Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра автоматизованих систем управління



**Звіт**

до виконаної лабораторної роботи № 2

з дисципліни

“Чисельні методи”

на тему:

***«Метод Штурма відокремлення коренів»***

Виконав

студент групи *ОІ-11 сп*

*Вальчевський П. В.*

Викладач:

*Сенета М. Я.*

Львів – 2023

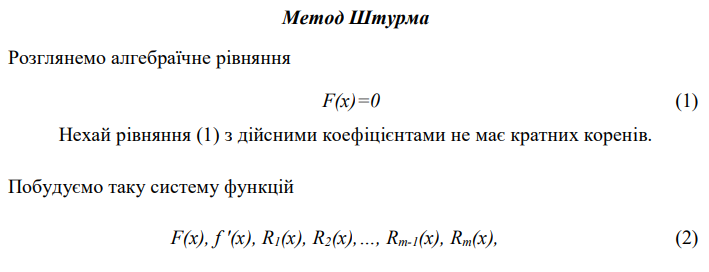
## Лабораторна робота № 2

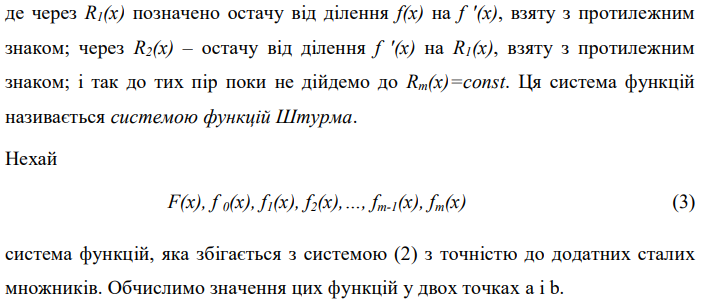
*Тема роботи:*  **«Метод Штурма відокремлення коренів».**

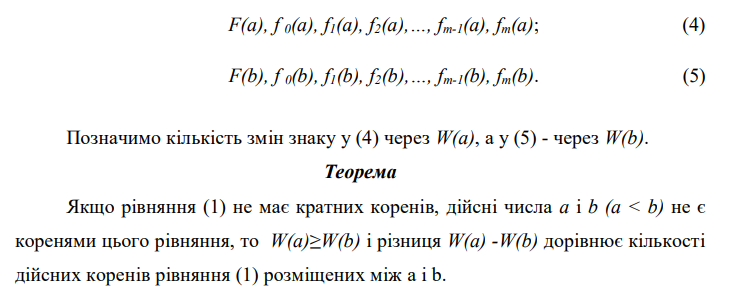
*Мета роботи:*вивчити і засвоїти методи відокремлення коренів.

