**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 8**

**УРАВНЕНИЯ ГИПЕРБОЛИЧЕСКОГО ТИПА**

**(Вариант 10)**

*Выполнил студент 3 курса МОиАИС*

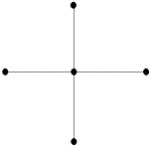
*Соколов Арсений*

**Задание**

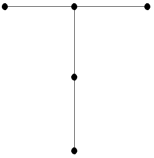
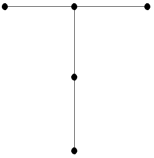
Решить волновое уравнение

явным методом и неявными методами второго порядка точности

Шаблон для явного метода:



Шаблон для неявного метода:

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

Неявные схемы решать с помощью прогонки.

Метод прогонки РАСПИСАТЬ подробно!

**Дано**

[*a*, *b*] = [0; 1],

[*c*, *d*] = [0; 10],

*f(x,t )*= 0,

Погрешность решения 0,01 (исходя из погрешности, порядка аппроксимации и условий сходимости для явных схем определить шаги по пространству и времени)

|  |  |  |  |
| --- | --- | --- | --- |
| № вар. | Начальные условия | Граничные условия | a |
| 10 |  |  | 1 |

**Решение**

*Схема крест (явная)*

Разностное уравнение

*,*

где

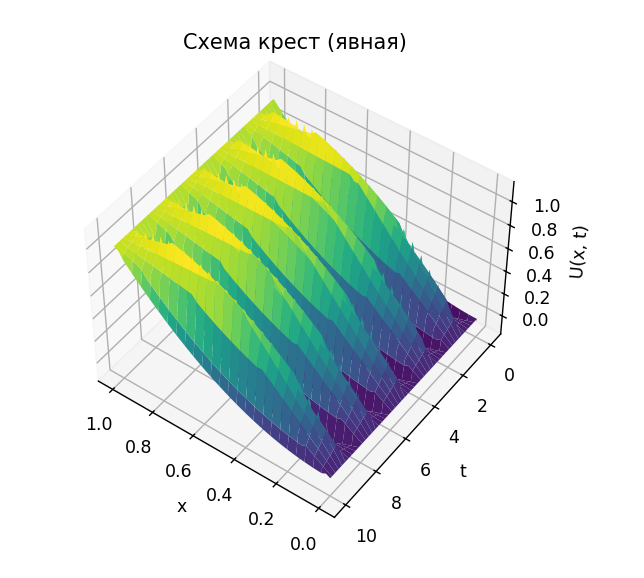
,

где

Решение начинается со второго слоя, на нулевом и первом слоях значения известны из начальных условий

На первом слое аппроксимируем производную

Тогда



*Схема 2 (неявная)*

Разностное уравнение

*(по времени на слое i) (по пространству на слое j+1)*

Неизвестные Из этого разностного соотношения получим систему уравнений относительно сеточной функции на *j+1-м* слое (*j = 1, 2 …*)

,

где   
Данную систему будем решать методом прогонки.

Найдем неизвестные помощью рекуррентного соотношения

Для этого выразим их и подставим в уравнение

,

получим

Это соотношение будет выполняться независимо от решения, если

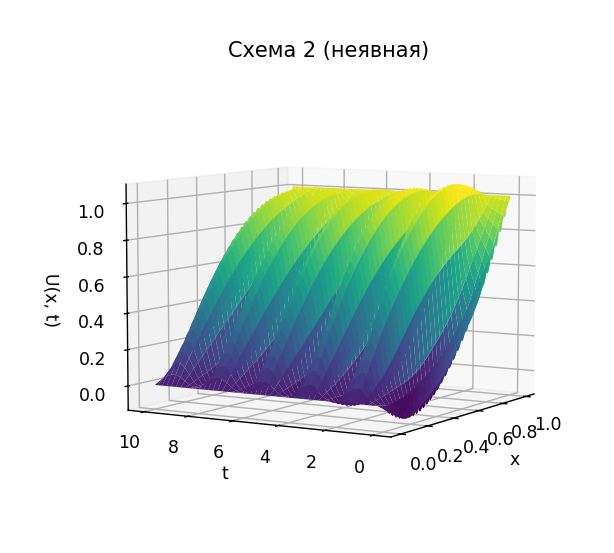
;

Граничные условия

;

Из начальных условий

Тогда



*Схема 3 (неявная)*

Разностное уравнение

,

где

Данную систему будем решать методом прогонки.

