Московский авиационный институт
(национальный исследовательский университет)

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

# Лабораторная работа №6

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 схему крест и неявную схему, решить начально-краевую задачу для дифференциального уравнения гиперболического типа. Аппроксимацию второго начального условия произвести с первым и со вторым порядком. 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 $\tau, h$.

Уравнение:

$$\frac{\partial^2 u}{\partial t^2} = 2\frac{\partial^2 u}{\partial x^2} + 4\frac{\partial u}{\partial x}$$

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

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

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

$$u_t(x,0) = \exp(-x)\sin(x)$$

Аналитическое решение:

$$U(x,t) = 0.5\exp(-x)\sin(x)\sin(2t)$$

# 2. Решение

- Явная схема (крест):
$$\frac{u_j^{k+1} - 2u_j^k + u_j^{k-1}}{\tau^2} = 2\frac{u_{j-1}^k - 2u_j^k + u_{j+1}^k}{h^2} + 2\frac{u_{j+1}^k - u_{j-1}^k}{2h}$$
- Неявная схема:
$$\frac{u_j^{k+1} - 2u_j^k + u_j^{k-1}}{\tau^2} = 2\frac{u_{j-1}^{k+1} - 2u_j^{k+1} + u_{j+1}^{k+1}}{h^2} + 2\frac{u_{j+1}^{k+1} - u_{j-1}^{k+1}}{2h}$$

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

Для решения задачи используются 2 вида аппроксимации начальных условий (для вычисления первого уровня):

$$u_j^1 = u_j^0 + \tau\frac{\partial u}{\partial t}_j^0$$

$$u_j^1 = u_j^0 + \tau \frac{\partial u^0}{\partial t_j} + \frac{\tau^2}{2}\left(\frac{\partial^2 u^0}{\partial t^2{}_j}\right) = u_j^0 + \tau \frac{\partial u^0}{\partial t_j} + \frac{\tau^2}{2}\left(2\frac{\partial^2 u^0}{\partial x^2{}_j} + 4\frac{\partial u^0}{\partial x_j}\right)$$

Второй вариант аппроксимации полностью совпадает с первым,

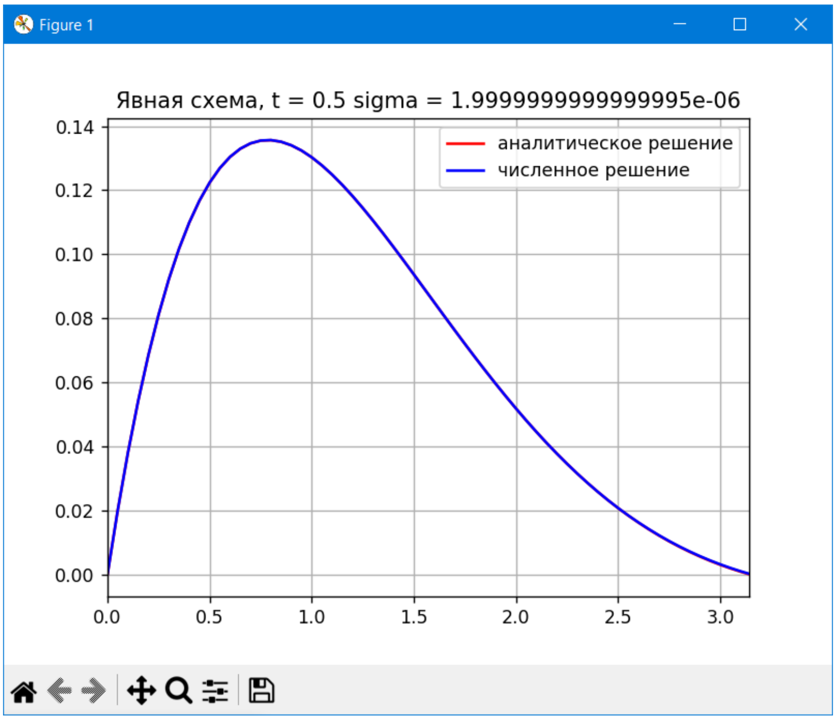
так как $\left(2\frac{\partial^2 u^0}{\partial x^2{}_j} + 4\frac{\partial u^0}{\partial x_j}\right) = 0$

