МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИЗМІСТ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра
інформатики та програмної інженерії
(повна назва кафедри, циклової комісії)

w

КУРСОВА РОБОТА

з «Основи програмування 2. Модульне програмування»
(назва дисципліни)
на тему: Розв’язання задач інтерполяції

Студентки 1 курсу, групи ІП-15
Кондрацької Соні Леонідівни

Спеціальності 121 «Інженерія програмного забезпечення»

Керівник Головченко Максим Миколайович
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Кількість балів: _____
Національна оцінка _____

Члени комісії

_____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

_____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

Київ- 2022 рік

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

(назва вищого навчального закладу)

Кафедра інформатики та програмної інженерії

Дисципліна Основи програмування

Напрям "ІПЗ"

Курс 1 Група ІП-15 Семестр 2

ЗАВДАННЯ

на курсову роботу студента

Кондрацької Соні Леонідівни

(прізвище, ім'я, по батькові)

1. Тема роботи Розв'язання задач інтерполяції

2. Строк здачі студентом закінченої роботи 12.06.2022

3. Вихідні дані до роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Дата видачі завдання 10.02.2022

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсової роботи	Термін виконання етапів роботи	Підписи керівника, студента
1.	Отримання теми курсової роботи	10.02.2022	
2.	Підготовка ТЗ	23.05.2022	
3.	Пошук та вивчення літератури з питань курсової роботи	20.05.2022	
4.	Розробка сценарію роботи програми	21.05.2022	
6.	Узгодження сценарію роботи програми з керівником	07.06.2022	
5.	Розробка (вибір) алгоритму рішення задачі	07.06.2022	
6.	Узгодження алгоритму з керівником	07.06.2022	
7.	Узгодження з керівником інтерфейсу користувача	07.06.2022	
8.	Розробка програмного забезпечення	06.06.2022	
9.	Налагодження розрахункової частини програми	06.06.2022	
10.	Розробка та налагодження інтерфейсної частини програми	06.06.2022	
11.	Узгодження з керівником набору тестів для контрольного прикладу	07.06.2022	
12.	Тестування програми	07.06.2022	
13.	Підготовка пояснювальної записки	09.06.2022	
14.	Здача курсової роботи на перевірку	12.06.2022	
15.	Захист курсової роботи	15.06.2022	

Студент _____

(підпис)

Керівник _____ Головченко Максим Миколайович

(підпис) (прізвище, ім'я, по батькові)

"___" _____2022 р.

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 58 сторінок, 16 рисунків, 13 таблиць, 10 посилань.

Об’єкт дослідження: задачі інтерполяції.

Мета роботи: дослідження методів знаходження апроксимуючого поліному та створення програмного забезпечення для зручного використання алгоритму за допомогою графічного інтерфейсу.

Вивчено метод розробки програмного забезпечення з використанням принципів ООП. Приведені змістовні постановки задач, їх індивідуальні математичні моделі, а також описано детальний процес розв’язання кожної з них.

Виконана програмна реалізація методу Лагранжа та системи Ейткена для знаходження апроксимуючого поліному та зображенні його на графіку.

ВСТУП

Дана робота присвячена розробці програми для розв’язку задач інтерполяції з вибором одного з двох методів – Лагранжа або схеми Ейткена, з використанням об’єктно-орієнтованого програмування. Задача програмного забезпечення полягає в текстовому та графічному відображенні шуканого поліному.

У розділі “Постановка задачі” будуть описані вхідні та вихідні дані програми та що вона повинна робити.

У розділі “Теоретичні відомості” буде коротко описано сутність методу Лагранжа та схеми Ейткена.

У розділі “Опис алгоритмів” буде описана алгоритмічна частина програми.

У розділі “Опис програмного забезпечення” будуть описані класи та зв'язки між ними, стандартні та користувацькі методи, що використовуватимуться у програмному забезпеченні.

У розділі “Тестування програмного забезпечення” за планом тестування буде перевірено працездатність та стресостійкість програми.

У розділі “Інструкція користувача” буде наведено інформацію про те, як користуватися програмою, формат вхідних та вихідних даних та системні вимоги до програмного забезпечення.

У розділі “Аналіз і узагальнення результатів” буде перевірено правильність роботи методів, оцінено їх асимптотичну складність та порівняно їх роботу.

1 ПОСТАНОВКА ЗАДАЧІ

Розробити програмне забезпечення, що буде знаходити апроксимуючий поліном для заданої системи точок наступними методами:

а) метод Лагранжа;

б) система Ейткена;

Вхідними даними для даної роботи є два масиви значень *(X та Y)* для 2-10ти точок , причому масив *X* не повинен містити дублікати.

Вихідними даними для даної роботи являється поліном *n-1* степені, що є розв’язком даної системи точок при обраному методі , який виводиться на екран. Де *n* – кількість точок.

Програмне забезпечення повинно видавати розв’язок за умови, що вхідні дані введені коректно і обраний метод розв’язку. Якщо це не так, то програма повинна вивести відповідне повідомлення. Якщо всі дані внесені правильно, то програмне забезпечення повинно також виводити графік знайденого поліному.

2 ТЕОРЕТИЧНІ ВІДОМОСТІ

2.1 Метод Лагранжа [10]

Інтерполяційна формула Лагранжа використовується для довільно заданих вузлів інтерполювання. Нехай у точках x_0 , x_1 ,..., x_n відомі значення функції $y = f(x)$. Тобто задана таблична функція

x	x_0	x_1	...	x_n
y	y_0	y_1	...	y_n

(2.1.1)

Потрібно побудувати поліном $Ln(x)$ степеня не вище n, що має в заданих вузлах x_0 , x_1 ,..., x_n ті ж самі значення, що і функція $f(x)$. Тобто такий, що $Ln(x_i) = yi$ $(i = 0, 1, 2, ..., n)$.

Будуватимемо багаточлен n-ого степеня $Ln(x)$ у вигляді лінійної комбінації $\sum_{i=0}^{n} c_i l_i(x)$багаточленів n-ї степені $l_i(x)$ $(i = 0, 1, 2, ..., n)$. Індекс i показує номер багаточлена. Для того, щоб цей многочлен був інтерполяційним для функції $f(x)$, достатньо зафіксувати як коефіцієнти c_i цієї лінійної комбінації, задані в табл. (2.1.1) значення $y_i=f(x_i)$, а від базисних багаточленів $l_i(x)$ вимагати виконання умови

$$l_i(x_j)=\delta_{ij}=\begin{cases}1 & ,якщо\ j=i\\ 0 & ,якщо\ j!=i\end{cases} \qquad (2.1.2)$$

де δ_{ij} – символ Кронекера.

У цьому випадку для багаточлена $L_n(x)\sum_{i=0}^{n} y_i l_i(x)$у кожному вузлі $x_i(i = 0, 1, 2,...,n)$ в силу (2.1.2), справедливо

$$L_n(x)=l_0(x_j)y_0+\ldots+l_{j-1}(x_j)y_{j-1}+l_j(x_j)y_j+l_{j+1}(x_j)y_{j+1}+\ldots+l_n(x_j)y_n=0+\ldots+0+y_j+0+\ldots+0=y_j$$

Щоб конкретизувати базисні багаточлени $li(x)$, врахуємо, що вони повинні відповідати умовам (2.2). Рівність нулю і-го багаточлена у всіх вузлах, крім і-го, означає, що $l_i(x)$ можна записати як$l_i(x)=A_i(x-x_0)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)$, а коефіцієнт A_i легко виходить із вимоги, що міститься в (2.1.2) $l_i(x_i) = 1$.

Підставляючи в вираз $l_i(x_i)$ значення $x=x_i$ і прирівнюючи результат

одиниці, отримуємо:

$$A_i = \frac{1}{(x_i - x_0)\dots(x_i - x_{i-1})(x_i - x_{i+1})\dots(x_i - x_n)}$$

Таким чином, базисні багаточлени Лагранжа мають вигляд:

$$l_i(x) = \frac{(x - x_0)(x - x_1)\dots(x - x_{i-1})(x - x_{i+1})\dots(x - x_n)}{(x_i - x_0)(x_i - x_1)\dots(x_i - x_{i-1})(x_i - x_{i+1})\dots(x_i - x_n)},$$

Тобто шуканий інтерполяційний багаточлен Лагранжа:

$$L_n(x) = \sum_{i=0}^{n} \frac{(x - x_0)(x - x_1)\dots(x - x_{i-1})(x - x_{i+1})\dots(x - x_n)}{(x_i - x_0)(x_i - x_1)\dots(x_i - x_{i-1})(x_i - x_{i+1})\dots(x_i - x_n)} y_i$$

2.2 Схема Ейткена [3]

В основі методу лежить багаторазове застосування лінійної інтерполяції. Відповідно до схеми Ейткена лінійна інтерполяція за точками $M_i(x_i,y_i)$ і $M_{i+1}(x_{i+1},y_{i+1})$ зводиться до обчислення визначника другого порядку

$$P_{i,i+1}(x) = \frac{1}{x_{i+1} - x_i}\begin{bmatrix} y_i & x_i - x \\ y_{i+1} & x_{i+1} - x \end{bmatrix}$$

При інтерполіруванні по трьох і більше точках послідовно обчислюються багаточлени

$$P_{i,i+1,i+2}(x) = \frac{1}{x_{i+2} - x_i}\begin{bmatrix} P_{i,i+1}(x) & x_i - x \\ P_{i+1,i+2}(x) & x_{i+2} - x \end{bmatrix}; P_{i,i+1,i+2,i+3}(x) = \frac{1}{x_{i+3} - x_i}\begin{bmatrix} P_{i,i+1,i+2}(x) & x_i - x \\ P_{i+1,i+2,i+3}(x) & x_{i+3} - x \end{bmatrix}$$

У загальному випадку інтерполяційний багаточлен n-го степеня, що приймає у точках x_i значення y_i ($i = \overline{0,n}$), записується таким чином:

$$P_{0,1,2,\dots,n}(x) = \frac{1}{x_n - x_0}\begin{bmatrix} P_{0,1,2,\dots,n-1}(x) & x_0 - x \\ P_{1,2,3,\dots,n}(x) & x_n - x \end{bmatrix}$$

3 ОПИС АЛГОРИТМІВ

Перелік всіх основних змінних та їхнє призначення наведено в таблиці 3.1

Таблиця 3.1 – Основні змінні та їхні призначення

Змінна	Призначення
arrX , arrY	Масиви координат точок, X та У відповідно.
size	Кількість точок(довжина масивів).
polynomial	Змінна для запису результату(шуканий поліном).
polyL	Поліном Лагранжа.
p	Базисні поліноми в методі Лагранжа та матриця в схемі Ейткена.
point	Символ-змінна X для розрахунку формул

3.1 Загальний алгоритм

1. Початок.

