Universitatea Tehnică a Moldovei

Facultatea Calculatoare Informatică şi Microelectronică

Departamentul Ingineria Software și Automatică

**RAPORT**

Lucrarea de laborator nr. 7

# La disciplina „Programarea Declarativa”

Tema: "Calcule matematice cu ajutorul librăriilor   
Numpy, SymPy și SciPy”

A efectuat: st. gr. SI-211 A. Chihai

A verificat: V. Rusu

Chișinău – 2023

**Mersul lucrarii:**

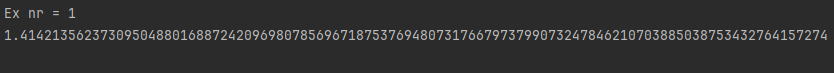
**Importurile pentru program**

import numpy as np  
import matplotlib.pyplot as plt  
from scipy import ndimage, integrate, optimize  
from skimage import io, util, color, restoration  
from scipy.ndimage import gaussian\_filter  
from sympy import (  
 symbols, expand, sin, cos, simplify,  
 limit, log, diff, Eq, solve, Not, Or, And, satisfiable,  
 Function, dsolve  
)  
from fractions import Fraction  
from numpy.lib.scimath import sqrt as scisqrt  
from decimal import Decimal, getcontext  
from scipy.ndimage import median\_filter

**Exerciții:**

**1. Calculați √2 cu 100 de zecimale.**

def ex1():  
 #setam numaru de cifre  
 getcontext().prec = 102  
 #functia pentru calcularea sqrt(2)  
 radacina\_doua = Decimal(2).sqrt()  
 print(radacina\_doua)

****

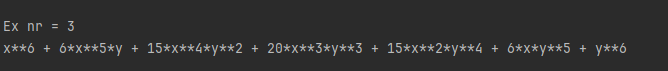
**2. Calculați 1/2 + 1/3 în aritmetica rațională.**

def ex2():  
 #declararea fractiilor  
 frac1 = Fraction(1, 2)  
 frac2 = Fraction(1, 3)  
 suma\_frac = frac1 + frac2  
 print(suma\_frac)

****

**3. Calculați forma extinsă a expresiei (𝑥 + 𝑦)6.**

def ex3():  
 #definirea variabilelor  
 x, y = symbols('x y')  
  
 #definirea expresiei  
 expresie = (x + y) \*\* 6  
  
 #rezultatul extins  
 rez\_extins = expand(expresie)  
 print(rez\_extins)

****

**4. Simplificați expresia trigonometrică sin(𝑥)/cos(𝑥).**

def ex4():  
 #definim variabila x  
 x = symbols('x')  
  
 #expresia trigonometrică  
 expresie\_trigonometrica = sin(x) / cos(x)  
  
 #simplificarea expresiei  
 expresie\_simplificata = simplify(expresie\_trigonometrica)  
  
 print(expresie\_simplificata)

****

**5. Calculați**



def ex5():  
 #definim variabila x  
 x = symbols('x')  
  
 #expresia  
 expresie = sin(x) / x  
  
 rez = limit(expresie, x, 0)  
 print(rez)

****

**6. Calculați derivata pentru log(𝑥) pentru 𝑥.**

def ex6():  
 x = symbols('x')  
 expresie = log(x)  
 derivata = diff(expresie, x)  
 print(derivata)

****

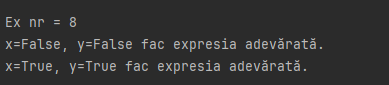
**7. Rezolvați sistemul de ecuații 2𝑥 + 3𝑦 = 5, 4𝑥 − 3𝑦 = −4.**

def ex7():  
 x, y = symbols('x y')  
 ecuatie1 = Eq(2 \* x + 3 \* y, 5)  
 ecuatie2 = Eq(4 \* x - 3 \* y, -4)  
 sol = solve((ecuatie1, ecuatie2), (x, y))  
 print(sol)

****

**8. Există valori booleene x, y care fac expresia (𝑥∨¬𝑦)∧(𝑦∨¬𝑥) adevărată?**

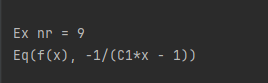
def ex8():  
 x, y = symbols('x y')  
  
 expresie = And(Or(x, Not(y)), Or(y, Not(x)))  
  
 sol = satisfiable(expresie, all\_models=True)  
  
 for model in sol:  
 print(f'x={model[x]}, y={model[y]} fac expresia adevărată.')