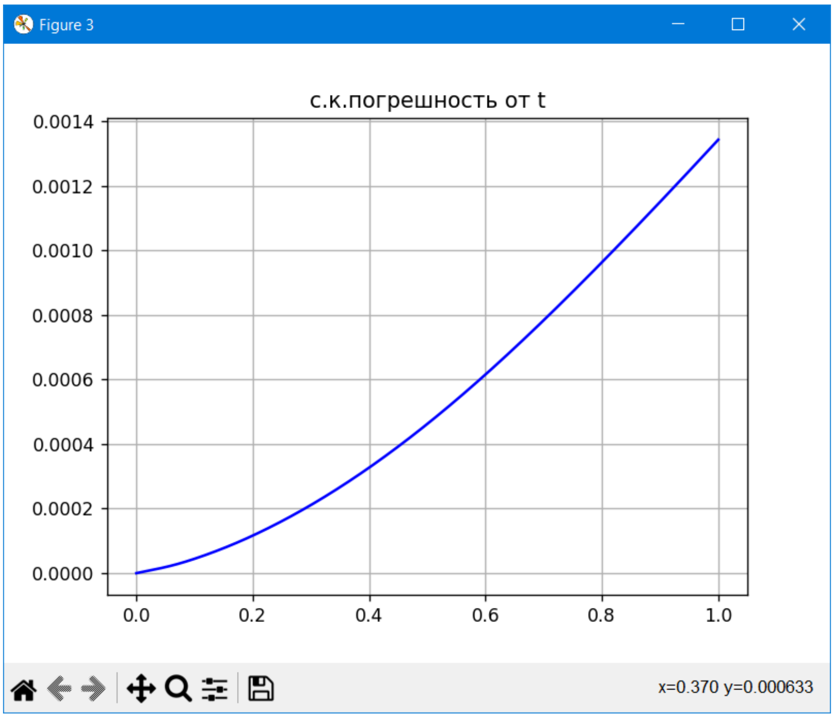
Погрешность между численным и аналитическим решением рассчитывается как среднеквадратическая.

Параметры задачи: $T = 1, h = 0.05, \tau = 0.00005$.

