Московский авиационный институт

(национальный исследовательский университет)

Институт № 8 «Информационные технологии и прикладная математика»

Лабораторная работа № 6

По курсу «Численные методы»

«Численное решение уравнений с частными производными гиперболического типа»

Выполнил:

Гришин П. Ф.

Группа:

М8О-401Б-21

Преподаватель:

Ревизников Д. Л.

Москва, 2025

Задача

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).

Описание метода

Классический пример уравнения гиперболического типа – волновое уравнение, например уравнение описания процесса малых поперечных колебаний струны:

$$\frac{\partial^2 u}{\partial t^2} = a^2 \frac{\partial^2 u}{\partial x^2}, \quad 0 < x < l, \quad t = 0.$$

При движении концов струны по заданным законам имеем первую начально-краевую задачу для волнового уравнения:

$$\begin{cases} \dfrac{\partial^2 u}{\partial t^2} = a^2 \dfrac{\partial^2 u}{\partial x^2}, \quad 0 < x < l, \quad t > 0; \\ u(0,t) = \varphi_0(t), \quad x = 0, \quad t > 0; \\ u(l,t) = \varphi_l(t), \quad x = l, \quad t > 0; \\ u(x,0) = \psi_1(x), \quad 0 \le x \le l, \quad t = 0; \\ \dfrac{\partial u(x,0)}{\partial t} = \psi_2(x), \quad 0 \le x \le l, \quad t = 0, \end{cases}$$

Заметим, что помимо начального распределения искомой функции, задается еще и распределение начальной скорости перемещения.
Если на концах струны заданы значения силы, которая по закону Гука пропорциональна значениям производной перемещения по пространственной переменной (то есть на концах заданы значения первых производных по переменной x), то ставится вторая начально-краевая задача для волнового уравнения:

$$\begin{cases} \dfrac{\partial^2 u}{\partial t^2} = a^2 \dfrac{\partial^2 u}{\partial x^2}, \quad 0 < x < l, \quad t > 0; \\ \dfrac{\partial u(0,t)}{\partial x} = \varphi_0(t), \quad x = 0, \quad t > 0; \\ \dfrac{\partial u(l,t)}{\partial x} = \varphi_l(t), \quad x = l, \quad t > 0; \\ u(x,0) = \psi_1(x), \quad 0 \le x \le l, \quad t = 0; \\ \dfrac{\partial u(x,0)}{\partial t} = \psi_2(x), \quad 0 \le x \le l, \quad t = 0. \end{cases}$$

В случае же, если концы закреплены упруго, ставится третья начально-краевая задача:

$$
\begin{cases}
\dfrac{\partial^2 u}{\partial t^2} = a^2 \dfrac{\partial^2 u}{\partial x^2}, \quad 0 < x < l, \quad t > 0; \\
\alpha \dfrac{\partial u(0,t)}{\partial x} + \beta u(0,t) = \varphi_0(t), \quad x = 0, \quad t > 0; \\
\gamma \dfrac{\partial u(l,t)}{\partial x} + \delta u(l,t) = \varphi_l(t), \quad x = l, \quad t > 0; \\
u(x,0) = \psi_1(x), \quad 0 \le x \le l, \quad t = 0; \\
\dfrac{\partial u(x,0)}{\partial t} = \psi_2(x), \quad 0 \le x \le l, \quad t = 0.
\end{cases}
$$

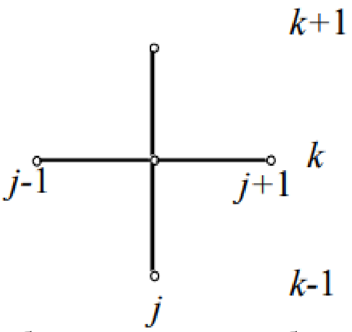
Рассмотрим первую начально-краевую задачу для волнового уравнения.
На пространственно-временной сетке будем аппроксимировать дифференциальное уравнение одной из конечно-разностных схем.

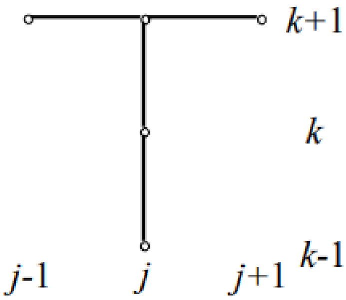
$$\frac{u_j^{k+1} - 2u_j^k + u_j^{k-1}}{\tau^2} = a^2 \frac{u_{j+1}^k - 2u_j^k + u_{j-1}^k}{h^2} + O\left(\tau^2 + h^2\right), \quad j = \overline{1, N-1}; \quad k = 1, 2, \ldots$$

