Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут”

Кафедра прикладної математики

Лабораторна робота №1

З дисципліни “Методи обчислення”

Виконав:

студент групи КМ-81

Верзун П.В.

Київ – 2020

# Завдання 1

Користуючись інтерполяційною схемою Ейткена, обчислити з заданою точністю значення даної функції в заданій точці x0.

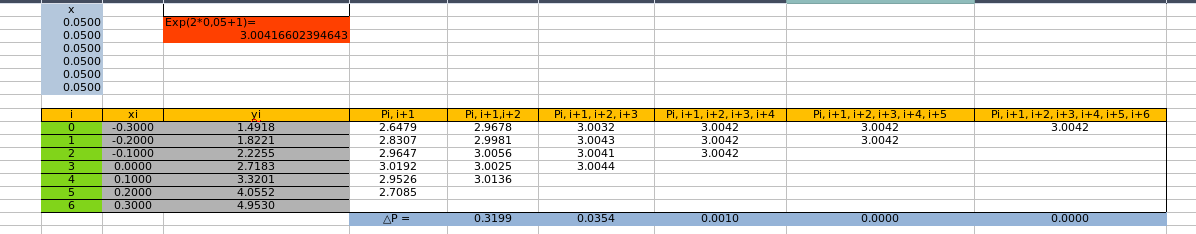
Функція:

Точність: 0.002

Вузли інтерполяції: [-0.3, -0.2, -0.1, 0, 0.1, 0.2, 0.3].

Шукаємо значення функції в точці x = 0.05

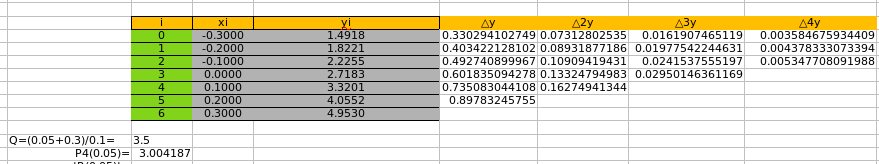
Для обрахунку були використані наступні формули:



Так як при обчисленні Pi, i+1, i+2, i+3, i+4, i+5, i+6 досягнута точність 0.001, яка менше тієї, що вказана в умові, вважаємо, що значення функції у вибраній точці приблизно дорівнює 3.0042. Дійсно, при перевірці калькулятором, значення функції в цій точці 3.0041, що співпадає з знайденим з заданою точністю (з точністю 0.0001).

# Завдання 2

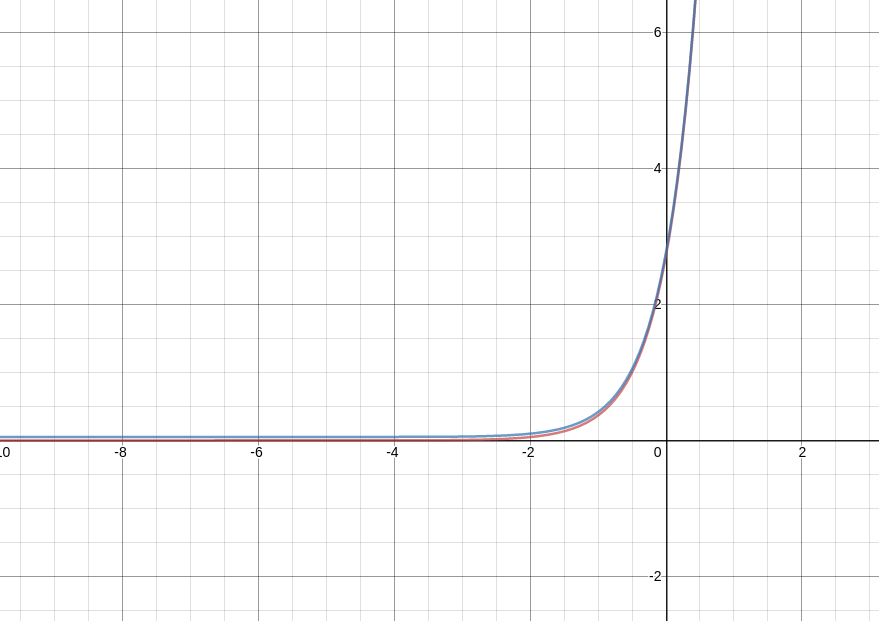
Користуючись інтерполяційною формулою Лагранжа або Ньютона, знайти значення функції для заданого аргумента. Оцінити похибку отриманого результату. Зобразити графіки Заданої функції та інтерполянти.