2. Зчитати введену систему точок:

    2.1. Зчитати систему координат точок.

    2.2. ПЕРЕВІРИТИ введені дані:

        2.2.1. ЯКЩО координати (елементи масивів x та y) точок – вірно записані раціональні числа, ТО перейти до наступного пункту. ІНАКШЕ видати відповідні повідомлення про помилки та перейти до 1.

        2.2.2. ЯКЩО кількість елементів x та y рівна та кількість точок >2 та <10, ТО перейти до наступного пункту. ІНАКШЕ видати відповідні повідомлення про помилки та перейти до 1.

2.2.3. ЯКЩО кількість точок >2 та <10, ТО перейти до наступного пункту. ІНАКШЕ видати відповідні повідомлення про помилки та перейти до 1.

2.2.4. ЯКЩО значення X не повторюються, ТО записати їх в $arrX, arrY$ відповідно. ІНАКШЕ видати відповідні повідомлення про помилки та перейти до 1.

3. Підключити обраний метод розв'язку:
   3.1. ЯКЩО обраний метод Лагранжа, ТО знайти поліном за методом Лагранжа. (пункт 3.2)
   3.2. ЯКЩО обрана схема Ейткена, ТО знайти поліном за схемою Ейткена. (пункт 3.3)
   3.3. ЯКЩО не обраний жодний метод, ТО вивести відповідне повідомлення та перейти до 3.
4. Вивести шуканий поліном та побудувати його графік.
5. ЯКЩО користувач хоче занести дані у файл, ТО:
   5.1. ЗАНЕСТИ поліном та введені координати у файл "polynom.txt".
6. КІНЕЦЬ

3.2 Алгоритм метода Лагранжа.

1. ПОЧАТОК
2. СТВОРЕННЯ змінної $polyL$ для відображення поліному і НАДАННЯ їй значення 0.
3. ЦИКЛ проходу по всім елементам $arrX, arrY (i = 0,1,..,size)$:
   3.1. Змінна p дорівнює нулю.
   3.2. ЦИКЛ проходу по елементам $(j = 0,1,..,size)$:
      3.2.1. ЯКЩО $i != j$, ТО:

3.2.1.1. ОБЧИСЛЕННЯ базисних поліномів: $p = p * (point - arrX[j]) / (arrX[i] - arrX[j])$

3.3. ОБЧИСЛЕННЯ поліному Лагранжа: $polyL = polyL + p * arrУ[i]$

4. СКЛАДАННЯ частин поліному бібліотекою SymPy і ОТРИМАННЯ кінцевого результат: $polynomial = sym.expand(polyL)$
5. КІНЕЦЬ

3.3 Алгоритм схеми Ейткена.

1. ПОЧАТОК
2. СТВОРЕННЯ матриці $p[size\,x\,size]$, заповнення нулями.
3. ЦИКЛ проходу по елементам $arrY$ ($j = 0,1,..,size$):

   3.1. ЗАПОВНЕННЯ першого рядка матриці елементами масиву $arrY$: $p[0][i] = arrУ[i]$

4. ЦИКЛ проходу по елементам $arrX$ та рядах матриці p ($k = 0,1,..,size-1$):

   4.1. ЦИКЛ проходу по елементам $arrX$ та стовпцям матриці p ($i = k+1,..,size$):

   4.1.1. ЗАПОВНЕННЯ матриці знайденими поліномами:

$$p[k+1][i] = ((point - arrX[k]) * p[k][i] - (point - arrX[i]) * p[k][k]) / (arrX[i] - arrX[k])$$

5. СКЛАДАННЯ частин шуканого поліному в останній комірці матриці: $polynomial = sym.expand(p[size-1][size-1])$
6. КІНЕЦЬ

4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Діаграма класів програмного забезпечення розміщена на рисунку 4.1.

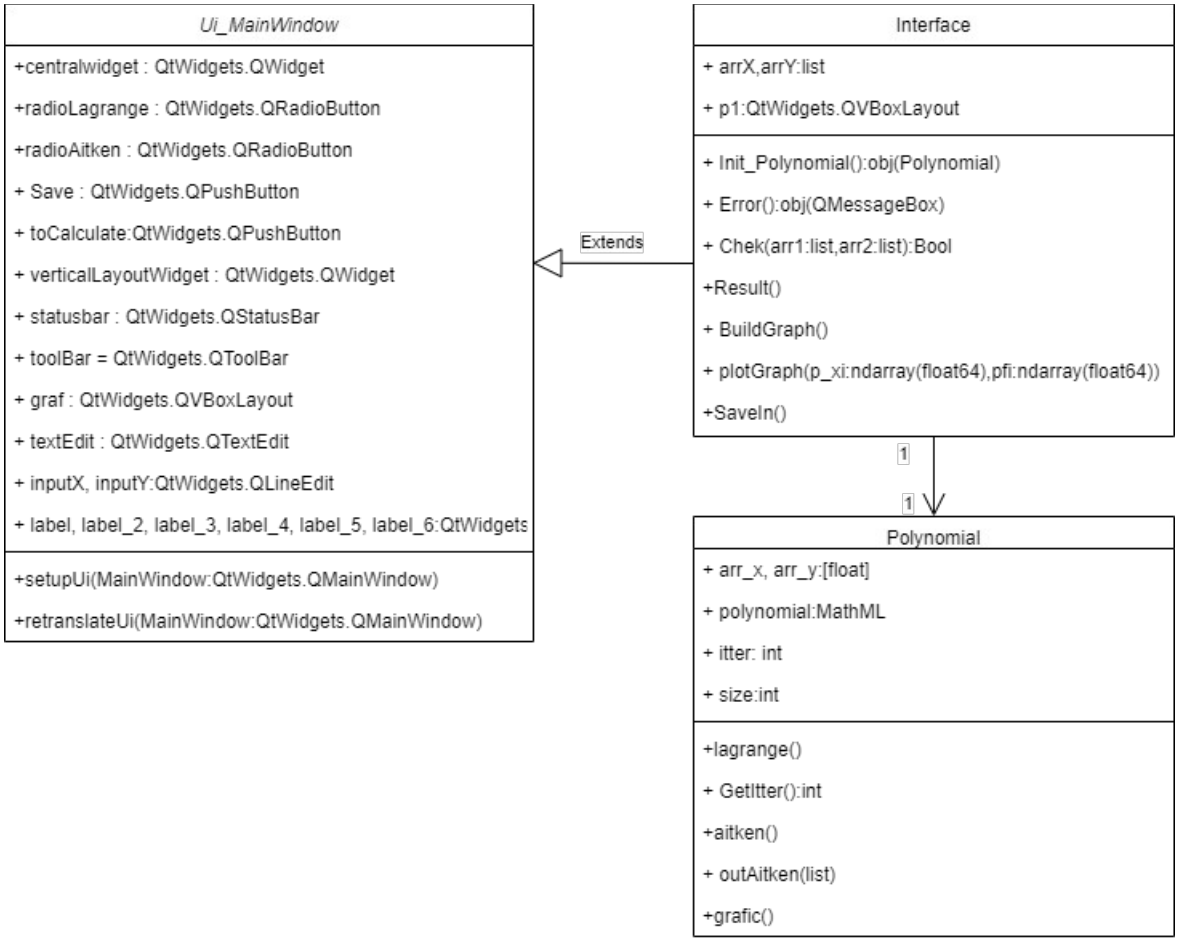


Рисунок 4.1 — Діаграма класів

4.2. Опис методів частин програмного забезпечення

4.3. Стандартні методи

У таблиці 4.1 наведено стандартні методи, використані при розробці програмного забезпечення.

Таблиця 4.1 – Стандартні методи

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
1	PyQt5	setObjectName	Встановлює ім’я вікну	string	-
2	sys	exit	Дозволяє розробнику завжди коректно завершувати роботу програми	QApplication.exec_: int	-
3	PyQt5	obj.exec_	Виконує код об’єкта	-	0:int
4	PyQt5	QMainWindow.resize	Встановити розмір головного вікна	w:int,h:int	-
5	PyQt5	obj.setStyleSheet	Встановити об’єкту зовнішній вигляд	args	-
6	PyQt5	obj.setGeometry	Встановити розміщення об’єкта відносно лівого верхнього кутка головного вікна, ширину та висоту	x:double, y:double, width:double, height:double	-

Продовження таблиці 4.1

7	PyQt5	Obj.setContentsMargins	Встановлює поля для використання навколо макета	(int,int,int,int)	-
8	PyQt5	QMainWindow.setCentralWidget	Встановлення віджета в якості центрального віджета вікна	widget:QWidget	-
9	PyQt5	setStatusBar	Встановлює рядок стану для головного вікна у рядок стану	QStatusBar	-
10	PyQt5	setReadOnly	Зробити об’єкт PyQt доступним/недоступним до зміни	flag:bool	-
11	PyQt5	addToolBar	Створює меню вікна	Qt.TopToolBarArea,QToolBar	-
12	PyQt5	setMaxLength	Встановлює максимальну довжину поля вводу	Arg:int	-
13	PyQt5	QPushButton.clicked.connect	Назначити функцію, що виконуватиметься при натиску кнопки	arg:function	-

Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
14	PyQt5	setWindowTitle	Змінює назву вікна	widget:QWidget	-
15	PyQt5	obj.setText	Встановити для об’єкта PyQt заданий текст	text:string	-
16	PyQt5	QMessageBox.setWindowTitle	Встановити назву випливаючого вікна з повідомленням	text:string	-
17	PyQt5	QMessageBox.setIcon	Встановити тип іконки для випливаючого вікна з повідомленням	Icon:QMessageBox.icon	-
18	PyQt5	QMessageBox.setInformativeText	Встановити додатковий текст для випливаючого вікна з повідмоленням	text:string	-
19	PyQt5	Obj.setStandardButtons	Встановити кнопку в випливваючому вікні	text:string	-

Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
20	PyQt5	Obj.text Obj.ToPlaneText	Повертає текст об’єкта PyQt	-	text:string
21	PyQt5	textEdit.clear()	Очищає вікно результату	-	-
22	PyQt5	Obj,isChecked()	Перевіряє стан радіокнопки	-	Flag:bool
23	pyqtgraph	setConfigOptions	Встановлення змінної pyqtgraph	Flag:bool	-
24	PyQt5	Obj.addWidget	Додає віджет в макет вікна	Obj pyqtgraph	-
25	pyqtgraph	pg.GraphicsLayoutWidget	Макет для управління макетом інтерфейса даних	-	GraphicsLayotWidgets
25	pyqtgraph	addPlot	Додає вікно графіка у віджет	GraphicsLayotWidgets	-
26	pyqtgraph	pg.setLabel	Зміна назви осі графіка	Str, text= str	
27	pyqtgraph	pg.showGrid	Встановлює видимість сітки графіку	flag:bool,flag:bool	-

Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
28	pyqtgraph	pg.mkPen	Встановлює характеристики маркера для малювання	color=str,width=float	-
29	pyqtgraph	Obj.plot	Малювання графіку	List,list,pen,symbol	-
30	list	len	Повертає кількість елементів у списку	List:list	size:int
31	Python	open	Відкрити файл	filename:string, mode:string	-
32	str	join	З’єднати в рядок список, кортеж або словник рядків, ставлячи між ними вказані символи	*args:iterable	line:string
33	Python	map	Привести всі елементи списку або кортежа до одного значення	type:type, *args:iterable	result:iterable
34	Python	filename.write	Вивести текст в файл filename	text:string	-

Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
35	SymPy	Symbol	Присвоює змінній символ для обчислення функцій	string	obj
36	SymPy	expand	З'єднує частини символічної функції	function	function
37	SymPy	lambdify	Переводить вираз SymPy в функцію Python	Symbol,function	string(lambda-function)
38	NumPy	linspace	Повертаєрівномірно розподілені числа в заданому інтервалі	Start:float,stop:float,num:int	list
39	Python	max	Визначає максимальне значення і списку	list	float
40	Python	min	Визначає мінімальне значення і списку	list	float
41	Python	str	Перетворює змінну та текст	int/float...	string
42	Python	all	Перевіряє чи всі елементи однакові	list	Flaf:bool

4.4. Користувацькі методи

У таблиці 4.2 наведено користувацькі методи, використані при розробці програмного забезпечення.

Таблиця 4.2 – Користувацькі методи

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
1	Ui_MainWindow	setupUi	Визначити основні об’єкти на головному вікні інтерфейсу	MainWindow: QMainWindow	-
2	Ui_MainWindow	retranslateUi	Назначити текст основним об’єктам інтерфейсу	MainWindow: QMainWindow	-
3	Polynomial	__init__	Конструктор з параметрами	List,list	-
4	Polynomial	lagrange	Метод Лагранжа	-	SymPy function
5	Polynomial	aitken	Схема Ейткена	-	SymPy function
6	Polynomial	grafic	Підготовка потрібних списків для побудови графіку	-	List,list

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
7	Interface	__init__	Конструктор без параметрів	-	-
8	Interface	Init_Polynomial	Створення об’єкту Polynomial за введеними даними	-	Polynomial
9	Interface	Error	Створення і налаштування випливаючих вікон	-	QMessageBox
10	Interface	Check	Перевірка введених координат на правильність	List,list	Flag:bool
11	Interface	Result	Функція що викликається принатисненні відповідної кнопки. Вивід результату	-	-
12	Interface	buildGraph	Створення та налаштування вікна для графіку	-	pyqtgraph
13	Interface	plotGraph	Побудова графіка за переданими масивами точкок	List,list	-
14	Interface	SaveIn	Збереження вхідних координат та результату в файл "polynom.txt"	-	-
15	Polynomial	outAitken	Вивід у консоль матриці схеми Ейткена	matrix(list)	-
16	Polynomial	GetItter	Повертає змінну-атрибут класу - itter	-	int

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 План тестування

Для подальшого проведення тестування розробимо план, за яким буде протестовано основний функціонал нашого програмного забезпечення, а також стресостійкість програми до введення у неї синтаксично та(або) логічно невірних даних.

a) Тестування правильності введених значень.
   1. Тестування при введенні нечисельних виразів в поля вводу координат.
   2. Тестування при введенні не однакової кількості координат X та У.
   3. Тестування при введенні точок кількість яких перевищує допустиме значення.(<2 або >10)
   4. Тестування при введенні координат X що повторюються.

b) Тестування коректності роботи методів розв’язку.
   1. Перевірка коректності роботи методу Лагранжа.
   2. Перевірка коректності роботи схеми Ейткена.

c) Тестування коректності роботи з файлами.
   1. Тестування коректності запису вхідних даних та результату у файл.

d) Тестування побудови графіків.

5.2 Приклади тестування

Проведемо тестування за допомогою введення в програму даних, необхідних для перевірки кожного пункту плану тестування. Кожному пункту плану виділимо окрему таблицю (таблиці 5.1 – 5.7).

Таблиця 5.1 – Приклад роботи програми при введенні нечисельних виразів в поле вводу координат точок

Мета тесту	Перевірити можливість введення некоректних даних.
Початковий стан програми	Відкрите вікно програми.
Вхідні дані	X: -0.2, 3, 1.4kl; У: 3 1 1
Схема проведення тесту	Введення невірних даних у поле вводу координат точок.
Очікуваний результат	Повідомлення про помилку формату даних.
Стан програми після проведення випробувань	Видано помилку «Некоректно введені дані».

Таблиця 5.2 – Приклад роботи програми при введенні не однакової кількості координат X та У

Мета тесту	Перевірити можливість введення некоректних даних.
Початковий стан програми	Відкрите вікно програми.
Вхідні дані	X: -6, 0, 5; У:7, 3.5
Схема проведення тесту	Введення не однакової кількості координат X та У.
Очікуваний результат	Повідомлення про логічну помилку
Стан програми після проведення випробувань	Видано помилку «Кількість значень X та У не співпадають або менше 2х».

Таблиця 5.3 – Приклад роботи програми при введенні точок кількість яких перевищує допустиме значення(<2 або >10)

Мета тесту	Перевірити можливість введення некоректних даних.
Початковий стан програми	Відкрите вікно програми.
Вхідні дані	a) X: -6 0 5; У:3.5 b) X:1 2 3 4 5 6 7 8 9 10 11; У:2 3 3 4 5 6 8 9 9 10 11
Схема проведення тесту	Введення більше 10ти або менше 2х точок.
Очікуваний результат	Повідомлення про логічну помилку
Стан програми після проведення випробувань	Видано помилку a) «Кількість значень X та У не співпадають або менше 2х». b) «Кількість значень більше 10»

Таблиця 5.4 – Приклад роботи програми при введенні координат X що повторюються

Мета тесту	Перевірити можливість введення некоректних даних.
Початковий стан програми	Відкрите вікно програми.
Вхідні дані	X:1 1 4 -0.4 У:3.3 -1 1 3
Схема проведення тесту	Введення координат X що дублюються.
Очікуваний результат	Повідомлення про логічну помилку
Стан програми після проведення випробувань	Видано помилку «Значення X не можуть повторюватись»

Таблиця 5.5 – Приклад коректності роботи методу Лагранжа

Мета тесту	Перевірити правильність роботи методів розв’язання задачі
Початковий стан програми	Відкрите вікно програми.
Вхідні дані	X: 1 0 -1 У: 3 1 1
Схема проведення тесту	Правильне заповнення полів координат і вибір методу Лагранжа для розв’язання задачі.
Очікуваний результат	Виведення поліному і його відображення на графіку.
Стан програми після проведення випробувань	Виведення поліному і його відображення на графіку.

Таблиця 5.6 – Приклад коректності роботи схеми Ейткена

Мета тесту	Перевірити правильність роботи методів розв’язання задачі
Початковий стан програми	Відкрите вікно програми.
Вхідні дані	X: 1 0 -1 У: 3 1 1
Схема проведення тесту	Правильне заповнення полів координат і вибір схеми Ейткена для розв’язання задачі.
Очікуваний результат	Виведення поліному і його відображення на графіку.
Стан програми після проведення випробувань	Виведення поліному і його відображення на графіку.

