Lab report 6 (Variant 2)

Student :

Herashchenko Serhii

Teacher:

Zhuravlova Zinaida Yuriyivna

25.11. 2024 p.

Task 1

1.1 Condition

Обчислити невласний інтеграл трьома способами.

$$\int_0^1 \ln x \, dx$$

1.2 Solution code

```
import numpy as np
from scipy.integrate import quad

def func(x):
    return x * np.log(x)

result_quad, error_quad = quad(func, 0, 1)

import sympy as sp
x = sp.symbols('x')
integral_sympy = sp.integrate(x * sp.ln(x), (x, 0, 1))

def transformed_func(t):
    return -np.exp(-2 * t) * t
result_transformed, error_transformed = quad(transformed_func, 0, np.inf)

print(f"quad: res = {result_quad}, error = {error_quad}")
print(f"sp.integrate: {integral_sympy.evalf()}")
print(f"quad(transformed_func): res = {result_transformed}, error = {error_transformed}")
```

1.3 Output

```
quad: res = -0.24999999999999992, error = 2.7755575615628904e-16
sp.integrate: -0.250000000000000
quad(transformed_func): res = -0.24999999999999986, error = 4.600476139877705e-10
```

Task 2

2.1 Condition

Обчислити подвійний інтеграл . Перевірити результат аналітично.

$$\int_{0}^{\frac{\pi}{2}} \int_{x-\frac{\pi}{3}}^{x+\frac{\pi}{3}} x \cos y \, dx dy$$

2.2 Solution code

```
import numpy as np
from scipy.integrate import dblquad
import sympy as sp

def integrand(y, x):
    return x * np.cos(y)

y_lower = lambda x: x - np.pi / 3
y_upper = lambda x: x + np.pi / 3

result_numeric, error_numeric = dblquad(integrand, 0, np.pi/2, y_lower, y_upper)

x, y = sp.symbols('x y')
integrand_symbolic = x * sp.cos(y)

inner_symbolic = sp.integrate(integrand_symbolic, (y, x - sp.pi/3, x + sp.pi/3))
analytic_integral = sp.integrate(inner_symbolic, (x, 0, sp.pi/2))

result_analytic = analytic_integral.evalf()

print(f"Numerical: {result_numeric}, error: {error_numeric}")
print(f"Analitic: {result_analytic}")
```

2.3 Output

```
Numerical: 0.9886482387824491, error: 1.7339490298217823e-14
Analitic: 0.988648238782449
```

Task 3

3.1 Condition

Знайти значення похідних першого і другого порядку функції $f(x) = \cos x$ в точці $x = \frac{\pi}{2}$ двома способами: за допомогою вбудованої функції та за різницевою формулою.

3.2 Solution code

```
import numpy as np
import sympy as sp
from scipy.misc import derivative

def f(x):
    return sp.cos(x)

x0 = sp.pi / 2

first_derivative = derivative(f, x0, dx=1e-6, n=1, order=3)
second_derivative = derivative(f, x0, dx=1e-6, n=2, order=5)

dx = sp.symbols('dx')

first_derivative_fd = sp.limit((f(x0 + dx) - f(x0)) / dx, dx, 0)

second_derivative_fd = sp.limit((f(x0 + dx) - 2 * f(x0) + f(x0 - dx)) / dx**2, dx, 0)

print(f"f' (scypy): {first_derivative}")
print(f"f'' (scypy): {second_derivative}")
print(f"f' (difference approximations): {first_derivative_fd}")
print(f"f'' (difference approximations): {second_derivative_fd}")
```

3.3 Output

```
f' (scypy): -0.999999999999833
f'' (scypy): -1.05879118406788E-10
f' (difference approximations): -1
f'' (difference approximations): 0
```

Task 4

4.1 Condition

Знайти розв'язок задачі Коші $\begin{cases} y''' + 3y'' + 8y - 2 = 0 \\ y(0) = 3, y'^{(0)} = -2, y''^{(0)} = 5 \end{cases}$. Побудувати графіки $y(x), y'(x), y''(x), x \in [0, 10]$.

