**International IT University**
Faculty of Computer technologies and cyber security
Department: MCM

# Report

In the discipline «Numerical Analysis»

Executed: Taldybayev B.A.

Group: IT3-2203

Lecturer: Шахан Н.Ш.

Almaty, 2025

Task 2: 1D Poisson Equation

1. We have formula:

$$\frac{\partial^2 U}{\partial x^2} = f(x),$$

where f(x) = 6x

2. Approximate by the finite difference method:

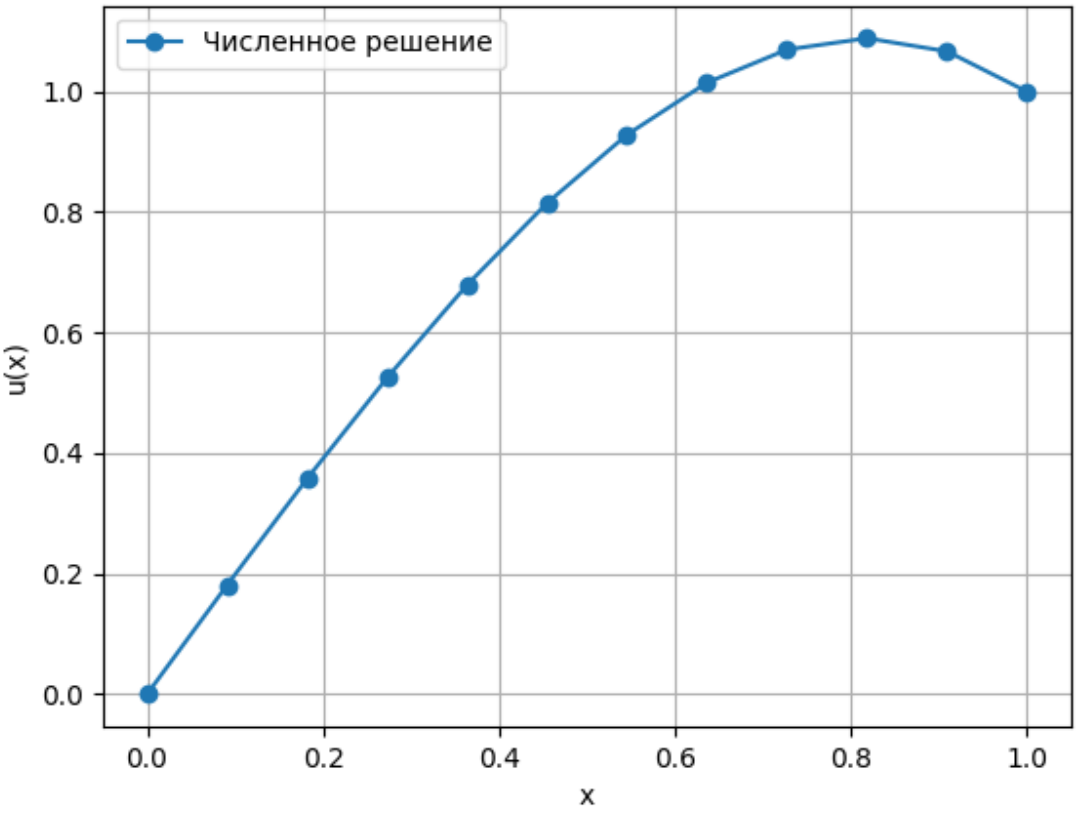
$$\frac{U_{i+1} - 2U_i + U_{i-1}}{h^2} = f(x_i)$$

3. Multiply to h^2 to get rid of the denominator. And multiply to -1 to get a standard view of Thomas's method:

$$-U_{i-1} + 2U_i - U_{i+1} = -h^2 f(x_i)$$

## Code and graph:

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


def thomas_algorithm(a, b, c, d):
    n = len(d)
    c_ = np.zeros(n - 1)
    d_ = np.zeros(n)

    c_[0] = c[0] / b[0]
    d_[0] = d[0] / b[0]

    for i in range(1, n - 1):
        c_[i] = c[i] / (b[i] - a[i - 1] * c_[i - 1])
    for i in range(1, n):
        d_[i] = (d[i] - a[i - 1] * d_[i - 1]) / (b[i] - a[i - 1] * c_[i - 1])

    u = np.zeros(n)
    u[-1] = d_[-1]
    for i in range(n - 2, -1, -1):
        u[i] = d_[i] - c_[i] * u[i + 1]

    return u


L = 1.0
N = 10
h = L / (N + 1)
A, B = 0, 1


def f(x):
    return 6 * x


x = np.linspace(h, L - h, N)
b = -2 * np.ones(N)
a = np.ones(N - 1)
c = np.ones(N - 1)
d = -h ** 2 * f(x)

d[0] -= A
d[-1] -= B

u = thomas_algorithm(a, b, c, d)

x_full = np.linspace(0, L, N + 2)
u_full = np.concatenate(([A], u, [B]))

plt.plot(x_full, u_full, 'o-', label="Численное решение")
plt.xlabel("x")
plt.ylabel("u(x)")
plt.legend()
plt.grid()
plt.show()
```


Численное решение
u(x)
x
0.0
0.2
0.4
0.6
0.8
1.0