МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І.І. МЕЧНИКОВА ФАКУЛЬТЕТ МАТЕМАТИКИ, ФІЗИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА МАТЕМАТИЧНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

**Звіт**

**з Лабораторної роботи №1**

**«Програмування за допомогою засобів для відображення графічної інформації»**

**з дисципліни «Комп’ютерна графіка»**

студента 3 курсу спеціальності «Комп’ютерна інженерія» Келера Олексія Олексійовича

Викладач: Канд. фіз.-матем.наук, доцент Петрушина Тетяна Іванівна

м. Одеса – 2022 рік

Зміст

[1. Практичне завдання 3](#_Toc2081593915)

[1.1. Постановка задачі ......................................................................................................3](#_Toc1245320494)

[1.2 Опис програми ............................................................................................................ 3](#_Toc428364491)

[2. Код програми 5](#_Toc651131016)

[2.1 Клас MainWindow 5](#_Toc1703449579)

[2.2 Клас Defaults 8](#_Toc2064055780)

[2.3 Клас Plotter 9](#_Toc767451346)

[Висновок 11](#_Toc1680839739)

#### Практичне завдання

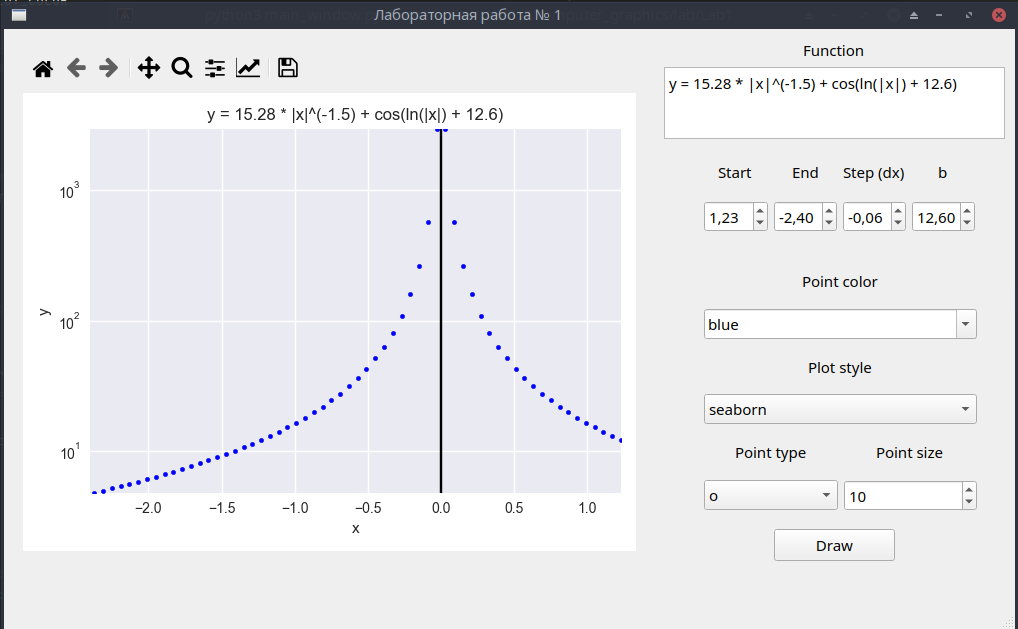
##### Постановка задачі Мета лабораторної роботи: вивчити можливості побудови графіків за допомогою мови програмування Python. Написати та налагодити програму побудови на екрані графіка заданої функції.

Варіант роботи № 16. Функція:



Мал. 1 Функція для варіанту № 16

##### 1.2 Опис програми У процесі виконання лабораторної роботи № 1 було написано програму з GUI (мал. 2) за допомогую мови программування Python.



Мал. 2 Приклад робочої програми

Написана програма дозволяє відобразити графік функції, зазначеної у варіанті. Також вона має певну гнучкість при використанні, а саме:

1. Можна вибрати початок, кінець та крок інтервалу. Чим менший крок, тим більше точок вигляду (x,y) відповідно і точність графіка. Також, оскільки в процесі написання коду для обчислення значень функції використовувалася бібліотека numpy, тому навіть велика кількість точок обчислювалася досить швидко, оскільки в цій бібліотеці підтримуються векторизовані операції, які тут і застосовуються.
2. Що стосується самого графіка функції - можна вибрати колір та розмір точок з яких він складається.

1. Також можна вибрати стиль графіка. Стиль відображає те, яким графік буде цілком, тобто, не тільки сама функція, але й задній фон, розмір шрифту, сітка на задньому фоні і т.д. За замовчуванням стоїть стиль seaborn. При натисканні на combo box з позначкою Plot style можна вибрати інший стиль.