Для обрахунку було використано першу інтерполяційну формулу Ньютона для рівновіддалених вузлів з кроком h:

,

де .

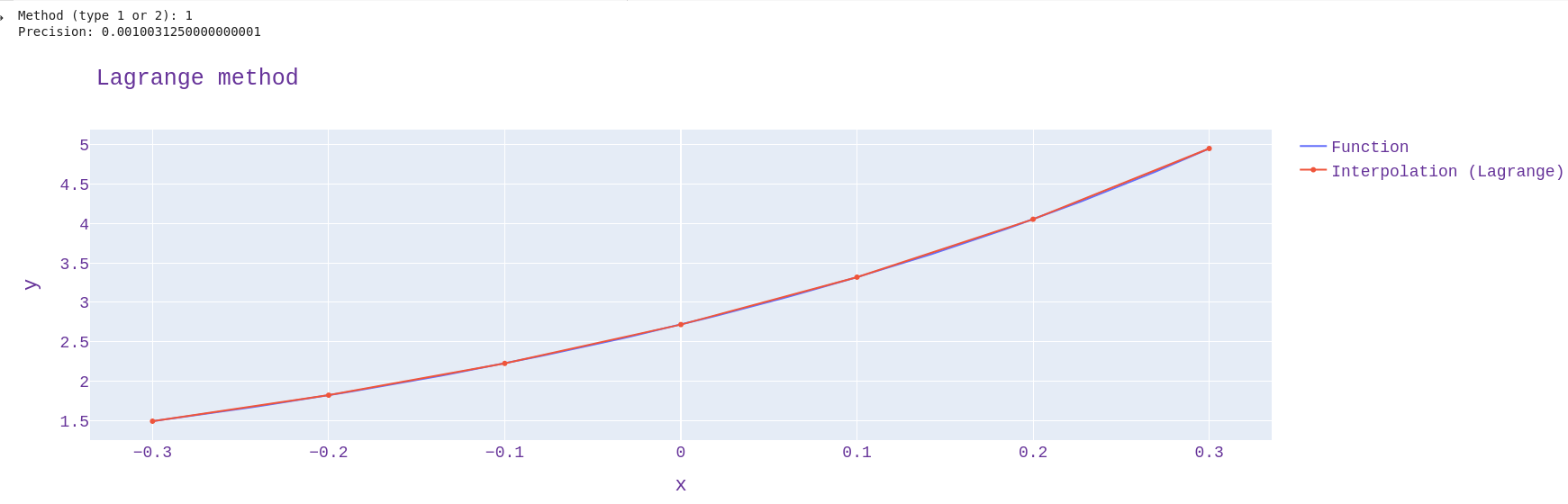


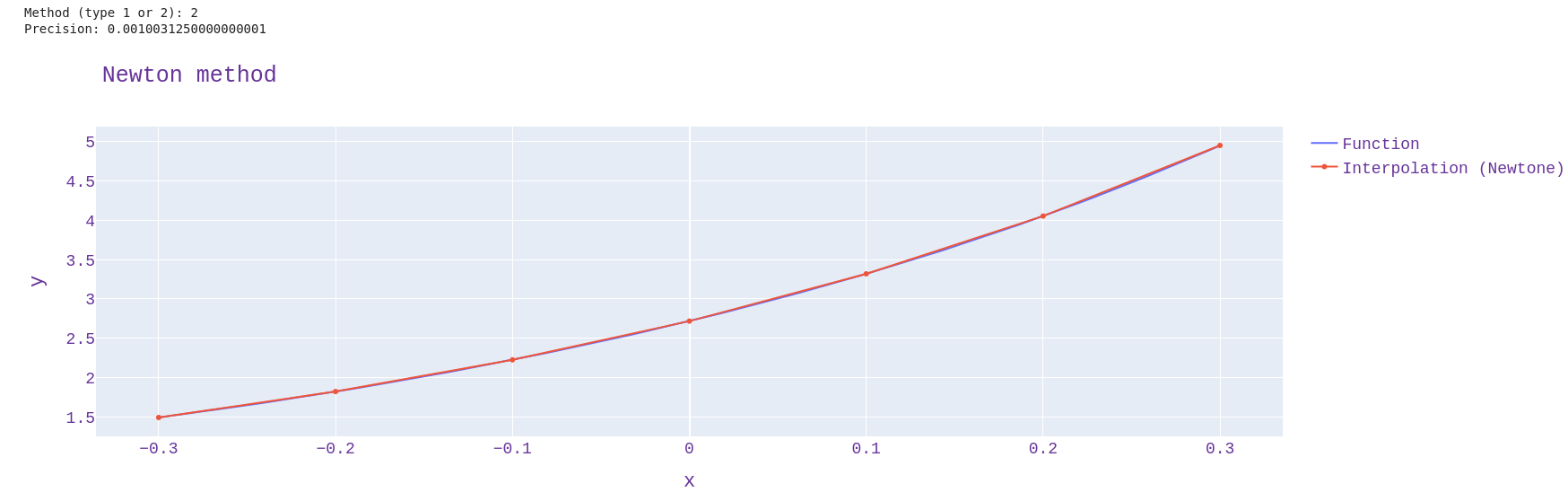
Як видно, на потрібному проміжку функція і інтерполянта накладаються одна на одну. Також відмічена точка з координатами x, який ми досліджуємо, та значенням функції у цій точці.