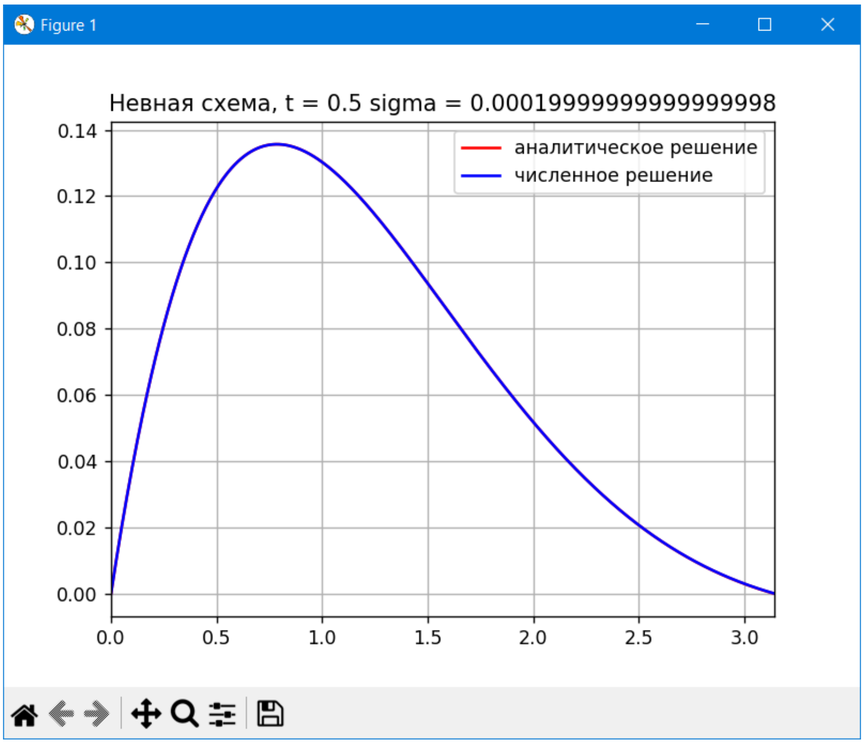
# 3. Вывод програмы

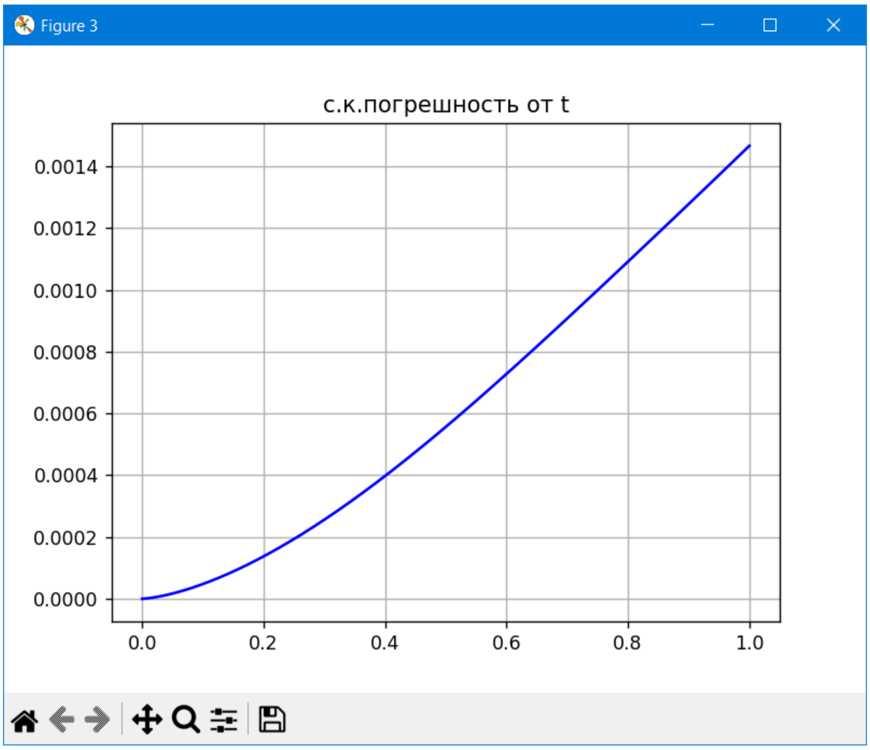
- Явная схема






- Неявная схема






# 4. Листинг

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

def psi(x):
    return 0

def dpsidt(x):
    return np.exp(-x)*np.sin(x)

def dpsidx(x):
    return 0

def dpsidxx(x):
    return 0

def phi0(t):
    return 0

def phi1(t):
    return 0

def true_fval(x, t):
    return 0.5*np.exp(-x)*np.sin(x)*np.sin(2*t)

```

```python
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,b, lb, ub, h, tau, T,apr):
    # 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])
    #заполним второй уровень
    if apr == 1:
        for j in range (len(x)):
            U[1,j] = U[0, j]+ tau*dpsidt(x[j])
    if apr == 2:
        for j in range (len(x)):
            U[1,j] = U[0, j]+tau*dpsidt(x[j])+(tau**2)*dpsidxx(x[j])+2*(tau**2)*dpsidx(x[j])
    # прямая схема
    for i in range(2, len(t)):
        for j in range(1, len(x)-1):
            U[i, j] = a*(tau**2)/(h**2)*U[i-1,j+1]+(2-2*a*(tau**2)/(h**2))*U[i-1,j]+a*(tau**2)/(h**2)*U[i-1,j-1]+b*(tau**2)/(2*h)*U[i-1,j+1]-b*(tau**2)/(2*h)*U[i-1,j-1]-U[i-2,j]
            U[i, 0] = phi0(t[i])
            U[i, -1] = phi1(t[i])