Таблиця 5.7 – Приклад роботи програми при записі вхідних даних та шуканого поліному у файл

Мета тесту	Перевірити коректність запису даних у файл
Початковий стан програми	Знайдений поліном.
Вхідні дані	X: 1 0 -1 У: 3 1 1 1.0*x**2 + 1.0*x + 1.0
Схема проведення тесту	Правильне заповнення полів координат, вибір і підтвердження алгоритму розв’язання, після розв’язання - натискання кнопки «Зберегти у файл» і перевірка вмісту файла “polynom.txt” в кореневому каталозі програми
Очікуваний результат	У файлі “polynom .txt” такий вміст: X: 1 0 -1 У: 3 1 1 polynomial is 1.0*x**2 + 1.0*x + 1.0
Стан програми після проведення випробувань	У файлі “polynom.txt” такий вміст: X: 1 0 -1 У: 3 1 1 polynomial is 1.0*x**2 + 1.0*x + 1.0

Таблиця 5.6 – Приклад роботи програми при зображенні графіка

Мета тесту	Перевірити коректність візуалізації задачі.
Початковий стан програми	Відкрите вікно програми.
Вхідні дані	X: 1 0 -1 У: 3 1 1 1.0*x**2 + 1.0*x + 1.0
Схема проведення тесту	Правильне заповнення полів координат, вибір і підтвердження алгоритму розв'язання та перевірка правильності зображеного графіку.
Очікуваний результат	Рисунок 5.1
Стан програми після проведення випробувань	Рисунок 5.2

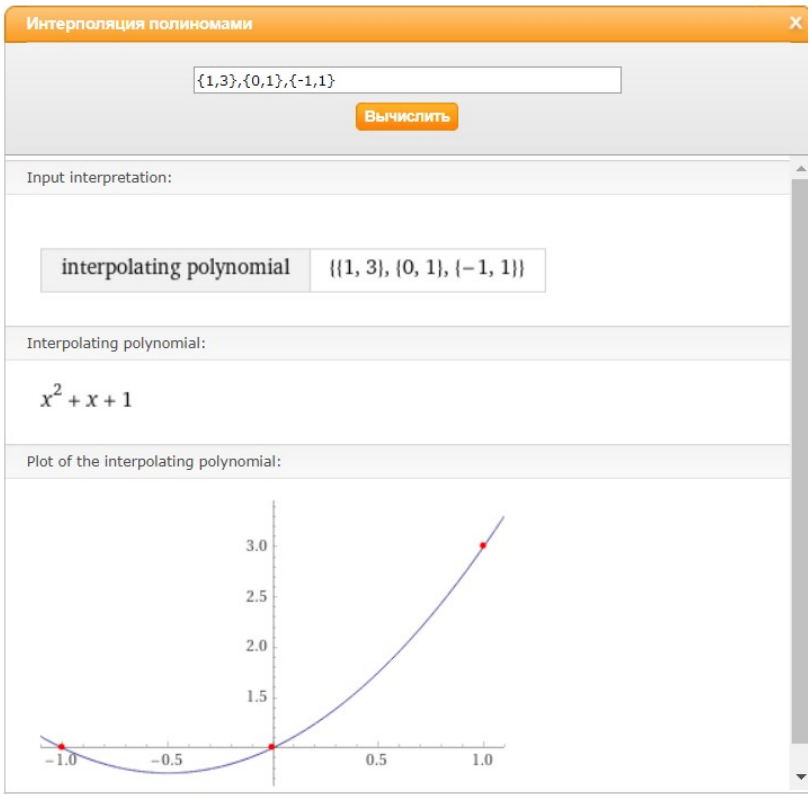


Рисунок 5.1 – Очікуваний результат побудови графіку, побудованого на основі шуканого поліному і введених даних

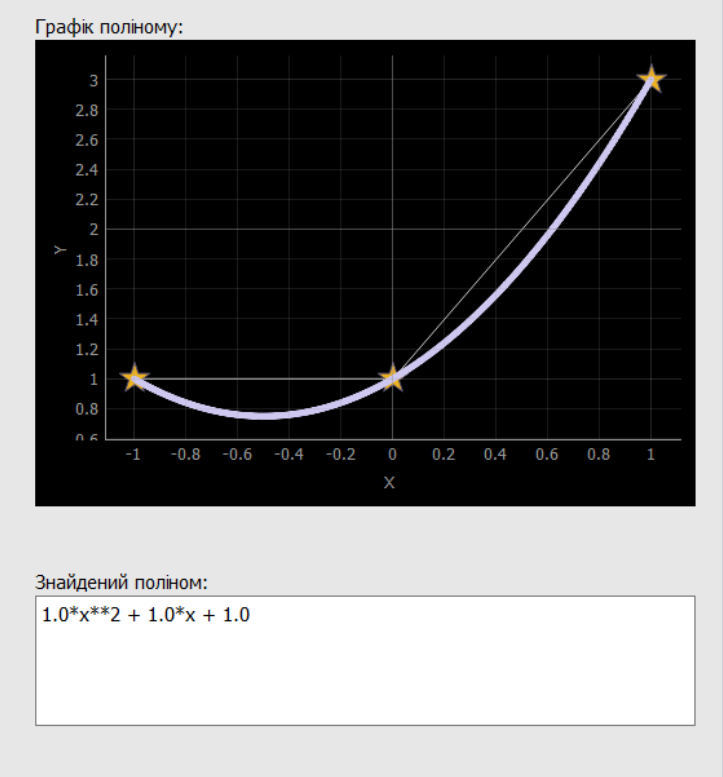


Рисунок 5.2 – Результат побудови графіку, побудованого на основі шуканого поліному і введених даних

6 ІНСТРУКЦІЯ КОРИСТУВАЧА

6.1 Робота з програмою

Після запуску виконавчого файлу з розширенням *.exe, відкривається головне вікно програми(Рисунок 6.1).

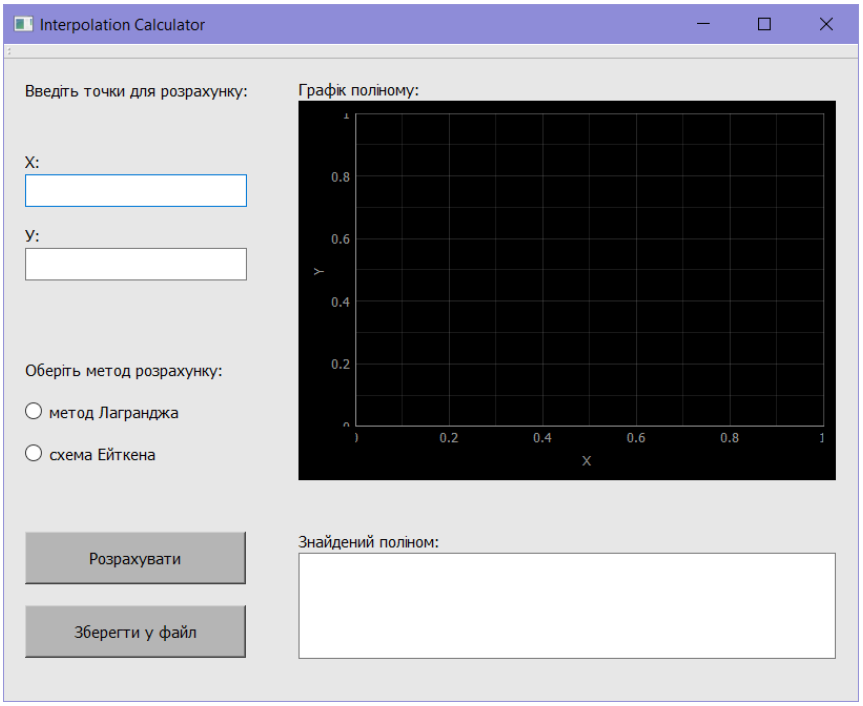


Рисунок 6.1 – Головне вікно програми

Далі за допомогою рядків вводу (Перший рядок для X координати, другий для У) необхідно ввести у кожний послідовність чисел через пробіл, що будуть оброблятися програмою як координати точок. Також радіо-кнопками обираємо метод розв’язку.(Рисунок 6.2)

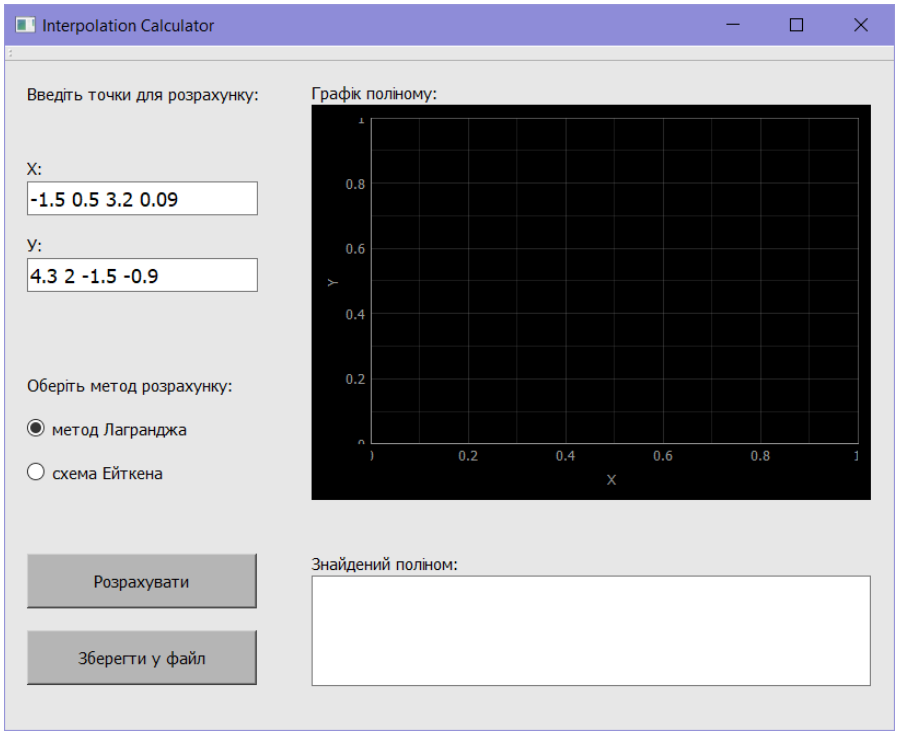


Рисунок 6.2 – Введення координат точок та вибір методу

При нажиманні на кнопку “Розрахувати” усі введені дані будуть перевірятися на коректність. Якщо вони не пройшли перевірку то з’явиться відповідне повідомлення. Приклади повідомлень містяться на наступних рисунках 6.3-6.6:

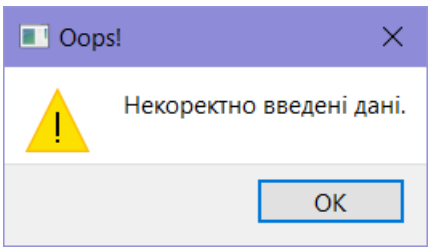
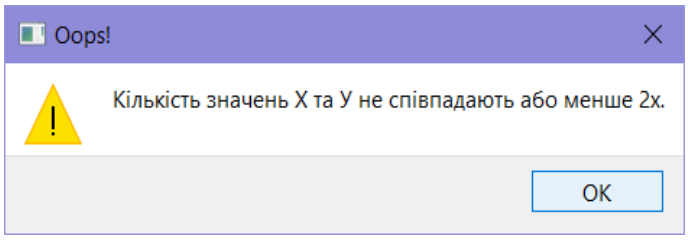


Рисунок 6.3 – Якщо були введені не потрібні знаки або символи



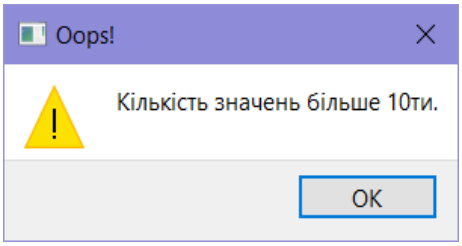


Рисунок 6.4 і 6.5– Обмеження кількості точок та перевірка: чи введені всі дані

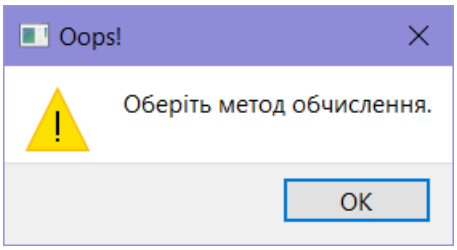


Рисунок 6.6 – Якщо жоден метод обчислення не був обраний

Якщо все правильно, то у правому нижньому вікні виведеться шуканий поліном, та буде показане зображення на графіку.(Рисунок 6.8)

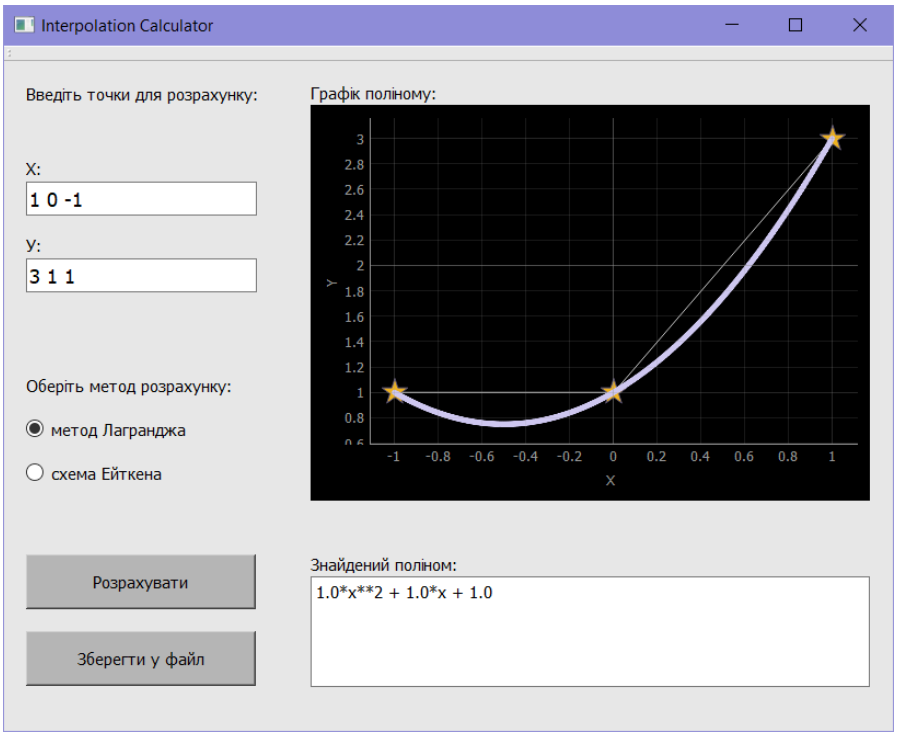


Рисунок 6.8 - Вивід результату

При виводі результату також випливає інформаційне вікно що містить число ітерацій, що відбулися у процесі програми. (Рисунок 6.9)

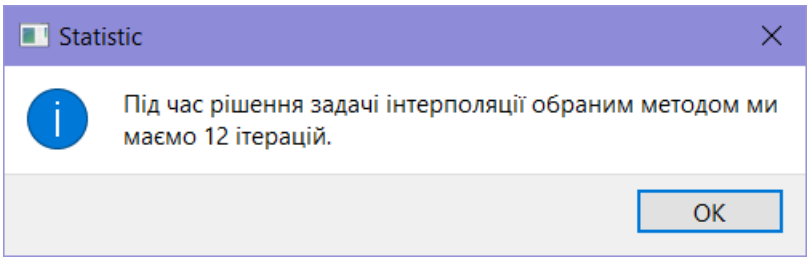


Рисунок 6.9 — Інформаційне вікно відображення статистики

Зберігання у файл “polynom.txt” відбувається за допомоги відповідної кнопки “Зберегти у файл”.

6.2 Формат вхідних та вихідних даних

Користувачем на вхід програми подається два масиви для представлення координат X та У, елементами яких є цілі числа та/або з плаваючою крапкою.

Результатом виконання програми є знайдений поліном та його графік.

6.3 Системні вимоги наведені в таблиці 6.1:

Таблиця 6.1 – Системні вимоги

	Мінімальні	Рекомендовані
Операційна система	Windows® XP/Windows Vista/Windows 7/ Windows 8/Windows 10 (з останніми оновленнями)	Windows 7/ Windows 8/Windows 10 (з останніми оновленнями)
Процесор	Intel® Pentium® III 1.0 GHz або AMD Athlon™ 1.0 GHz	Intel® Pentium® D або AMD Athlon™ 64 X2
Оперативна пам'ять	256 MB RAM (для Windows® XP) / 1 GB RAM (для Windows Vista/Windows 7/ Windows 8/Windows 10)	2 GB RAM
Відеоадаптер	Intel GMA 950 з відеопам'яттю об'ємом не менше 64 МБ (або сумісний аналог)	
Дисплей	800x600	1024x768 або краще
	Мінімальні	Рекомендовані
Прилади введення	Клавіатура, комп’ютерна миша	

Продовження таблиці 6.1

	Мінімальні	Рекомендовані
Додаткове програмне забезпечення	Бібліотека NumPy 1.22 або вище Бібліотека SymPy 1.6.2. або вище Бібліотека Pyqtgraph 0.12.4 або вище Набір модулів PyQt5 GPL v3	

7 АНАЛІЗ РЕЗУЛЬТАТІВ

Головною задачею курсової роботи була реалізація програми для знаходження поліному такими методами: Лагранжа, схемою Ейткена.

Критичні ситуації у роботі програми виявлені не були. Під час тестування було виявлено, що більшість помилок виникало тоді, коли користувачем вводилися нечислові вхідні дані або не був обраний метод рішення і відповідно, завдання неможливо розв’язати. Тому всі дані, які вводить користувач, ретельно перевіряються на валідність і лише потім подаються на обробку програмі.

Для перевірки та доведення достовірності результатів виконання програмного забезпечення скористаюся сервісом wolframalpha.

а) Метод Лагранжа

Результат виконання методу Лагранжа наведено на рисунках 7.1 і 7.2.

