Московский авиационный институт
(национальный исследовательский университет)

Институт №8 "Информационные технологии и прикладная математика"
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №5
по дисциплине:
Численные методы
Вариант №5

Выполнил: студент группы М8О-409Б-20
Искренкова А.В.
Принял: Пивоваров Е.Д.
Оценка:____

Москва, 2023 г.

1. Задание

Используя явную и неявную конечно-разностные схемы, а также схему Кранка - Николсона, решить начально-краевую задачу для дифференциального уравнения параболического типа. Осуществить реализацию трех вариантов аппроксимации граничных условий, содержащих производные: двухточечная аппроксимация с первым порядком, трехточечная аппроксимация со вторым порядком, двухточечная аппроксимация со вторым порядком. В различные моменты времени вычислить погрешность численного решения путем сравнения результатов с приведенным в задании аналитическим решением $U(x,t)$. Исследовать зависимость погрешности от сеточных параметров τ, h.

Уравнение:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \sin(\pi x)$$

$$u(0,t) = 0$$

$$u(1,t) = 0$$

$$u(x,0) = 0$$

Аналитическое решение:

$$U(x,t) = \frac{1}{\pi^2}(1 - \exp(-\pi^2 t))\sin(\pi x)$$

2. Решение

- Явная схема:

$$\frac{u_j^{k+1} - u_j^k}{\tau} = \frac{u_{j-1}^k - 2u_j^k + u_{j+1}^k}{h^2} + f(x_j, t^k)$$

- Неявная схема:

$$\frac{u_j^{k+1} - u_j^k}{\tau} = \frac{u_{j-1}^{k+1} - 2u_j^{k+1} + u_{j+1}^{k+1}}{h^2} + f(x_j, t^{k+1})$$

- Гибридная схема (при $\theta = 0.5$, схема Кранка-Николсона)

$$\frac{u_j^{k+1} - u_j^k}{\tau} = (\theta)\left(\frac{u_{j-1}^{k+1} - 2u_j^{k+1} + u_{j+1}^{k+1}}{h^2} + f(x_j, t^k)\right) + (1-\theta)\left(\frac{u_{j-1}^k - 2u_j^k + u_{j+1}^k}{h^2} + f(x_j, t^k)\right)$$

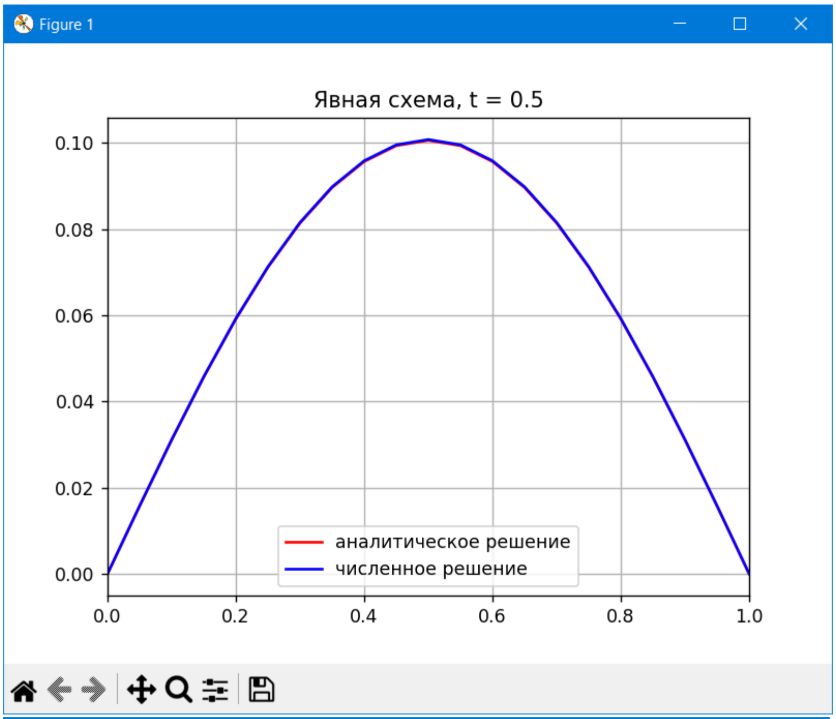
Для решения задачи не используется аппроксимация граничных условий, так как они нулевые.