****

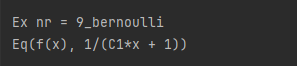
**9. Rezolvați ecuația diferențială a lui Bernoulli** 

**Rezolvați aceeași ecuație folosind hint='Bernoulli'. Ce observati?**

def ex9\_Simplu():  
 x = symbols('x')  
 f = Function('f')  
 ecuatie = x \* Derivative(f(x), x) + f(x) - f(x) \*\* 2  
 sol\_simpla = dsolve(ecuatie)  
 print(sol\_simpla

****

def ex9\_Bernoulli():  
 x = symbols('x')  
 f = Function('f')  
 ecuatie = x \* Derivative(f(x), x) + f(x) - f(x) \*\* 2  
 sol\_bernoulli = dsolve(ecuatie, hint='Bernoulli')  
 print(sol\_bernoulli)

****

**10. Folosind funcția quad() din librăria scipy, scrieți un program care rezolvă următoarea**

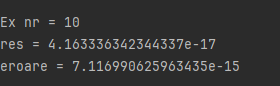
**integrală numerică:**



**De ce este important să avem o estimare a**

**preciziei (sau a erorii) integralei numerice?**

def ex10():  
 def functie(x):  
 return np.cos(2 \* np.pi \* x)  
  
 res, eroare = quad(functie, 0,1)  
 print(f"res = {res}")  
 print(f"eroare = {eroare}")

****

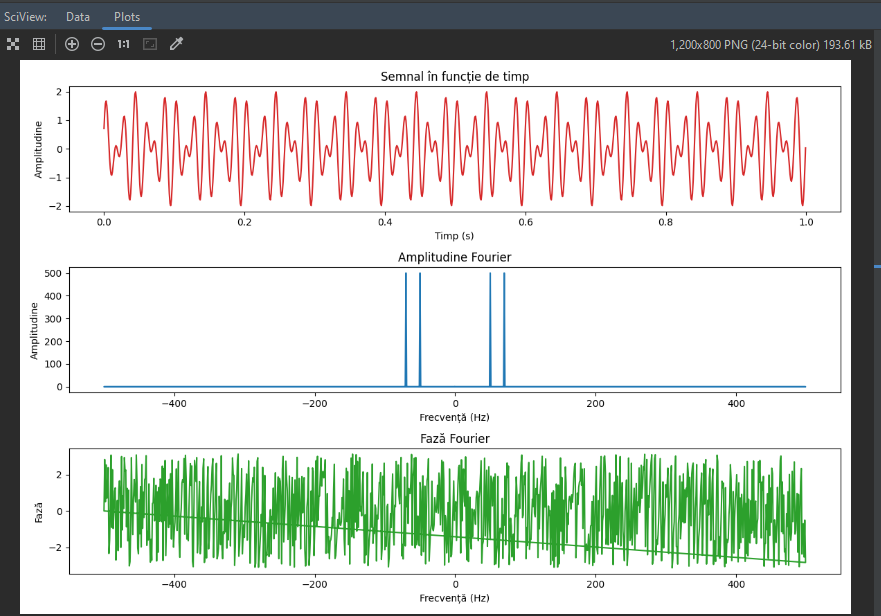
**11. Creați un semnal ca o suprapunere a unei unde sinusoidale de 50 Hz și 70 Hz (cu o ușoară**

**schimbare de fază între ele). Apoi transformați semnalul Fourier și trasați valoarea absolută**

**a coeficienților (complexi) discreți de transformare Fourier în funcție de frecvență**

**(observați vârfuri la 50Hz și 70Hz).**

import numpy as np  
import matplotlib.pyplot as plt  
  
# Setăm parametrii semnalului  
frecventa1 = 50 # Hz  
frecventa2 = 70 # Hz  
durata = 1.0 # secunde  
frecventa\_esantionare = 1000 # Hz  
faza\_relativa = np.pi / 4 # ușoară schimbare de fază  
  
