Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Отчёт по лабораторной работе №16

По дисциплине «Методы численного анализа»

По теме «Метод сеток решения волнового уравнения»

Вариант 6

Выполнил:

студент гр. 653504

Куликов А.Д.

Проверил:

Анисимов В.Я.

Минск 2018

**Цель работы**

Изучить метод разностных аппроксимаций для волнового уравнения; Составить программы волнового уравнения по разработанным алгоритмам;

Получить численное решение волнового уравнения.

## **Краткие теоретические сведения**

**Разностная схема для одномерного волнового уравнения**

Рассмотрим одномерную математическую модель распространения колебаний на струне. Пусть струна в деформированном состоянии распространяется на интервале оси x и u(x, t) — перемещение по времени в направлении y точки, изначально лежащей на оси x. Функция перемещения u(x, t) определяется следующей математической моделью:

|  |  |  |
| --- | --- | --- |
|  |  | (1) |
|  |  |  |
|  |  | (2) |

|  |  |  |
| --- | --- | --- |
|  |  | (3) |
|  |  |  |
|  |  | (4) |

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

Постоянная c и функция I(x) — заданы.

Уравнение (1) известно, как волновое уравнение (уравнение колебаний струны). Так как это уравнение в частных производных содержит вторую производную по времени, необходимо задать два начальных условия. Условие (2) начальную форму струны, а условие (3) означает, что начальная скорость струны равна нулю. Кроме того, уравнение (1) дополняется граничными условиями (4) и (5). Эти два условия означают, что струна закреплена на концах, т.е. перемещения равны нулю.

## **Расчетная сетка**

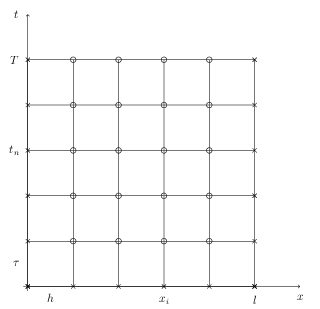
Для построения разностной схемы надо прежде всего ввести сетку в области изменения независимых переменных и задать шаблон, т.е. множество точек сетки, участвующих в аппроксимации дифференциального выражения. Введем равномерную сетку по переменному x с шагом h

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

и сетку по переменному t с шагом τ:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Точки  , образуют узлы пространственно-временной сетки



Разностная схема.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | ◦ |  |
|  | ◦ | ◦ | ◦ |
|  |  | ◦ |  |
|  |  |  |  |

## **Разностная схема**

Простейшей разностной аппроксимацией уравнения (1) и граничных условий (4) и (5) является следующая система уравнений:

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

Разностное уравнение (6) имеет второй порядок погрешности аппроксимации по τ и по h. Решение  выражается явным образом через значения на предыдущих слоях:

|  |  |  |
| --- | --- | --- |
|  |  | (8) |

Здесь мы ввели параметр

|  |  |  |
| --- | --- | --- |
|  |  |  |

который называют *числом Куранта*.

**Первое начальное условие:**

|  |  |  |
| --- | --- | --- |
|  |  | (9) |

## **Аппроксимация второго начального условия**

Построим такую аппроксимацию. Уравнение

|  |  |  |
| --- | --- | --- |
|  |  | (10) |

аппроксимирует уравнение  со вторым порядком.

Из (10) имеем . Отсюда получаем

|  |  |  |
| --- | --- | --- |
|  |  | (11) |
|  |  |  |

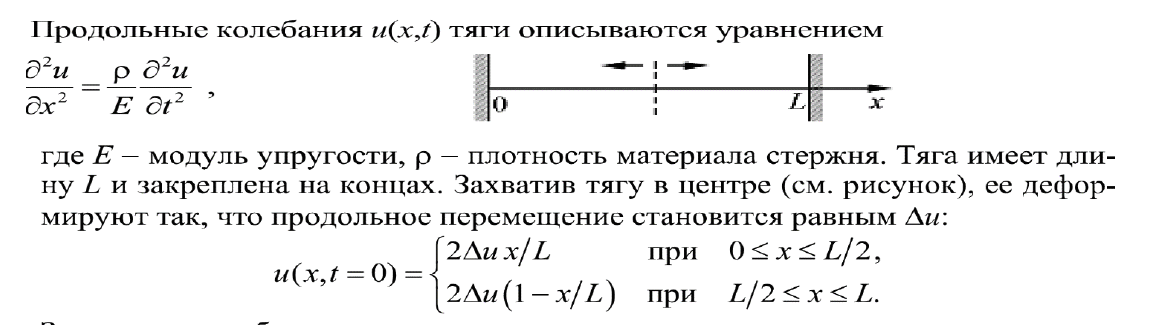
Совокупность уравнений (6), (7), (9) и (11) составляет разностную схему, аппроксимирующую исходную задачу (1) – (5).