***Порядок виконання роботи***

1. **Основні теоретичні відомості**







**Номер варіанту – 3.**

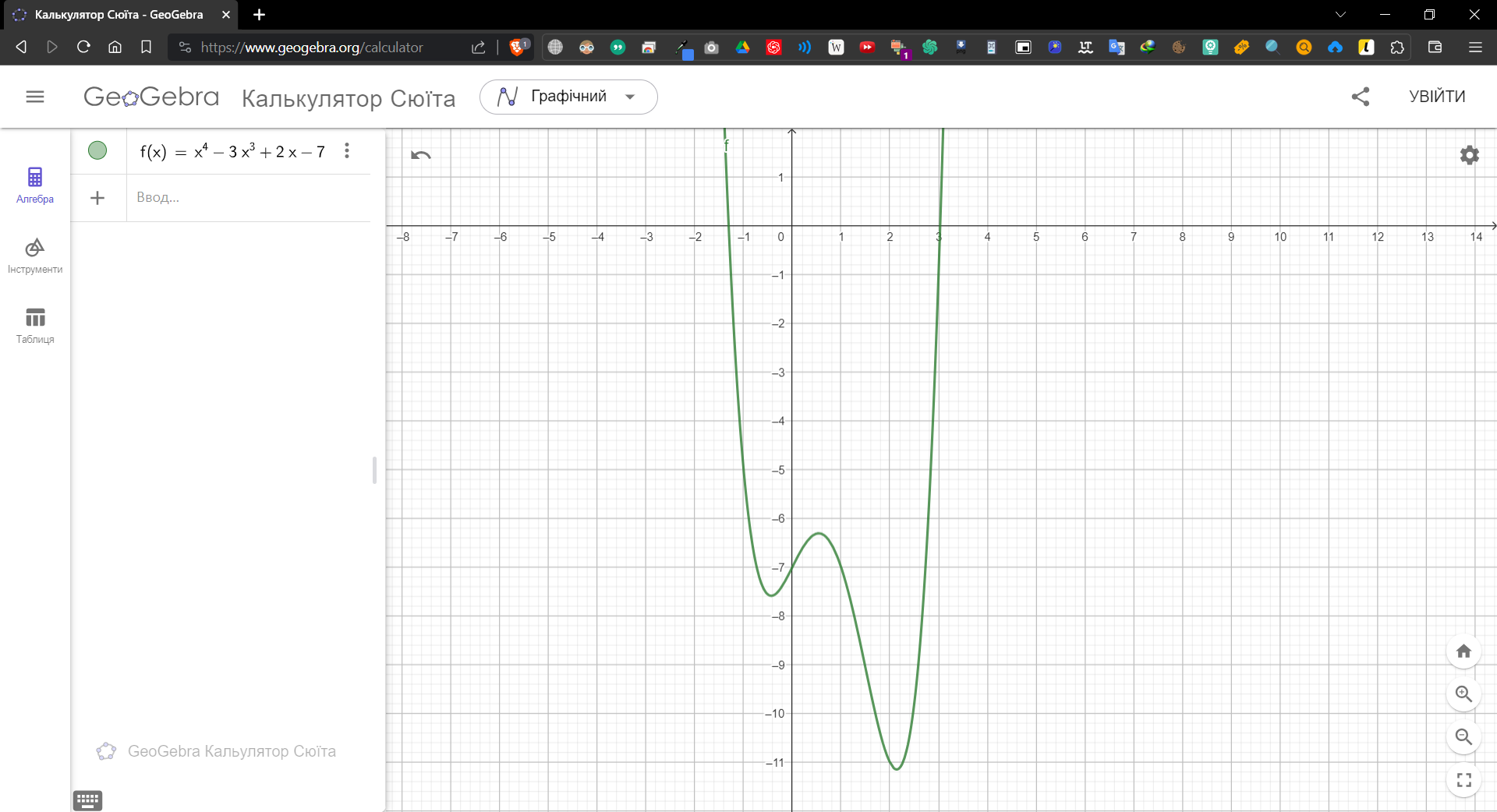
1. ****
2. **Умова завдання**

Відокремити кожен з коренів рівняння f(x)=0, використовуючи метод Штурма з довжиною проміжків не більше 1. Отриманий результат перевірити за допомогою графіка функції.

Для побудови графіка можна використати довільну програму (напр., Advanced Grapher або Wolfram).

Доповнити текст програми таким чином, щоб у результатах виконання програми було відображено інформацію: група, ПІБ студента, номер ЛР і варіанту завдання.

1. **Алгоримт розв’язання завдання**
2. Створення методів для обрахунку усіх функцій.
3. Створення методу для визначення списку знаків.
4. Створення методу для обрахунку кількості змін знаку.
5. Створення методу для визначення діапазонів розміщення коренів.
6. Створення методу для обрахунку коренів функції (для перевірки).
7. Створення методу для відображення графіку функції (для перевірки).
8. Створення логіки в основній функції шляхом створення відповідних змінних й структур даних та послідовне використання раніше створених методів.
9. Вивід результату.
10. Перегляд й аналіз результату. Відлагодження програми при некоректній роботі.
11. Коментування блоків коду.
12. **Виконане завдання вручну**
13. За допомогою онлайн-сервісу GeoGebra ( <https://www.geogebra.org/calculator> ), будую графік функції й визначаю проміжок [-3; 4], де приблизно розміщуються корені функції. Функція записана у вигляді: Графік функції:



1. Обраховую похідну функції: .
2. Далі знаходжу остачу від ділення, але результат беру з протилежним знаком:
3. Оскільки, остача від ділення не число, продовжую дану процедуру, але з іншими функціями:
4. Оскільки, остача від ділення не число, продовжую дану процедуру, але з іншими функціями:
5. При останній дії було одержано число, отже визначено кількість функцій у системі функцій. Створюємо таблицю, куди записуємо значення зміни знаків функцій на певному значенні з проміжку.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **X** | **F(x)** | **F(x)’** | **R1(x)** | **R2(x)** | **R3(x)** | **W** |
| -3 | + | - | + | - | - | 3 |
| -2 | + | - | + | - | - | 3 |
| -1 | - | - | + | - | - | 2 |
| 0 | - | + | + | - | - | 2 |
| 1 | - | - | + | - | - | 2 |
| 2 | - | - | + | + | - | 2 |
| 3 | - | + | + | + | - | 2 |
| 4 | + | + | + | + | - | 1 |

1. Для перевірки можна використати очний метод – кількість змін знаків поступово спадає на одиницю. Також, кількість коренів можна визначити за допомогою різниці нижньої межі та верхньої: . Записавши діапазон на зміні кількості знаків у функціях, можна отримати таке: -2 < x < -1 та 3 < x < 4, що співпадає з діапазоном реальних коренів, які можна побачити на графіку функції. Зауважу, що коли у таблиці з’являється «0», це означає, що ми потрапили на корінь та знаки можуть не чергуватися, якщо нуль поміж однакових знаків, але у моєму випадку такого не сталося.
2. **Код програмної реалізації**
3. *Файл main.py (реалізація алгоритму та опис створених методів)*

import numpy as np # Для зберігання й обчислення діпазону значення Х і У функції.

import matplotlib.pyplot as plt # Для побудуви графіку (полотно).

from sympy import symbols, Eq, real\_roots # Для обрахунку коренів.

# Обрахунок функції зі значенням змінної.

def getFx(x=float) -> float:

return round(x \*\* 4 - 3 \* x \*\* 3 + 2 \* x - 7, 2)

# Обрахунок похідної функції зі значенням змінної.

def getFxDerivative(x=float) -> float:

return round(4 \* x \*\* 3 - 9 \* x \*\* 2 + 2, 2)

# Обрахунок першої остачі зі значенням змінної.

def getR1Fx(x=float) -> float:

return round(27.0 / 16 \* x \*\* 2 - 3.0 / 2 \* x + 53.0 / 8, 2)

# Обрахунок другої остачі зі значенням змінної.

def getR2Fx(x=float) -> float:

return round(1664.0 / 81 \* x - 5680.0 / 243, 2)

# Обрахунок третьої остачі зі значенням змінної.

def getR3Fx(x=float) -> float:

return round(-1229211.0 / 173056, 2)

# Повернення ітераційного списку при певному значенні змінної.

def getIterationList(x=float) -> list:

iterationList = [getFx(x), getFxDerivative(x), getR1Fx(x), getR2Fx(x), getR3Fx(x)]

return iterationList

# Повернення знаку числа.

def getSignNumber(number=float) -> str:

if number > 0:

return "+"

elif number < 0:

return "-"

else:

return "0"

# Повернення списку знаків ітераційного списку.

def getSignIterationList(iterationList=list) -> list:

signIterationList = []

for i in iterationList:

signIterationList.append(getSignNumber(i))

return signIterationList

# Обрахунок кількості змін знаків при ітерації.

def getCountChangeInSignIterationList(signIterationList=list) -> int:

counter = 0

for i in range(len(signIterationList) - 1):

if signIterationList[i] != "0" and signIterationList[i] != signIterationList[i + 1]:

counter += 1

return counter

# Повернення списку з діапазоном коренів.

def getRangeOfRootsList(xList=list, countSignChangeList=list) -> list:

rangeOfRootsList = ["Кількість коренів: " + str(int(countSignChangeList[0] - countSignChangeList[-1]))]

counterRoots = 0

for i in range(len(countSignChangeList) - 1):

if countSignChangeList[i] - countSignChangeList[i + 1] != 0:

counterRoots += 1

rangeOfRootsList.append(

"Діапазон № " + str(counterRoots) + ": " + str(xList[i]) + " < x < " + str(xList[i + 1]))

return rangeOfRootsList

# Вивід таблиць зі значеннями або знаками функції.

def getTablesOfResultLists(message=str, xList=list, resultList=list, countChangeList=list) -> None:

print(message)

sepValue = " | "

lengthLine = 45

line = "\t" + "=" \* lengthLine

print("\tx", "f(x)", "f(x)'", "R1", "R2", "R3", "W", sep=sepValue)

print(line)

for i in range(len(xList)):

print("\t", xList[i], end=sepValue)

for j in range(len(resultList[i])):

print(resultList[i][j], sep=sepValue, end=sepValue)

print(countChangeList[i])

print(line)

