МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П.Королёва»
(Самарский университет)

Институт информатики, математики и электроники
Факультет информатики
Кафедра технической кибернетики

Отчет по курсовой работе

Дисциплина «Численные методы математической физики»

Тема: «**РЕШЕНИЕ ЗАДАЧ МАТЕМАТИЧЕСКОЙ ФИЗИКИ МЕТОДОМ КОНЕЧНЫХ РАЗНОСТЕЙ**»

Вариант № 40

Выполнил студент:	Белоусов А. А.
Группа:	6409-010302D
Проверил:	Дегтярев А. А.

Самара 2019

ЗАДАНИЕ К КУРСОВОЙ РАБОТЕ

1. Осуществить математическую постановку краевой задачи для физического процесса, описанного в предложенном варианте курсовой работы.

2. Осуществить построение разностной схемы, приближающей полученную краевую задачу. При этом следует согласовать с преподавателем тип разностной схемы.

3. Провести теоретическое исследование схемы: показать, что схема аппроксимирует исходную краевую задачу, и найти порядки аппроксимации относительно шагов дискретизации; исследовать устойчивость схемы и сходимость сеточного решения к решению исходной задачи математической физики.

4. Разработать алгоритм численного решения разностной краевой задачи

5. Разработать компьютерную программу, реализующую созданный алгоритм, с интерфейсом, обеспечивающим следующие возможности: диалоговый режим ввода физических, геометрических и сеточных параметров задачи; графическую визуализацию численного решения задачи.

6. Используя разработанную программу и тестовый пример, согласованный с преподавателем, провести экспериментальное исследование фактической сходимости сеточного решения к точному (вычисленному с помощью ряда Фурье). Проводя измельчение сетки, сравнить экспериментальную скорость убывания погрешности сеточного решения со скоростью, полученной при теоретическом исследовании схемы.

7. Оформить отчет о проделанной работе.

ВАРИАНТ 40

Разработать программу расчета на промежутке времени $0 < t \leq T$ малых поперечных колебаний прямоугольной однородной мембраны шириной l_x и длиной l_y. Колебания мембраны возбуждаются начальным отклонением

$$u(x, y, t = 0) = \alpha(x, y), 0 \leq x \leq l_x, 0 \leq y \leq l_y.$$

Края мембраны $x = 0, x = l_x, y = 0$ и $y = l_y$ жестко закреплены, а реакция окружающей среды пренебрежимо мала. Начальные скорости точек мембраны равны нулю.

Поверхностная плотность мембраны и величина натяжения, возникающего в ней в процессе колебаний, равны ρ и η соответственно.

Для решения описанной задачи математической физики применить метод разделения переменных. Для расчетов использовать представление решения задачи в виде ряда Фурье по собственным функциям оператора Лапласа, удовлетворяющим соответствующим краевым условиям.

При проведении расчетов использовать значения параметров l_x, l_y, T, ρ, η, а также выражение функции $\alpha(x, y)$, указанные преподавателем.

Для численного решения описанной задачи математической физики использовать следующие разностные схемы:

- простейшую явную конечно-разностную схему;

Значения параметров, указанные преподавателем:

$$\begin{aligned}
l_x &= 4, \\
l_y &= 1, \\
T &= 10, \\
\rho &= 1, \\
\eta &= 1, \\
\alpha(x, y) &= p(x, y) \sin(\frac{\pi y}{l_y}) \\
p(x, y) &= -\frac{x^2}{4} + x
\end{aligned}$$

РЕФЕРАТ

Отчёт: 25 страниц, 6 рисунков, 1 таблица, 4 источника, 1 приложение.

УРАВНЕНИЯ МАТЕМАТИЧЕСКОЙ ФИЗИКИ, КРАЕВАЯ ЗАДАЧА, УРАВНЕНИЕ ПОПЕРЕЧНЫХ КОЛЕБАНИЙ ПРЯМОУГОЛЬНОЙ МЕМБРАНЫ, МЕТОД КОНЕЧНЫХ РАЗНОСТЕЙ, ЯВНАЯ КОНЕЧНО-РАЗНОСТНАЯ СХЕМА, УСТОЙЧИВОСТЬ, АППРОКСИМАЦИЯ, СХОДИМОСТЬ

Целью курсовой работы является построение и исследование разностных схем для решения краевой задачи колебаний прямоугольной мембраны.

Для решения задачи была использована явная конечно-разностная схема. Проведено теоретическое исследование аппроксимации и устойчивости разностной схемы. Сделан вывод о сходимости сеточного решения к точному решению исходной задачи.

Разработана компьютерная программа, обеспечивающая расчет и графическую визуализацию процесса колебаний мембраны.

Приведены графические результаты численного решения задачи колебаний мембраны.

Программа написана на языке Python в среде разработки PyCharm, операционная система Windows.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Характеризуя метод конечных разностей, необходимо выделить его достоинства и недостатки в сравнении с другими методами.

К достоинствам метода конечных разностей следует отнести его высокую универсальность, например, значительно более высокую, чем у аналитических методов. Применение этого метода нередко характеризуется относительной простотой построения решающего алгоритма и его программной реализации. Зачастую удается осуществить распараллеливание решающего алгоритма.