получая явную схему, с помощью которой решение u^{k+1} определяется сразу, так как значения на нижних временных слоях должны быть известны. Порядок аппроксимации для этой схемы равен двум и по пространственной, и по временной переменным, а сама схема условно устойчива с условием $\sigma = a^2t^2/h^2$
Или же используя схему:

$$\frac{u_j^{k+1} - 2u_j^k + u_j^{k-1}}{\tau^2} = a^2 \frac{u_{j+1}^{k+1} - 2u_j^{k+1} + u_{j-1}^{k+1}}{h^2} + O\left(\tau + h^2\right), \quad j = \overline{1, N-1}; \quad k = 1, 2, \ldots$$

Получая неявную абсолютно устойчивую схему, которая сводится приведением к СЛАУ с трехдиагональной матрицей, решаемой методом прогонки.
Приведем шаблоны описанных схем:





В обоих вариантах необходимо знать значения u_j^{k-1} и u_j^k на нижних временных слоях.

Для k=1:

$u_j^0 = \psi_1(x_j)$, $j = \overline{0, N}$,

Для определения же u_j^1 воспользуемся простейшей аппроксимацией второго начального условия:

$$\frac{u_j^1 - u_j^0}{\tau} = \psi_2(x_j)$$

откуда получим выражение:

$$u_j^1 = \psi_1(x_j) + \psi_2(x_j)\tau$$

Для повышения порядка аппроксимации можно воспользоваться другим способом. Для этого разложим u_j^1 в ряд Тейлора:

$$u_j^1 = u(x_j, 0+\tau) = u_j^0 + \left.\frac{\partial u}{\partial t}\right|_j^0 \tau + \left.\frac{\partial^2 u}{\partial t^2}\right|_j^0 \frac{\tau^2}{2} + O(\tau^3)$$

и воспользуемся исходным уравнением:

$$\left.\frac{\partial^2 u}{\partial t^2}\right|_j^0 = a^2 \left.\frac{\partial^2 u}{\partial x^2}\right|_j^0 = a^2 \psi_1''(x_j)$$

откуда получим функцию со вторым порядком точности:

$$u_j^1 = \psi_1(x_j) + \psi_2(x_j)\tau + a^2 \psi_1''(x_j)\frac{\tau^2}{2}$$

Вариант

$$\frac{\partial^2 u}{\partial t^2} + 2\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + 2\frac{\partial u}{\partial x} - 3u,$$

$$u(0,t) = \exp(-t)\cos(2t),$$

$$u(\frac{\pi}{2}, t) = 0,$$

$$u(x,0) = \exp(-x)\cos x,$$

$$u_t(x,0) = -\exp(-x)\cos x.$$

Аналитическое решение: $U(x,t) = \exp(-t-x)\cos x \cos(2t)$

Решение и код

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

def phi0(t):
        return np.exp(-t)*np.cos(2*t)
         def phi1(t):
             return 0
        def psi1(x):
        return np.exp(-x)*np.cos(x)
def psi2(x):
        return -np.exp(-x) * np.cos(x)
def solution(x, t):
        return np.exp(-t-x)*np.cos(x)*np.cos(2*t)

def ddpsi1(x):
        return -np.exp(-x) * np.sin(x) - np.exp(-x) * np.cos(x)
def dpsi1(x):
        return 2 * np.exp(-x) * np.sin(x)

#/////////////////////////////////////////////////////

N = 10
K = 50
T = 1
l = np.pi / 2 h =
l/N
tau = T/K
sigma = tau**2/(h ** 2) x =
np.linspace(0,l,N)
```

```
t = np.linspace(0,T,K)
Xp, Tp = np.meshgrid(x, t)

print(sigma)
print(h)
```

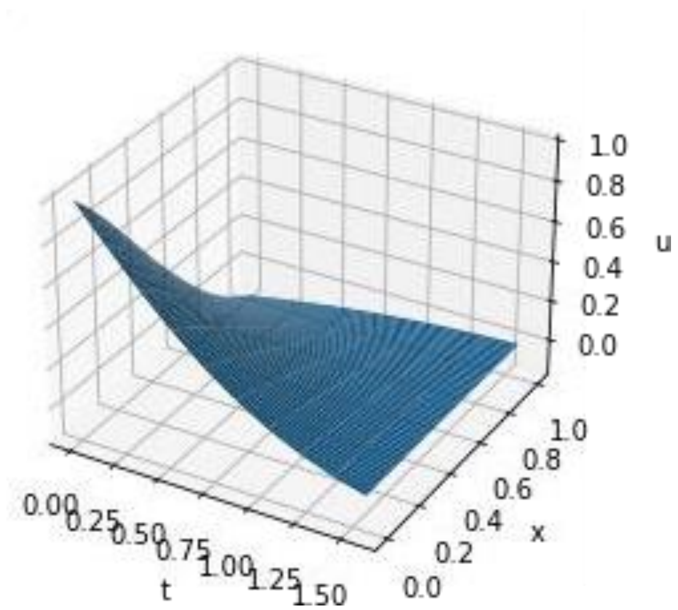
#///

0.016211389382774045
0.15707963267948966

#//

