**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 8**

**УРАВНЕНИЯ ГИПЕРБОЛИЧЕСКОГО ТИПА**

**(Вариант 5)**

*Выполнил студент 3 курса ПМ*

*Бут Дмитрий*

***Постановка задачи:***

***Цель работы***: усвоить сущность и методы решения ***линейного дифференциального уравнения 2-го порядка гиперболического типа***.

Численное решение дифференциального уравнения в частных производных предполагает получение двумерной числовой таблицы приближенных значений *Uij* искомой функции *U*(*t,x)* с заданной точностью для некоторых значений аргументов

*xj ∈* [*a*, *b*], *ti ∈* [*c*, *d*]

Численное решение таких дифференциальных уравнений находят методами конечных разностей.

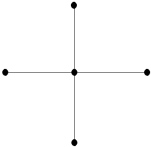
Погрешность решения, найденного этими методами, оценивается величиной O(*τ p, h q*)*,* где *p*, *q* - порядок аппроксимации метода.

***Задание.***

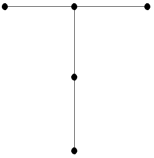
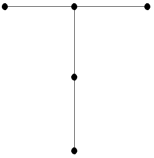
Решить волновое уравнение

явным методом и неявными методами второго порядка точности

Шаблон для явного метода:



Шаблон для неявного метода:



Вывести результаты в виде трёхмерных графиков U(x,t).

Неявные схемы решать с помощью прогонки.

**Метод прогонки РАСПИСАТЬ подробно!**

Входные данные:

[*a*, *b*] = [0; 1], [*c*, *d*] = [0; 10], *f(x,t)*=0

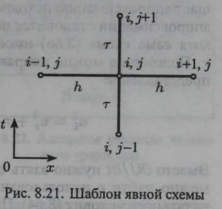
Погрешность решения 0,01



***Результаты расчетов***

**Явный метод (явная трёхслойная схема типа крест).**

Используемый шаблон:



Заменим в исходном уравнении вторые производные искомой функции по и их конечно-разностными соотношениями с помощью значений сеточной функции в узлах сетки ():

Отсюда можно найти явное выражение для значения сеточной функции на (j+ 1)-м слое:

Где .

Здесь, как обычно в трехслойных схемах, для определения неизвестных значений на (j + 1)-м слое нужно знать решения на j-м и (j - 1)-м слоях. Поэтому начать счет по формулам можно лишь для второго слоя, а решения на нулевом и первом слоях должны быть известны.

Они находятся с помощью начальных условий.

На нулевом слое имеем:

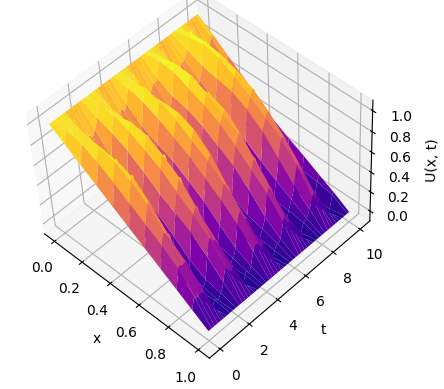
Для получения решения на первом слое воспользуемся вторым начальным условием. Производную заменим конечно-разностной аппроксимацией. В простейшем случае полагают:

Из этого соотношения можно найти значения сеточной функции на первом временном слое:

Рассмотренная разностная схема решения задачи условно устойчива. Необходимое и достаточное условие устойчивости имеет вид:

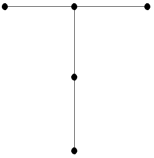
Отсюда возьмём

Получив точки, построим трёхмерный график:



**Неявный метод.**

Используем шаблон:



Заменим в исходном уравнении вторые производные искомой функции по и их конечно-разностными соотношениями с помощью значений сеточной функции в узлах сетки ():

Введём параметр :

Приведём к стандартному трёхдиагональному виду:

(1)

Тогда , , ,

Применим метод прогонки:

Пусть , тогда

Используя это соотношение, получим выражение через :

Подставим в уравнение (1):

где  — правая часть *i*-го уравнения. Это соотношение будет выполняться независимо от решения, если потребовать:

Отсюда следует:

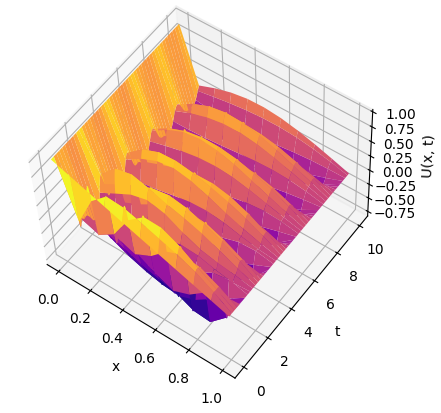
Здесь для определения неизвестных значений на (i + 1)-м слое нужно знать решения на i-м слое. Поэтому начать счет по формулам можно лишь для первого слоя, а решения на нулевом слое должны быть известны. Для этого используем начальные условия

Имеем:

Тогда

Также имеем граничное условие, задающее значение функции в последнем узле:

Получив точки, построим трёхмерный график:



**Неявный метод.**

Используем шаблон:



Вторую производную по t в уравнение аппроксимируем, как и ранее, по трехточечному шаблону с помощью значений сеточной функции на слоях j - 1, j, j + 1.

Производную по х заменяем полусуммой ее аппроксимации на (j + 1)-м

и (j - 1)-м слоях (рис. 8.23):

Из этого соотношения можно получить систему уравнений относительно неизвестных значений сеточной функции на (j + 1)-м слое:

(2)

Где

Полученная неявная схема устойчива и сходится со скоростью . Систему линейных алгебраических уравнений можно, в частности, решать методом прогонки.

Тогда , , ,

Применим метод прогонки:

Пусть , тогда

Используя это соотношение, получим выражение через :

Подставим в уравнение (1):

где  — правая часть *i*-го уравнения. Это соотношение будет выполняться независимо от решения, если потребовать:

Отсюда следует:

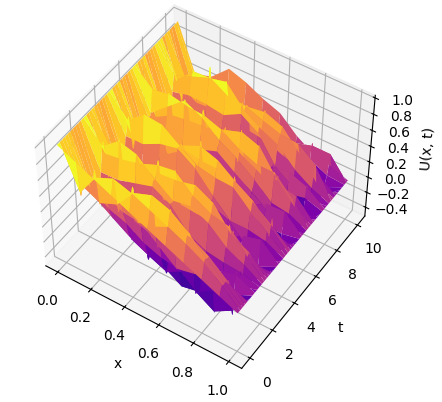
Здесь для определения неизвестных значений на (i + 1)-м слое нужно знать решения на i-м слое. Поэтому начать счет по формулам можно лишь для первого слоя, а решения на нулевом слое должны быть известны. Для этого используем начальные условия

Имеем:

Тогда

Также имеем граничное условие, задающее значение функции в крайнем узле:

Получив точки, построим трёхмерный график:



**ПРИЛОЖЕНИЕ**

***Программа реализации явного метода***

import matplotlib.pyplot as plt  
import numpy as np  
  
  
def urmat(ux0, ux1, ut0, ut1, D, h, tau):  
 x = np.arange(0, 1 + h, h)  
 t = np.arange(0, 10 + tau, tau)  
 u = np.zeros((len(x), len(t)))  
 for i in range(len(x)):  
 u[i][0] = ux0(x[i])  
 u[i][1] = u[i][0] + tau \* ux1(x[i])  
 for j in range(len(t)):  
 u[0][j] = ut0(t[j])  
 u[-1][j] = ut1(t[j])  
 l = (D \* tau / h) \*\* 2  
 for j in range(1, len(t) - 1):  
 for i in range(1, len(x) - 1):  
 u[i][j + 1] = 2 \* (1 - l) \* u[i][j] + l \* (u[i + 1][j] + u[i - 1][j]) - u[i][j - 1]  
  
 X, T = np.meshgrid(x, t)  
 fig = plt.figure()  
 ax = fig.add\_subplot(111, projection='3d')  
 ax.plot\_surface(X, T, u.T, cmap='plasma')  
 ax.set\_xlabel('x')  
 ax.set\_ylabel('t')  
 ax.set\_zlabel('U(x, t)')  
 plt.show()  
  
  
ux0 = lambda x: x \*\* 2  
ux1 = lambda x: 2  
ut0 = lambda t: 0  
ut1 = lambda t: 1  
D = 1  
h = 0.1  
tau = 0.01  
urmat(ux0, ux1, ut0, ut1, D, h, tau)