## **Вычислительный алгоритм**

Теперь мы можем сформулировать вычислительный алгоритм:

1. Вычисляем , используя (9).
2. Вычисляем , используя  (11) и задаем граничные условия (7) при n=0.
3. Для всех временных слоев n=1,2,…,K−1
   1. находим, используя (8)
   2. задаем граничные условия (7).

**Задание 1**



Затем тяга освобождается. Рассчитать колебания при заданных параметрах

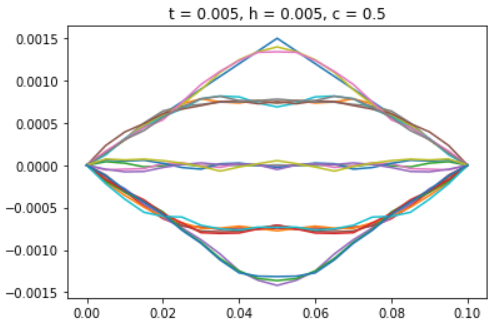
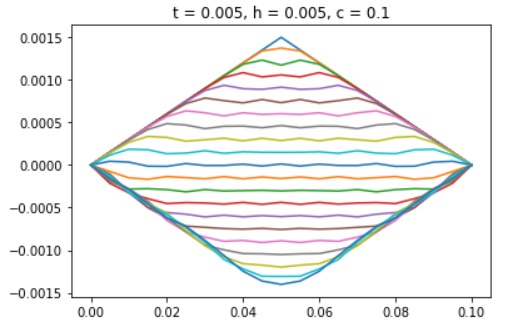
**Решение**

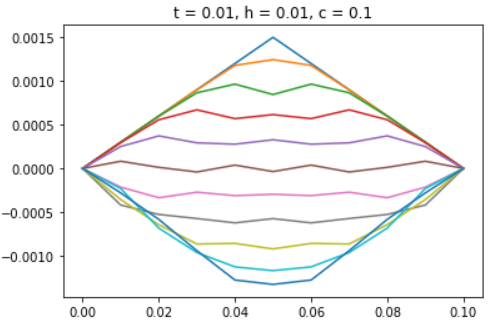
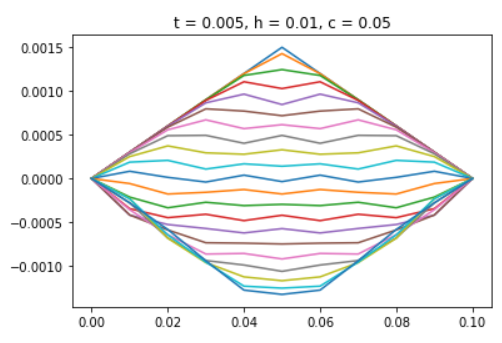
Составим разностную схему решения волнового уравнения:

. Отсюда

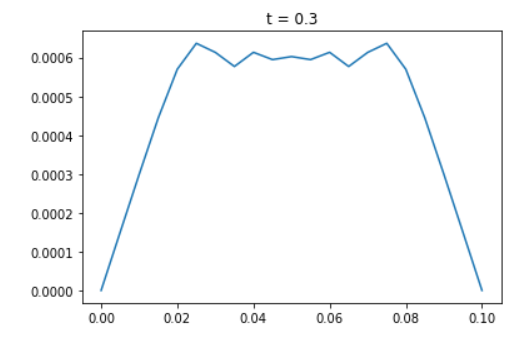
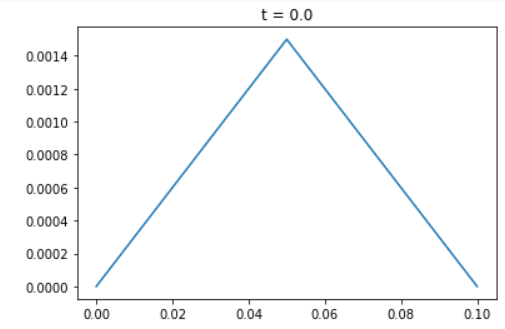
Подставив начальные условия, найдем значения сетки с помощью явной схемы.