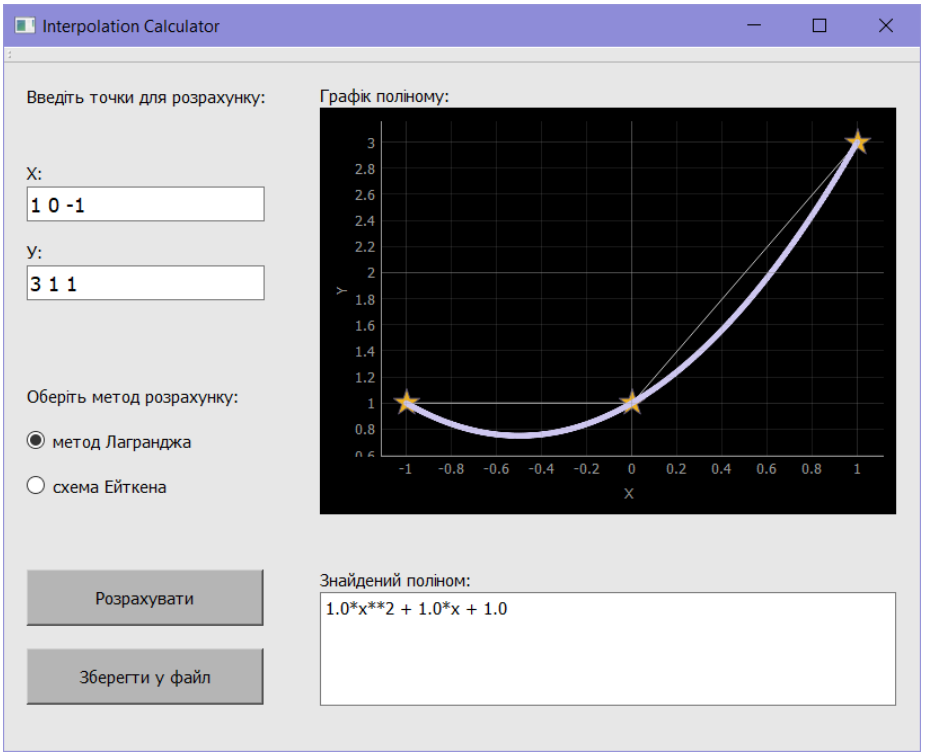


Рисунок 7.1 – Результат виконання методу Лагранжа програмою

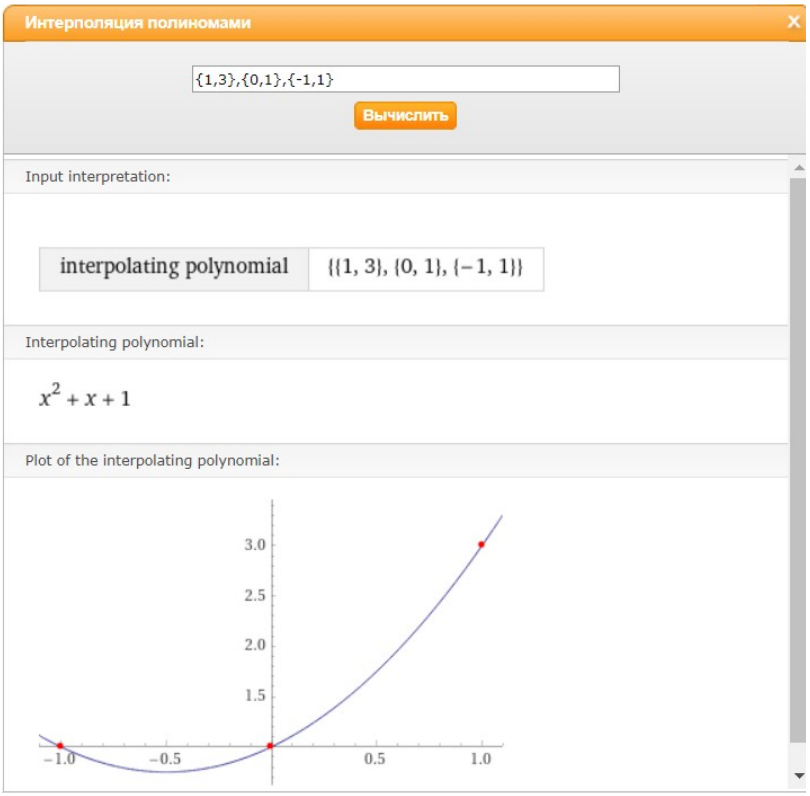


Рисунок 7.2 – Перевірка правильності знаходження поліному сервісом wolframalpha

Оскільки результат виконання збігається з результатом в wolframalpha (рисунок 7.2), то даний метод працює вірно.

б) Схема Ейткена

Результат виконання схеми Ейткена наведено на рисунку 7.3.

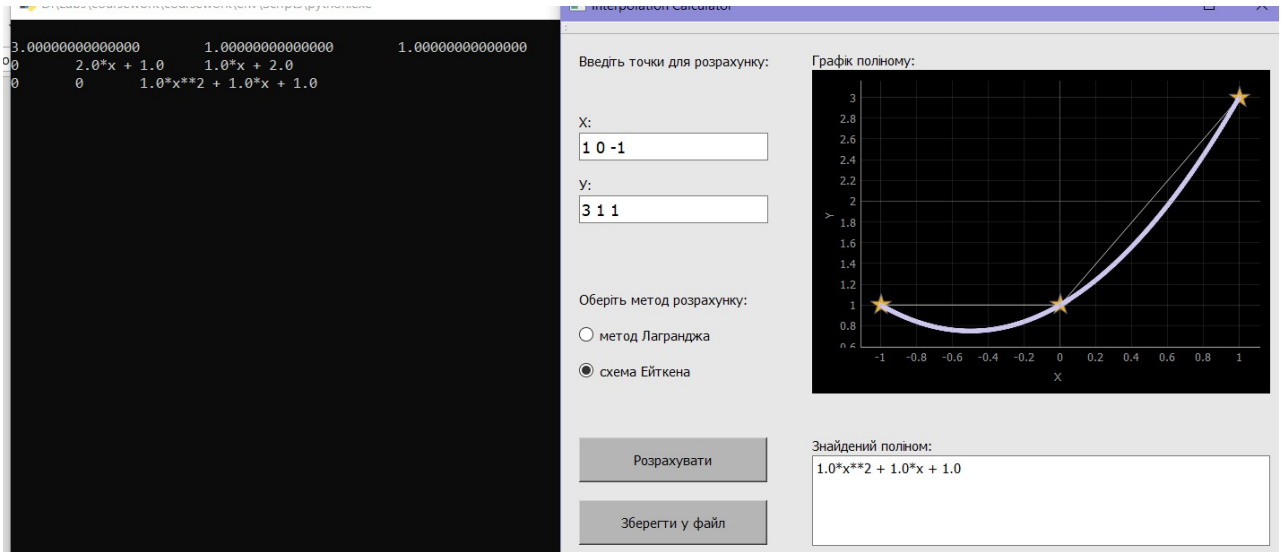


Рисунок 7.3 – Результат виконання схеми Ейткена програмою

Оскільки результат виконання збігається з результатом методу Лагранжа (рисунок 7.2), то даний метод працює вірно.

Для проведення тестування ефективності програми було створено два масиви випадкових координат X та У розміром по *n* елементів, тобто *n точок*.

Результати тестування ефективності алгоритмів розв’язання задач інтерполяції в таблиці 7.1:

Таблиця 7.1 – Тестування ефективності методів

Кількість точок	Параметри тестування	Метод	
		Лагранжа	Схема Ейткена
2	Кількість ітерацій	6	4
10	Кількість ітерацій	110	64
50	Кількість ітерацій	2550	1324
100	Кількість ітерацій	10100	5149
1000	Кількість ітерацій	1001000	501499
5000	Кількість ітерацій	25005000	12507499

Візуалізація результатів табилиці 7.1 наведено на рисунках 7.1 та 7.2:

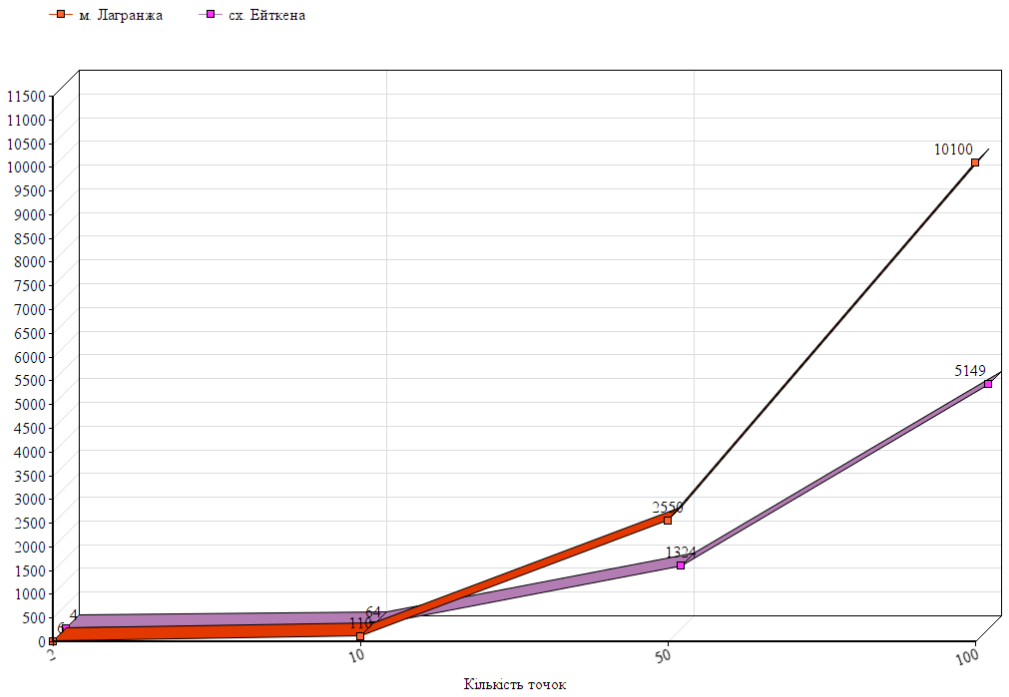


Рисунок 7.1 – Графік залежності кількості ітерацій методу від розміру вхідної системи

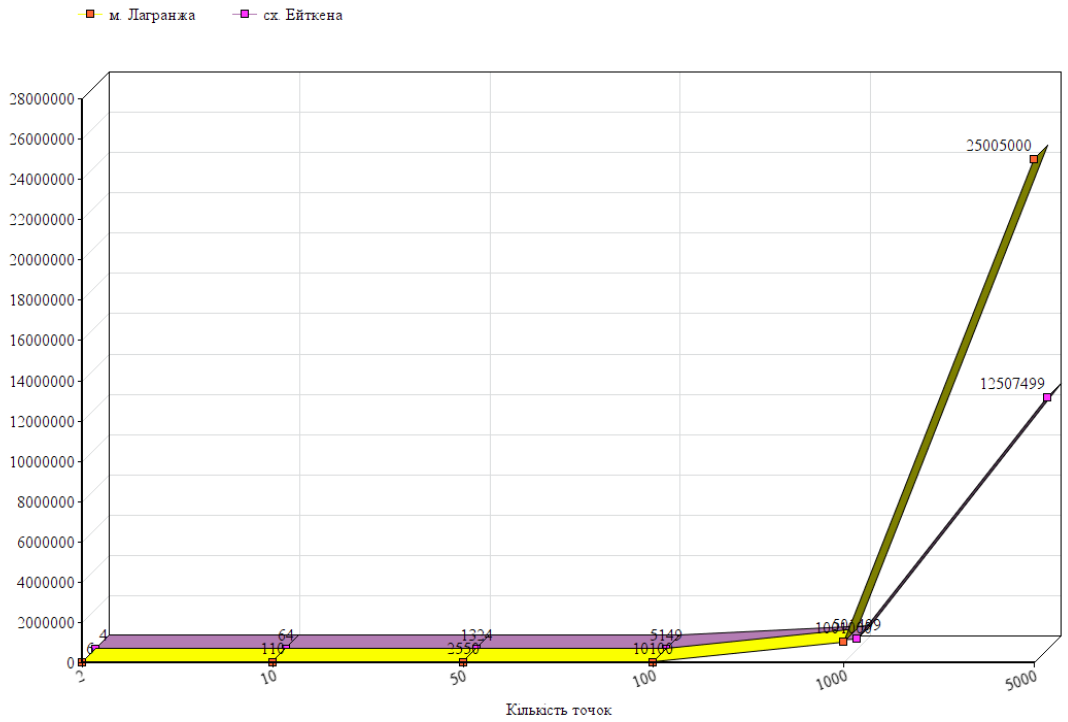


Рисунок 7.2 – Графік залежності кількості ітерацій методу від розміру вхідної системи

Нехай V – кількість точок, а отже кількість елементів в масивах координат. Тоді:

1. Обчислювальна складність методу Лагранжа – V^2.