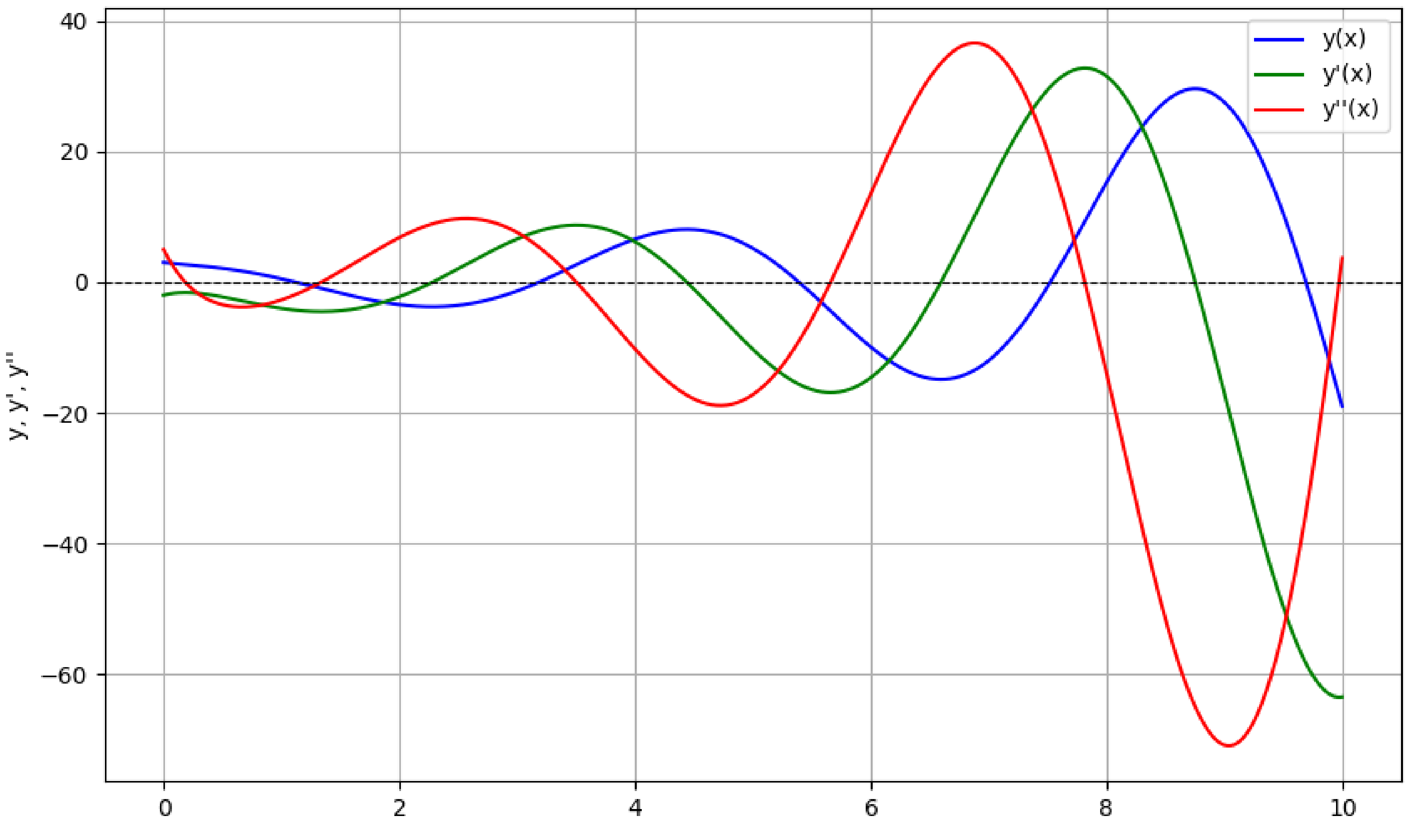
4.2 Solution code

```
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt
import sympy as sp

# ======================================

x = sp.symbols('x')
C1, C2, C3 = sp.symbols('C1 C2 C3')
y = sp.Function('y')

eq = sp.Eq(y(x).diff(x, 3) + 3 * y(x).diff(x, 2) + 8 * y(x) - 2, 0)

general_solution = sp.dsolve(eq)
print(f"General solution:\n{general_solution} \n")

y_general = general_solution.rhs

ics = {
    y(0): 3,
    y(x).diff(x).subs(x, 0): -2,
    y(x).diff(x, 2).subs(x, 0): 5,
}

constants = sp.solve([y_general.subs(x, 0) - 3,
                      y_general.diff(x).subs(x, 0) + 2,
                      y_general.diff(x, 2).subs(x, 0) - 5], [C1, C2, C3])

specific_solution = y_general.subs(constants)
print(f"Specific solution:\ny(x) = {specific_solution}")

# ======================================

def ode_system(t, y):
    dydt = [y[1],
            y[2],
            -3*y[2] - 8*y[0] + 2]
    return dydt

y0 = [3, -2, 5]

t_span = (0, 10)
t = np.linspace(0, 10, 500)
solution = solve_ivp(ode_system, t_span, y0, t_eval=t, method='RK45')

y = solution.y[0]
y_prime = solution.y[1]
y_double_prime = solution.y[2]

# ======================================

plt.figure(figsize=(10, 6))

plt.plot(t, y, label="y(x)", color='blue')

plt.plot(t, y_prime, label="y'(x)", color='green')
```

```
plt.plot(t, y_double_prime, label="y''(x)", color='red')

plt.xlabel("x")
plt.ylabel("y, y', y''")
plt.axhline(0, color='black', linewidth=0.8, linestyle='--')
plt.legend()
plt.grid()
plt.show()
```

