Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Розрахунково-графічне завдання #4

Чисельне розв’язання інтегральних рівнянь

Виконав:

Захаров Дмитро Олегович

Група МП-41

Харків – 2025

Зміст

1 Постановка задачі **2**

2 Опис методів **2**

2.1 Метод квадратур 2

3 Імплементація **3**

3.1 Методологія 3

3.2 Код на Python 3

1 Постановка задачі

Розв'язати методом квадратур інтегральне рівняння

$$y(x) - \int_0^1 \cos(0.5xt)y(t)dt = \sqrt{1+x^2}$$

2 Опис методів

2.1 Метод квадратур

Нехай ми маємо інтегральне рівняння Фредгольма 2-го роду

$$y(x) = \lambda \int_a^b K(x,t)y(t)dt + f(x), \quad x \in [a,b],$$

Метод квадратур полягає у заміні визначеного інтегралу $\int_a^b F(x)dx$ на суму $\sum_{i=1}^n A_i F(x_i) + R(F)$, де $R(F)$ — мала нев'язка. Таким чином, розіб'ємо відрізок $[a,b]$ на n частин таким чином, що $x_i = a + (i-1)h$ для $h = \frac{b-a}{n-1}$, та позначимо $y_i := y(x_i)$, $f_i := f(x_i)$, $K_{i,j} := K(x_i, x_j)$. Тоді, використовуючи заміну, маємо:

$$y_i = f_i + \lambda \sum_{j=1}^n A_j K_{i,j} y_j + R_i.$$

Відкинувши R_i та позначивши $v_i \approx y(x_i)$ — наближене значення y у точці x_i, отримаємо систему лінійних алгебраїчних рівнянь

$$v_i = f_i + \lambda \sum_{j=1}^n A_j K_{i,j} v_j, \quad i = 1, \ldots, n.$$

Коли ми знайдемо v_i, то зможемо знайти наближений розв'язок $\widehat{y}$ за формулою

$$\widehat{y}(x) = f(x) + \lambda \sum_{j=1}^n A_j K(x, x_j) v_j.$$

3 Імплементація

3.1 Методологія

В нашому прикладі маємо:

$$f(x) = \sqrt{1 + x^2}, \quad K(x, t) = \cos(0.5xt), \quad a = 0, \quad b = 1.$$

Розділяємо відрізок на n частин, отримуючи $x_i = ih$ для $h = \frac{1}{n}$. Скористаємось формулою трапецій. Тоді, $A_1 = A_n = \frac{1}{2}h$ і $A_j = h$ для всіх інших j. Таким чином, можемо знайти наближене значення $y(x)$ в точках x_i за допомогою системи лінійних алгебраїчних рівнянь

$$v_i = \sqrt{1 + h^2 i^2} + h \sum_{j=1}^{n} k_j \cos(0.5h^2 ij) v_j, \quad k_j = \begin{cases} \frac{1}{2}, & j \in \{1, n\}, \\ 1, & j \notin \{1, n\}. \end{cases}$$

3.2 Код на Python

Наступний код розв'язує зазане інтегральне рівняння методом квадратур:

```python
from typing import Callable

# Some math-related imports
import numpy as np
from scipy.sparse import lil_matrix
from scipy.sparse.linalg import spsolve


def solve_fredholm(
    n: int,
    a: float,
    b: float,
    kernel: Callable[[np.ndarray, np.ndarray], np.ndarray],
    f_func: Callable[[np.ndarray], np.ndarray],
) -> Callable[[np.ndarray], np.ndarray]:
```

```python
    """
    Solves the Fredholm integral equation of the second kind:
    """

def solve_fredholm(
    n: int,
    a: float,
    b: float,
    kernel: Callable[[np.ndarray, np.ndarray], np.ndarray],
    f_func: Callable[[np.ndarray], np.ndarray],
) -> Callable[[np.ndarray], np.ndarray]:
    """
    Solves the Fredholm integral equation of the second kind:
    """

    h = (b - a) / n  # Step size
    xs = np.linspace(a, b, n+1)  # Grid points

    weights = h * np.ones(n+1)  # Coefficients for the integral trapezoi
    weights[0] = weights[n] = 0.5 * h # Trapezoidal rule for endpoints

    b = np.zeros(n+1) # Initialize the solution vector
    M = np.zeros((n+1, n+1)) # Coefficient matrix

    # Now, we fill the matrix A and the vector v
    for i in range(n+1):
        b[i] = f_func(xs[i]) # Free term is easy to compute

        for j in range(n+1):
            delta = 1.0 if i == j else 0.0
            M[i,j] = delta - kernel(xs[i], xs[j]) * weights[j]

    v = np.linalg.solve(M, b) # Solve the linear system

    def resultant_fn(x: np.ndarray) -> np.ndarray:
        """
```

```
        Returns the resultant function y(x).
        """

        return f_func(x) + np.sum(weights * kernel(xs, x) * v)

    return resultant_fn
```