#### Код програми

Програмний код поділений на класи з описом найважливіших моментів. Програму було написано мовою програмування Python 3 версії 3.10. У процесі написання прогріммі були використані наступні бібліотеки:

1. matplotlib 3.5.1
2. numpy 1.21.4
3. PyQt5 5.15.6

Дизайн програми був створений у QtDesigner.

##### 2.1 Клас MainWindow

# Підключаємо необхідні бібліотеки  
import numpy as np  
import sys  
import matplotlib  
  
from typing import Tuple  
from matplotlib import pyplot as plt  
# noinspection PyUnresolvedReferences  
from matplotlib.backends.backend\_qt5agg import FigureCanvas  
from matplotlib.backends.backend\_qt5agg import NavigationToolbar2QT as NavigationToolbar  
# noinspection PyUnresolvedReferences  
from PyQt5 import uic, QtWidgets  
  
from src.plotter import Plotter  
from src.default\_values import Defaults  
from src.function import function  
  
# Вказуємо шлях до головної форми  
uimain = Defaults.UI\_PATH  
form, base = uic.loadUiType(uifile=uimain)  
matplotlib.use("QT5Agg")  
  
  
class MainWindow(form, base):  
  
 def \_\_init\_\_(self):  
  
 super(base, self).\_\_init\_\_()  
 self.setupUi(self)  
  
 # вказуємо стиль, тип та колір графіка  
 self.plot\_point\_style = Defaults.DEFAULT\_POINT\_STYLE  
 self.plot\_type = Defaults.DEFAULT\_PLOT\_TYPE  
 self.plot\_color = Defaults.DEFAULT\_PLOT\_COLOR  
  
 # У наступних класах ми підключаємо до форми віджети  
 # та виставляємо значення за замовчуванням  
 self.\_\_setup\_default\_gui\_values()  
 self.\_\_connect\_canvas()  
 self.\_\_connect\_widgets()  
  
 # Вказуємо назву вікна основної форми  
 self.setWindowTitle(Defaults.WINDOW\_TITLE)  
  
 def \_\_connect\_canvas(self) -> None:  
  
 # створюємо область для малювання,  
 # місце на якому функція буде намальована  
 # і меню з інструментами  
 self.figure = plt.figure()  
 self.canvas = FigureCanvas(self.figure)  
 self.toolbar = NavigationToolbar(self.canvas, self)  
  
 lay = QtWidgets.QVBoxLayout(self.plot\_widget)  
 lay.addWidget(self.toolbar)  
 lay.addWidget(self.canvas)  
  
 def \_\_connect\_widgets(self) -> None:  
  
 # Приєднуємо віджети до їх обробників  
 self.plot\_graph\_button.clicked.connect(self.process\_draw\_button\_click)  
  
 self.line\_color\_combo\_box.activated[str].connect(self.\_\_set\_line\_color)  
 self.point\_types\_combobox.activated[str].connect(self.\_\_set\_plot\_point\_style)  
 self.plot\_style\_combo\_box.activated[str].connect(self.\_\_set\_plot\_style)  
  
 def \_\_setup\_default\_gui\_values(self) -> None:  
  
 # Виставляємо значення за замовчуванням для віджетів  
  
 # У текстовому полі вводимо функцію для варіанта №16  
 self.function\_body\_text\_edit.setPlainText(Defaults.FUNCTION\_NAME + '12.60)')  
  
 # Задаємо мінімальне, максимальне значення за замовчуванням для  
 # початку інтервалу  
 self.start\_spinbox.setMinimum(Defaults.MIN\_START\_SB\_VAL)  
 self.start\_spinbox.setMaximum(Defaults.MAX\_START\_SB\_VAL)  
 self.start\_spinbox.setValue(Defaults.DEFAULT\_START\_SB\_VAL)  
  
 # Задаємо мінімальне значення, максимальне значення,  
 # та значення за замовчуванням для кінця інтервалу  
 self.end\_spinbox.setMinimum(Defaults.MIN\_END\_SB\_VAL)  
 self.end\_spinbox.setMaximum(Defaults.MAX\_END\_SB\_VAL)  
 self.end\_spinbox.setValue(Defaults.DEFAULT\_END\_SB\_VAL)  
  
 # Задаємо мінімальне значення, максимальне значення  
 # та значення за замовчуванням  
 # для кроку графіка  
 self.step\_spinbox.setMinimum(Defaults.MIN\_STEP\_SB\_VAL)  
 self.step\_spinbox.setMaximum(Defaults.MAX\_STEP\_SB\_VAL)  
 self.step\_spinbox.setValue(Defaults.DEFAULT\_STEP\_SB\_VAL)  
  