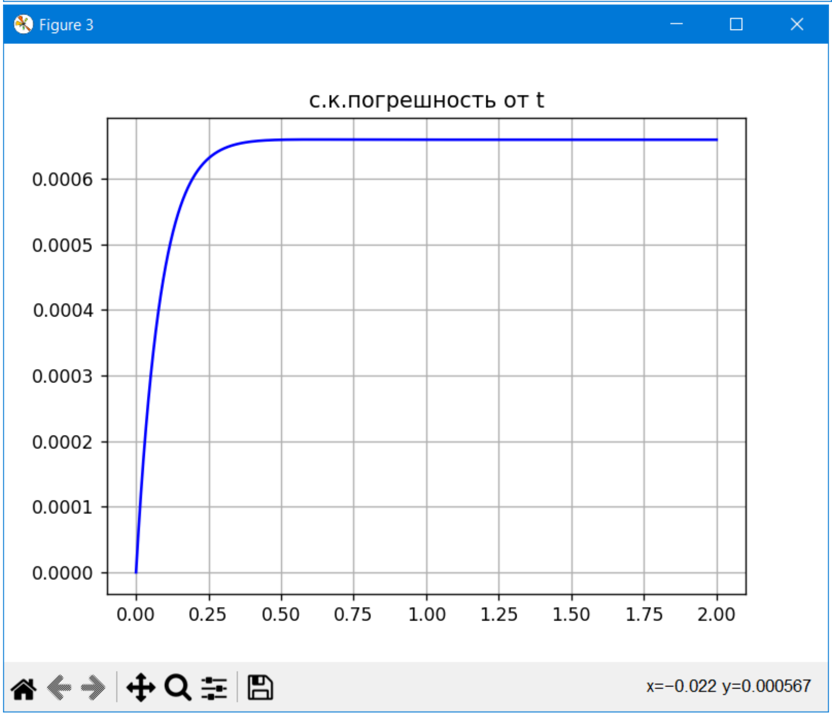
Погрешность между численным и аналитическим решением рассчитывается как среднеквадратическая.

Параметры задачи: $T = 2, h = 0.05, \tau = 0.0005$.