**Завдання 3**

Зробити програмне забезпечення задачі інтерполяції функції Лагранжа та Ньютона.

Приклад роботи програми:





Код програми:

import plotly.graph\_objs as go

import numpy as np

import math

import mpmath

import functools as ft

def Lagrange(x,y,t):

z=0

for j in range(len(y)):

p1=1; p2=1

for i in range(len(x)):

if i==j:

p1=p1\*1; p2=p2\*1

else:

p1=p1\*(t-x[i])

p2=p2\*(x[j]-x[i])

z=z+y[j]\*p1/p2

return z

def get\_max\_of\_error(x\_points, max\_of\_abs\_n\_diff):

curr\_point = x\_points[0]

points\_number = 100

step = (x\_points[-1] - x\_points[0])/points\_number

max\_error = -math.inf

while curr\_point <= x\_points[-1]:

differences = list(map(lambda x: abs(curr\_point - x), x\_points))

mult\_differences = ft.reduce(lambda x, y: x/2 + y/2, differences)

max\_error = max(max\_error, max\_of\_abs\_n\_diff \*

mult\_differences)

curr\_point += step

return max\_error

class Newton(object):

def \_\_init\_\_(self, x, y):

self.x = x

self.y = y

def newton(self, tmp\_x):

table = np.zeros([len(self.x), len(self.x) + 1], dtype=float)

for i in range(len(self.x)):

table[i][0] = self.x[i]

table[i][1] = self.y[i]

for i in range(2, table.shape[1]):

for j in range(i - 1, table.shape[0]):

table[j][i] = (table[j][i - 1] - table[j - 1][i - 1]) / (self.x[j] - self.x[j - i + 1])

tmp\_y = []

for ans\_x in tmp\_x:

ans\_y = 0

for i in range(table.shape[0]):

tmp = table[i][i + 1]

for j in range(i):

tmp \*= (ans\_x - self.x[j])

ans\_y += tmp

tmp\_y.append(ans\_y)

return tmp\_y

def plot\_interpolation():

answer = input('Method (type 1 or 2): ')

x=np.array([-0.3, -0.2, -0.1, 0, 0.1, 0.2, 0.3], dtype=float)

y=np.array([1.4918, 1.8221, 2.2255, 2.7183, 3.3201, 4.0552, 4.9530], dtype=float)

if answer == '1':

xnew=np.linspace(np.min(x),np.max(x),100)

ynew=[Lagrange(x,y,i) for i in xnew]

print(f"Precision: {get\_max\_of\_error(x, 0.002)}")

fig = go.Figure()

fig.add\_trace(go.Scatter(x=xnew, y=ynew, name='Function'))

fig.add\_trace(go.Scatter(x=x, y=y, name='Interpolation (Lagrange)'))

fig.update\_layout(

title="Lagrange method",

xaxis\_title="x",

yaxis\_title="y",

font=dict(

family="Courier New, monospace",

size=18,

color="RebeccaPurple"

)

)

fig.show()

elif answer == '2':

s = Newton(x, y)

xnew=np.linspace(np.min(x),np.max(x),100)

ynew=s.newton(xnew)

print(f"Precision: {get\_max\_of\_error(x, 0.002)}")

fig = go.Figure()

fig.add\_trace(go.Scatter(x=xnew, y=ynew, name='Function'))

fig.add\_trace(go.Scatter(x=x, y=y, name='Interpolation (Newtone)'))

fig.update\_layout(

title="Newton method",

xaxis\_title="x",

yaxis\_title="y",

font=dict(

family="Courier New, monospace",

size=18,

color="RebeccaPurple"

)

)

fig.show()

plot\_interpolation()