Лабораторная работа №4

Ларин Егор. 4 группа 3 курс

22 декабря 2022 г.

Теория

Задача

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + t^2 \sin xt + x \cos xt, 0 < x < 1, 0 < t \leq 0.5,$$

$$\begin{cases} u(x,0) = 0, 0 \leq x \leq 1, \\ u(0,t) = 0, 0 \leq t \leq \frac{1}{2}, \\ u(1,t) = \sin(t), 0 \leq t \leq \frac{1}{2}. \end{cases}$$

Точное решение

$$u(x,t) = \sin(xt) \tag{1}$$

Сетка

- τ – шаг сетки по времени, $N_2 = \frac{1}{2\tau}$.
- h – шаг сетки по пространству, $N_1 = \frac{1}{h}$.
- $\varphi_i^j = f(x_i, t_{j+\frac{1}{2}})$

Граничные условия

$$\begin{cases} y_i^0 = 0, i = \overline{0, N_1} \\ y_0^j = 0, j = \overline{0, N_2} \\ y_{N_1}^j = \sin(j\tau), j = \overline{0, N_2} \end{cases}$$

Разностная схема

1. Явная $\frac{y_i^{j+1} - y_i^j}{\tau} = \Lambda y_i^j + \varphi_i^j$
2. Чисто неявная $\frac{y_i^{j+1} - y_i^j}{\tau} = \Lambda y_i^{j+1} + \varphi_i^j$
3. Кранка-Николсона $\frac{y_i^{j+1} - y_i^j}{\tau} = \frac{1}{2}(\Lambda y_i^j + \Lambda y_i^{j+1}) + \varphi_i^j$

Устойчивость и порядок аппроксимации

- Явная – $O(\tau + h^2)$, не является абсолютно устойчивой.
- Чисто – неявная $O(\tau + h^2)$, является абсолютно устойчивой.
- Кранка-Николсона – $O(\tau^2 + h^2)$, является абсолютно устойчивой.

Листинг кода

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

a = 0.
b = 1.
c = 0.
d = 0.5

draw_plots = False

def u(x, t):
    return np.sin(t * x)

def u0(x):
    return 0

def u1(t):
    return 0.

def u2(t):
    return np.sin(t)

def f(x, t):
    return t**2 * np.sin(x*t) + x * np.cos(x * t)

def TDMA(a, b, c, f):
    beta = []
    l = len(c) - 1
    alfa = [b[0] / c[0]]
    beta.append(f[0] / c[0])
    for i in range(1, l):
        check = (c[i] - a[i - 1] * alfa[i - 1])
        if abs(check) < 1e-19: exit()
        alfa.append(b[i] / check)
        beta.append((f[i] + a[i - 1] * beta[i - 1]) / check)
```

```
    beta.append((f[l] + a[l - 1] * beta[l - 1]) / (c[l] - a[l -1] * alfa[l - 1]))
    y = [0] * (l + 1)
    y[l] = beta[l]
    for i in range(l - 1, -1, -1):
        y[i] = alfa[i] * y[i + 1] + beta[i]
    return np.array(y)

def plot(y):
    data = np.array(y)
    length = data.shape[0]
    width = data.shape[1]
    x, y = np.meshgrid(np.arange(width), np.arange(length))
    ax = plt.axes(projection='3d')
    ax.plot_surface(x, y, data)
    if draw_plots:
        plt.show()

def build_exact(h, tau):
    n = int((b - a) / h) + 1
    m = int((d - c) / tau) + 1
    ys = np.zeros((n, m))
    for i in range(n):
        for j in range(m):
            ys[i, j] = u(h*i, tau*j)
    return ys

def build_explicit(h, tau):
    xs = np.linspace(a, b, int((b - a) / h) + 1)
    ts = np.linspace(c, d, int((d - c) / tau) + 1)
    ys = np.zeros((len(xs), len(ts)))