 # Задаємо мінімальне значення, максимальне значення  
 # та значення за замовчуванням  
 # для коефіцієнту b  
 self.b\_value\_spinbox.setMinimum(Defaults.MIN\_B\_COEF\_SB\_VAL)  
 self.b\_value\_spinbox.setMaximum(Defaults.MAX\_B\_COEF\_SB\_VAL)  
 self.b\_value\_spinbox.setValue(Defaults.DEFAULT\_B\_COEF\_SB\_VAL)  
  
 self.point\_size\_spinbox.setMinimum(Defaults.MIN\_POINT\_SIZE)  
 self.point\_size\_spinbox.setMaximum(Defaults.MAX\_POINT\_SIZE)  
 self.point\_size\_spinbox.setValue(Defaults.DEFAULT\_POINT\_SIZE)  
  
 # Встановлення допустимих значень для кольорів функції  
 self.line\_color\_combo\_box.addItems(  
 Defaults.POINT\_COLORS  
 )  
  
 # Встановлення допустимих значень  
 # для вигляду графіка функції  
 self.point\_types\_combobox.addItems(  
 Defaults.POINT\_STYLES  
 )  
  
 # Встановлення допустимих значень для виду полотна  
 self.plot\_style\_combo\_box.addItems(  
 Defaults.PLOT\_STYLES  
 )  
  
 # Встановлюємо тип графіка вибраний у комбобоксі  
 self.plot\_style\_combo\_box.setCurrentText(  
 self.plot\_type  
 )  
  
 # Наступні три функції є оборотниками подій,  
 # що укладаються у:  
 # 1. Виборі кольору для графіка функціїю  
 # 2. Виду лінії функції.  
 # 3. Загального вигляду графіка.  
 def \_\_set\_line\_color(self, line\_color: str) -> None:  
 self.plot\_color = line\_color  
  
 def \_\_set\_plot\_point\_style(self, plot\_point\_style: str) -> None:  
 self.plot\_point\_style = plot\_point\_style  
  
 def \_\_set\_plot\_style(self, plot\_type: str) -> None:  
 self.plot\_type = plot\_type  
  
 def \_\_get\_func\_values(self) -> Tuple[np.ndarray, np.ndarray, float]:  
  
 # Тут ми отримуємо початок та кінець інтервалу для графіка,  
 # а також його крок.  
 start: float = self.start\_spinbox.value()  
 end: float = self.end\_spinbox.value()  
 step: float = self.step\_spinbox.value()  
 b\_coefficient: float = self.b\_value\_spinbox.value()  
  
 # Формуємо масиви, що відповідають координатам точок.  
 # Масив х створюємо за допомогою функції np.arange()  
 # в яку передаємо початок, кінець та крок інтервалу.  
 # Масив y створюємо шляхом передачі масиву х у функцію function  
 # яка є функцією з варіанту № 16.  
 x: np.ndarray = np.arange(start, end, step, dtype='float32')  
 y: np.ndarray = function(x, b\_coefficient)  
  
 return x, y, b\_coefficient  
  
 def \_\_perform\_plotting(self) -> None:  
  
 # Oтримуємо масиви з координатами х та у.  
 x, y, b = self.\_\_get\_func\_values()  
  
 # За допомогою класу Plotter та його методу plot\_function  
 # малюємо нашу функцію із зазначеними параметрами.  
 temp = Defaults.FUNCTION\_NAME + f'{b})'  
 self.function\_body\_text\_edit.setPlainText(temp)  
  
 Plotter.plot\_function(  
 self.canvas, self.figure,  
 x, y,  
 self.start\_spinbox.value(), self.end\_spinbox.value(),  
 self.plot\_point\_style, self.plot\_color,  
 temp, self.point\_size\_spinbox.value(),  
 Defaults.DEFAULT\_PLOT\_XLABEL, Defaults.DEFAULT\_PLOT\_YLABEL,  
 self.plot\_type  
 )  
  
 # Функція обробки натискання на кнопку Draw.  
 def process\_draw\_button\_click(self) -> None:  
 self.\_\_perform\_plotting()  
  
# Функція з варіанта №16.  
# Спочатку ми намагаємося обчислити значення для масиву у  
# за допомогою векторизованих операцій із бібліотеки numpy.  
# Якщо відбувається помилка в процесі обчислення,  
# її перехоплює блок except виводить відповідне повідомлення  
# та повертає None.  
def function(x: np.ndarray, b: float) -> np.ndarray | None:  
  
 try:  
 y = 15.28 \* np.power(np.abs(x), -1.5) + np.cos(np.log(np.abs(x)) + b)  
 return y  
  
 except Exception as e:  
 print(f'Error in function evaluation:\n{repr(e)}')  
 return None

if \_\_name\_\_ == '\_\_main\_\_':  
  
 app = QtWidgets.QApplication(sys.argv)  
 win = MainWindow()  
 win.show()  
 sys.exit(app.exec\_())