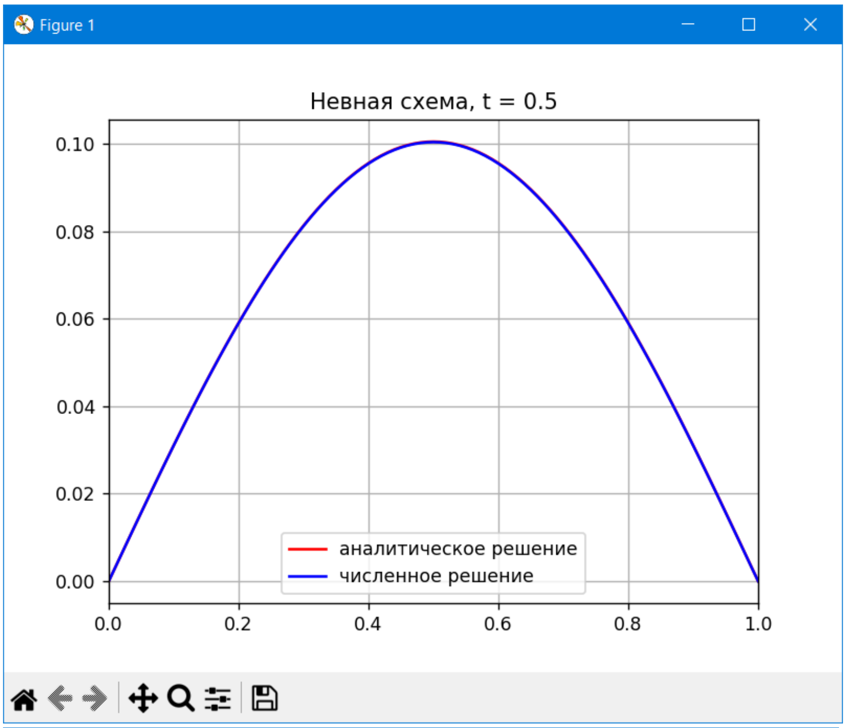
3. Вывод программы

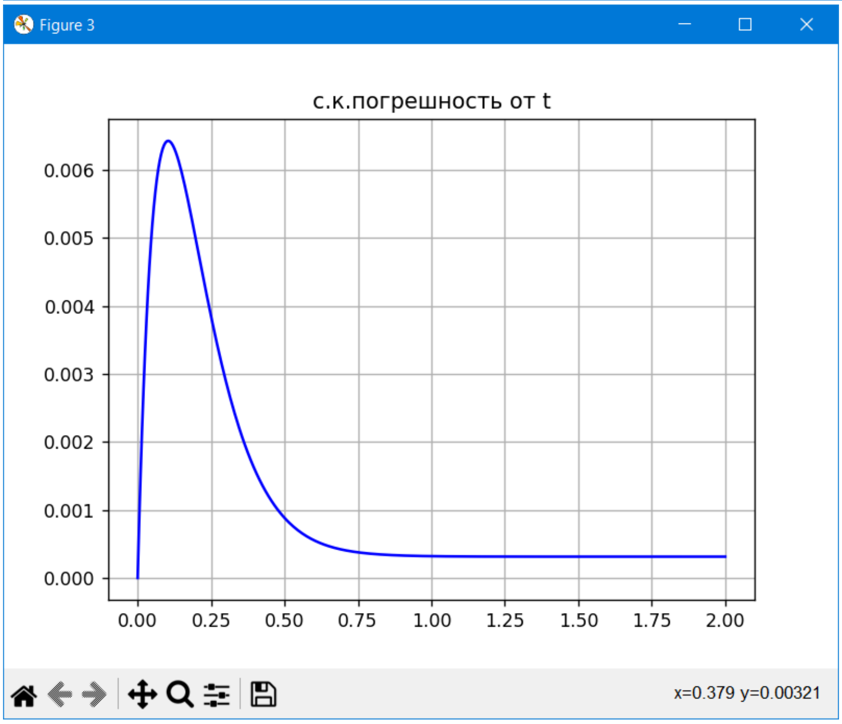
- Явная схема





- Неявная схема





- Схема Кранка-Николсона

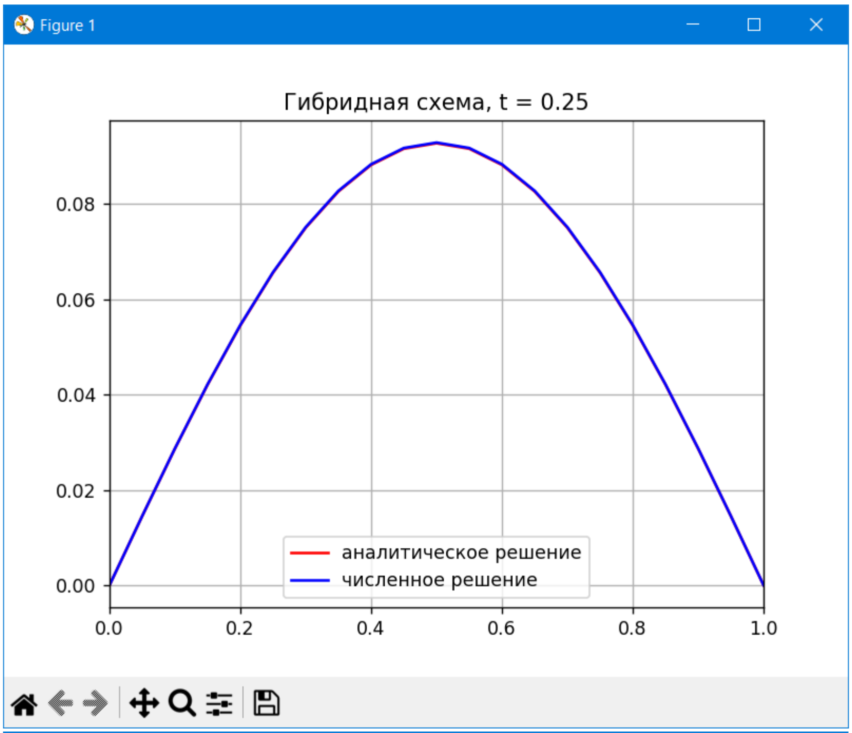


Figure 1
Гибридная схема, t = 0.25
0.08
0.06
0.04
0.02
0.00
0.0
0.2
0.4
0.6
0.8
1.0
аналитическое решение
численное решение