4.3Output

```
General solution:
Eq(y(x), C1*exp(x*(-1 + 1/(2*(2*sqrt(6) + 5)**(1/3)) + (2*sqrt(6) +
5)**(1/3)/2))*sin(sqrt(3)*x*(-(2*sqrt(6) + 5)**(1/3) + (2*sqrt(6) + 5)**(-1/3))/2) +
C2*exp(x*(-1 + 1/(2*(2*sqrt(6) + 5)**(1/3)) + (2*sqrt(6) + 5)**(1/3)/2))*cos(sqrt(3)*x*(-
(2*sqrt(6) + 5)**(1/3) + (2*sqrt(6) + 5)**(-1/3))/2) + C3*exp(-x*((2*sqrt(6) + 5)**(-1/3)
+ 1 + (2*sqrt(6) + 5)**(1/3))) + 1/4)

Specific solution:
y(x) = (-7*sqrt(3)*(2*sqrt(6) + 5)**(2/3)/24 - sqrt(2)*(2*sqrt(6) + 5)**(1/3)/12 +
sqrt(3)*(2*sqrt(6) + 5)**(1/3)/8 + sqrt(2)*(2*sqrt(6) + 5)**(2/3)/3)*exp(x*(-1 +
1/(2*(2*sqrt(6) + 5)**(1/3)) + (2*sqrt(6) + 5)**(1/3)/2))*sin(sqrt(3)*x*(-(2*sqrt(6) +
5)**(1/3) + (2*sqrt(6) + 5)**(-1/3))/2) + (-sqrt(6)*(2*sqrt(6) + 5)**(2/3)/9 -
sqrt(6)*(2*sqrt(6) + 5)**(1/3)/36 + (2*sqrt(6) + 5)**(1/3)/8 + 7*(2*sqrt(6) +
5)**(2/3)/24 + 11/6)*exp(x*(-1 + 1/(2*(2*sqrt(6) + 5)**(1/3)) + (2*sqrt(6) +
5)**(1/3)/2))*cos(sqrt(3)*x*(-(2*sqrt(6) + 5)**(1/3) + (2*sqrt(6) + 5)**(-1/3))/2) + 1/4
+ (-7*(2*sqrt(6) + 5)**(2/3)/24 - (2*sqrt(6) + 5)**(1/3)/8 + sqrt(6)*(2*sqrt(6) +
5)**(1/3)/36 + 11/12 + sqrt(6)*(2*sqrt(6) + 5)**(2/3)/9)*exp(-x*((2*sqrt(6) + 5)**(-1/3)
+ 1 + (2*sqrt(6) + 5)**(1/3)))
```



Task 5

5.1 Condition

Розв'язати СЛАУ $Ax = b$ при $A = \begin{pmatrix} 5 & 2 & -1 \\ 3 & 0 & 2 \\ 1 & -3 & 6 \end{pmatrix}$, $b = \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}$ трьома способами.