```
def analitic(x,t): u =
        [0]*K
        for i in range(K): u[i] =
                [0]*N
        for i in range(K):
                for j in range(N):
                        u[i][j] = solution(x[j], t[i])
        return u

u = analitic(x,t) fig =
plt.figure()
ax = plt.axes(projection ='3d') ax.set_xlabel('t')
ax.set_ylabel('x') ax.set_zlabel('u')
ax.plot_surface(Xp,Tp,np.array(u)) plt.show()
```

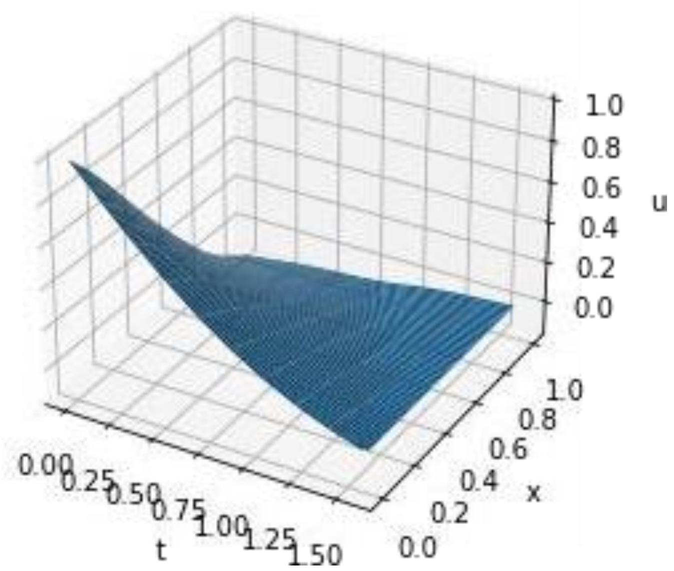
#//



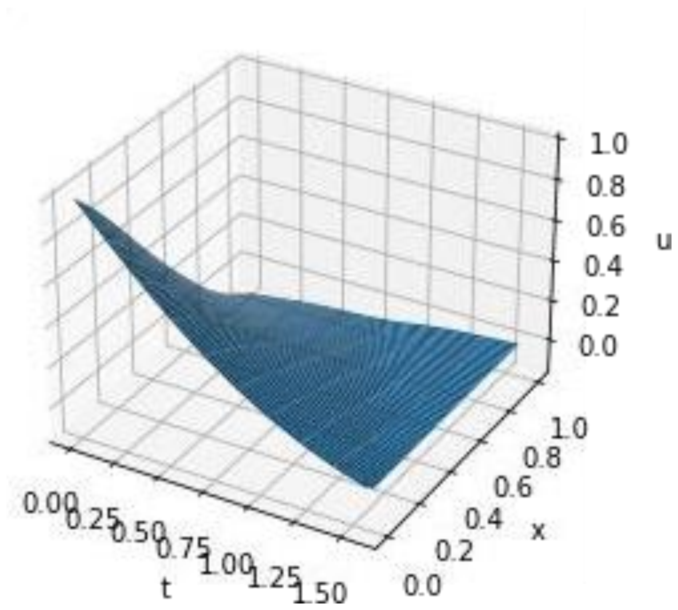
#//

```
def explicit_solve(l, N, K, T, app):  u = [0]*K
        for i in range(K): u[i] =
                [0]*N
        for k in range(K):
                u[k][0] = phi0(tau * k)
                u[k][-1] = phi1(tau * k)
        for j in range(N): u[0][j] = psi1(j
                * h)
```

```
            if app == 1:
                u[1][j] = psi1(j * h) + tau * psi2(j * h)
            if app == 2:
                u[1][j] = psi1(j * h) + tau * psi2(j * h) + tau**2*(ddpsi1(j*h)+2
*dpsi1(j*h)-3*psi1(j*h))/2
    for k in range(2, K):
        for j in range(1, N - 1):
            u[k][j] = ((-3*tau**2-2*tau**2/h**2+2+2*tau)*u[k-1][j]+(tau**2/h*
*2+tau**2/h)*u[k-1][j+1]+(tau**2/h**2-tau**2/h)*u[k-1][j-1]-u[k-2][j])/(1+2*t    au)
    return(u)