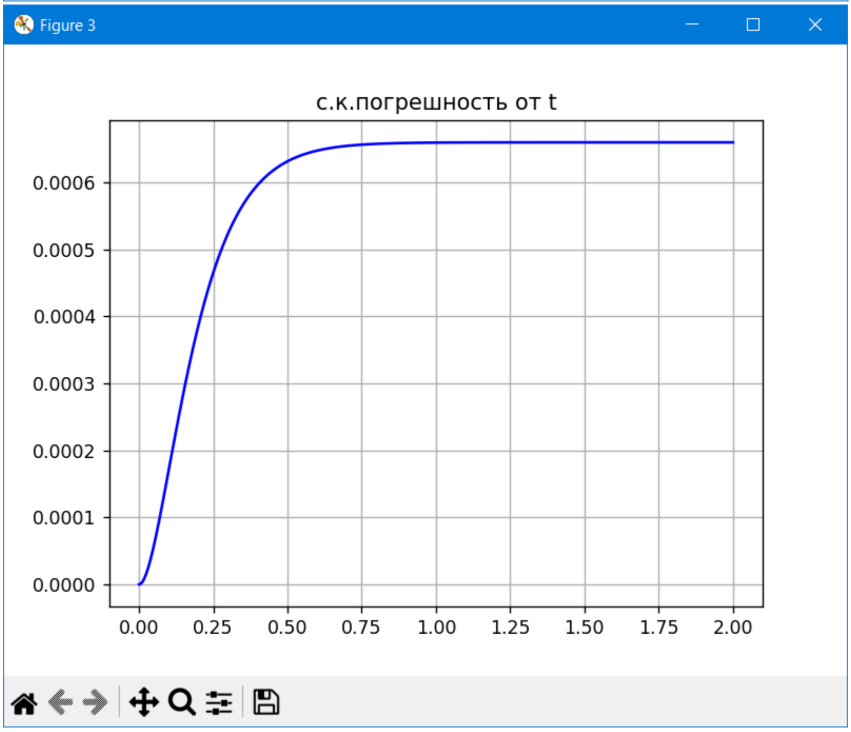


Figure 3
с.к.погрешность от t
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