Найдем неизвестные помощью рекуррентного соотношения

Для этого

,

где

Определим коэффициенты прогонки и

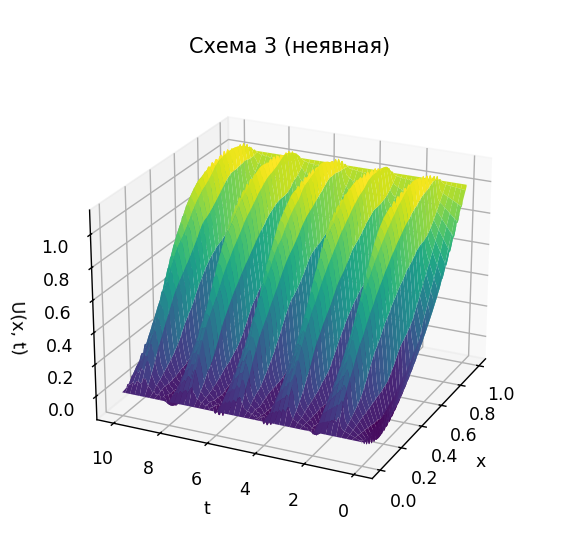
* (из граничного условия )
* ;

для

Учтем второе граничное условие

Из начальных условий

Тогда

**

**ПРИЛОЖЕНИЕ 1**

***Программа решения волнового уравнения по схеме крест***

import numpy as np  
import matplotlib.pyplot as plt  
from mpl\_toolkits.mplot3d import Axes3D  
  
# Параметры задачи  
a, b = 0, 1  
c, d = 0, 10  
a\_squared = 1 # Уточните значение a^2, если оно задано  
N = 50 # Число узлов по x  
M = 500 # Число узлов по t  
h = (b - a) / N  
tau = (d - c) / M  
lambda\_ = (a\_squared \* tau \*\* 2) / h \*\* 2  
  
# Инициализация сетки  
u = np.zeros((M + 1, N + 1))  
  
# Начальные условия  
x = np.linspace(a, b, N + 1)  
u[0, :] = x \*\* 2  
u[1, :] = x \*\* 2 - tau  
  
# Применение явной схемы "крест"  
for j in range(1, M):  
 for i in range(1, N):  
 u[j+1, i] = 2 \* (1 - lambda\_) \* u[j, i] + lambda\_ \* (u[j, i+1] + u[j, i-1]) - u[j-1, i]  
  
 # Граничные условия  
 u[j+1, 0] = 0  
 u[j+1, N] = 1  
  
# Визуализация результатов  
t = np.linspace(c, d, M + 1)  
X, T = np.meshgrid(x, t)  
  
fig = plt.figure()  
ax = fig.add\_subplot(111, projection='3d')  
ax.plot\_surface(X, T, u, cmap='viridis')  
  
ax.set\_xlabel('x')  
ax.set\_ylabel('t')  
ax.set\_zlabel('U(x, t)')  
ax.set\_title('Схема крест (явная)')  
  
plt.show()

**ПРИЛОЖЕНИЕ 2**

***Программа решения волнового уравнения по схеме 2***

import numpy as np  
import matplotlib.pyplot as plt  
from mpl\_toolkits.mplot3d import Axes3D  
  
# Параметры задачи  
a, b = 0, 1  
c, d = 0, 10  
D = 1 # Уточните значение D, если оно задано  
N = 100 # Число узлов по x  
M = 1000 # Число узлов по t  
h = (b - a) / N  
tau = (d - c) / M  
lambda\_ = (D \* tau / h) \*\* 2  
  