# Вивід діапазону коренів

def getInformationRangeOfRoots(message=str, rangeOfRootsList=list) -> None:

print(message)

for i in rangeOfRootsList:

print(f"\t{i}")

# Вивід графіку функції.

def drawFx(a=int, b=int) -> None:

# Константи підібрані вручну.

num = 40 # Кількість Х для подальшого зображення графіку.

aY = -12.5 # Нижня межа У.

bY = 4 # Верхня межа У.

# Діапазон Х.

X = np.linspace(a, b, num)

# Обраховуємий діапазон У.

Y = X \*\* 4 - 3 \* X \*\* 3 + 2 \* X - 7

# Малювання графіку.

plt.plot(X, Y, label='Графік функції', color='b')

plt.xlabel('X')

plt.ylabel('Y')

plt.title('ЛР № 2, Варіант № 3, Вальчевський П. В., група ОІ-11 сп')

plt.legend()

plt.grid(True)

plt.xlim(a, b)

plt.ylim(aY, bY)

plt.axvline(0, color='r', linewidth=1) # Вісь X.

plt.axhline(0, color='r', linewidth=1) # Вісь Y.

plt.show()

# Обрахунок й вивід коренів функції.

def calcAndOutputRoots() -> None:

# Оголошуємо символ X для подальшого обчислення рівняння.

X = symbols('X')

# Вивід лише дійсних коренів від рівняння.

print("Корені рівняння (обчислено за допомогою бібліотеки для перевірки):")

counter = 0

for root in real\_roots(Eq(X \*\* 4 - 3 \* X \*\* 3 + 2 \* X - 7, 0)):

counter += 1

print(f"\tКорінь № {counter} : {round(root.evalf(), 2)}")

# Виконання програми.

if \_\_name\_\_ == "\_\_main\_\_":

try:

getResult = [] # Список значень усіх функцій при кожному значенню х (ітерації).

getResultSign = [] # Список знаків усіх функцій при кожному значенню х (ітерації).

getResultChangeSign = [] # Список кількості змін знаків кожної функції при ітераціях.

getValuesX = [] # Список значень х.

getRangeOfRoots = [] # Список діапазону зміни кореня функції.

a = -3 # Нижня межа.

b = 4 # Верхня межа.

# Обрахунок логіки програми.

for x in range(a, b + 1):

getValuesX.append(x)

getResult.append(getIterationList(x))

getResultSign.append(getSignIterationList(getResult[-1]))

getResultChangeSign.append(getCountChangeInSignIterationList(getResultSign[-1]))

getRangeOfRoots = getRangeOfRootsList(getValuesX, getResultChangeSign)

# Вивід результатів.

print(f"ЛР № 2, Варіант № 3, Вальчевський П. В., група ОІ-11 сп, взято відрізок [{a}; {b}]")

lengthLineForOutput = 100

lineForOutput = "-" \* lengthLineForOutput

print(lineForOutput)

nameTableOne = "Таблиця № 1 значення функції:"

getTablesOfResultLists(nameTableOne, getValuesX, getResult, getResultChangeSign)

print(lineForOutput)

nameTableTwo = "Таблиця № 2 знаки функції:"

getTablesOfResultLists(nameTableTwo, getValuesX, getResultSign, getResultChangeSign)

print(lineForOutput)

nameInformationBlock = "Висновки програми:"

getInformationRangeOfRoots(nameInformationBlock, getRangeOfRoots)

print(lineForOutput)

calcAndOutputRoots()

print("Графік функції відображено в іншому вікні!")

drawFx(a, b)

except Exception:

print("Помилка при виконанні програми. Перевірте коректність запису алгоритму!")