4. Листинг

```python
import matplotlib
import numpy as np
import math
import matplotlib.pyplot as plt
from math import sqrt

def psi(x):
    return 0

def phi0(t):
    return 0

def phi1(t):
    return 0

def true_fval(x, t):
    return (1 / (math.pi ** 2)) * (1 - np.exp((-math.pi ** 2) * t)) * np.sin(math.pi * x)

def f(x):
    return np.sin(math.pi * x)

def norma(a):
    norm = 0
    for i in range(len(a)):
        norm += a[i] ** 2
    return sqrt(norm)

# метод прогонки
def tridiagonal(a, b, c, d):
    n = len(d)
    x = np.zeros(n)
    p = [-c[0] / b[0]]
    q = [d[0] / b[0]]
    for i in range(1, n):
        p.append(-c[i] / (b[i] + a[i] * p[i - 1]))
        q.append((d[i] - a[i] * q[i - 1]) / (b[i] + a[i] * p[i - 1]))
    x[-1] = q[-1]
    for i in reversed(range(n - 1)):
        x[i] = p[i] * x[i + 1] + q[i]
    return x

def ExScheme(a, lb, ub, h, tau, T):
    # разбиение осей
    x = np.arange(lb, ub + h, h)
    t = np.arange(0, T + tau, tau)
    # строим конечно-разностную сетку
    U = np.zeros((len(t), len(x)))
    # заполним первый уровень
    for j in range(len(x)):
        U[0, j] = psi(x[j])
```

```
    # прямая схема
    for i in range(1, len(t)):
        for j in range(1, len(x)-1):
            U[i, j] = a*tau/(h**2)*U[i-1,j-1]+(1-2*a*tau/(h**2))*U[i-1,j]+a*tau/(h**2)*U[i-1,j+1]+tau*f(x[j])
            U[i, 0] = phi0(t[i])
            U[i, -1] = phi1(t[i])

    return U


def ImScheme(a, lb, ub, h, tau, T):
    x = np.arange(lb, ub + h, h)
    t = np.arange(0, T + tau, tau)
    U = np.zeros((len(t), len(x)))
    for j in range(len(x)):
        U[0, j] = psi(x[j])

    for i in range(1, len(t)):
        aa = np.zeros(len(x)-2)
        bb = np.zeros(len(x)-2)
        cc = np.zeros(len(x)-2)
        dd = np.zeros(len(x)-2)
        dd[0] = -(U[i - 1, 1] + a * tau / (h ** 2) * phi0(t[i])) - tau * f(x[1])
        dd[-1] = -(U[i - 1, len(x) - 1] + a * tau / (h ** 2) * phi1(t[i])) - tau * f(x[len(x) - 2])
        bb[0] = -(1 + 2 * a * tau / (h ** 2))
        bb[-1] = -(1 + 2 * a * tau / (h ** 2))
        cc[0] = a * tau / (h ** 2)
        aa[-1] = a * tau / (h ** 2)
        for j in range(1, len(x) - 3):
            aa[j] = a * tau / (h ** 2)
            bb[j] = -(1 + 2 * a * tau / (h ** 2))
            cc[j] = a * tau / (h ** 2)
            dd[j] = -U[i - 1, j+1] - tau * f(x[j+1])
        xx = tridiagonal(aa, bb, cc, dd)
        for j in range(1,len(x)-1):
            U[i, j] = xx[j-1]

    return U


def Hybrid(a, lb, ub, h, tau, T,teta):
    x = np.arange(lb, ub + h, h)
    t = np.arange(0, T + tau, tau)
    U = np.zeros((len(t), len(x)))
    for j in range(len(x)):
```

```
        U[0, j] = psi(x[j])

    for i in range(1, len(t)):
        aa = np.zeros(len(x) - 2)
        bb = np.zeros(len(x) - 2)
        cc = np.zeros(len(x) - 2)
        dd = np.zeros(len(x) - 2)
        dd[0] = -(1-teta)*a*tau/(h**2)*U[i-1,0]-(1-(1-teta)*2*a*tau/(h**2))*U[i-1,1]-(1-teta)*a*tau/(h**2)*U[i-1,2]\
                -a*tau/(h**2)*teta*phi0(t[i])-tau*f(x[1])
        dd[-1] = -(1-teta)*a*tau/(h**2)*U[i-1,len(x)-3]-(1-(1-teta)*2*a*tau/(h**2))*U[i-1,len(x)-2]-(1-teta)*a*tau/(h**2)*U[i-1,len(x)-1]\
                -a*tau/(h**2)*teta*phi1(t[i])-tau*f(x[len(x)-2])
        bb[0] = -(1 + 2 * a * tau / (h ** 2)*teta)
        bb[-1] = -(1 + 2 * a * tau / (h ** 2)*teta)
        cc[0] = a * tau / (h ** 2)*teta
        aa[-1] = a * tau / (h ** 2)*teta
        for j in range(1, len(x) - 3):
            aa[j] = a * tau / (h ** 2)*teta
            bb[j] = -(1 + 2 * a * tau / (h ** 2)*teta)