5.2 Solution code

```
import numpy as np
import scipy.linalg as linalg

A = np.array([[5, 2, -1], [3, 0, 2], [1, -3, 6]])
B = np.array([3, 2, 1])

x1 = np.linalg.solve(A, B)
print("numpy.linalg.solve:", x1)

x2 = linalg.solve(A, B)
print("scipy.linalg.solve:", x2)

A_inv = np.linalg.inv(A)
x3 = np.dot(A_inv, B)
print("inverse matrix:", x3)
```

5.3 Output

```
numpy.linalg.solve: [0.57142857 0.14285714 0.14285714]
scipy.linalg.solve: [0.57142857 0.14285714 0.14285714]
inverse matrix: [0.57142857 0.14285714 0.14285714]
```

Task 6

6.1 Condition

Знайти псевдорозв'язок перевизначеної СЛАР $Ax = b$ при $A = \begin{pmatrix} 3 & -2 \\ 5 & 1 \\ 2 & 0 \end{pmatrix}, b = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ чотирьома способами.

6.2 Solution code

```
import numpy as np
import scipy.linalg as linalg

A = np.array([[3, -2], [5, 1], [2, 0]])
B = np.array([1, 1, 1])

x1, resids, rank, s = np.linalg.lstsq(A, B, rcond=None)
print("(numpy.linalg.lstsq):", x1)

x2 = np.dot(np.linalg.pinv(A), B)
print("(numpy.linalg.pinv):", x2)

x3 = np.dot(np.linalg.inv(np.dot(A.T, A)), np.dot(A.T, B))
print("(normal equation):", x3)

U, S, Vt = np.linalg.svd(A, full_matrices=False)
S_inv = np.linalg.inv(np.diag(S))
x4 = np.dot(Vt.T, np.dot(S_inv, np.dot(U.T, B)))
print("(SVD):", x4)
```

6.3 Output

```
(numpy.linalg.lstsq): [ 0.25925926 -0.14814815]
(numpy.linalg.pinv): [ 0.25925926 -0.14814815]
(normal equation): [ 0.25925926 -0.14814815]
(SVD): [ 0.25925926 -0.14814815]
```

Task 7

7.1 Condition

Знайти спектральну норму матриці $A = \begin{pmatrix} 5 & 2 & -1 \\ 3 & 0 & 2 \\ 1 & -3 & 6 \end{pmatrix}$ двома способами: за допомогою вбудованої функції та по визначенню.

7.2Solution code

```
import numpy as np
import scipy.linalg as linalg

A = np.array([[5, 2, -1], [3, 0, 2], [1, -3, 6]])

spectral_norm_scipy = linalg.norm(A, ord=2)
print("scipy.linalg.norm:", spectral_norm_scipy)

A_T_A = np.dot(A.T, A)
eigenvalues = np.linalg.eigvals(A_T_A)
spectral_norm_definition = np.sqrt(np.max(eigenvalues))
print("by definition:", spectral_norm_definition)
```