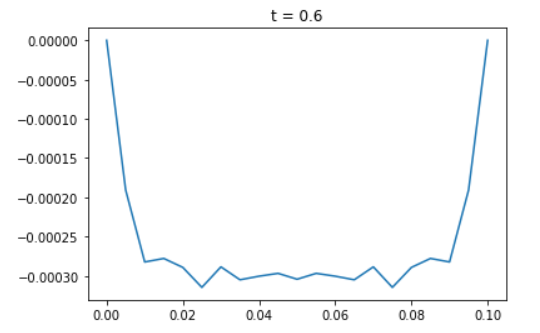
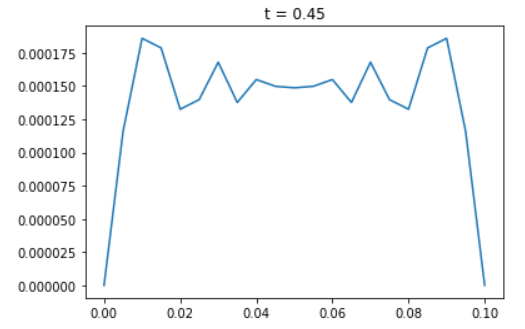
Результат работы программы разных шагах по

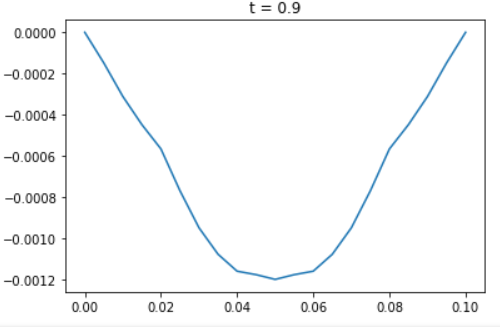
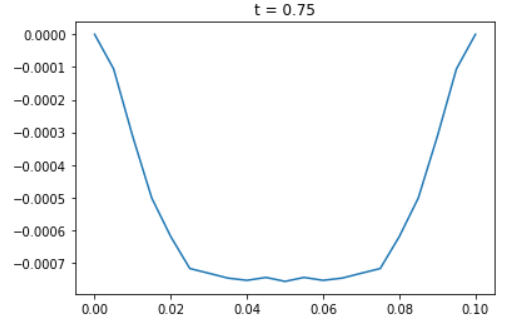




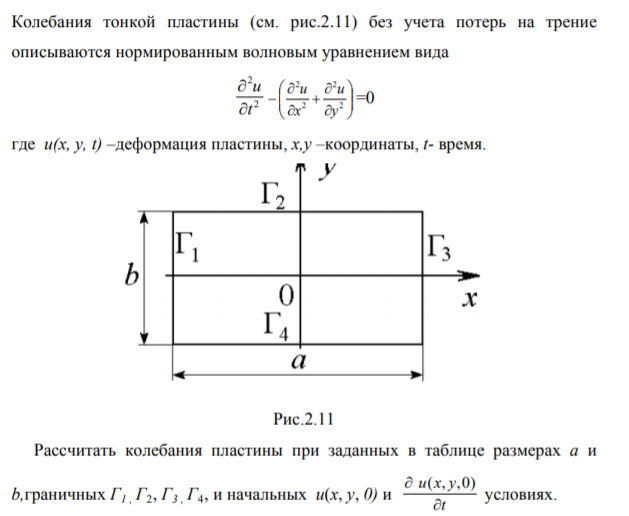
Колебания струны в разные моменты времени при :







**Задание 2**

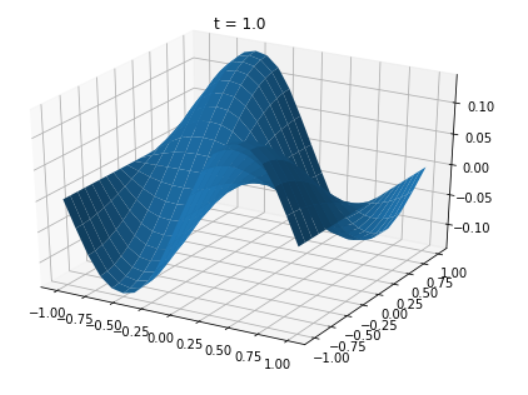
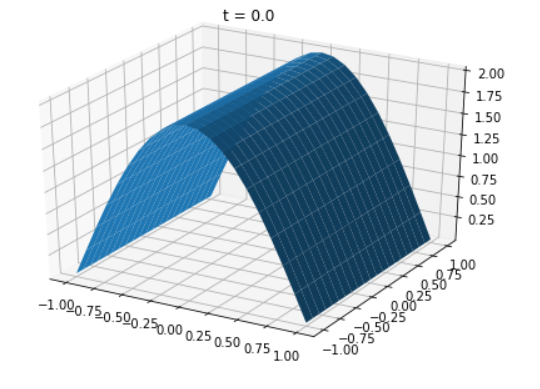