    for i in range(len(xs)):
        ys[i, 0] = u0(xs[i])

    for j in range(len(ts) - 1):
        ys[0, j + 1] = u1(ts[j + 1])

        for i in range(1, len(xs) - 1 ):
            a_i = (ys[i + 1, j] - 2 * ys[i, j] + ys[i - 1, j]) / (h * h)
            b_i = f(xs[i], ts[j])
            ys[i, j + 1] = tau * (a_i + b_i) + ys[i, j]

        ys[len(xs) - 1, j + 1] = u2(ts[j + 1])

    plot(ys)
```

```python
    return ys

def build_implicit(h, tau):
    xs = np.linspace(a, b, int((b - a) / h) + 1)
    ts = np.linspace(c, d, int((d - c) / tau) + 1)
    ys = np.zeros((len(xs), len(ts)))

    for i in range(len(xs)):
        ys[i, 0] = u0(xs[i])

    for j in range(len(ts)-1):
        c_i = [1]
        c_i = c_i + [1 / tau + 2 / (h * h)] * (len(xs) - 2)
        c_i.append(1)

        a_i = []
        a_i = a_i + [-1 / (h * h)] * (len(xs) - 2)
        a_i.append(0)

        b_i = [0]
        b_i = b_i + [-1 / (h * h)] * (len(xs) - 2)

        f_i = [u1(ts[j + 1])]
        for i in range(1, len(xs) - 1):
            l = f(xs[i], ts[j + 1])
            f_i.append((ys[i, j] / tau) + l)
        f_i.append(u2(ts[j + 1]))
        ys[:, j + 1] = TDMA(-np.array(a_i), -np.array(b_i), c_i, f_i)

    plot(ys)
    return ys


def build_nickolson(h, tau):
    xs = np.linspace(a, b, int((b - a) / h) + 1)
    ts = np.linspace(c, d, int((d - c) / tau) + 1)
    ys = np.zeros((len(xs), len(ts)))

    for i in range(len(xs)):
        ys[i, 0] = u0(xs[i])

    for j in range(len(ts)-1):
        c_i = [1]
        c_i = c_i + [1 / tau + 1 / (h * h)] * (len(xs) - 2)
        c_i.append(1)
```

```
        a_i = []
        a_i = a_i + [-1 / (2 * h * h)] * (len(xs) - 2)
        a_i.append(0)

        b_i = [0]
        b_i = b_i + [-1 / (2 * h * h)] * (len(xs) - 2)

        f_i = [u1(ts[j + 1])]
        for i in range(1, len(xs) - 1):
            l = f(xs[i], ts[j] + 0.5 * tau)
            v = (1 / (2 * h * h)) * (ys[i + 1, j] - 2 * ys[i, j] + ys[i - 1, j])
            f_i.append((ys[i, j] / tau) + l + v)
        f_i.append(u2(ts[j + 1]))
        ys[:, j + 1] = TDMA(-np.array(a_i), -np.array(b_i), c_i, f_i)