# Generăm semnalul  
timp = np.linspace(0, durata, int(durata \* frecventa\_esantionare), endpoint=False)  
semnal = np.sin(2 \* np.pi \* frecventa1 \* timp) + np.sin(2 \* np.pi \* frecventa2 \* timp + faza\_relativa)  
  
# Efectuăm transformata Fourier  
transformata = np.fft.fft(semnal)  
frecvente = np.fft.fftfreq(len(transformata), 1 / frecventa\_esantionare)  
  
# Plotăm cele trei grafice separate  
plt.figure(figsize=(12, 8))  
  
# Semnal în funcție de timp  
plt.subplot(3, 1, 1)  
plt.plot(timp, semnal, color='tab:red')  
plt.title('Semnal în funcție de timp')  
plt.xlabel('Timp (s)')  
plt.ylabel('Amplitudine')  
  
# Amplitudinea transformatei Fourier în funcție de frecvență  
plt.subplot(3, 1, 2)  
plt.plot(frecvente, np.abs(transformata), color='tab:blue')  
plt.title('Amplitudine Fourier')  
plt.xlabel('Frecvență (Hz)')  
plt.ylabel('Amplitudine')  
  
# Faza transformatei Fourier în funcție de frecvență  
plt.subplot(3, 1, 3)  
plt.plot(frecvente, np.angle(transformata), color='tab:green')  
plt.title('Fază Fourier')  
plt.xlabel('Frecvență (Hz)')  
plt.ylabel('Fază')  
  
plt.tight\_layout()  
plt.show()

****

**12.Găsiți valoarea minimă a lui 𝑥 pentru optimizarea expresiei**



**Apelați funcția scipy.optimize.fmin care ia ca argument o funcție f pentru a minimiza și o**

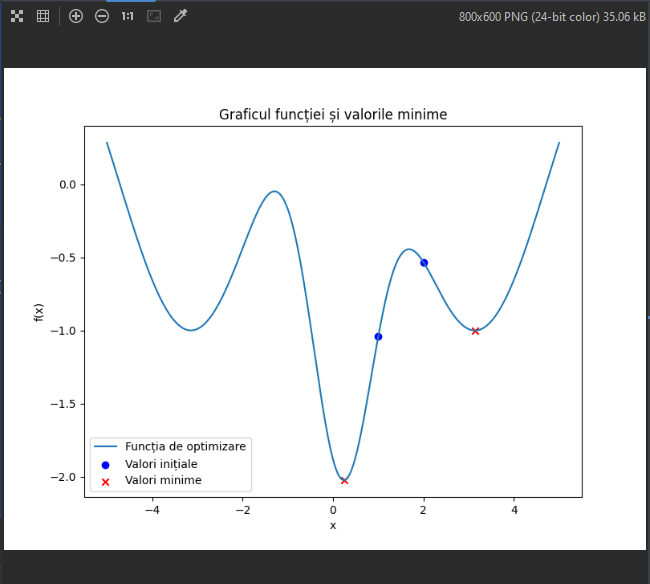
**valoare inițială x0 de la care să pornească căutarea pentru minim și care returnează valoarea**

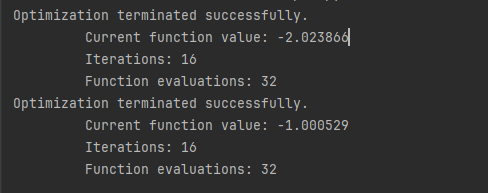
**lui x pentru care f(x) este (local) minimizat. Repetați căutarea minimului pentru două valori**

**(x0 = 1.0 și, respectiv, x0 = 2.0) pentru a demonstra că în funcție de valoarea de pornire**

**putem găsi minime diferite ale funcției f.**

# Definim funcția de optimizat  
import numpy as np  
from matplotlib import pyplot as plt  
from scipy.optimize import fmin  
  
  
  
def functie\_optimizare(x):  
 return np.cos(x) - 3 \* np.exp(-(x - 0.2)\*\*2)  
  