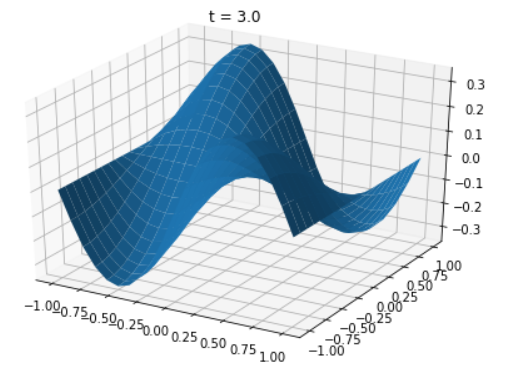
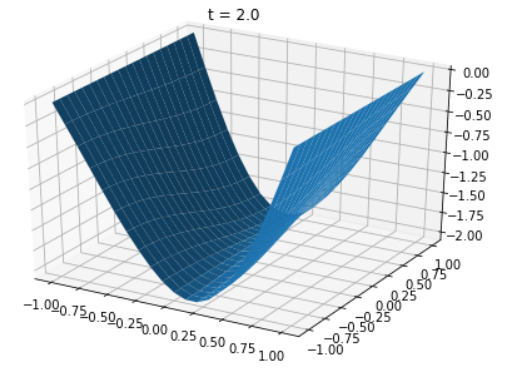
**Решение**

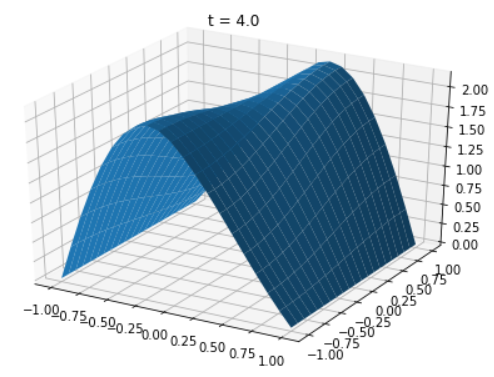
**Граничные условия первого рода**

**Аппроксимация граничных условий второго рода**

Результат работы программы при в разные моменты времени:







**Вывод**

В результате данной работы был изучен метод сеток численного решения одномерного и двумерного уравнений теплопроводности, а также составлен алгоритм и программный продукт, решающий эту задачу.

**Исходный код программы**

**Задание 1.**

import numpy as np

import sympy as sp

from matplotlib import pyplot as plt

L = 0.1

du = 0.0015

E = 82e9

rho = 9700

tau = 0.005

h = 0.005

T = 1

c = 0.1

xs, ts = np.arange(0, L + h, h), np.arange(0, T + tau, tau)

nx, nt = len(xs), len(ts)

u = np.zeros((nt, nx))

xs\_center = xs[1:-1]

gamma = tau / h \* c

for i, x in enumerate(xs):

coeff = x / L if x <= L / 2 else (1 - x / L)

u[0][i] = 2 \* du \* coeff

for i, x in enumerate(xs\_center, 1):

u[1][i] = u[0][i] + gamma\*\*2 / 2 \* (u[0][i + 1] - 2 \* u[0][i] + u[0][i - 1])

for n, t in enumerate(ts[2:], 2):

for i, x in enumerate(xs\_center, 1):

u[n][i] = 2 \* u[n - 1][i] - u[n - 2][i] + gamma\*\*2 \* (u[n - 1][i + 1] - 2 \* u[n - 1][i] + u[n - 1][i - 1])

for n in range(0, nt, 10):

plt.title('t = {}, h = {}, c = {}'.format(tau, h, gamma))

plt.plot(xs, u[n])

plt.show()

%matplotlib notebook

%matplotlib notebook

from matplotlib import animation

fig = plt.figure()

ax = plt.axes(xlim=(0, L), ylim=(-du, du))

line, = ax.plot([], [], lw=2)

# initialization function: plot the background of each frame

def init():

line.set\_data([], [])

return line,

# animation function. This is called sequentially

def animate(i):

line.set\_data(xs, u[i])

return line,

# call the animator. blit=True means only re-draw the parts that have changed.

anim = animation.FuncAnimation(fig, animate, init\_func=init,

frames=600, interval=30, blit=True)

plt.show()

%matplotlib inline

for n in range(0, nt, 30):

plt.plot(xs, u[n])

plt.title('t = {}'.format(ts[n]))

plt.show()