**Скріншот отриманого результату**

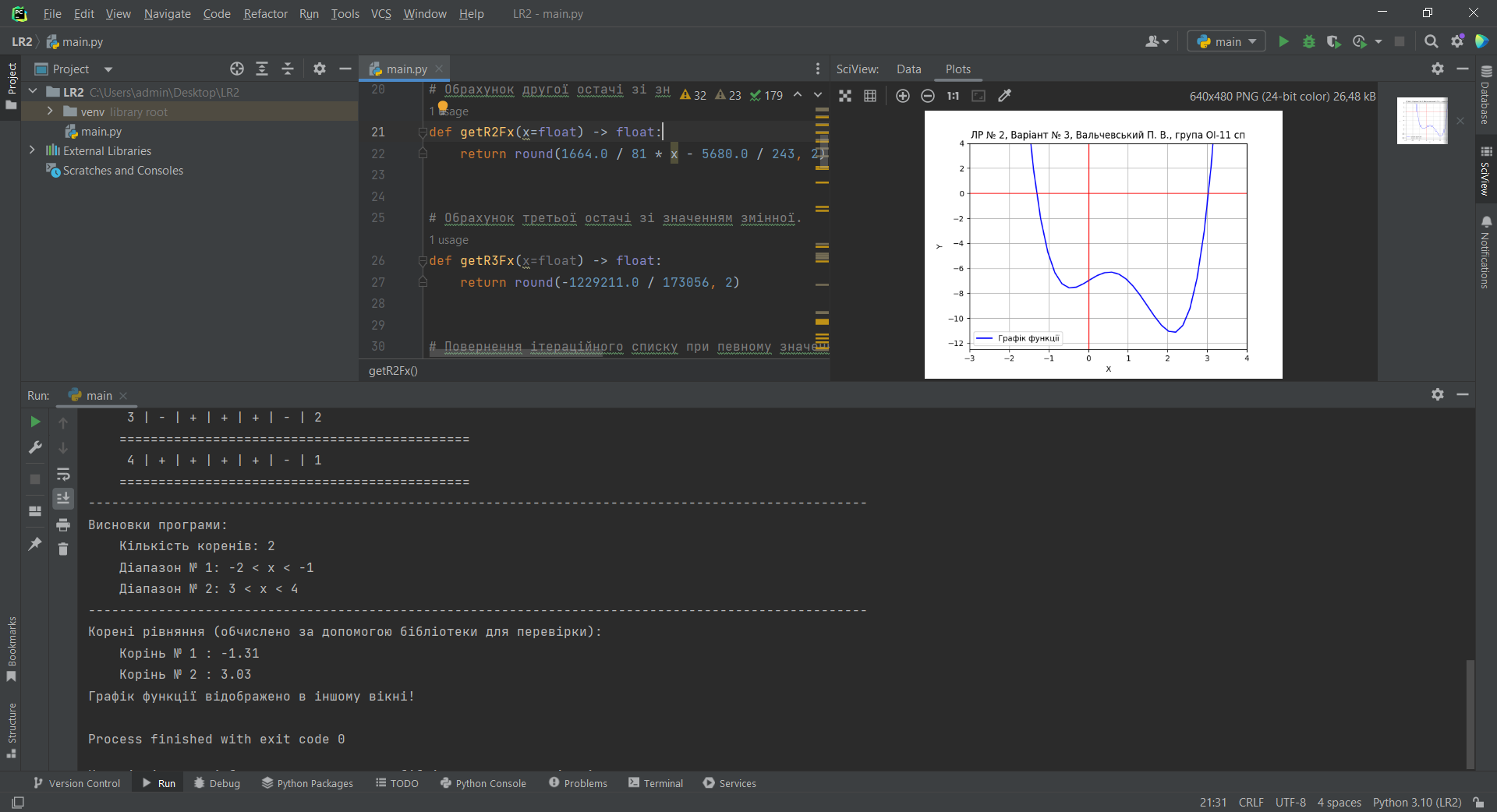
1. 

Рисунок 1 Знімок з середовища розробки (для підтвердження створення графіку функції).

