**Міністерство освіти і науки України**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”**

**Кафедра прикладної математики**

РОЗРАХУНКОВО-ГРАФІЧНА РОБОТА

з кредитного модуля

«Програмування 1. Основи програмування»

на тему:

«Програма чисельного диференціювання (Обчислення першої похідної)»

Виконав: Чорний Дмитро Станіславович

група КМ-02 Факультет прикладної математики

N залікової книжки КМ-0224

Керівник: Олефір О.С. ( )

«\_\_\_» \_\_\_\_\_\_\_\_\_ 2020р.

Захищена з оцінкою\_\_\_\_\_\_\_\_\_

ЗМІСТ

[1.](#_heading=h.30j0zll) ПОСТАНОВКА ЗАДАЧІ 3

[1.1](#_heading=h.1fob9te) Дослідження проблеми 3

[1.2](#_heading=h.3znysh7) Формулювання завдання 4

[2.](#_heading=h.2et92p0) ПРОЕКТУВАННЯ ПРОГРАМНИХ ЗАСОБІВ 5

[2.1](#_heading=h.tyjcwt) Методи вирішення задач 5

[2.2](#_heading=h.3dy6vkm) Проектування алгоритмів 6

[2.3](#_heading=h.1t3h5sf) Структура програмного забезпечення 8

[2.4](#_heading=h.4d34og8) Опис розроблених алгоритмів 9

[2.5](#_heading=h.2s8eyo1) Засоби керування програмами 10

[3.](#_heading=h.17dp8vu) РЕАЛІЗАЦІЯ ПРОГРАМИ 11

[3.3](#_heading=h.3rdcrjn) Опис вхідних даних 11

[3.4](#_heading=h.26in1rg) Опис результатів 12

[3.5](#_heading=h.lnxbz9) Опис контрольних прикладів 12

[3.6](#_heading=h.35nkun2) Експериментальні розрахунки 14

[ВИСНОВКИ 15](#_heading=h.1ksv4uv)

[ЛІТЕРАТУРА 16](#_heading=h.44sinio)

[ДОДАТОК А 17](#_heading=h.2jxsxqh)

[ДОДАТОК Б 25](#_heading=h.z337ya)

1. ПОСТАНОВКА ЗАДАЧІ
   1. Дослідження проблеми

Поняття похідної виникло через велику кількість задач природничих наук і математики, які зводилися до обчислення границь одного й того ж типу. Найголовніші серед них — обчислення швидкості прямолінійного руху точки та побудова дотичної до графіка функції. Але, якщо не зважати на механічний та геометричний зміст, а виділити спільних метод їх розв'язку, до поняття похідної. **Похідною** функції  у точці **y = f(x)** у точці **х** називається границя (якщо ця границя існує) відношення приросту функції до приросту аргументу, що прямує до нуля так що:

Формула 1.1

Проте зі збільшенням задач, які покладалися на технічних спеціалістів, з часом виникало все більше потреби обчислювати значення похідної в конкретній точці, тому і виникла необхідність обчислювати необхідні величини швидко, та з необхідною точністю. Це спонукало математиків та інженерів розробляти та впроваджувати нові методи розв’язання задач, а кібернетиків та програмістів – імплементувати актуальні рішення для виконання рутинних задач комп’ютером, або іншою обчислювальною технікою. Іншим викликом для математиків та статистиків став той факт, що на практиці рідко хто має справу з чітко визначеною функцією, натомість, існує певна кількість дискретних величин – значень деякої функцій у конкретних точках, які і потрібно дослідити. Тому для обрахування похідної в точці від таблично заданої функції було необхідно цю функцію інтерполювати, тобто розкласти в многочлен. Проте майже будь-яка апроксимація за визначенням дає лише приблизний образ прототипованої функції, а отже для оперування результатом розрахунків необхідно і бути усвідомленим щодо помилки, яку може вміщувати у собі формула. Актуальним це залишається і для машинного обчислення чисельного значення похідної, бо комп’ютер, оперуючи з числами, залежить від представлення чисел у пам’яті, а отже теж може генерувати похибку.

* 1. Формулювання завдання

Отже, мету цього дослідження можна сформулювати наступним чином:

Розробити програму чисельного диференціювання, яка буде обраховувати значення першої похідної у точці від таблично заданої функції. Крім того, розширити функціонал цієї програми для аналітично заданої функції, організувати ввід інформації користувачем, та вивід обрахованої інформації на екран. Обрахунок та вивід помилки обчислень. Вивід інформаційного повідомлення щодо формату даних. Збереження таблично заданої функції на час роботи програми задля можливості багатократно проводити обрахунки без необхідності повторювати ввід таблиці кожного разу.

Крім того, відповідно до вимог навчальної програми, алгоритм потрібно розробити за допомогою мови програмування Python, а також необхідно розробити макет можливого інтерфейсу для роботи програми та підготувати пояснювальну записку щодо етапів розробки програми.

1. ПРОЕКТУВАННЯ ПРОГРАМНИХ ЗАСОБІВ
2. Методи вирішення задач

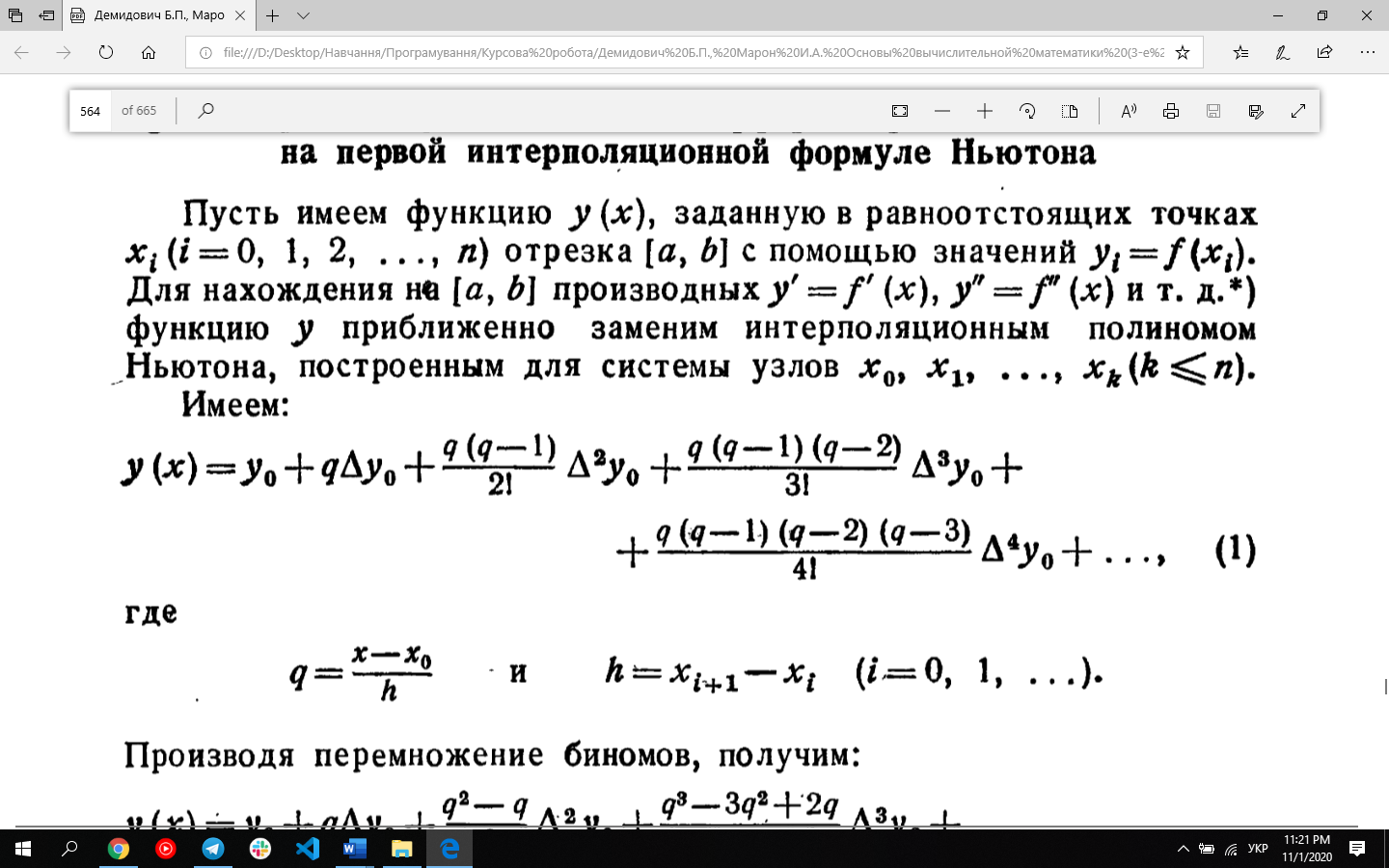
Відповідно до поставленої мети завдання можна розділити на декілька підзадач:

1. Інтерполяція функції
2. Обрахування значення похідної
3. Обрахування помилки

Серед можливих варіантів інтерполяції функції[2], було обрано обрахувати приблизне значення функції за допомогою поліному Ньютона(Формула 2.1), через його наступні переваги:

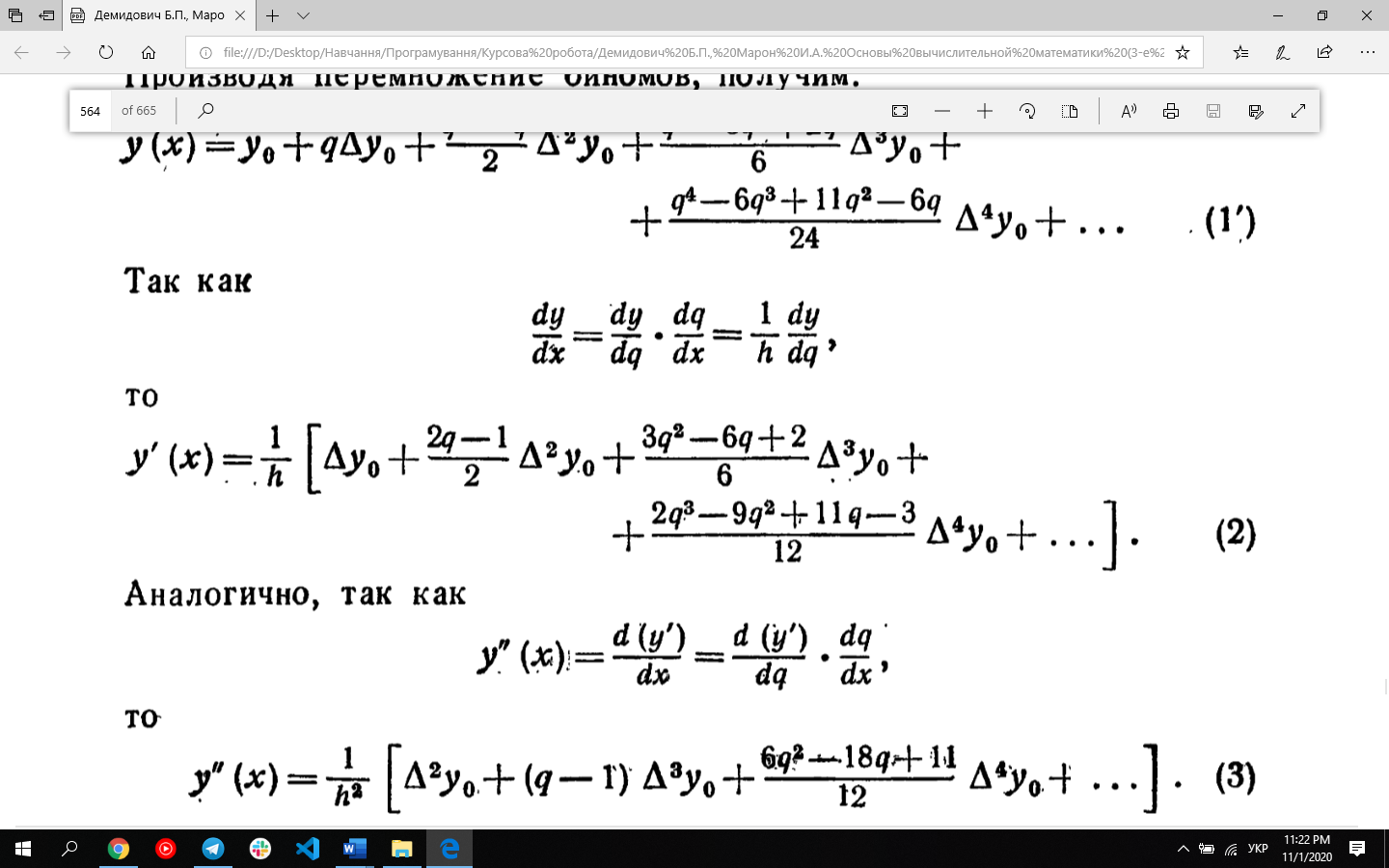
1. Зі збільшенням точок інтерполяції точність зростає
2. Відсутня необхідність перераховувати поліном при додаванні кожної точки, як наприклад при побудові поліному Лагранжа
3. Надійність методу, та доступність його опису в джерелах

Проте під час розрахунків можливих шляхів вирішення другої підзадачі, було прийняте важливе структурне вирішення – об’єднати першу та другу підзадачу, та використати формулу, яка сама по собі є похідною від формули поліному Ньютона.[1]



