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 1: 1D Advection Equation

1. We have formula:

$$\frac{\partial U}{\partial t} + a\frac{\partial U}{\partial x} = \mu\frac{\partial^2 U}{\partial x^2}$$

- U (x, t) — искомая функция,
- a — коэффициент переноса,
- μ — коэффициент вязкости,
- x ∈ [0, L]— пространство,
- t > 0 — время.

2. Approximation of the time derivative (step forward)

$$\frac{\partial U}{\partial x} \approx \frac{U_{i+1}^n - U_i^n}{\Delta x}$$

Approximation of the spatial derivative (step back)

$$\frac{\partial U}{\partial x} \approx \frac{U_i^n - U_{i-1}^n}{\Delta x}$$

The second derivative in space is approximated by the "forward-backward" scheme (centered difference):

$$\frac{\partial^2 U}{\partial x^2} \approx \frac{U_{i+1}^n - 2U_i^n + U_{i-1}^n}{\Delta x^2}$$

So in the end:

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} + a\frac{U_i^n - U_{i-1}^n}{\Delta x} = \mu\frac{U_{i+1}^n - 2U_i^n + U_{i-1}^n}{\Delta x^2}$$

We need to move U_i^{n+1}, to the left side, so we multiply the equation by Δt and move U_i^{n+1}, to the left side:

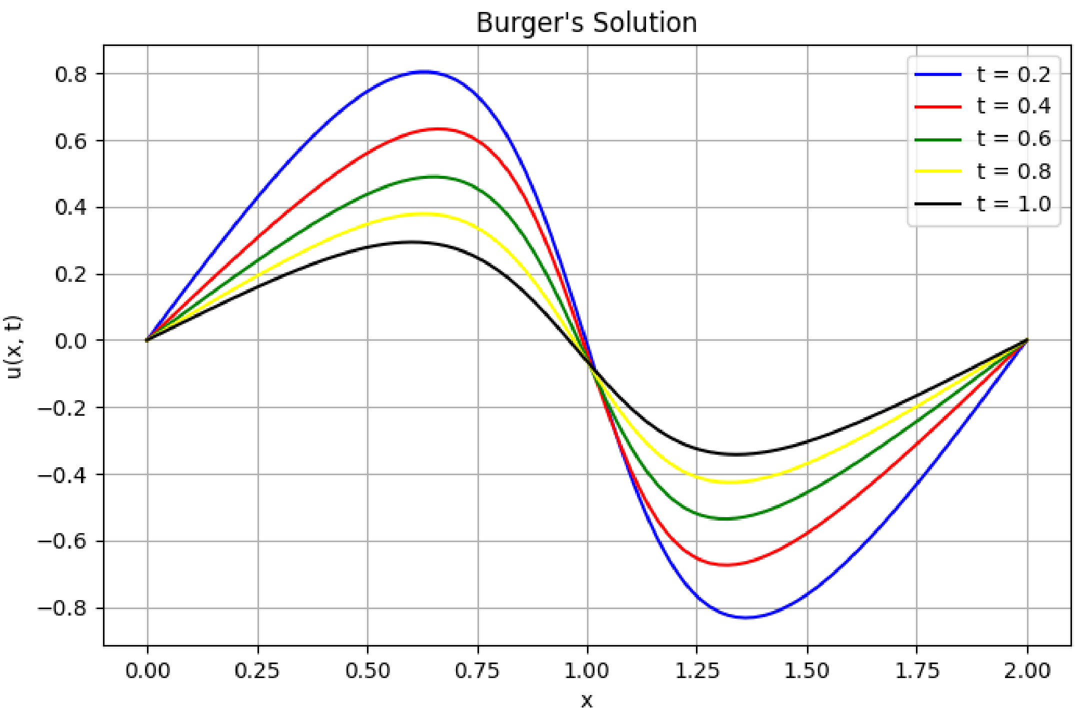
$$U_i^{n+1} - U_i^n = -a\frac{\Delta t}{\Delta x}(U_i^n - U_{i-1}^n) + \mu\frac{\Delta t}{\Delta x^2}(U_{i+1}^n - 2U_i^n + U_{i-1}^n)$$

$$U_i^{n+1} = U_i^n - a\frac{\Delta t}{\Delta x}(U_i^n - U_{i-1}^n) + \mu\frac{\Delta t}{\Delta x^2}(U_{i+1}^n - 2U_i^n + U_{i-1}^n)$$

3. Code part:

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

x0, xn = 0, 2
N = 100
delta_x = (xn - x0) / N

t0, tn = 0, 1
delta_t = 0.001
N_t = int((tn - t0) / delta_t)

nu = 0.1

X = np.linspace(x0, xn, N + 1)
T = np.arange(t0, tn + delta_t, delta_t)

U = np.zeros((N_t + 1, N + 1))
U[0, :] = np.sin(np.pi * X)

for i in range(1, N_t + 1):
    U_next = U[i - 1].copy()
    for j in range(1, N):
        convective = U[i - 1, j] * (U[i - 1, j] - U[i - 1, j - 1]) /
delta_x
        diffusive = nu * (U[i - 1, j + 1] - 2 * U[i - 1, j] + U[i - 1, j -
1]) / delta_x**2
        U_next[j] = U[i - 1, j] - delta_t * convective + delta_t *
diffusive
    U[i] = U_next.copy()

plot_times = [0.2, 0.4, 0.6, 0.8, 1.0]
plot_indices = [int(t / delta_t) for t in plot_times]
colors = ['blue', 'red', 'green', 'yellow', 'black']

plt.figure(figsize=(8, 5))
for i, idx in enumerate(plot_indices):
    plt.plot(X, U[idx], color=colors[i], label=f"t = {plot_times[i]:.1f}")

plt.xlabel("x")
plt.ylabel("u(x, t)")
plt.title("Burger's Solution")
plt.legend()
plt.grid()
plt.show()
```

Graph:



Conclusion:

The graphs at different time points (t = 0.2, 0.4, 0.6, 0.8, 1.0) show that the initial sine wave changes over time. The solution gradually smooths out, and sharp gradients disappear.

This behavior is caused by numerical diffusion, which leads to a gradual loss of sharpness in the waveform. Additionally, the combined effect of convection and viscosity influences the wave's shape, causing it to evolve differently than in a purely advective case. The choice of numerical parameters affects the accuracy and stability of the solution.