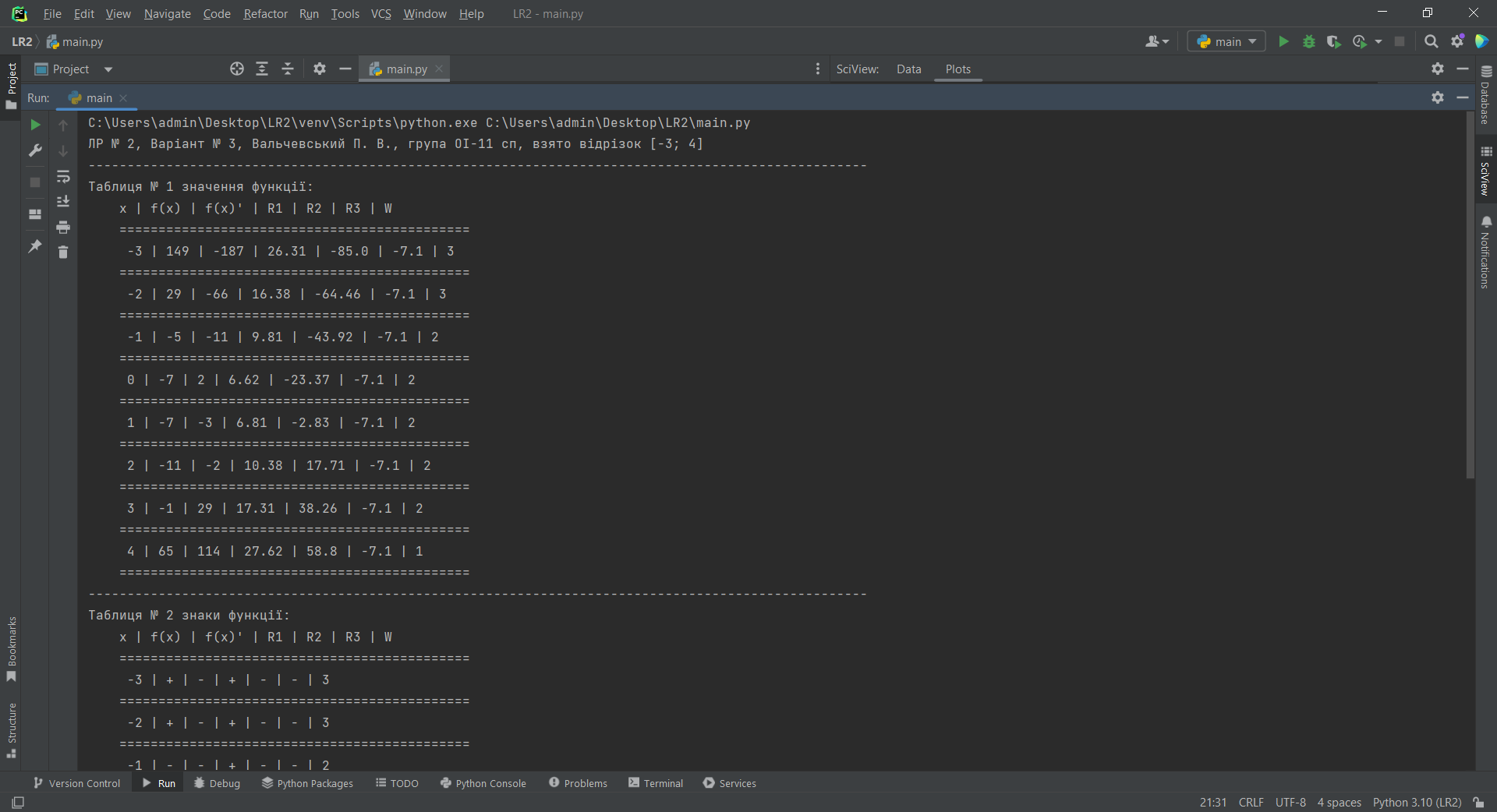


Рисунок 2 Результати розрахунків під час виконання програми.