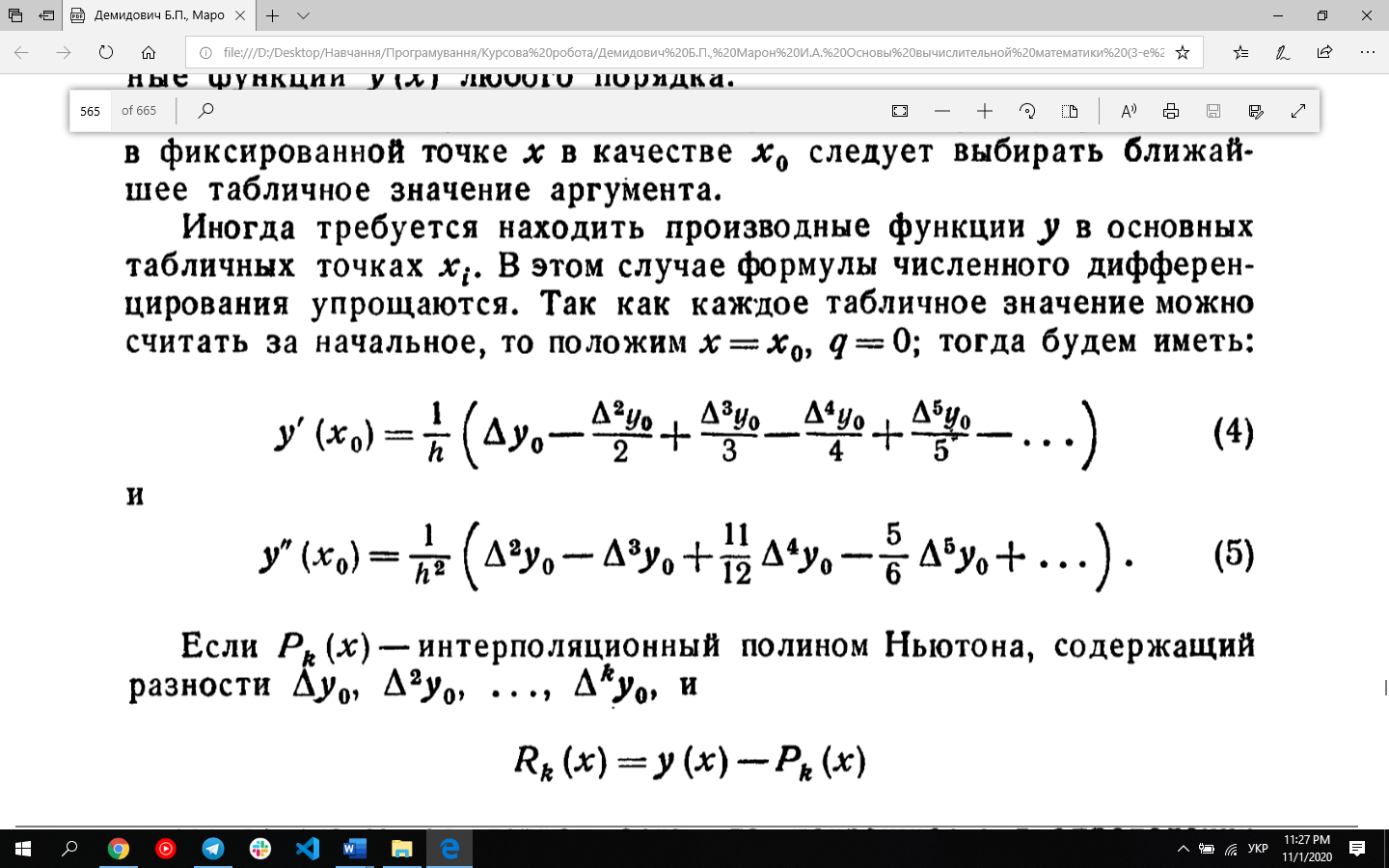
Формула 2.1 – Формула поліному Ньютона

Виведення формули 2.2 наведено на сторінці 563 підручника Демидовича Б.П. та Марона И.А. «Основы вычислительной математики».



Формула 2.2

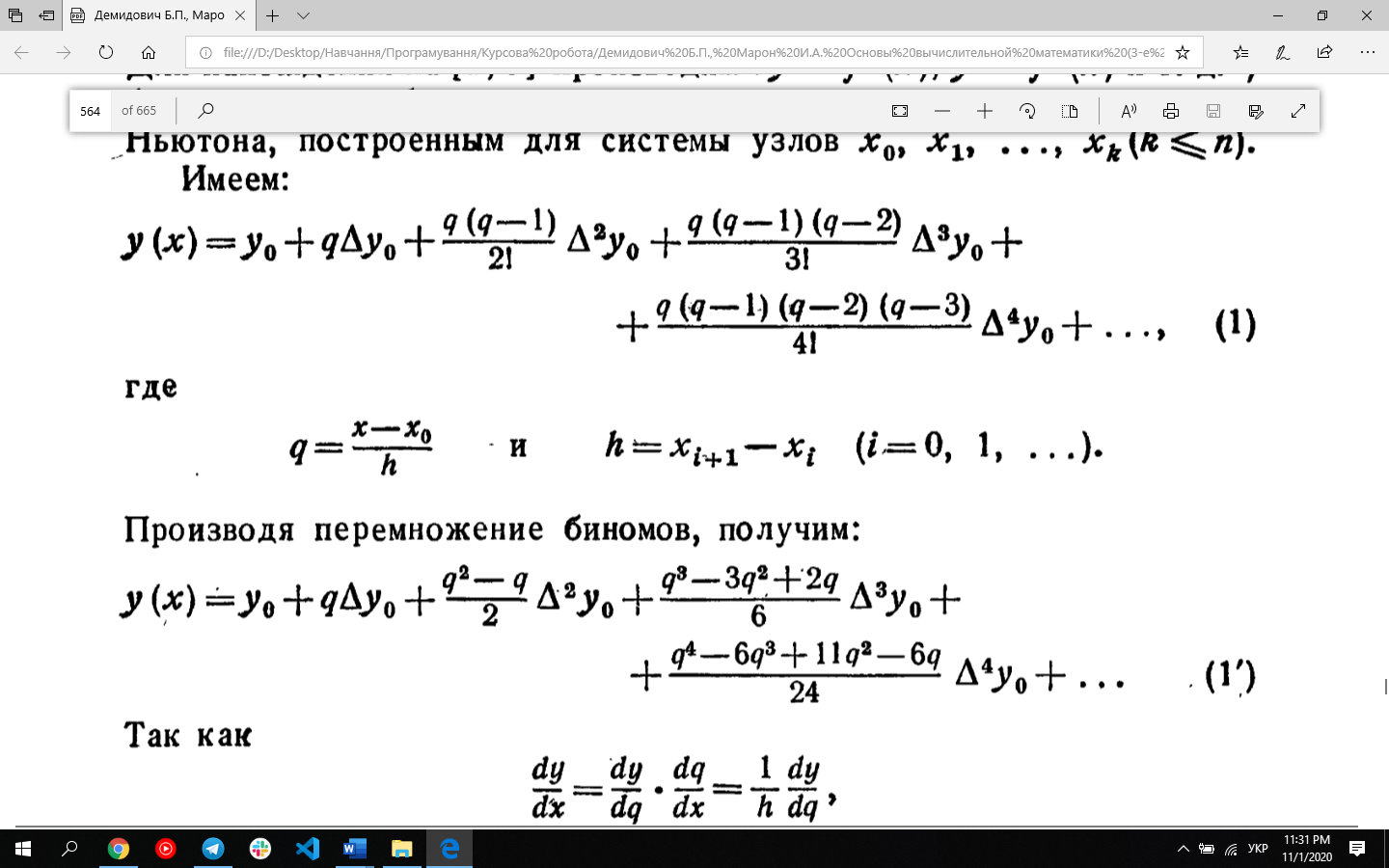
Змінена елементарними перетвореннями формула 2.2 і була використана як основна для розрахунку чисельного значення похідної в точці. Проте за умови того, що точка диференціювання – це одна із таблично заданих точок доречно використати спрощену формулу 2.3.



Формула 2.3

Варто зауважити, що формула працює для рівновіддалених точок, тому серед вимог до введених даних є вимога до рівновіддаленості точок по осі х.

Для кращого розуміння формул 2.1, 2.2, 2.3 варто зауважити:

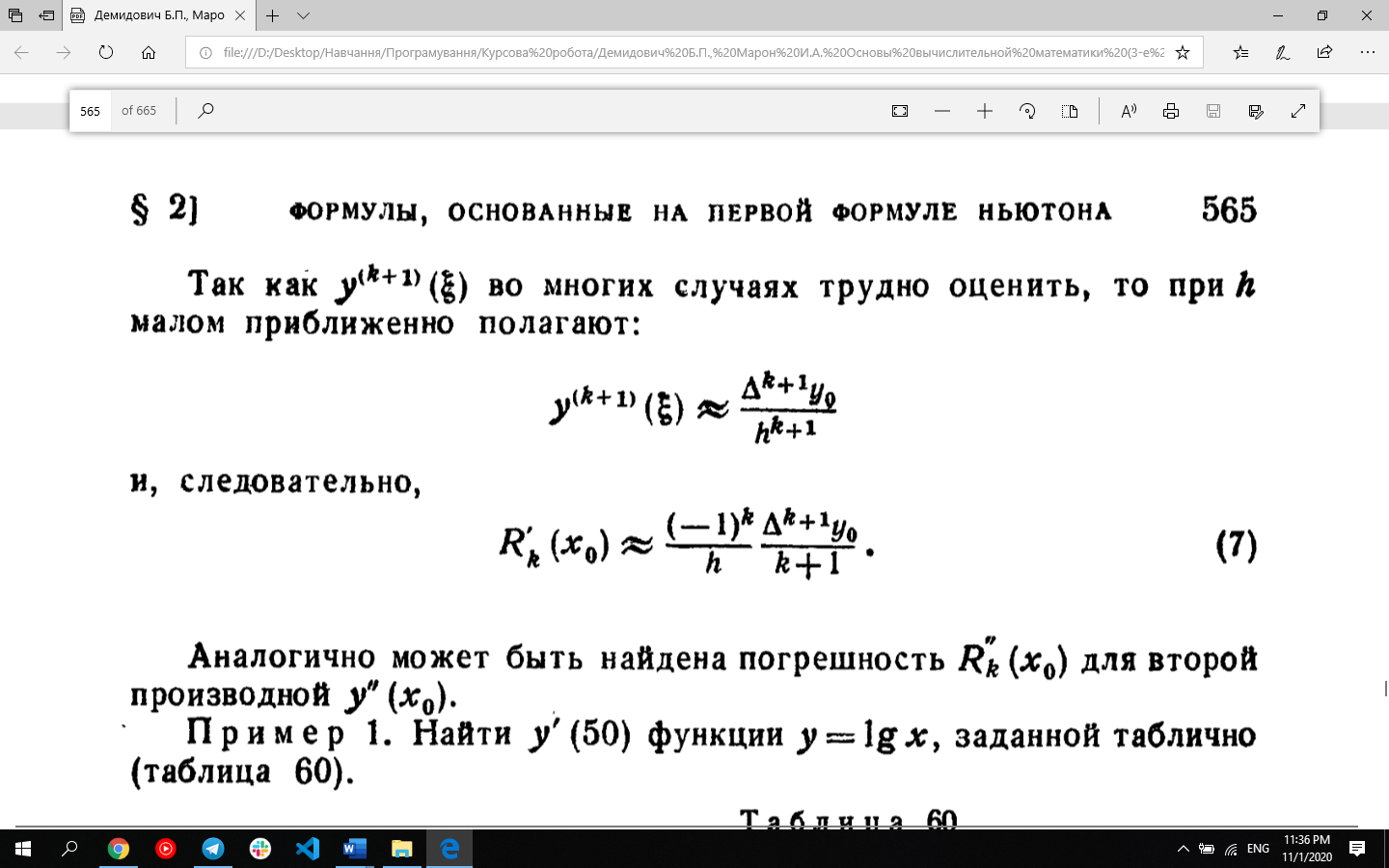


Формула 2.4 та 2.5 відповідно.

Де, x0 – найближче табличне значення до точки диференціювання,

а – скінченні різниці n-го порядку.

Формула 2.6 – для знаходження помилки, взята з того ж джерела[1].



Формула 2.6

1. Проектування алгоритмів

Для алгоритмізації процесу чисельного диференціювання було прийнято наступні архітектурні рішення:

1. Підпрограми для: Введення числа з перевіркою, Введення відповіді на бінарне питання «Так»/«Ні», та обрахування значення похідної з помилкою, було виведено в окремі функції.
2. За допомогою циклу програма має виконуватися доти, доки користувач не вирішить зупинити роботу з програмою.
3. Всередині основного циклу, реалізовані два окремих цикли, які являють собою дві підпрограми і ніколи не виконуються одночасно – перша для роботи з таблично заданими функціями, друга для роботи з аналітично заданими функціями.
4. На початку програми користувача зустрічає інформаційне повідомлення та повідомлення з вимогами до даних.
5. Структура програмного забезпечення

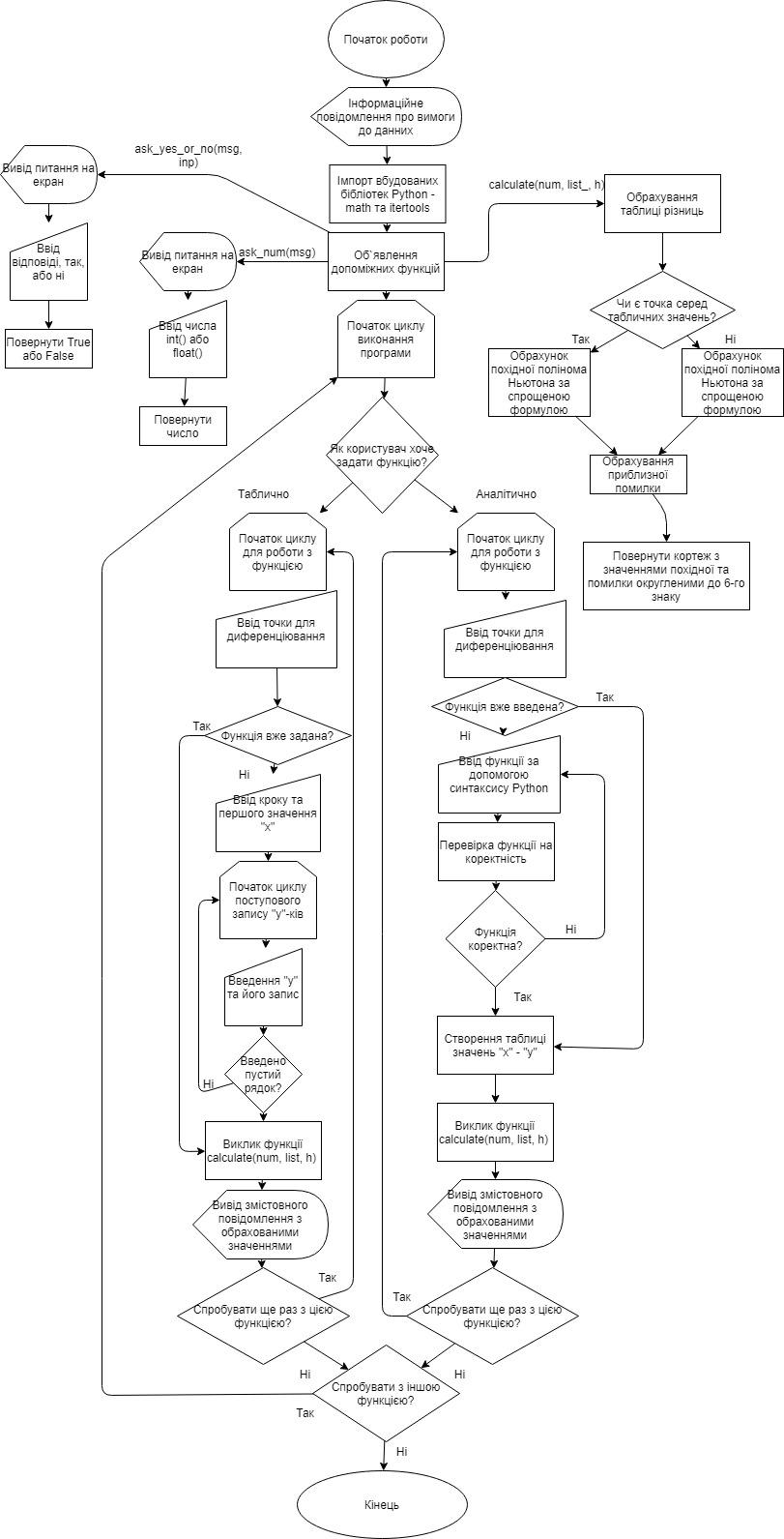


Рисунок 2.1 Блок-схема

1. Опис розроблених алгоритмів

Робота починається з інформаційного повідомлення щодо формату та вимог до даних. Далі визначаються три функції:

Функція ask\_yes\_or\_no(msg, inpmsg) яка приймає на вхід два рядки – перше та друге повідомлення. Далі функція запускає цикл який триває доти, доки користувач не введе коректне значення, попередньо питаючи користувача питання з відповідями «так» чи «ні».

Функція ask\_num(msg, type\_ = “float”) яка приймає на вхід один рядок, та аргумент «int» або «float». Функція в циклі перевіряє відповідність введеного числа до заданого типу. Цикл переривається коли число підходить до типу та повертає число.

Функція calculate(num, list\_, h) яка приймає на вхід точку диференціювання, список з парами «х» та «у» та різницю між значеннями «х» - h. Далі функція будує таблицю скінченних різниць, перевіряє чи точка диференціювання знаходиться в таблиці, якщо так, обчислює значення за спрощеною формулою, якщо ні, то обчислює значення за простою формулою. Далі функція обчислює значення помилки, та повертає пару з значень похідної та помилки відповідно.

Ініціюється цикл, імпортуються необхідні бібліотеки, користувач обирає як він хоче ввести функцію – таблично, чи аналітично. Якщо користувач вводить функцію таблично – він обирає перший «х» та крок, а далі по черзі вводить усі значення «у». Коли користувач вводить пустий рядок, ввід зупиняється та програма обраховує значення.

Якщо користувач бажає ввести функцію таблично, він вводить її за допомогою синтаксису мови Python. Функція перевіряється на коректність, та обчислюються 14 точок значення функції в околі точки диференціювання. Далі аналогічно з обчисленням табличної функції розраховується пара значень.