# Generăm un set de puncte pentru x  
x\_valori = np.linspace(-5, 5, 1000)  
  
# Calculăm valorile corespunzătoare funcției  
y\_valori = functie\_optimizare(x\_valori)  
  
# Găsim minimul pentru valoarea inițială x0 = 1.0  
x0\_1 = 1.0  
x\_min\_1 = fmin(functie\_optimizare, x0\_1)  
  
# Găsim minimul pentru valoarea inițială x0 = 2.0  
x0\_2 = 2.0  
x\_min\_2 = fmin(functie\_optimizare, x0\_2)  
  
# Trasăm graficul funcției și locurile valorilor minime  
plt.figure(figsize=(8, 6))  
plt.plot(x\_valori, y\_valori, label='Funcția de optimizare')  
plt.scatter([x0\_1, x0\_2], [functie\_optimizare(x0\_1), functie\_optimizare(x0\_2)],  
 color='blue', label='Valori inițiale', marker='o')  
plt.scatter([x\_min\_1, x\_min\_2], [functie\_optimizare(x\_min\_1), functie\_optimizare(x\_min\_2)],  
 color='red', label='Valori minime', marker='x')  
  
# Adăugăm etichete și titlu  
plt.xlabel('x')  
plt.ylabel('f(x)')  
plt.title('Graficul funcției și valorile minime')  
plt.legend()  
  
# Afișăm graficul  
plt.show()

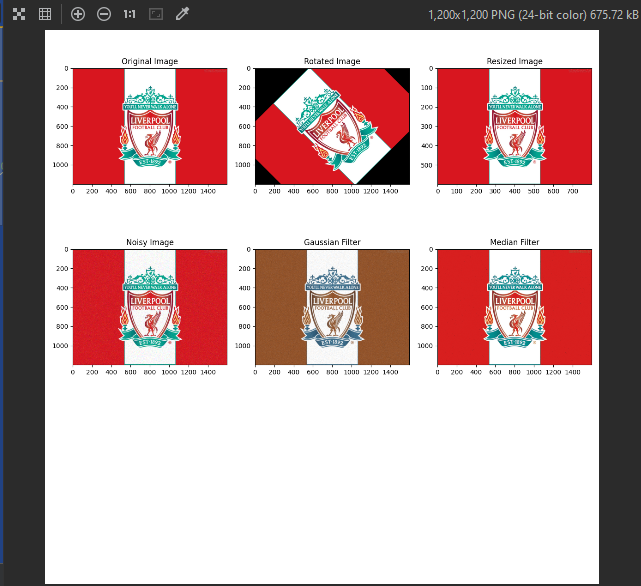
****

****

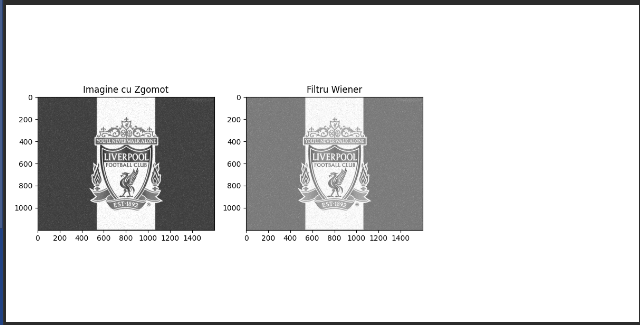
**13. Efectuați diverse manipulări asupra unei imagini : schimbați orientarea, rezoluția**

**(scipy.ndimage oferă manipularea tablourilor n-dimensionale ca imagini). Generați zgomot asupra imaginii, apoi folosiți pe rând filtrele Gaussian, median, Wiener. Observați eficacitatea fiecărui filtru**

import numpy as np  
import matplotlib.pyplot as plt  
from skimage import io, color, util, restoration  
from scipy import ndimage  
from scipy.ndimage import gaussian\_filter, median\_filter  
  
# Download an example image (replace with your own image if needed)  
image\_url = "https://external-content.duckduckgo.com/iu/?u=http%3A%2F%2Fwww.pixelstalk.net%2Fwp-content%2Fuploads%2F2016%2F07%2FLiverpool-HD-Picture.jpg&f=1&nofb=1&ipt=8b74c59ece6d23db798e3beb0bc6a79c33f90b6df005db0588fbf46e7c8b07e7&ipo=images"  
image = io.imread(image\_url)  
  