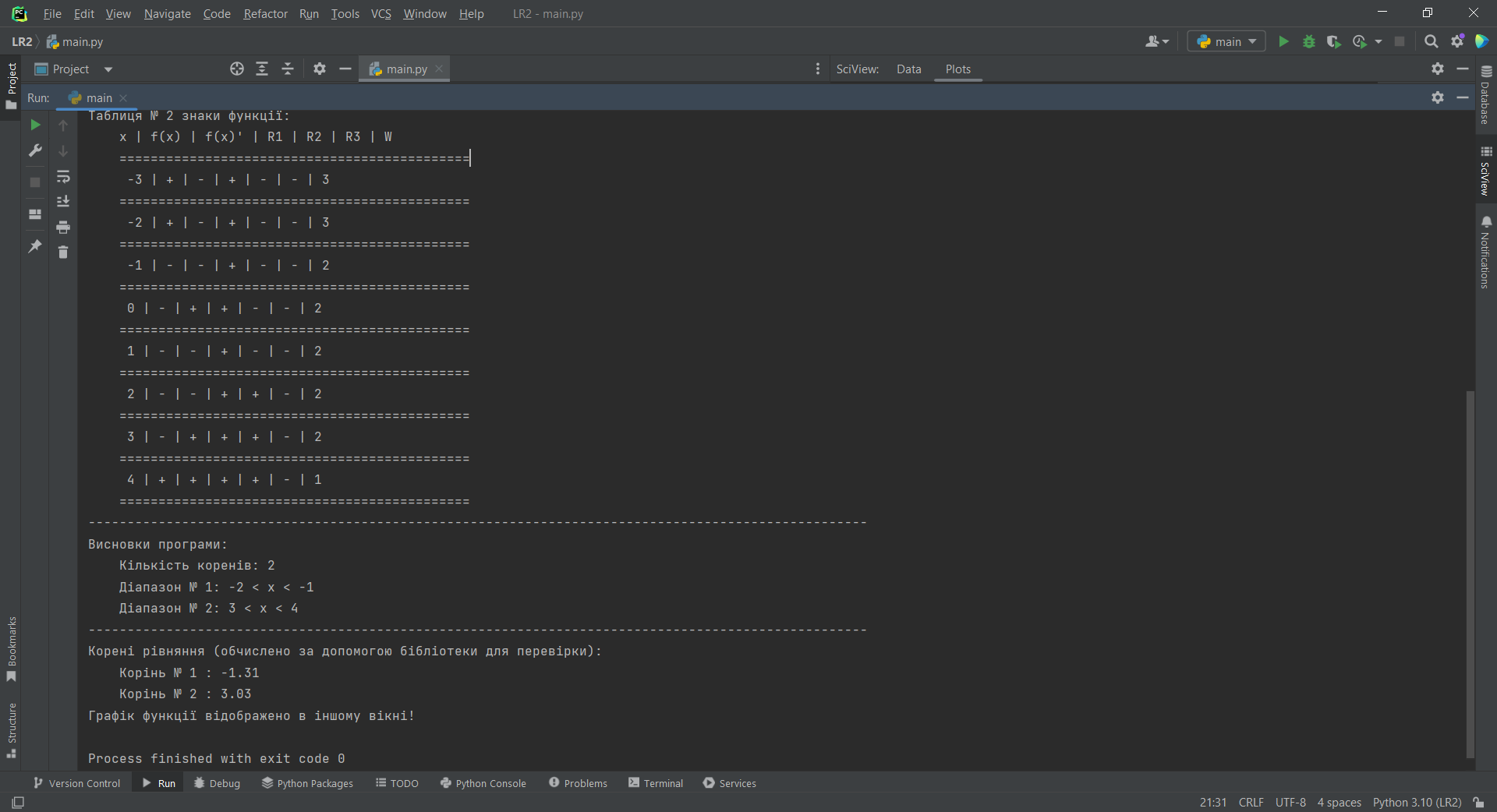


Рисунок 3 Результати розрахунків під час виконання програми (продовження).