            cc[j] = a * tau / (h ** 2)*teta
            dd[j] = -(1-teta)*a*tau/(h**2)*U[i-1,j]-(1-(1-teta)*2*a*tau/(h**2))*U[i-1,j+1]-(1-teta)*a*tau/(h**2)*U[i-1,j+2]-tau*f(x[j+1])
        xx = tridiagonal(aa, bb, cc, dd)
        for j in range(1, len(x) - 1):
            U[i, j] = xx[j - 1]

    return U

def plot_ex(a, lb, ub, h, tau, T, k):
    x = np.arange(lb, ub + h, h)
    t = np.arange(0, T + tau, tau)
    plt.figure(1)
    plt.title('Явная схема, t = ' + str(t[k]))
    plt.grid()
    plt.plot(x, true_fval(x, t[k]), color='red', label='аналитическое решение')
    U = ExScheme(a, lb, ub, h, tau, T)
    plt.plot(x, U[k, :], color='blue', label='численное решение')
    plt.legend()
    plt.xlim((0, ub))
    plt.figure(2)
    plt.title('Погрешность на срезе t ='+ str(t[k])+' от x')
    plt.grid()
    eps = []
    for i in range(len(x)):
        a = np.abs(true_fval(x[i], t[k]) - U[i, :])
        eps = np.append(eps, a)
    plt.plot(x, a, color='green')
```

```
    plt.figure(3)
    plt.title('с.к.погрешность от t')
    plt.grid()
    eps = []
    for i in range(len(t)):
        a = true_fval(x,t[i]) - U[i, :]
        eps = np.append(eps, norma(a))
    plt.plot(t, eps, color='blue')

    plt.show()
    return

def plot_im(a, lb, ub, h, tau, T, k):
    x = np.arange(lb, ub + h, h)
    t = np.arange(0, T + tau, tau)
    plt.figure(1)
    plt.title('Невная схема, t = ' + str(t[k]))
    plt.grid()
    plt.plot(x, true_fval(x, t[k]), color='red', label='аналитическое решение')
    U = ImScheme(a, lb, ub, h, tau, T)
    plt.plot(x, U[k, :], color='blue', label='численное решение')
    plt.legend()
    plt.xlim((0, ub))
    plt.figure(2)
    plt.title('Погрешность на срезе t ='+ str(t[k])+' от x')
    plt.grid()
    eps = []
    for i in range(len(x)):
        a = np.abs(true_fval(x[i], t[k]) - U[i, :])
        eps = np.append(eps, a)
    plt.plot(x, a, color='green')
    plt.figure(3)
    plt.title('с.к.погрешность от t')
    plt.grid()
    eps = []
    for i in range(len(t)):
        a = true_fval(x, t[i]) - U[i, :]
        eps = np.append(eps, norma(a))
    plt.plot(t, eps, color='blue')

    plt.show()
    return

def plot_Hybrid(a, lb, ub, h, tau, T, k,teta):
    x = np.arange(lb, ub + h, h)
    t = np.arange(0, T + tau, tau)
    plt.figure(1)
    plt.title('Гибридная схема, t = ' + str(t[k]))
    plt.grid()
```

```
    plt.plot(x, true_fval(x, t[k]), color='red', label='аналитическое решение')
    U = Hybrid(a, lb, ub, h, tau, T,teta)
    plt.plot(x, U[k, :], color='blue', label='численное решение')
    plt.legend()
    plt.xlim((0, ub))
    plt.figure(2)
    plt.title('Погрешность на срезе t ='+ str(t[k])+' от x')
    plt.grid()
    eps = []
    for i in range(len(x)):
        a = np.abs(true_fval(x[i], t[k]) - U[i, :])
        eps = np.append(eps, a)
    plt.plot(x, a, color='green')
    plt.figure(3)
    plt.title('с.к.погрешность от t')
    plt.grid()
    eps = []
    for i in range(len(t)):
        a = true_fval(x, t[i]) - U[i, :]
        eps = np.append(eps, norma(a))
    plt.plot(t, eps, color='blue')

    plt.show()
    return

plot_ex(1,0,1,0.05,0.0005,2,1000)
plot_im(1, 0, 1, 0.01, 0.005, 2, 100)
plot_Hybrid(1,0,1,0.05,0.0005,2,500,0.5)

```

5. Вывод

В ходе выполнения лабораторной работы были изучены явная, неявная и гибридная схемы решений начально-краевой задачи для дифференциального уравнения параболического типа. Также были изучены три варианта аппроксимации граничных условий. Были получены результаты в графическом представлении и подсчитаны погрешности для каждого варианта решения.