    return U


def ImScheme(a,b, lb, ub, h, tau, T,apr):
    x = np.arange(lb, ub + h, h)
```

```python
    t = np.arange(0, T + tau, tau)
    U = np.zeros((len(t), len(x)))
     # заполним первый уровень
    for j in range(len(x)):
        U[0, j] = psi(x[j])
    #заполним второй уровень
    if apr == 1:
        for j in range (len(x)):
            U[1,j] = U[0, j]+ tau*dpsidt(x[j])
    if apr == 2:
        for j in range (len(x)):
            U[1,j] = U[0, j]+ tau*dpsidt(x[j])+(tau**2)*dpsidxx(x[j])+2*(tau**2)*dpsidx(x[j])

    for i in range(2, 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] = -2*U[i-1,1]+U[i-2,1] - (a * (tau**2) / (h ** 2) - b*(tau**2)/(2*h)) * phi0(t[i])
        dd[-1] = -2*U[i-1,len(x)-1]+U[i-2,len(x)-1] - (a * (tau**2) / (h ** 2) + b*(tau**2)/(2*h)) * phi1(t[i])
        bb[0] = -(1 + 2 * a * (tau**2) / (h ** 2))
        bb[-1] = -(1 + 2 * a * (tau**2) / (h ** 2))
        cc[0] = a * (tau**2) / (h ** 2) + b*(tau**2)/(2*h)
        aa[-1] = a * (tau**2) / (h ** 2) - b*(tau**2)/(2*h)
        for j in range(1, len(x) - 2):
            aa[j] = a * (tau**2) / (h ** 2) - b*(tau**2)/(2*h)
            bb[j] = -(1 + 2 * a * (tau**2) / (h ** 2))
            cc[j] = a * (tau**2) / (h ** 2) + b*(tau**2)/(2*h)
            dd[j] = -2*U[i - 1, j+1] + U[i-2,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,b, lb, ub, h, tau, T,k,apr):
    x = np.arange(lb, ub + h, h)
    t = np.arange(0, T + tau, tau)
    sigma = a*(tau**2)/(h**2)
    plt.figure(1)
    plt.title('Явная схема, t = ' + str(t[k])+ ' sigma = ' + str(sigma))
    plt.grid()
    plt.plot(x, true_fval(x, t[k]), color='red', label='аналитическое решение')
    U = ExScheme(a,b, lb, ub, h, tau, T,apr)
    plt.plot(x, U[k, :], color='blue', label='численное решение')
```

```python
    plt.legend()
    plt.xlim((0, ub))

    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,b, lb, ub, h, tau, T,k,apr):
    x = np.arange(lb, ub + h, h)
    t = np.arange(0, T + tau, tau)
    sigma = a*(tau**2)/(h**2)
    plt.figure(1)
    plt.title('Невная схема, t = ' + str(t[k])+ ' sigma = ' + str(sigma))
    plt.grid()
    plt.plot(x, true_fval(x, t[k]), color='red', label='аналитическое решение')
    U = ImScheme(a,b, lb, ub, h, tau, T,apr)
    plt.plot(x, U[k, :], color='blue', label='численное решение')
    plt.legend()
    plt.xlim((0, ub))

    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(2,4,0,np.pi,0.05,0.00005,1,10000,2)
plot_im(2,4,0,np.pi,0.005,0.00005,1,10000,2)
```

# 5. Вывод

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