# Rotate the image  
image\_rotated = ndimage.rotate(image, angle=45, reshape=False)  
  
# Resize the image  
image\_resized = ndimage.zoom(image, zoom=(0.5, 0.5, 1))  
  
# Generate noise on the image  
image\_noisy = util.random\_noise(image, mode='s&p', rng=None)  
  
# Apply the Gaussian filter  
image\_gaussian = gaussian\_filter(image\_noisy, sigma=1)  
  
# Apply the Median filter  
image\_median = median\_filter(image\_noisy, size=3)  
  
# Convert the image to grayscale  
image\_gray = color.rgb2gray(image\_noisy)  
  
# Create a Gaussian PSF with the same number of channels as the input image  
psf = restoration.estimate\_sigma(image\_gray)  
  
  
# Display the images  
plt.figure(figsize=(12, 12))  
  
plt.subplot(3, 3, 1)  
plt.imshow(image)  
plt.title('Original Image')  
  
plt.subplot(3, 3, 2)  
plt.imshow(image\_rotated)  
plt.title('Rotated Image')  
  
plt.subplot(3, 3, 3)  
plt.imshow(image\_resized)  
plt.title('Resized Image')  
  
plt.subplot(3, 3, 4)  
plt.imshow(image\_noisy)  
plt.title('Noisy Image')  
  
plt.subplot(3, 3, 5)  
plt.imshow(image\_gaussian)  
plt.title('Gaussian Filter')  
  
plt.subplot(3, 3, 6)  
plt.imshow(image\_median)  
plt.title('Median Filter')  
  
  
plt.tight\_layout()  
plt.show()

****

import numpy as np  
import matplotlib.pyplot as plt  
from skimage import io, color, util, restoration  
  
# Descarcă o imagine de exemplu (înlocuiește cu propria imagine dacă este necesar)  
url\_imagine = "https://external-content.duckduckgo.com/iu/?u=http%3A%2F%2Fwww.pixelstalk.net%2Fwp-content%2Fuploads%2F2016%2F07%2FLiverpool-HD-Picture.jpg&f=1&nofb=1&ipt=8b74c59ece6d23db798e3beb0bc6a79c33f90b6df005db0588fbf46e7c8b07e7&ipo=images"  
imagine = io.imread(url\_imagine)  
  
# Converteste imaginea la tonuri de gri  
imagine\_gri = color.rgb2gray(imagine)  
  
# Generează zgomot pe imagine  
imagine\_cu\_zgomot = util.random\_noise(imagine\_gri, mode='s&p', rng=None)  
  
# Creează un PSF Gaussian simplu (2D)  
psf = np.ones((5, 5)) / 25 # Presupunând un filtru de medie simplu  
  
# Aplică filtrul Wiener  
imagine\_wiener = restoration.wiener(imagine\_cu\_zgomot, psf, balance=0.1)  
  
# Afișează imaginile  
plt.figure(figsize=(12, 6))  
  
plt.subplot(1, 3, 1)  
plt.imshow(imagine\_cu\_zgomot, cmap='gray')  
plt.title('Imagine cu Zgomot')  
  
plt.subplot(1, 3, 2)  
plt.imshow(imagine\_wiener, cmap='gray')  
plt.title('Filtru Wiener')  
  
plt.tight\_layout()  
plt.show()

****

**Concluzie:**

Prin aceste exerciții, am descoperit și folosit unelte puternice în Python pentru a rezolva probleme matematice și științifice. Aceste instrumente, precum NumPy, SciPy și SymPy, ne-au ajutat să facem calcule, să manipulăm date și să rezolvăm diverse tipuri de probleme.

Am acoperit o varietate de sarcini, de la calcule simple precum rădăcina pătrată a lui 2, până la probleme mai complexe, cum ar fi rezolvarea ecuațiilor sau prelucrarea imaginilor. Folosind aceste biblioteci, am înțeles cum să abordăm și să rezolvăm diverse probleme matematice și științifice într-un mod simplu și eficient.

Top of Form

Bottom of Form