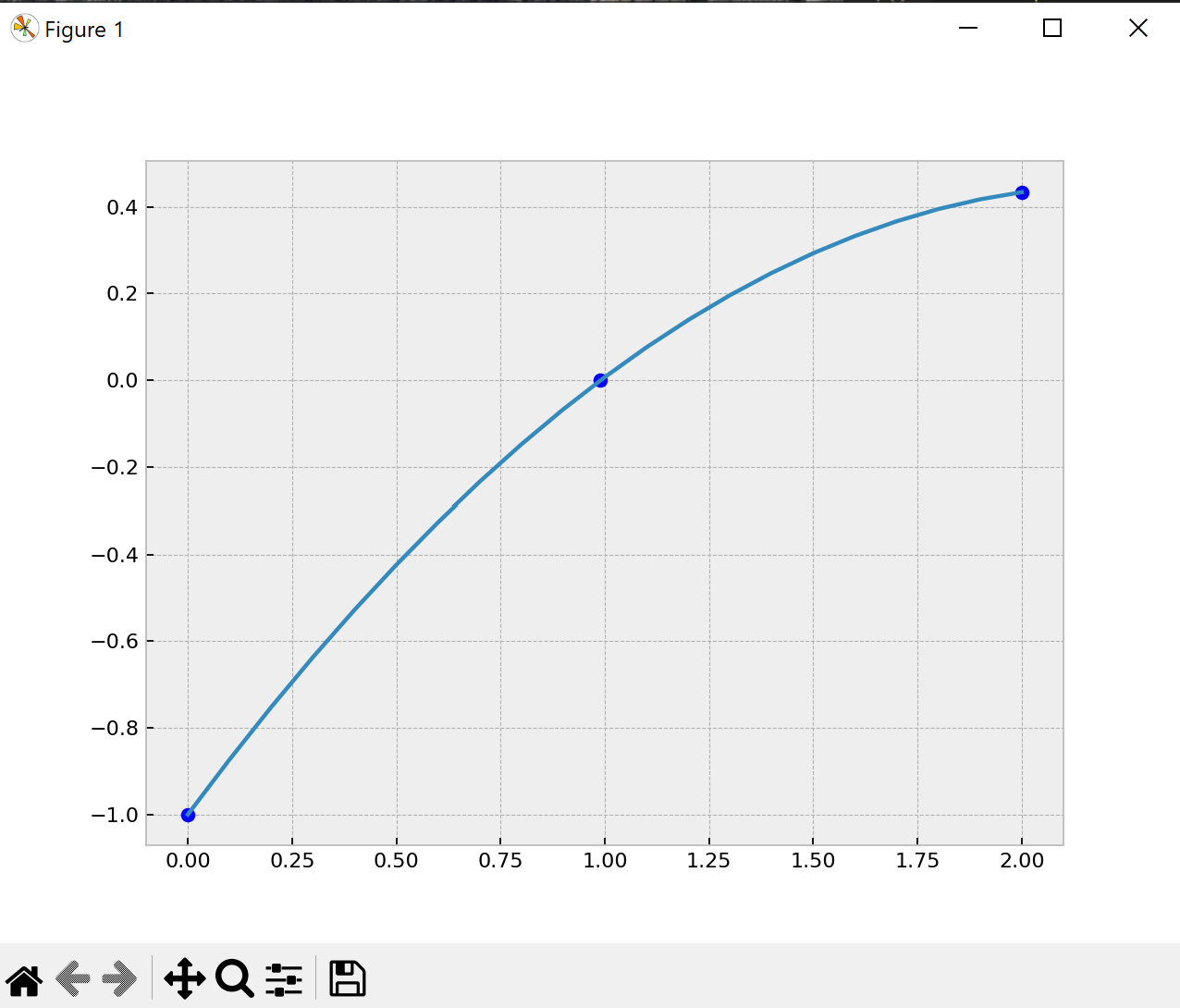
mplt.plot(x, y, 'bo')

mplt.plot(newx, newy)

mplt.show()

**Вывод:**

****

****

**КОД:**

import matplotlib.pyplot as mplt

import numpy as np

mplt.style.use('bmh');

# for jupiter

# %matplotlib inline

# x, y data points

x = np.array([0, 0.99, 2]);

y = np.array([-1, 0, 0.434]);

*def* divided\_differences\_table(*x*, *y*):

    # divided differences table

    n = len(*y*);

    coeff = np.zeros([n,n]) # returns new array given zeros

    # y column 1st

    coeff[:,0] = *y* # setting y to first column

    # for loop to set divided diff table

    for j in range(1, n):

        for i in range(n-j):

            coeff[i][j] = (coeff[i+1][j-1] - coeff[i][j-1]) / (*x*[i+j] - *x*[i])

        print(*f*"{j}:{coeff[i][j]}");

    return coeff

*def* evaluate\_newton\_poly(*coeff*, *xdata*, *x*):

    # evaluate newton polynomial

    n = len(*xdata*) - 1

    p = *coeff*[n]

    # main loop

    for i in range(1, n+1):

        p = *coeff*[n-i] + (*x* - *xdata*[n-i])\*p

    print(*f*"at {*x*}: evaluated: {p}");

    return p

# get diveded difference coefficient

get\_div\_diff = divided\_differences\_table(x,y)[0,:];

# evaluate new data points

newx = np.arange(x[0], x[-1]+0.1, 0.1) # (start=x[0], stop=x[-1]+0.1 or as we wish, scale=(as we please)pref=0.1);

newy = evaluate\_newton\_poly(get\_div\_diff, x, newx);

mplt.figure(*figsize*=(8, 6), *dpi*=80) # 80x80

mplt.plot(x, y, 'bo')

mplt.plot(newx, newy)

mplt.show()