#////////////////////////////////////////////////////

#Аппроксимация второго начального условия с первым порядком

exp1 = explicit_solve(l, N, K, T, 1) fig =
plt.figure()
ax = plt.axes(projection ='3d') ax.set_xlabel('t')
ax.set_ylabel('x') ax.set_zlabel('u')
ax.plot_surface(Xp, Tp, np.array(exp1)) plt.show()
#////////////////////////////////////////////////////
```

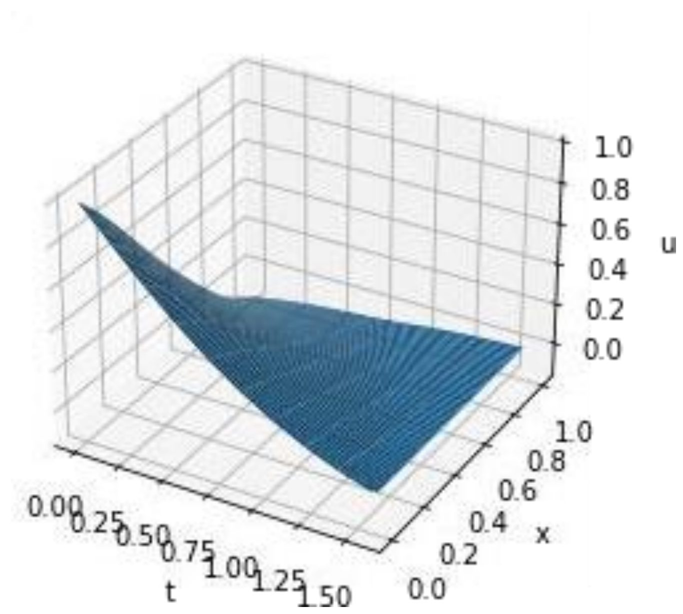



```
#////////////////////////////////////////////////////
#Аппроксимация второго начального условия со вторым порядком

exp2 = explicit_solve(l, N, K, T, 2) fig =
plt.figure()
ax = plt.axes(projection ='3d') ax.set_xlabel('t')
ax.set_ylabel('x') ax.set_zlabel('u')
ax.plot_surface(Xp, Tp, np.array(exp2)) plt.show()

#////////////////////////////////////////////////////
```




```
#//////////////////////////////////////////////////
#неявная схема

def tma(a, b, c, d): n = len(a)
    p, q = [], []
    p.append(-c[0] / b[0])
    q.append(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 = [0 for _ in range(n)]
    x[n - 1] = q[n - 1]
    for i in range(n-2, -1, -1):
        x[i] = p[i] * x[i+1] + q[i]
    return x

def implicit_solve():
    a = np.zeros(N)
    b = np.zeros(N)
    c = np.zeros(N)
    d = np.zeros(N) u =
    [0]*K
    for i in range(K): u[i] =
        [0]*N
    for j in range(N): u[0][j] = psi1(j
        * h)
        u[1][j] = psi1(j * h) + tau * psi2(j * h)
    for k in range(2, K):
        for j in range(1, N-1): a[j] = 2 - h * 2
            b[j] = 2 * (h ** 2) * (-2 / (2 * tau) - 1 / (tau ** 2) - 3) - 4 c[j] = h * 2 + 2
            d[j] = -4*(h**2)*u[k - 1][j] / (tau ** 2) + (2 * (h ** 2) / (tau
** 2) - 2 * (h ** 2) / tau) * u[k - 2][j] a[0] = 0
        b[0] = h
        c[0] = 0
```

```
        d[0] = phi0((k)*tau)*h a[-1] =
        0
        b[-1] = h
        c[-1] = 0
        d[-1] = phi1(tau*(k))*h

        u[k] = tma(a, b, c, d)
    return u

imp = implicit_solve()

fig = plt.figure()
ax = plt.axes(projection ='3d') ax.set_xlabel('t')
ax.set_ylabel('x') ax.set_zlabel('u')
ax.plot_surface(Xp, Tp, np.array(imp)) plt.show()
#//////////////////////////////////////////////////////
```




```
#//////////////////////////////////////////////////////
def pogr(res):
    return np.sqrt(sum([sum([(u[i][j]-res[i][j])**2 for j in range(len(x))])
for i in range(len(t))])) pogr(exp1)

#//////////////////////////////////////////////////////

0.5272028672614416

#/////////////////////////////////////////////////

pogr(exp2)

#/////////////////////////////////////////////////

0.5135453802385277

#////////////////////////////////////////////////

pogr(imp)

#//////////////////////////////////////////////////////

0.5371863521392726

#//////////////////////////////////////////////////////
```

Выводы

В ходе выполнения данной работы для решения начально-краевой задачи для дифференциального уравнения гиперболического типа были реализованы явная схема крест и неявная схема. Также реализованы два варианта аппроксимации второго начального условия - с первым и со вторым порядком. Для вычисления погрешности была найдена среднеквадратическая ошибка при помощи сравнения результатов с приведенным точным решением.