Обраховані значення виводяться на екран, так користувач висловлює своє бажання повторити свою роботу з цією ж функцією, або повторити програму, або не робити нічого з цих двох. У першому разі він просто вводить іншу точку диференціювання, у другому – точку та функцію, у третьому – програма зупиняє свою роботу.

1. Засоби керування програмами

Серед опцій користувача керування програмою є:

1. Введення таблиці значень
2. Введення аналітичної функції (У разі не розпізнавання функції бажаним є введення табличних значень)
3. Введення точки диференціювання
4. Можливість декілька разів повторити роботу з функцією, без повторного введення.

В той ж час можливий варіант користувацького інтерфейсу:

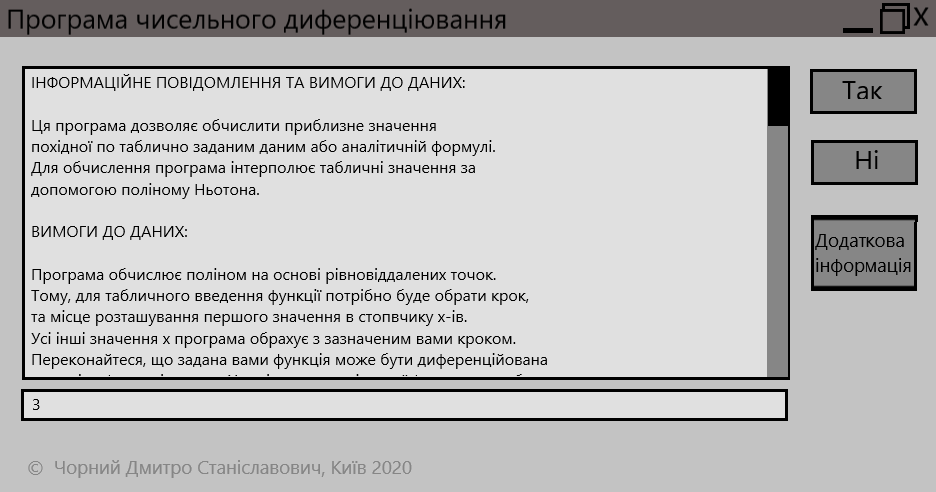


Рисунок 2.2

Або ж робота може виконуватися з командного рядка.

1. РЕАЛІЗАЦІЯ ПРОГРАМИ
2. Опис вхідних даних

Основна вимога до вхідних даних – таблиця рівновіддалених дискретних значень функції. Ввід реалізується наступним чином:

користувач спочатку вводить крок (нехай h = 0.1) потім вводить перший «x» (нехай x = 0) та поступово вводить значення усіх «у» (Нехай у = 0, 1, 2, 3, 4, 5). Тоді буде сформована така таблиця:

|  |  |
| --- | --- |
| х | у |
| 0 | 0 |
| 0.1 | 1 |
| 0.2 | 2 |
| 0.3 | 3 |
| 0.4 | 4 |
| 0.5 | 5 |

Таблиця 3.1

Також користувач вводить точку для диференціювання.

У разі вводу аналітичної функції користувач вводить точку диференціювання (наприклад 0.5) та функцію за допомогою синтаксису мови Python та модулю math. (Наприклад: math.sin(x)+x), тоді буде сформована така таблиця:

|  |  |
| --- | --- |
| х | у |
| -0.1 | -1.99 |
| 0 | 0 |
| 0.1 | 1.99 |
| 0.2 | 0.398 |
| 0.3 | 0.595 |
| 0.4 | 0.789 |
| 0.5 | 0.979 |
| 0.6 | 1.164 |
| 0.7 | 1.344 |
| 0.8 | 1.517 |
| 0.9 | 1.683 |
| 1 | 1.841 |
| 1.1 | 1.991 |
| 1.2 | 2.132 |

Таблиця 3.2

1. Опис результатів

Як результат розрахунків програма виводить змістовне повідомлення з обрахованими значеннями похідної та помилки:

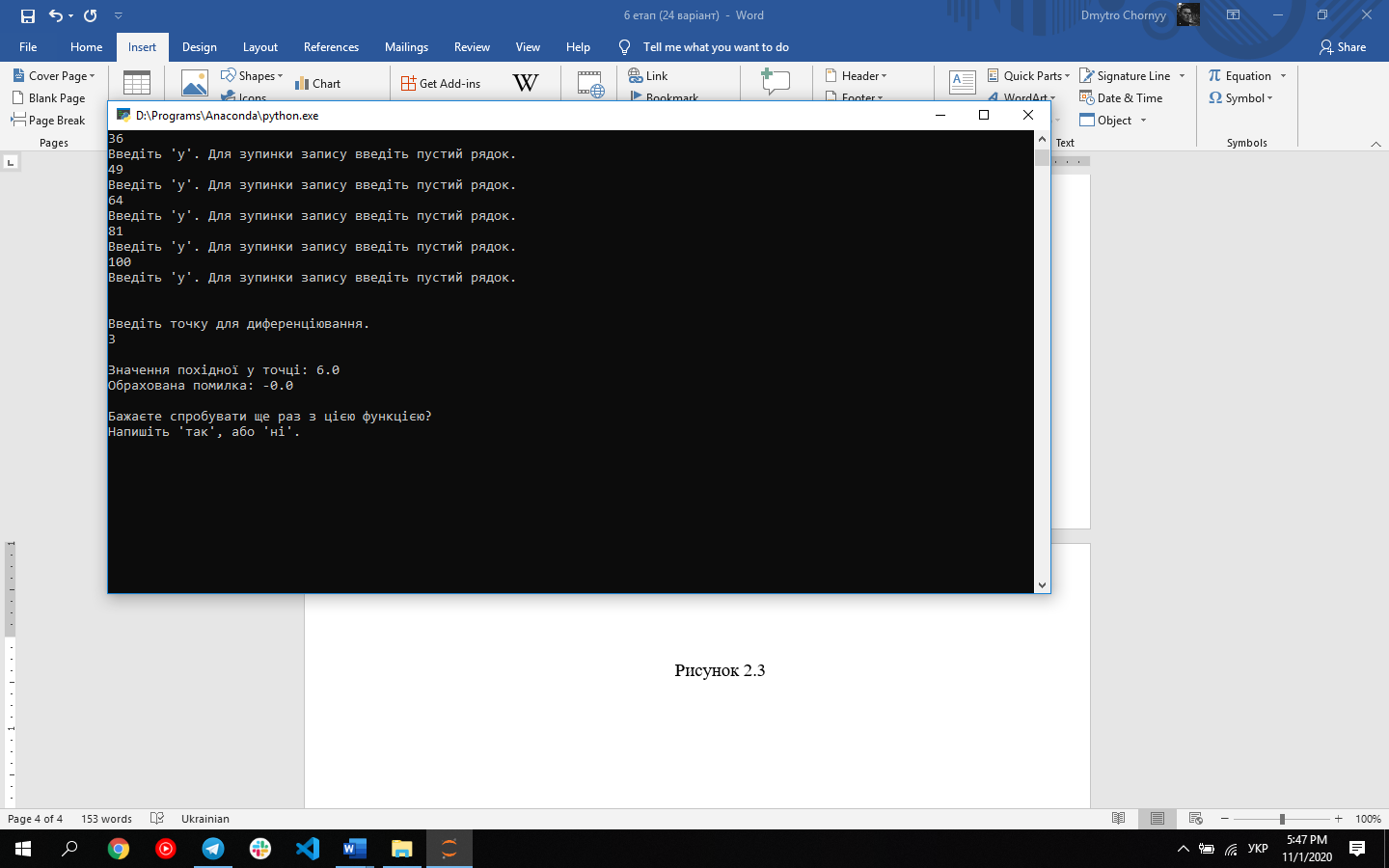


Рисунок 3.1

Також програма пропонує спробувати розрахунок ще раз.