    plot(ys)
    return ys

print(f'Explicit: {np.max(np.abs(build_exact(0.1,0.1) -build_explicit(0.1,0.1)))}')
print(f'Explicit: {np.max(np.abs(build_exact(0.1,0.005) - build_explicit(0.1,0.005)))}')
print(f"Implicit: {np.max(np.abs(build_exact(0.1,0.1) - build_implicit(0.1,0.1)))}")
print(f"Nickolson: {np.max(np.abs(build_exact(0.1,0.1) - build_nickolson(0.1, 0.1)))}")
```

Графики

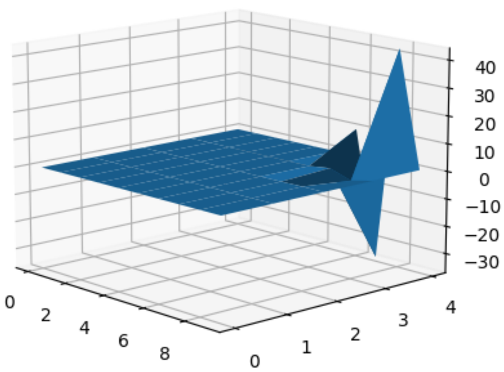

Рис. 1: Явная неустойчивая

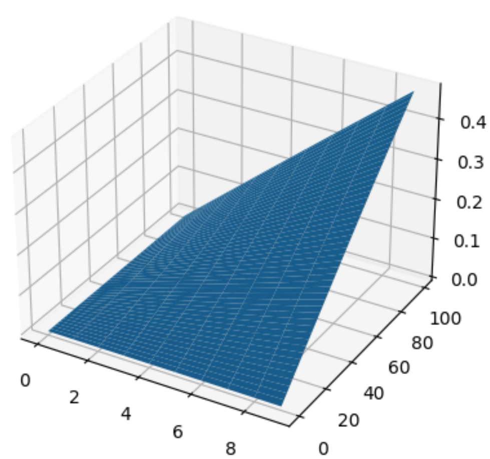

Рис. 2: Явная устойчивая

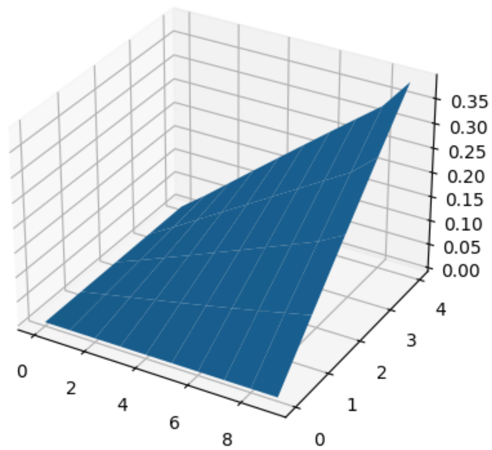

Рис. 3: Чисто неявная

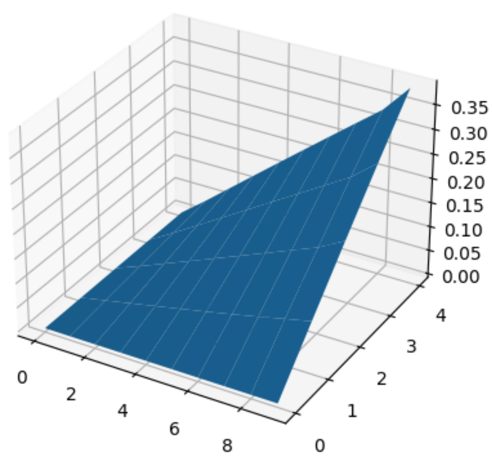

Рис. 4: Кранка-Николсона

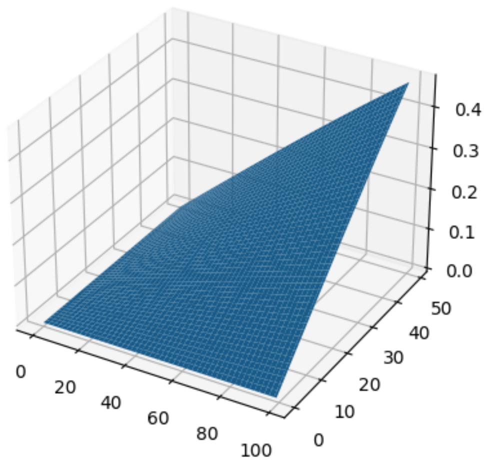

Рис. 5: Точное решение

Результаты вычислительного эксперимента

Схема	$\max \lvert y_i^j - u(x_i, t_j) \rvert$
Явная (нейстойчивая)	18.08729087566136
Явная (устойчивая)	2.770470845148143e-05
Чисто неявная	0.0004969953632136814
Кранка-Николсона	0.00017371480558825425