Доведення: Найскладнішою частиною алгоритму є арифметичний цикл з одним вкладеним циклом. Число для позначення номеру елемента на початку – 0, на кожній ітерації циклу номер збільшується на 1, цикл завершується коли коли будуть пройдені всі елементи масиву,кількість яких дорівнює V, а отже виконується $V-1$ разів, має складність O(V). Всередині цього циклу – арифметичний цикл, який виконується також V разів, а отже асимптотична складність алгоритму - $O(V)*O(V)=O(VV)=O(V^2)$.

Нехай V – кількість точок, а отже кількість елементів в масивах координат, а також є розмірністю матриці.

2. Обчислювальна складність схеми Ейткена – V^2.

Доведення: Найскладнішою частиною алгоритму є цикл перебору рядів матриці в яку будуть записуватись поліноми різних степенів . Його виконання закінчується, коли буде пройдено $V-1$. Всередині цього циклу є цикл перебору стовпців матриці. На кожній його ітерації кількість компонент зменшується на $k+1$, де k- поточний номер ряду, тому цикл виконається $\frac{(V-1)^2}{2}$ разів. Асимптотична складність всього алгоритму - $O(V)*O((V-1)^2)=O(V^2)$

За результатами тестування можна зробити такі висновки:

а) Всі розглянуті методи дозволяють знаходити найкоротші остовні дерева в великих та надвеликих графах.

б) Обчислювальна складність всіх методів є приблизно однаковою.

в) Швидкість виконання обох методів залежить від кількості введених точок.

г) З розглянутих методів найоптимальнішим для практичного використання є схема Ейткена, оскільки вона виконується найшвидше.

ВИСНОВКИ

У процесі виконання курсової роботи я визначила поставлену переді мною задачу.

Після цього визначила теоретичні відомості з математичним описом понять задачі.

Далі я розробила алгоритми рішення задач інтерполяції двома методами, такими як метод Лагранжа та схема Ейткена, і програмне забезпечення мовою Python з використанням об’єктно-орієнтованого програмування, що реалізує знаходження потрібного результату. Також був створений графічний інтерфейс, на основі набору бібліотек PyQt5 і додатка QT Designer, для взаємодії користувача з елементами програми.

Щоб перевірити працездатність програми я створила тести для можливих варіантів роботи, за допомогою яких визначила що програма працює коректно у всіх випадках.

Також для полегшення роботи з програмою була створена інструкція користувача.

ПЕРЕЛІК ПОСИЛАНЬ

1. Вержбицький В.М. Численні методи (математичний аналіз та прості диференціальні рівняння). - М.: Вища школа, 2000.
2. Демидович Б.П., Марон І.А. Основи обчислювальної математики. - М.: Наука, 1970.
3. Березін І.С., Жідков Н.П. Методи обчислень, Т.1. М: ГІФМЛ, 1962. - с.88 :URL .
4. Гарднер, Ф. М. Інтерполяція в цифрових модемах – Частина II: Реалізація та продуктивність,е / F. M. Gardner // IEEE Trans. Commun. – 1993р., с. 998–1008.
5. Інтерполяційна схема Ейткена :URL .
6. Інтерполяційна схема Ейткена 2 :URL .
7. Використання інтерполяції Ньютона та методу Ейткена для розв’язування диференціального рівняння першого порядку :URL .
8. Лістунов С.Б .Дослідження методу рішення задач інтерполяції функції методом поліному лагранжа. С. 36 :URL .
9. Робота з бібліотекою Pyqtgraph :URL .
10. Формула Лагранжа :URL
11. Документація SymPy :URL
12. Документація NumPy :URL
13. Документація PyQT5: URL

Додаток А Технічне завдання

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра
інформатики та програмної інженерії

Затвердив

Керівник Головченко Максим Миколайович

«7» червня 2022 р.

Виконавець:

Студент Кондрацька Соня Леонідівна

«23» травня 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання курсової роботи

на тему: “Розв’язання задач інтерполяції”

з дисципліни:

«Основи програмування»

Київ 2022

1. *Мета*: Метою курсової роботи є розробка ефективного програмного забезпечення для розв’язку задач інтерполяції методом Лагранжа та схемою Ейткена

2. *Дата початку роботи*: «23» квітня 2022 р.

3. *Дата закінчення роботи*: «___»_____ 2022 р.

4. *Вимоги до програмного забезпечення*.

   a) Функціональні вимоги:

   – Можливість задавати систему координат від двох до десяти точок самостійно.

   – Можливість перевірки внесених даних на коректність.

   – Можливість бачити відповідне інформаційне повідомлення в разі некоректно введених даних або при не обраному методі.

   – Можливість обирати метод розв’язку задачі: метод Лагранжа або схему Ейткена.

   – Можливість знаходити шуканий поліном обраним методом.

   – Можливість бачити графічне розв’язання задачі на графіку.

   – Можливість зберігання розв’язку та вхідних координат в текстовий файл.

   b) Нефункціональні вимоги:

   – Можливість підтримки Windows 10 та вище.

   – Все програмне забезпечення та супроводжуюча технічна документація повинні задовольняти наступним ДЕСТам:

ГОСТ 29.401 - 78 - Текст програми. Вимоги до змісту та оформлення.

ГОСТ 19.106 - 78 - Вимоги до програмної документації.

ГОСТ 7.1 - 84 та ДСТУ 3008 - 2015 - Розробка технічної документації.

5. *Стадії та етапи розробки*:

1) Об'єктно-орієнтований аналіз предметної області задачі (до__.__.202_ р.)

2) Об'єктно-орієнтоване проектування архітектури програмної системи (до __.__.202_р.)

3) Розробка програмного забезпечення (до __.__.202_р.)

4) Тестування розробленої програми (до __.__.202_р.)

5) Розробка пояснювальної записки (до __.__.202_ р.).

6) Захист курсової роботи (до __.__.202_ р.).

6. *Порядок контролю та приймання*. Поточні результати роботи над КР регулярно демонструються викладачу. Своєчасність виконання основних етапів графіка підготовки роботи впливає на оцінку за КР відповідно до критеріїв оцінювання.

ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду програми вирішення задачі інтерполяції

(Найменування програми (документа))

Електронний носій

(Вид носія даних)

11 арк, 43 КБ

(Обсяг програми (документа), арк., Кб)

студента групи ІП-15 І курсу
Кондрацької С.Л.

1.coursework.py

```
from interface import *

if __name__ == "__main__":
    import sys
    app=QtWidgets.QApplication(sys.argv)
    myWin = Interface()
    myWin.show()
    sys.exit(app.exec_())
```

2.interface.py

```
from polynomialClass import *
from window import *
from pyqtgraph import PlotWidget, plot
class Interface(Ui_MainWindow,QtWidgets.QMainWindow):#Робота з
інтерфейсом
    def __init__(self):
        super(Interface, self).__init__()
        self.setupUi (self)
        self.p1= self.buildGraph()
        self.toCalculate.clicked.connect(self.Result)
        self.Save.clicked.connect(self.SaveIn)

    def Init_Polynomial(self):#Приймання програмою введених значень координат
        strX=self.inputX.text().split()
        strY=self.inputY.text().split()
        if self.Check(strX,strY):#Метод для перевірки значень
            self.arrX = list(map(float,strX))
            self.arrY = list(map(float,strY))
            return Polynomial(self.arrX,self.arrY)
```

```
    return 0
def Message(self):#Ф-ція визначення помилки
    Message=QMessageBox()
    Message.setWindowTitle("Oops!")
    Message.setText("Некоректно введені дані.")
    Message.setIcon(QMessageBox.Warning)
    Message.setStandardButtons(QMessageBox.Ok)
    return Message

def Check(self,arr1,arr2):#Метод для перевірки значень
    error=self.Message()
    for i in range(len(arr1)):
        if arr1[i].lstrip('-').replace('.','',1).isdigit()==False :
            error.exec_()
            return False
    for i in range(len(arr2)):
        if arr2[i].lstrip('-').replace('.','',1).isdigit()==False :
            error.exec_()
            return False
    if len(arr1)!=len(arr2) or len(arr1)<2:
        error.setText("Кількість значень X та У не співпадають або менше 2х.")
        error.exec_()
        return False
    elif len(arr1)>10:
        error.setText("Кількість значень більше 10ти.")
        error.exec_()
        return False
    for i in range(len(arr1)-1):
        for j in range(i+1,len(arr1)):
            if arr1[i]==arr1[j]:
```

```
                error.setText("Значення X не можуть повторюватись.")
                error.exec_()
                return False
        if all(i==arr2[0] for i in arr2):
            error.setText("Значення Y не можуть бути всі однакові.")
            error.exec_()
            return False
        return True

    def Result(self):#Ф-ція для виводу результату, що викликається при
нажиманні кнопки РОЗРАХУВАТИ
        if self.Init_Polynomial()!=0:
            self.textEdit.clear()
            func=self.Init_Polynomial()
            if self.radioLagrange.isChecked():#Перевірка на вибір методу
                polynom=func.lagrange()
            elif self.radioAitken.isChecked():
                polynom=func.aitken()
            else:
                error=self.Message()
                error.setText("Оберіть метод обчислення.")
                error.exec_()
                return
            self.textEdit.insertPlainText(str(polynom))#Вивід результату
            p_xi,pfi=func.grafic() #Виклик ф-ції з класу polynomialClass
            self.plotGraph(p_xi,pfi)
            itter=func.GetIter()
            statistic=self.Message()
            statistic.setWindowTitle("Statistic")
            statistic.setText("Під час рішення задачі інтерполяції обраним методом
```

```
ми маємо "+str(itter)+ " ітерацій.")
            statistic.setIcon(QMessageBox.Information)
            statistic.exec_()

    def buildGraph(self): #Створення і підготовка полотна для графіка
        pg.setConfigOptions (antialias = True) #створення змінної pyqtgraph, antialias
=згладжування кривої
        win = pg.GraphicsLayoutWidget () # макет pg для реалізації автоматичного
управління макетом інтерфейса даних
        self.graf.addWidget(win)
        p1 = win.addPlot () # Додаємо вікно графіка
        p1.setLabel ('left', text = 'Y')
        p1.showGrid (x = True, y = True)
        p1.setLogMode (x = False, y = False)
        p1.setLabel ('bottom', text = 'X')
        return p1

    def plotGraph(self,p_xi,pfi):#Малювання графіка з заданими значеннями
        pen = pg.mkPen(color=(206, 199, 240),width=5)
        self.p1.plot(self.arrX,self.arrY,symbolBrush=(237, 177,
32),symbol='star',symbolPen=(86, 80, 115), symbolSize=25 ,clear=True)
        self.p1.plot(p_xi,pfi,pen=pen)

    def SaveIn(self):
        with open("polynom.txt", 'w') as file:
            file.write("x:\n"+self.inputX.text())
            file.write("\n y:\n"+self.inputY.text())
            file.write("\npolynomial is "+self.textEdit.toPlainText())
```