7.3 Output

```
scipy.linalg.norm: 7.210836116862384
by definition: 7.210836116862384
```

Task 8

8.1 Condition

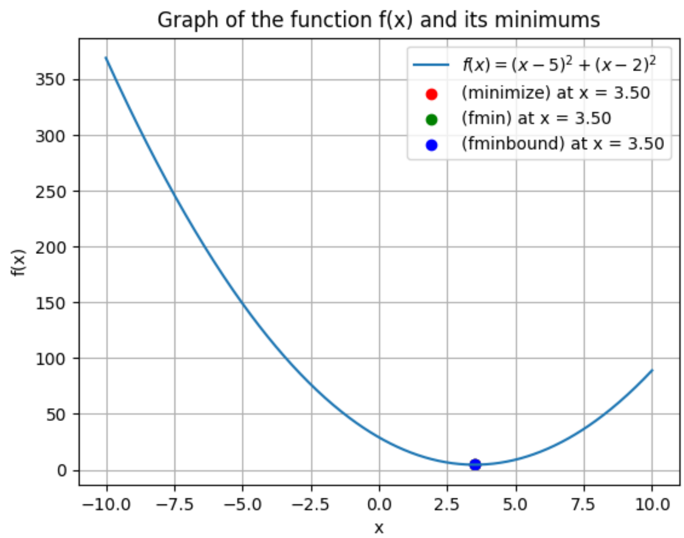
Знайти мінімум функції однієї змінної $f(x) = (x-5)^2 + (x-2)^2$ трьома способами. Побудувати графік функції.

8.2Solution code

```
import numpy as np
import scipy.optimize as opt
import matplotlib.pyplot as plt

def f(x):
    return (x - 5)**2 + (x - 2)**2

result1 = opt.minimize(f, x0=0)
print("scipy.optimize.minimize:", result1.x)

result2 = opt.fmin(f, x0=0)
print("scipy.optimize.fmin:", result2)

def df(x):
    return 2 * (x - 5) + 2 * (x - 2)

result3 = opt.fminbound(f, -10, 10)
print("gradient descent(fminbound):", result3)

x_vals = np.linspace(-10, 10, 400)
y_vals = f(x_vals)

plt.plot(x_vals, y_vals, label=r'$f(x) = (x-5)^2 + (x-2)^2$')
plt.scatter(result1.x, f(result1.x), color='red', label=f'(minimize) at x = 
{result1.x[0]:.2f}')
plt.scatter(result2, f(result2), color='green', label=f'(fmin) at x = {result2[0]:.2f}')
plt.scatter(result3, f(result3), color='blue', label=f'(fminbound) at x = {result3:.2f}')
plt.title('Graph of the function f(x) and its minimums')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.legend()
plt.grid(True)
plt.show()
```

8.3 Output

```
scipy.optimize.minimize: [3.49999999]
Optimization terminated successfully.
         Current function value: 4.500000
         Iterations: 28
         Function evaluations: 56
scipy.optimize.fmin: [3.5]
gradient descent(fminbound): 3.5
```



Task 9

9.1 Condition

Побудувати графіки інтерполяційного полінома Лагранжа, а також інтерполяційних сплайнів 1-го та 3-го ступенів для функції, заданої таблично:

x	0	1	2	3	4	5
$f(x)$	5	-1	3	2	0	8

9.2Solution code

```
import numpy as np
import scipy.interpolate as interp
import matplotlib.pyplot as plt

x_data = np.array([0, 1, 2, 3, 4, 5])
y_data = np.array([5, -1, 3, 2, 0, 8])

lagrange_interpolator = interp.lagrange(x_data, y_data)

linear_spline = interp.PchipInterpolator(x_data, y_data)

cubic_spline = interp.CubicSpline(x_data, y_data)

x_vals = np.linspace(0, 5, 400)

y_lagrange = lagrange_interpolator(x_vals)
y_linear = linear_spline(x_vals)
y_cubic = cubic_spline(x_vals)

plt.figure(figsize=(10, 6))

plt.plot(x_vals, y_lagrange, label='Lagrange polynomial', color='blue', linestyle='-')

plt.plot(x_vals, y_linear, label='linear spline', color='green', linestyle='--')

plt.plot(x_vals, y_cubic, label='cubic spline', color='red', linestyle='-.')

plt.scatter(x_data, y_data, color='black', zorder=5, label='Given points')

plt.xlabel('x')
plt.ylabel('f(x)')
plt.legend()
plt.grid(True)

plt.show()
```

9.3 Output