##### 2.2 Клас Defaults

import matplotlib.markers  
from typing import List  
from matplotlib import pyplot as plt  
  
  
# Клас для стандартних значень,  
# які виставляються при запуску програми.  
class Defaults:  
  
 UI\_PATH: str = 'ui/lab1\_main\_window.ui'  
  
 WINDOW\_TITLE: str = 'Лабораторная работа № 1'  
  
 FUNCTION\_NAME: str = 'y = 15.28 \* |x|^(-1.5) + cos(ln(|x|) + '  
  
 MIN\_START\_SB\_VAL: float = -100.0  
 MAX\_START\_SB\_VAL: float = 100.0  
 DEFAULT\_START\_SB\_VAL: float = 1.23  
  
 MIN\_END\_SB\_VAL: float = -100  
 MAX\_END\_SB\_VAL: float = 100  
 DEFAULT\_END\_SB\_VAL: float = -2.4  
  
 MIN\_STEP\_SB\_VAL: int = -10.0  
 MAX\_STEP\_SB\_VAL: int = 100.0  
 DEFAULT\_STEP\_SB\_VAL: int = -0.06  
  
 MIN\_B\_COEF\_SB\_VAL: float = -100.0  
 MAX\_B\_COEF\_SB\_VAL: float = 100.0  
 DEFAULT\_B\_COEF\_SB\_VAL: float = 12.6  
  
 MIN\_POINT\_SIZE = 1  
 MAX\_POINT\_SIZE = 100  
 DEFAULT\_POINT\_SIZE = 10  
  
 DEFAULT\_PLOT\_COLOR: str = 'blue'  
 DEFAULT\_PLOT\_TYPE: str = 'seaborn'  
 DEFAULT\_POINT\_STYLE: str = 'o'  
  
 DEFAULT\_PLOT\_TITLE: str = 'Function graph'  
 DEFAULT\_PLOT\_XLABEL: str = 'x'  
 DEFAULT\_PLOT\_YLABEL: str = 'y'  
  
 POINT\_COLORS: tuple = (  
 'blue', 'red', 'green', 'cyan',  
 'magenta', 'yellow', 'black', 'white'  
 )  
  
 POINT\_STYLES: List[str] = matplotlib.markers.MarkerStyle.filled\_markers  
 PLOT\_STYLES: List[str] = plt.style.available

##### 2.3 Клас Plotter

import numpy as np  
from matplotlib import pyplot as plt  
  
  
class Plotter:  
  
 @staticmethod  
 def plot\_function(canvas, figure,  
 x: np.ndarray, y: np.ndarray,  
 start: float, end: float,  
 plot\_point\_style: str, color\_: str,  
 label\_: str, point\_size: int,  
 xlabel\_: str, ylabel\_: str,  
 type\_: str) -> None:  
  
 # Установка типу графіка  
 plt.style.use(type\_)  
  
 # Якщо кількість елементів масивів x та y по-різному порушуємо помилку.  
 if x.shape != y.shape:  
 raise ValueError('x and y must have same shape and number of elements!')  
  
 # Очистка графіка  
 figure.clf()  
  
 # Малюємо координатні осі  
 plt.axhline(y=0, c='black')  
 plt.axvline(x=0, c='black')  
  
  
 # Малюємо функцію із заданими параметрами  
 crds = zip(x,y)  
  
 for coord\_x, coord\_y in crds:  
 plt.scatter(coord\_x, coord\_y, color=color\_, s=point\_size, marker=plot\_point\_style)  
  
 # Установка титульного значення та міток по осі X та Y та легенди  
 plt.title(label\_)  
 plt.xlabel(xlabel\_)  
 plt.ylabel(ylabel\_)  
 #plt.legend()  
  
 # Установка титульного значення та міток по осі X та Y та легенди  
 plt.xlim(end, start)  
 plt.ylim(min(y), max(y))  
  
 # Масштаб графіка  
 plt.yscale('symlog', linthresh=0.01)  
  
 # Вміщення графіка на полотно  
 plt.tight\_layout()  
  
 # Виведення графіка на полотні  
 canvas.draw()

# **Висновок**

У цій лабораторній роботі було створено додаток для малювання графіків функції програмування Python 3. У процесі написання програмного коду були використані наступні бібліотеки: matplotlib, numpy, PyQt5. Дизайн для програми був створений у QtDesigner.