# Инициализация сетки  
u = np.zeros((M + 1, N + 1))  
  
# Начальные условия  
for i in range(N + 1):  
 x\_i = i \* h  
 u[0, i] = x\_i \*\* 2  
  
for i in range(N + 1):  
 x\_i = i \* h  
 u[1, i] = x\_i \*\* 2 - tau  
  
# Прогонка  
for j in range(1, M):  
 alpha = np.zeros(N + 1)  
 beta = np.zeros(N + 1)  
  
 # Прямой ход  
 for i in range(1, N):  
 alpha[i] = lambda\_ / (1 + 2 \* lambda\_ - lambda\_ \* alpha[i - 1])  
 beta[i] = (2 \* u[j, i] - u[j - 1, i] + lambda\_ \* beta[i - 1]) / (1 + 2 \* lambda\_ - lambda\_ \* alpha[i - 1])  
  
 # Граничное условие на правой границе  
 u[j + 1, N] = 1  
  
 # Обратный ход  
 for i in range(N - 1, 0, -1):  
 u[j + 1, i] = alpha[i] \* u[j + 1, i + 1] + beta[i]  
  
 # Граничное условие на левой границе  
 u[j + 1, 0] = 0  
  
# Визуализация результатов  
  
x = np.linspace(a, b, N + 1)  
t = np.linspace(c, d, M + 1)  
X, T = np.meshgrid(x, t)  
  
fig = plt.figure()  
ax = fig.add\_subplot(111, projection='3d')  
ax.plot\_surface(X, T, u, cmap='viridis')  
  
ax.set\_xlabel('x')  
ax.set\_ylabel('t')  
ax.set\_zlabel('U(x, t)')  
ax.set\_title('Схема 2 (неявная)')  
  
plt.show()

**ПРИЛОЖЕНИЕ 3**

***Программа решения волнового уравнения по схеме 3***

import numpy as np  
import matplotlib.pyplot as plt  
from mpl\_toolkits.mplot3d import Axes3D  
  
# Параметры задачи  
a, b = 0, 1  
c, d = 0, 10  
a\_squared = 1 # Уточните значение a^2, если оно задано  
N = 100 # Число узлов по x  
M = 1000 # Число узлов по t  
h = (b - a) / N  
tau = (d - c) / M  
lambda\_ = (a\_squared \* tau \*\* 2) / (2 \* h \*\* 2)  
  
# Инициализация сетки  
u = np.zeros((M + 1, N + 1))  
  
# Начальные условия  
x = np.linspace(a, b, N + 1)  
u[0, :] = x \*\* 2  
u[1, :] = x \*\* 2 - tau  
  
# Прогонка  
for j in range(1, M):  
 alpha = np.zeros(N + 1)  
 beta = np.zeros(N + 1)  
  
 # Прямой ход  
 for i in range(1, N):  
 alpha[i + 1] = -lambda\_ / (lambda\_ \* alpha[i] - (1 + 2 \* lambda\_))  
 beta[i + 1] = ((1 + 2 \* lambda\_) \* u[j - 1, i] - lambda\_ \* (u[j - 1, i + 1] + u[j - 1, i - 1]) - 2 \* u[  
 j, i] - lambda\_ \* beta[i]) / (lambda\_ \* alpha[i] - (1 + 2 \* lambda\_))  
  
 # Граничное условие на правой границе  
 u[j + 1, N] = 1  
  
 # Обратный ход  
 for i in range(N - 1, 0, -1):  
 u[j + 1, i] = alpha[i + 1] \* u[j + 1, i + 1] + beta[i + 1]  
  
 # Граничное условие на левой границе  
 u[j + 1, 0] = 0  
  
# Визуализация результатов  
t = np.linspace(c, d, M + 1)  
X, T = np.meshgrid(x, t)  
  
fig = plt.figure()  
ax = fig.add\_subplot(111, projection='3d')  
ax.plot\_surface(X, T, u, cmap='viridis')  
  
ax.set\_xlabel('x')  
ax.set\_ylabel('t')  
ax.set\_zlabel('U(x, t)')  
ax.set\_title('Схема 3 (неявная)')  
  
plt.show()