1. Опис контрольних прикладів

Для ознайомлення з темою обрахунку похідних спочатку були проаналізовані методи розрахунку похідних від функцій за допомогою табличних значень похідних:

Приклад 1

Знайти похідну функції:

За властивості лінійності похідною, отримаємо:



Далі скористаємося таблицею похідних елементарних функцій:



Отже, отримуємо відповідь:



Приклад 2

Знайти похідну функції:

Похідна суми дорівнює сумі похідних і винесемо константу в другому доданку:



Далі скористаємося таблицею похідних елементарних функцій:



Отже, отримуємо відповідь:



Приклад 3

Знайти похідну функції:

За правилом диференціювання маємо:



Застосовуючи таблицю похідних для степеневої і логарифмічною функцій і перетворюючи отриманий вираз, отримаємо:





Отже, отримуємо відповідь:



Ці розрахунки допомогли в проектуванні програми, та перетворенні формули, наведеної вище, на формулу, більш придатну для обрахунку за допомогою алгоритму.

Приклади обрахунку похідних вже за допомогою інтерполювання методом Ньютона наведені в підручнику «Основы вычислительной математики» на сторінках 565-567.[1]

1. Експериментальні розрахунки

Скріншоти тестування програми, та порівняння з істинними значеннями наведені у Додатку Б.

ВИСНОВКИ

Отже, у результаті виконання розрахунково-графічної роботи була досліджена тема чисельного диференціювання (обчислення першої похідної). Далі, у відповідності до плану виконання розрахункової роботи, був детально вивчений метод вирішення поставленої задачі, обрахунок контрольних прикладів, спроектована архітектура майбутньої програми та можливий варіант користувацького інтерфейсу. Далі була розроблений сам алгоритм, написаний у формі програми мовою Python. Наприкінці був оформлений звіт та пояснювальна записка. Щодо кожного виконаного етапу відбувалося звітування у відповідності до зазначених дат.

Було набуто навички проектування, розробки та тестування програмного забезпечення мовою Python, а також ознайомлення з повним циклом алгоритмізації рішення прикладної задачі – від дослідження проблеми, і до експериментальних розрахунків, калібровки програми та оформлення пояснювальної записки.

ЛІТЕРАТУРА

1. Демидович Б.П. Основы вычислительной математики / Демидович Б.П. Марон И.А. / ст. 562-576
2. Численные методы / Университет ИТМО / [Електронний ресурс] - http://aco.ifmo.ru/el\_books/numerical\_methods/lectures/glava3.html

ДОДАТОК А

Текст програми

print("\n\n\n","\t\t\t\t\t ", "РОЗРАХУНКОВО-ГРАФІЧНА РОБОТА", sep = "")

print("\t\t\t\t\t\t ", "З дисципліни:\n\t\t\t\t\t 'Програмування' 1-ий семестр.", sep = "")

print("\n\n\n","\t\t\t\t ", "Тема:'Програма чисельного диференціювання'", sep = "")

print("\n\n\n","\t\t\t", "Виконав","\t\t\t\t\t\t\t\t", "Чорний Д.С.", sep = "")

print("\n\n\n","\t\t\t\t\t\t\t", "2020","\n\n\n" , sep = "")

print("ІНФОРМАЦІЙНЕ ПОВІДОМЛЕННЯ ТА ВИМОГИ ДО ДАНИХ:",

"",

"Ця програма дозволяє обчислити приблизне значення\nпохідної по таблично заданим даним або аналітичній формулі.\nДля обчислення програма інтерполює табличні значення за\nдопомогою поліному Ньотона.",

"",

"ВИМОГИ ДО ДАНИХ:",

"",

"Програма обчислює поліном на основі рівновіддалених точок.\nТому, для табличного введення функції потрібно буде обрати крок,\nта місце розташування першого значення в стопвчику х-ів.\nУсі інші значення х програма обрахує з зазначеним вами кроком.\nПереконайтеся, що задана вами функція може бути диференційована\nу точці диференціювання. У разі вводу аналітичної формули вам буде\nзапропоновано ввести функцію за допомогою синтаксису мови Python.\nДалі, буде обрахована таблиця значень з кроком 0.1. Якщо ви бажаєте\nобрати інший крок, або функція не визначена в місцях з кроком 0.1\nвід точки диференціювання рекомендованим є ввести функцію\nвже в табличному вигляді. Точка має знаходитися в проміжку від\nнайпершого табличного х і до останнього.",

sep="\n")

import math

import itertools

def ask\_yes\_or\_no(firstmsg="", inputmsg=""):

print(firstmsg)

yes\_or\_not = True

while yes\_or\_not:

y\_o\_n = input(inputmsg)

if (y\_o\_n == "Так") or (y\_o\_n == "так"):

yes\_or\_not = False

return True

elif (y\_o\_n == "Ні") or (y\_o\_n == "ні"):

yes\_or\_not = False

return False

else:

print("Ви ввели некоректне значення. Введіть 'так' або 'ні'.")

def ask\_num(msg, type\_="float"):

num\_stay = True

while num\_stay:

try:

if type\_ == "int":

num = int(input(msg+"\n"))

num\_stay = False

return num

else:

num = float(input(msg+"\n"))

num\_stay = False

return num

except ValueError:

print("Ви ввели непідходяще значення. Введіть значення числом.")

def calculate(num, list\_, h):

is\_in\_table = False

for i in list\_:

if num == i[0]:

place = list\_.index(i)

is\_in\_table = True

elif i == list\_[0] and num < i[0]:

return ("Х не входить у проміжок заданих таблично даних." , "Х не входить у проміжок заданих таблично даних.")

elif i == list\_[-1] and num > i[0]:

return ("Х не входить у проміжок заданих таблично даних." , "Х не входить у проміжок заданих таблично даних.")

elif (list\_[list\_.index(i)-1][0] < num) and (num < i[0]):

if (num-list\_[list\_.index(i)-1][0] > i[0]-num) and (i != list\_[-1]):

place = list\_.index(i)

else:

place = list\_.index(i)-1

dif\_table = [[]] dif\_table = [[]]

for i in range(len(list\_)-1):

dif\_table[0].append(list\_[i+1][1]-list\_[i][1])

while len(dif\_table[-1])!=1:

dif\_table.append([])

for i in range(len(dif\_table[-2])-1):

dif\_table[-1].append(dif\_table[-2][i+1]-dif\_table[-2][i])

if is\_in\_table:

res = 0

for i in range(math.ceil((len(list\_)-place)/2)):

res += ((-1)\*\*i)\*((dif\_table[i][place])/(i+1))

res = res/h

error = (((-1)\*\*(len(list\_)-place-3))\*dif\_table[len(list\_)-place-2][place])/(h\*(len(list\_)-place-1))

else:

res = 0

q = (num-list\_[place][1])/h

for i in range(math.ceil((len(list\_)-place)/2)):

if i == 0:

res += dif\_table[i][place]

elif i == 1:

res += ((2\*q-1)/2)\*dif\_table[i][place]

else:

jopa = 1

for j in range(1, i+1):

jopa =jopa\*(q-j)

jui = 0

coefs = list(itertools.combinations(range(1, i+1), i-1))

for j in range(i):

jack = q

for c in coefs[j]:

jack=jack\*(q-c)

jui = jui+jack

res += ((jopa + jui)/(math.factorial(i+1)))\*dif\_table[i][place]

res = res/h

jopa = 1

for j in range(1, (len(list\_)-place-1)):

jopa =jopa\*(q-j)

jui = 0

coefs = list(itertools.combinations(range(1, (len(list\_)-place+1)), (len(list\_)-place-1)))

for j in range((len(list\_)-place-1)):

jack = q

for c in coefs[j-1]:

jack=jack\*(q-c)

jui = jui+jack

error = (((-1)\*\*(len(list\_)-place-3))\*dif\_table[len(list\_)-place-2][place])/(h\*(len(list\_)-place-1))

return (round(res, 6), round(error, 6))

stay = True

while stay:

# питання введення функції таблично, чи аналітично

if ask\_yes\_or\_no("Бажаєте ввести функцію таблично?", "Напишіть 'так' для табличного введення, або 'ні' для аналітичного.\n"):

point\_func\_stay = True

der\_stay = False

point\_stay = True

else:

point\_func\_stay = False

der\_func\_stay = True

der\_stay = True

while point\_func\_stay:

#Введення функції таблично

if point\_stay:

print("\nВведіть крок табличних значень, перший 'х' з таблиці та кожен 'у' відповідно. \nДля зупинки запису введіть пустий рядок.")

numbers = []

h = ask\_num("Введіть крок табличних значень.")

counter = 0

x = ask\_num("Введіть перший 'х'. (Усі інші будуть обраховані у відповідності з кроком.)")

while point\_stay:

inp = input("Введіть 'у'. Для зупинки запису введіть пустий рядок.\n")

if inp == "":

if len(numbers)==0:

print("Ви не ввели ні одного y. Спробуйте ще.")

else:

point\_stay = False

else:

try:

numbers.append(tuple([x+(h\*counter), float(inp)]))

counter +=1

except ValueError:

counter +=1

print("Значення в некоректному форматі. Введіть число.")