**Задание 2.**

# coding: utf-8

# In[15]:

import numpy as np

import sympy as sp

import math

from matplotlib import pyplot as plt

import plotly.plotly as py

import plotly.graph\_objs as go

import plotly

# In[16]:

a = 2

b = 2

h = 0.1

tau = 0.05

T = 4

gamma = tau / h

# In[17]:

xs = np.arange(-a / 2, a / 2 + h, h)

ys = np.arange(-b / 2, b / 2 + h, h)

ts = np.arange(0, T + tau, tau)

nx = len(xs)

ny = len(ys)

nt = len(ts)

u = np.zeros((nt, ny, nx))

# In[18]:

u[0, :] = 2 \* np.cos(math.pi \* xs / a)

# In[19]:

for i in range(0, ny):

for j in range(1, nx - 1):

if i == 0:

y\_coeff = -2 \* u[0, i, j] + 2 \* u[0, i + 1, j]

elif i == ny - 1:

y\_coeff = -2 \* u[0, i, j] + 2 \* u[0, i - 1, j]

else:

y\_coeff = u[0, i - 1, j] - 2 \* u[0, i, j] + u[0, i + 1, j]

u[1, i, j] = u[0, i, j] + gamma\*\*2 / 2 \* (

u[0, i, j - 1] - 2 \* u[0, i, j] + u[0, i, j + 1] + \

y\_coeff

) + tau \* np.tan(np.sin(2 \* math.pi \* xs[j] / a)) \* np.sin(math.pi \* ys[i] / b)

#u[1, 0] = (u[1, 1] + u[1, 2]) / 2

#u[1, -1] = u[1, -2]

# In[20]:

for t in range(2, nt):

for i in range(0, ny):

for j in range(1, nx - 1):

if i == 0:

y\_coeff = -2 \* u[t - 1, i, j] + 2 \* u[t - 1, i + 1, j]

elif i == ny - 1:

y\_coeff = -2 \* u[t - 1, i, j] + 2 \* u[t - 1, i - 1, j]

else:

y\_coeff = u[t - 1, i - 1, j] - 2 \* u[t - 1, i, j] + u[t - 1, i + 1, j]

u[t, i, j] = 2 \* u[t - 1, i, j] - u[t - 2, i, j] + gamma\*\*2 \* (

u[t - 1, i, j - 1] - 2 \* u[t - 1, i, j] + u[t - 1, i, j + 1] + y\_coeff

)

#u[t, 0] = (u[t, 1] + u[t, 2]) / 2

#u[t, -1] = u[t, -2]

# In[21]:

def plot(u):

x\_grid, y\_grid = np.meshgrid(xs, ys)

surface = go.Surface(x=x\_grid, y=y\_grid, z=u)

data = [surface]

layout = go.Layout(

title='Parametric Plot',

scene=dict(

xaxis=dict(

gridcolor='rgb(255, 255, 255)',

showbackground=True,

backgroundcolor='rgb(230, 230,230)'

),

yaxis=dict(

title='t',

gridcolor='rgb(255, 255, 255)',

showbackground=True,

backgroundcolor='rgb(230, 230,230)'

),

zaxis=dict(

title='u(x, t)',

gridcolor='rgb(255, 255, 255)',

showbackground=True,

backgroundcolor='rgb(230, 230,230)'

)

)

)

fig = go.Figure(data=data, layout=layout)

plotly.offline.plot(fig, auto\_open=True)

# In[22]:

import time

#for i in range(0, 10):

# plot(u[i])

# time.sleep(0.5)

for i in range(0, nt, 30):

plot(u[i])

time.sleep(0.5)

# In[34]:

get\_ipython().run\_line\_magic('matplotlib', 'notebook')

get\_ipython().run\_line\_magic('matplotlib', 'notebook')

import mpl\_toolkits.mplot3d.axes3d as p3

import matplotlib.animation as animation

x\_grid, y\_grid = np.meshgrid(xs, ys)

fig = plt.figure()

ax = p3.Axes3D(fig)

line = ax.plot\_surface(x\_grid, y\_grid, u[0])

# initialization function: plot the background of each frame

def init():

line = ax.plot\_surface(x\_grid, y\_grid, u[0])

return line,

# animation function. This is called sequentially

def animate(i):

ax.clear()

line = ax.plot\_surface(x\_grid, y\_grid, u[i])

return line,

ax.set\_xlim3d([-a/2, a/2])

ax.set\_xlabel('X')

ax.set\_ylim3d([-b/2, b/2])

ax.set\_ylabel('Y')

ax.set\_zlim3d([-2.5, 2.5])

ax.set\_zlabel('Z')

# call the animator. blit=True means only re-draw the parts that have changed.

anim = animation.FuncAnimation(fig, animate, init\_func=init,

frames=800, interval=20, blit=True)

plt.show()