3. polynomialClass.py

```
import sympy as sym
import numpy as np
import pyqtgraph as pg
class Polynomial:
    def GetIter(self):
        return self.itter
    def __init__(self,arrX=[],arrY=[]):
        self.arr_x=arrX
        self.arr_y=arrY
        self.size=len(self.arr_x)
    def lagrange(self):#Знаходження поліному методом Лагранжа
        point=sym.Symbol('x')#Використовуємо бібліотеку sympy для позначення
point як символ
        polyL = 0
        self.itter=0
        for i in range(self.size):
          p = 1
          self.itter+=1
          for j in range(self.size):
              self.itter+=1
              if i != j:
                  p = p * (point - self.arr_x[j]) / (self.arr_x[i] - self.arr_x[j]) #Формула
базисного поліному
          polyL = polyL + p * self.arr_y[i]
        self.polynomial=sym.expand(polyL) #Поєднання частин формули
        return self.polynomial
    def aitken(self): #Знаходження поліному схемою Ейткена
```

```
        point=sym.Symbol('x')
        self.itter=0
        p = [[0]*self.size for i in range(self.size)]
        for i in range(self.size):
            p[0][i]=self.arr_y[i]
            self.itter+=1
        for k in range(0,self.size-1):
            self.itter+=1
            for i in range(k+1,self.size):
                self.itter+=1
                p[k+1][i] =
((point-self.arr_x[k])*p[k][i]-(point-self.arr_x[i])*p[k][k])/(self.arr_x[i]-self.arr_x[k])
        self.outAitken(p)
        self.polynomial=sym.expand(p[self.size-1][self.size-1])
        return self.polynomial
    def grafic(self):#Метод знаходження меж графіка і функції для його побудови
        px=sym.lambdify(sym.Symbol('x'),self.polynomial)#Перетворення функції в
лямбда-функцію для швидкого обчислення числових значень
        a=min(self.arr_x)
        b=max(self.arr_x)
        p_xi=np.linspace(a,b,50)#Масив значень x в потрібних межах
        pfi=px(p_xi)#Масив значень функції в точках x
        return p_xi,pfi
    def outAitken(self,matrix):#Додаткова функція(не метод класу) для виведення
схеми Ейткена у вигляді таблиці
        print()
        for row in range(self.size):
            for elem in range(self.size):
                if row==0:
                  print(round(matrix[row][elem],3),end='\t')
```

```
            else:
              print(sym.expand(matrix[row][elem]),end='\t')
          print()
        print()
```

4.window.py

#вікно програми

створеної QT Designer

```
from PyQt5.QtWidgets import Qapplication,
QMainWindow,
QMessageBox,QWidget,QVBoxLayout
from PyQt5 import QtCore, QtGui, QtWidgets,uic


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(812, 625)
        MainWindow.setStyleSheet("\n"
"background-color: rgb(231, 231, 231);\n"
"")
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.radioLagrange = QtWidgets.QRadioButton(self.centralwidget)
        self.radioLagrange.setGeometry(QtCore.QRect(20, 320, 161, 31))
        self.radioLagrange.setStyleSheet("font: 9pt \"MS Shell Dlg 2\";")
        self.radioLagrange.setObjectName("radioLagrange")
```

```
        self.radioAitken = QtWidgets.QRadioButton(self.centralwidget)
        self.radioAitken.setGeometry(QtCore.QRect(20, 360, 161, 31))
        self.radioAitken.setStyleSheet("font: 9pt \"MS Shell Dlg 2\";")
        self.radioAitken.setObjectName("radioAitken")
        self.Save = QtWidgets.QPushButton(self.centralwidget)
        self.Save.setGeometry(QtCore.QRect(20, 520, 210, 50))
        self.Save.setStyleSheet("font: 9pt \"MS Shell Dlg 2\";\n"
"background-color: rgb(181, 181, 181);")
        self.Save.setObjectName("Save")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(20, 280, 201, 31))
        self.label.setStyleSheet("font: 9pt \"MS Shell Dlg 2\";")
        self.label.setObjectName("label")
        self.label_2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(280, 19, 150, 20))
        self.label_2.setStyleSheet("font: 9pt \"MS Shell Dlg 2\";")
        self.label_2.setObjectName("label_2")
        self.label_3 = QtWidgets.QLabel(self.centralwidget)
        self.label_3.setGeometry(QtCore.QRect(20, 20, 220, 20))
        self.label_3.setStyleSheet("font: 8pt \"MS Shell Dlg 2\";\n"
"font: 9pt \"MS Shell Dlg 2\";")
        self.label_3.setObjectName("label_3")
        self.label_4 = QtWidgets.QLabel(self.centralwidget)
        self.label_4.setGeometry(QtCore.QRect(20, 90, 55, 16))
        self.label_4.setStyleSheet("font: 9pt \"MS Shell Dlg 2\";")
        self.label_4.setObjectName("label_4")
        self.label_5 = QtWidgets.QLabel(self.centralwidget)
        self.label_5.setGeometry(QtCore.QRect(280, 449, 150, 20))
        self.label_5.setStyleSheet("font: 9pt \"MS Shell Dlg 2\";")
        self.label_5.setObjectName("label_5")
```

```
        self.toCalculate = QtWidgets.QPushButton(self.centralwidget)
        self.toCalculate.setGeometry(QtCore.QRect(20, 450, 210, 50))
        self.toCalculate.setStyleSheet("font: 9pt \"MS Shell Dlg 2\";\n"
"background-color: rgb(181, 181, 181);\n"
"")
        self.toCalculate.setObjectName("toCalculate")
        self.label_6 = QtWidgets.QLabel(self.centralwidget)
        self.label_6.setGeometry(QtCore.QRect(20, 160, 55, 16))
        self.label_6.setStyleSheet("font: 9pt \"MS Shell Dlg 2\";")
        self.label_6.setObjectName("label_6")
        self.inputX = QtWidgets.QLineEdit(self.centralwidget)
        self.inputX.setGeometry(QtCore.QRect(20, 110, 211, 31))
        self.inputX.setStyleSheet("font: 11pt \"MS Shell Dlg 2\";\n"
"background-color: rgb(255, 255, 255);")
        self.inputX.setObjectName("inputX")
        self.inputY = QtWidgets.QLineEdit(self.centralwidget)
        self.inputY.setGeometry(QtCore.QRect(20, 180, 211, 31))
        self.inputY.setStyleSheet("font: 11pt \"MS Shell Dlg 2\";\n"
"background-color: rgb(255, 255, 255);")
        self.inputY.setObjectName("inputY")
        self.textEdit = QtWidgets.QTextEdit(self.centralwidget)
        self.textEdit.setGeometry(QtCore.QRect(280, 470, 511, 101))
        self.textEdit.setStyleSheet("background-color: rgb(255, 255, 255);\n"
"font: 9pt \"MS Shell Dlg 2\";")
        self.textEdit.setObjectName("textEdit")
        self.verticalLayoutWidget = QtWidgets.QWidget(self.centralwidget)
        self.verticalLayoutWidget.setGeometry(QtCore.QRect(280, 40, 511, 361))
        self.verticalLayoutWidget.setObjectName("verticalLayoutWidget")
        self.graf = QtWidgets.QVBoxLayout(self.verticalLayoutWidget)
        self.graf.setContentsMargins(0, 0, 0, 0)
```

```
        self.graf.setObjectName("graf")
        MainWindow.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        self.toolBar = QtWidgets.QToolBar(MainWindow)
        self.toolBar.setObjectName("toolBar")
        MainWindow.addToolBar(QtCore.Qt.TopToolBarArea, self.toolBar)
        self.inputX.setMaxLength(60)
        self.inputY.setMaxLength(60)
        self.textEdit.setReadOnly(True)
        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Interpolation
Calculator"))
        self.radioLagrange.setText(_translate("MainWindow", "метод Лагранжа"))
        self.radioAitken.setText(_translate("MainWindow", "схема Ейткена"))
        self.Save.setText(_translate("MainWindow", "Зберегти у файл"))
        self.label.setText(_translate("MainWindow", "Оберіть метод розрахунку:"))
        self.label_2.setText(_translate("MainWindow", "Графік поліному:"))
        self.label_3.setText(_translate("MainWindow", "Введіть точки для
розрахунку:"))
        self.label_4.setText(_translate("MainWindow", "X:"))
        self.label_5.setText(_translate("MainWindow", "Знайдений поліном:"))
        self.toCalculate.setText(_translate("MainWindow", "Розрахувати"))
        self.label_6.setText(_translate("MainWindow", "У:"))
        self.toolBar.setWindowTitle(_translate("MainWin
```

```
dow", "toolBar"))
```