#Введення точки диференціювання

num = ask\_num("\nВведіть точку для диференціювання.")

#Обрахунок значення

result = calculate(num, numbers, h)

print("\nЗначення похідної у точці:", result[0])

print("Обрахована помилка:", result[1])

##Спробувати ще раз з цією функцією?

if not ask\_yes\_or\_no("\nБажаєте спробувати ще раз з цією функцією?", "Напишіть 'так', або 'ні'.\n"):

point\_func\_stay = False

else:

point\_stay = False

while der\_func\_stay:

#Ввід точки для диференціювання

num = ask\_num("\nВведіть точку для диференціювання.")

#Введення аналітично

while der\_stay:

func\_ = input("Введіть функцію від x за допомогою синтаксису мови Python.\nЗвертайтеся до модулю math через 'math.'\n")

try:

x = num

eval(func\_)

def func(x):

return eval(func\_)

der\_stay = False

except:

print("\nНеправильно введена функція або функція не визначена у точці диференціювання. Спробуйте ще.")

#Переведення фунціїї в табличну форму

iterations = 14

numbers = []

for i in range(math.floor(iterations/2)):

try:

numbers.append(tuple([num-math.floor(iterations/2)+(i/10), func(num-math.floor(iterations/2)+(i/10))]))

except:

pass

for i in range(math.ceil(iterations/2)):

try:

numbers.append(tuple([num+(i/10), func(num+(i/10))]))

except:

pass

#Обрахунок значення похідної від поліному

result = calculate(num, numbers, 0.1)

print("\nЗначення похідної у точці:", result[0])

print("Обрахована помилка:", result[1])

#Спробувати ще раз з цією функцією?

if not ask\_yes\_or\_no("\nБажаєте спробувати ще раз з цією функцією?", "Напишіть 'так', або 'ні'.\n"):

der\_func\_stay = False

else:

der\_stay = False

#Запустити програму ще раз?

if not ask\_yes\_or\_no("\nБажаєте спробувати ще раз з іншою функцією?", "Напишіть 'так', або 'ні'.\n"):

stay = False

ДОДАТОК Б

Тестові приклади

Функціональність програми була перевірена на тестових даних, взятих з підручника «Основы вычислительной математики» за авторством Демидовича Б.П. та Марона И.А., які знаходяться на сторінці 566.

Тестові дані:

|  |  |  |
| --- | --- | --- |
|  | Час, с | Відстань, см |
| 0 | 0 | 0 |
| 1 | 0.01 | 1.519 |
| 2 | 0.02 | 6.031 |
| 3 | 0.03 | 13.397 |
| 4 | 0.04 | 23.396 |
| 5 | 0.05 | 35.721 |
| 6 | 0.06 | 50.0 |
| 7 | 0.07 | 65.798 |
| 8 | 0.08 | 82.635 |
| 9 | 0.09 | 100 |

Введення даних в програму:

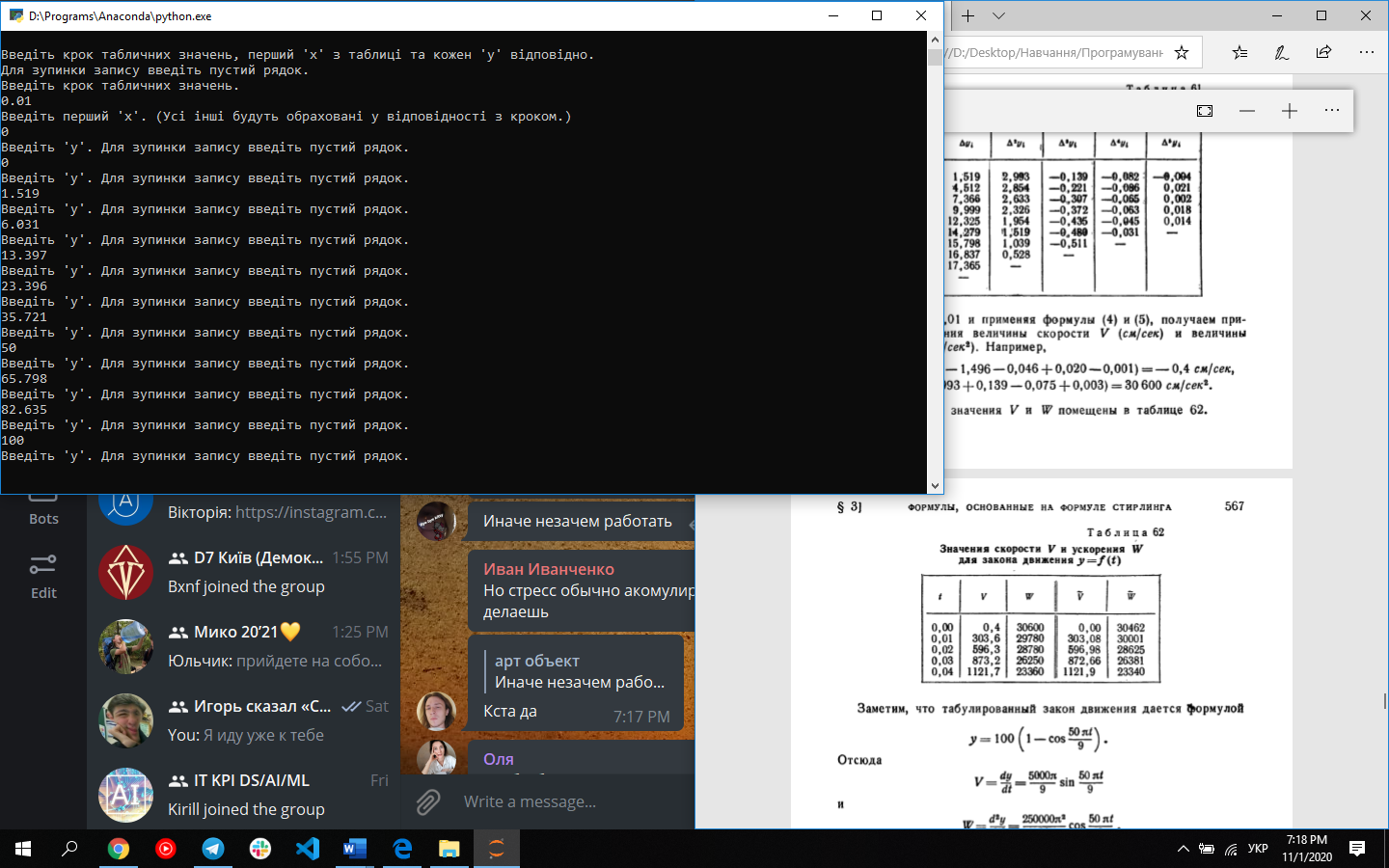


Рисунок Б.1

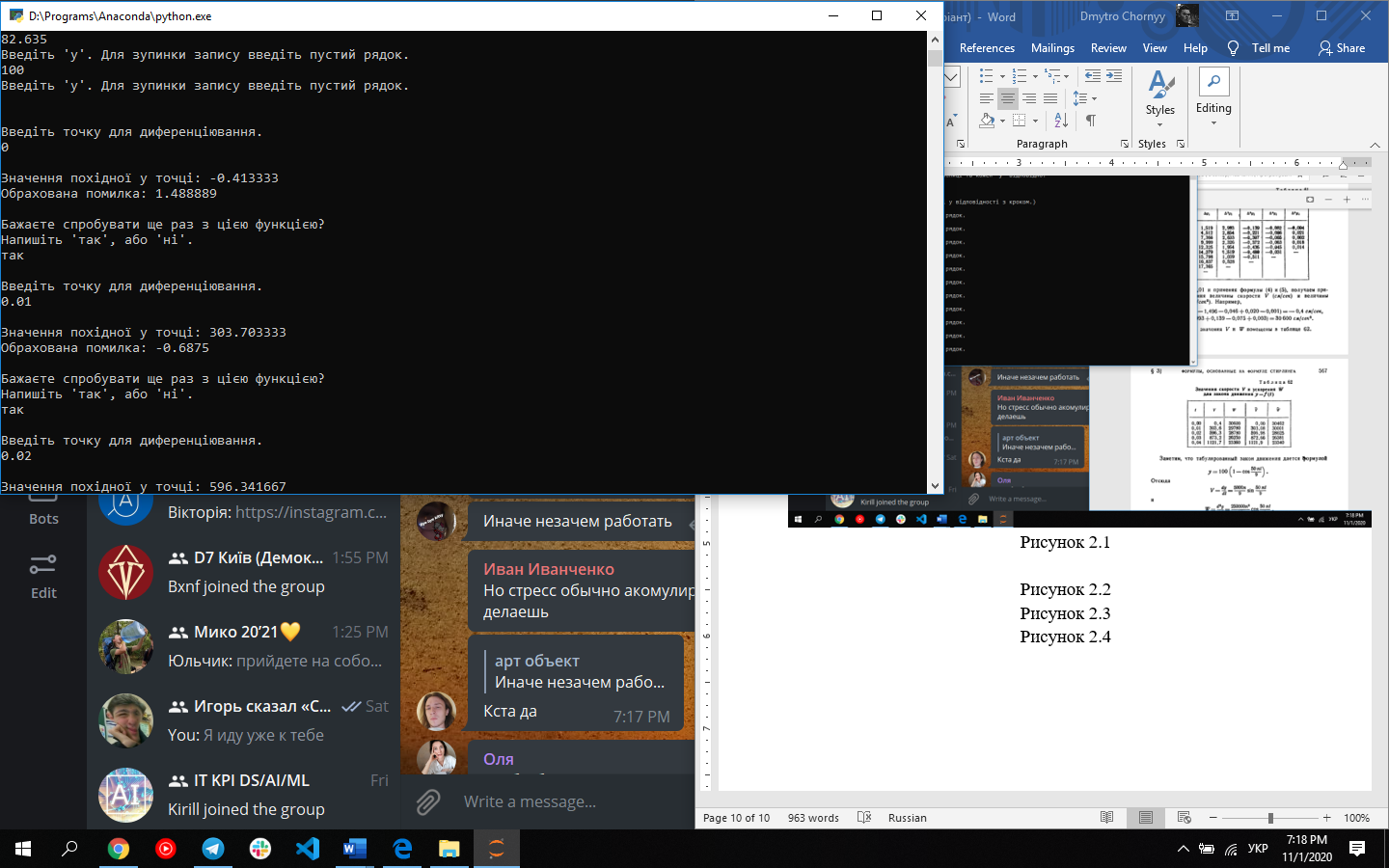


Рисунок Б.2

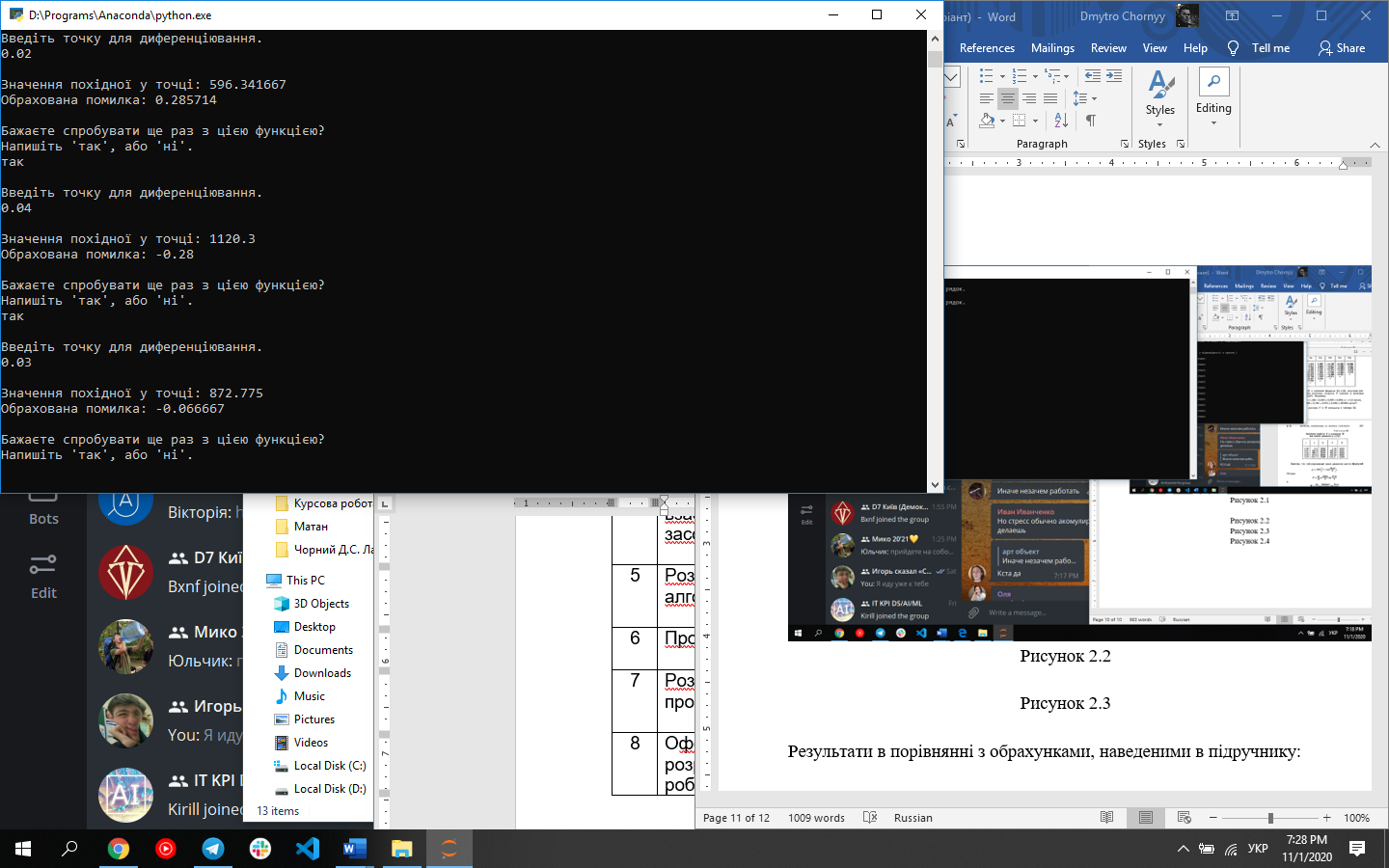


Рисунок Б.3

Результати в порівнянні з обрахунками, наведеними в підручнику:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Значення «у» | Результат роботи програми\* | Обрахована помилка\* | Обраховані значення за допомогою формули | Істинні значення похідних |
| 0 | -0,4133 | 1.4889 | 0,04 | 0 |
| 0,01 | 303,703 | -0.6875 | 303,6 | 303,08 |
| 0,02 | 596,342 | 0,285 | 596,3 | 596,98 |
| 0,03 | 872.775 | -0.0667 | 873,2 | 872,66 |
| 0,04 | 1120,3 | -0.28 | 1121,7 | 1121,9 |

Таблиця Б.1

\*невідповідність у 1-2% від обрахованих у підручнику найймовірніше пов’язана з особливостями представлення чисел у пам’яті комп’ютера.