Наступний код виконує розв'язок інтегрального рівняння, заданого вище:

```
import numpy as np
import matplotlib.pyplot as plt

from solver import solve_fredholm

if __name__ == "__main__":
    # Problem parameters
    a, b = 0.0, 1.0
    n = 20 # Number of grid points
    f_func = lambda x: np.sqrt(1 + x**2)
    kernel = lambda x, y: np.cos(0.5 * x * y)

    # Solve the Fredholm integral equation
    y_solution = solve_fredholm(n, a, b, kernel, f_func)

    # Plot the solution
    x_vals = np.linspace(a, b, n)
    y_vals = np.array([y_solution(x_vals[i]) for i in range(n)], dtype=np

    plt.plot(x_vals, y_vals, label="Numerical Solution", color='blue', li
    plt.title("Solution to the Fredholm Integral Equation")
    plt.xlabel("x")
    plt.ylabel("y(x)")
    plt.legend()
    plt.grid()
    plt.savefig("fredholm_solution.pdf", dpi=300)
    plt.show()
```

Результат зображено на Рисунку 1.

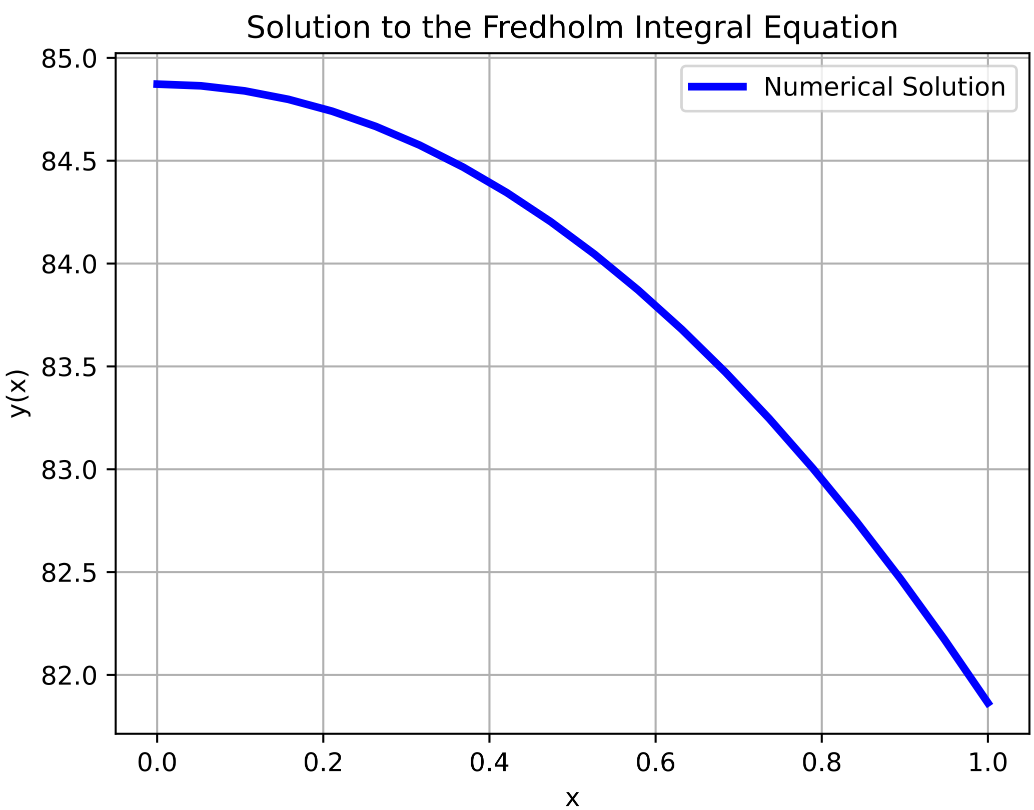

Рис. 1: Графік розв'язку інтегрального рівняння Фредгольма 2-го роду для $n = 20$