***Программа реализации неявного метода (неявная схема «Т»)***

import matplotlib.pyplot as plt  
import numpy as np  
  
  
def urmat(ux0, ux1, ut0, ut1, d, h, tau):  
 x = np.arange(0, 1 + h, h)  
 t = np.arange(0, 10 + tau, tau)  
 u = np.zeros((len(x), len(t)))  
  
 for i in range(len(x)):  
 u[i][0] = ux0(x[i])  
 u[i][1] = tau \* ux1(x[i]) + u[i][0]  
 for j in range(len(t)):  
 u[0][j] = ut0(t[j])  
 u[-1][j] = ut1(t[j])  
 l = (d \* tau / h)\*\*2  
 for j in range(1, len(t) - 1):  
 D = [u[0][j + 1]]  
 for i in range(1, len(x) - 1):  
 D.append(-2 \* u[i][j] + u[i][j - 1])  
 a, b = [0], [0]  
 for i in range(1, len(x) - 1):  
 a.append(-l / (l \* a[i - 1] - (2 \* l + 1)))  
 b.append((D[i] - l \* b[i - 1]) / (l \* a[i - 1] - (2 \* l + 1)))  
 for i in range(len(x) - 2, 0, -1):  
 u[i][j + 1] = a[i] \* u[i + 1][j + 1] + b[i]  
 X, T = np.meshgrid(x, t)  
 fig = plt.figure()  
 ax = fig.add\_subplot(111, projection='3d')  
 ax.plot\_surface(X, T, u.T, cmap='plasma')  
 ax.set\_xlabel('x')  
 ax.set\_ylabel('t')  
 ax.set\_zlabel('U(x, t)')  
 plt.show()  
  
  
ux0 = lambda x: x\*\*2  
ux1 = lambda x: 2  
ut0 = lambda t: 0  
ut1 = lambda t: 1  
D = 1  
h = 0.1  
tau = 0.01  
urmat(ux0, ux1, ut0, ut1, D, h, tau)

***Программа реализации неявного метода***

import matplotlib.pyplot as plt  
import numpy as np  
  
  
def urmat(ux0, ux1, ut0, ut1, d, h, tau):  
 x = np.arange(0, 1 + h, h)  
 t = np.arange(0, 10 + tau, tau)  
 u = np.zeros((len(x), len(t)))  
 for i in range(len(x)):  
 u[i][0] = ux0(x[i])  
 u[i][1] = tau \* ux1(x[i]) + u[i][0]  
 for j in range(len(t)):  
 u[0][j] = ut0(t[j])  
 u[-1][j] = ut1(t[j])  
 l = ((d \* tau / h)\*\*2) / 2  
 for j in range(1, len(t) - 1):  
 D = [u[0][j + 1]]  
 for i in range(1, len(x) - 1):  
 D.append((1 + 2 \* l) \* u[i][j - 1] - l \* (u[i + 1][j - 1] + u[i - 1][j - 1]) - 2 \* u[i][j])  
 a, b = [0], [0]  
 for i in range(1, len(x) - 1):  
 a.append(-l / (l \* a[i - 1] - (1 + 2 \* l)))  
 b.append((D[i] - l \* b[i - 1]) / (l \* a[i - 1] - (1 + 2 \* l)))  
 for i in range(len(x) - 2, 0, -1):  
 u[i][j + 1] = a[i] \* u[i + 1][j + 1] + b[i]  
  
 X, T = np.meshgrid(x, t)  
 fig = plt.figure()  
 ax = fig.add\_subplot(111, projection='3d')  
 ax.plot\_surface(X, T, u.T, cmap='plasma')  
 ax.set\_xlabel('x')  
 ax.set\_ylabel('t')  
 ax.set\_zlabel('U(x, t)')  
 plt.show()  
  
  
ux0 = lambda x: x\*\*2  
ux1 = lambda x: 2  
ut0 = lambda t: 0  
ut1 = lambda t: 1  
D = 1  
h = 0.1  
tau = 0.01  
urmat(ux0, ux1, ut0, ut1, D, h, tau)