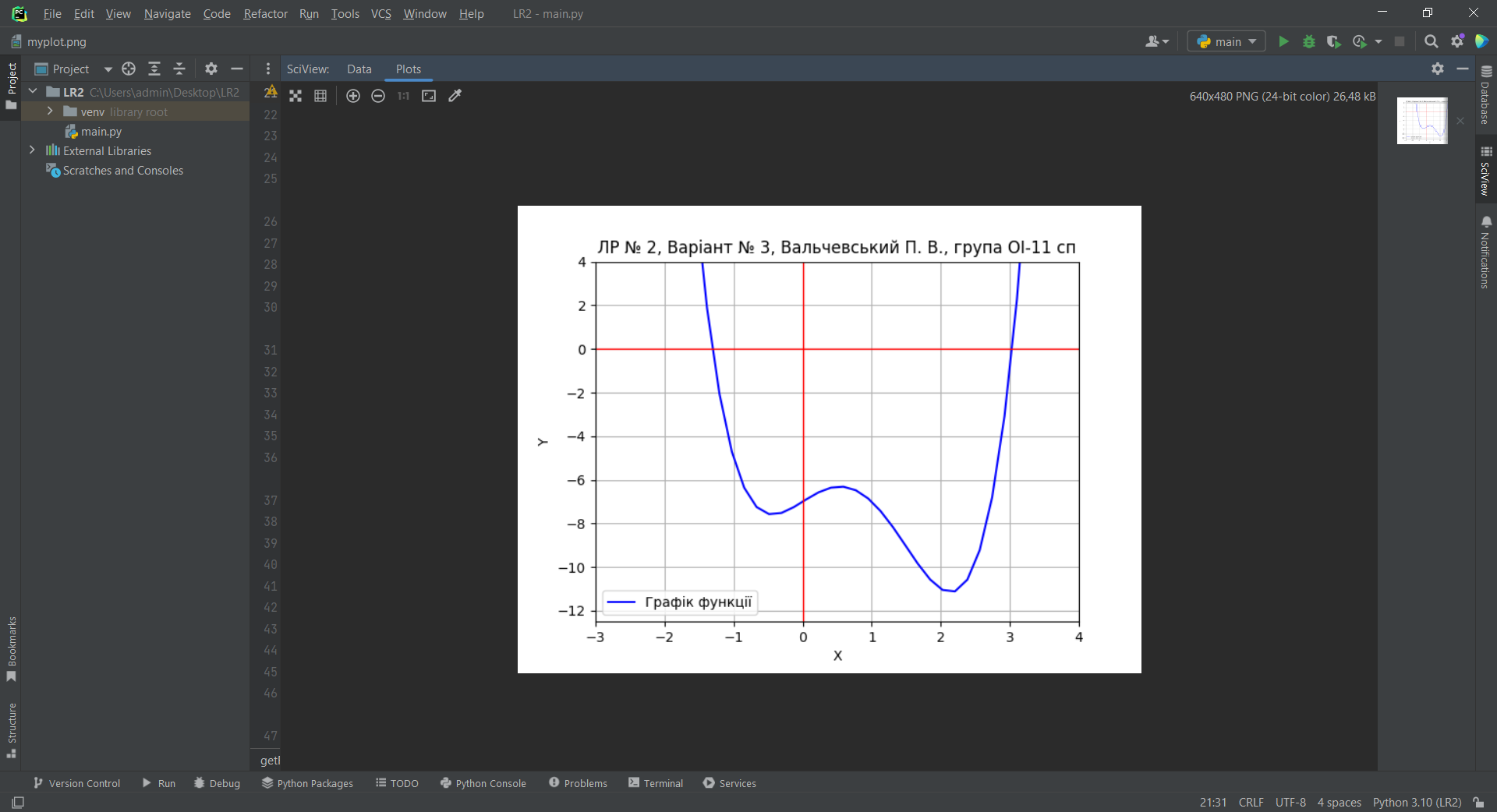


Рисунок 4 Створений графік функції (в окремому вікні).

1. **Висновки (з аналізом результату)**

Згідно результатів та порівняння з результатами розв’язаних вручну під час виконання програми – алгоритм виконується правильно та без помилок (з урахуванням виняткових ситуацій).

Реалізовуються умови для коректного обчислення, тобто, кожне подальше обчислення буде залежати від результатів попереднього й якщо дані не будуть правильно обчислені або не відповідатимуть обмеженням, то програма виведе повідомлення про виняткову ситуацію.

Для користувача буде вивід таблиці зі значенням функцій та зміною знаків. Далі буде виведено кількість коренів й діапазони їх розміщення. Також, для додаткової перевірки було використано можливості мови Python та обчислено корені рівняння через відповідну бібліотеку, а також було побудовано відповідний графік для функції через інші бібліотеки, що абсолютно співпадає з вище представленим через GeoGebra. Усі вище згадані бібліотеки мають коментар-пояснення у програмному коді.

Програма була відлагоджена й виконана у програмному середовищі PyCharm 2023.2.1 за допомогою мови програмування Python 3.10.4.