К числу недостатков метода следует отнести: проблематичность его использования на нерегулярных сетках; очень быстрый рост вычислительной трудоемкости при увеличении размерности задачи (увеличении числа неизвестных переменных); сложность аналитического исследования свойств разностной схемы.

Суть метода конечных разностей состоит в замене исходной(непрерывной) задачи математической физики ее дискретным аналогом (разностной схемой), а также последующим применением специальных алгоритмов решения дискретной задачи.

В настоящей работе метод конечных разностей применен для численного решения задачи колебаний прямоугольной мембраны. Проведено теоретическое исследование аппроксимации разностной схемы. Сделан вывод о сходимости сеточного решения к точному решению исходной задачи. Разработана компьютерная программа, реализующая алгоритм численного решения. Приведены графические результаты численного решения задачи.

1 Постановка краевой задачи

Построим математическую модель поперечных колебаний тонкой однородной мембраны. Уравнение свободных поперечных колебаний мембраны имеет вид:

$$\frac{\partial^2 u}{\partial t^2} = \alpha^2 (\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}). \tag{1}$$

В условии задачи указано, что края мембраны жестко закреплены. Отсюда следуют граничные условия:

$$u|_{x=0}, \quad u|_{y=0}, \quad u|_{x=l_x}, \quad u|_{y=l_y} = 0. \tag{2}$$

По условию, в начальный момент времени отклонение мембраны задано функцией $\alpha(x, y)$, а начальная скорость точек мембраны равна нулю. Отсюда получаем начальные условия:

$$u|_{t=0} = p(x, y) \sin(\frac{\pi y}{l_y});$$

$$\frac{\partial u}{\partial t}|_{t=0} = 0.$$

Производим следующую замену:

$$u(x, y, t) = \nu(x, t) \sin(\frac{\pi y}{l_y}). \tag{3}$$

Подставляем в исходное уравнение, переходим к виду:

$$\frac{\partial^2 \nu}{\partial t^2} = \alpha^2 (\frac{\partial^2 \nu}{\partial x^2} - \frac{\pi^2}{l_y} \nu(x, t)). \tag{4}$$

Пересчитываем начальные условия для $\nu(x, y)$:

$$\nu|_{t=0} = p(x, y);$$

$$\frac{\partial \nu}{\partial t}|_{t=0} = 0.$$

Таким образом, имея уравнение, набор граничных и начальных условий, получим математическую модель, описывающую данную задачу:

$$u(x, y, t) = \nu(x, t) \sin(\frac{\pi y}{l_y}); \tag{5}$$

$$\begin{cases} \dfrac{\partial^2 \nu}{\partial t^2} = \alpha^2 \left(\dfrac{\partial^2 \nu}{\partial x^2} - \left(\dfrac{\pi}{l_y} \right)^2 \nu(x, t) \right), \quad 0 \le x \le l_x; \\ \nu|_{x=0}, \quad \nu|_{x=l_x} = 0; \\ \nu|_{t=0} = p(x); \\ \dfrac{\partial \nu}{\partial t}|_{t=0} = 0; \end{cases} \tag{6}$$

2 Решение краевой задачи с помощью простейшей явной схемы

2.1 Построение простейшей явной схемы

Для построения простейшей неявной разностной схемы для задачи (6) заменим все непрерывные соотношения их сеточными аналогами. В данном случае будем использовать равномерную сетку, определяемую как следующее множество узлов (x_i, t_k):

$$\begin{aligned} &x_i = ih_x; i = \overline{0, I}; h_x = \frac{L}{I}, \\ &t_k = kh_t; k = \overline{0, K}; h_t = \frac{T}{K}. \end{aligned} \tag{7}$$

Заменим частные производные, входящие в состав (6) следующими разностными соотношениями:

$$\frac{\partial \nu(x_i, t_k)}{\partial t} \approx \frac{\nu(x_i, t_k) - \nu(x_i, t_{k-1})}{h_t}, k = \overline{1, I}, i = \overline{0, I}, \tag{8}$$

$$\frac{\partial^2 \nu(x_i, t_k)}{\partial t^2} \approx \frac{\nu(x_i, t_{k-1}) - 2\nu(x_i, t_k) + \nu(x_i, t_{k+1})}{h_t^2}, i = \overline{1, I-1}, k = \overline{0, K}. \tag{9}$$

$$\frac{\partial^2 \nu(x_i, t_k)}{\partial x^2} \approx \frac{\nu(x_{i-1}, t_k) - 2\nu(x_i, t_k) + u(x_{i+1}, t_k)}{h_x^2}, i = \overline{1, I-1}, k = \overline{0, K}. \tag{10}$$

Заменим правую часть начального условия следующей сеточной функцией:

$$p(x_i) = \psi_i, i = \overline{1, I}. \tag{11}$$

После произведенных преобразований запишем общий вид простейшей явной разностной схемы для задачи (6):

$$\begin{cases} \dfrac{\nu_i^{k-1} - 2\nu_i^k + \nu_i^{k+1}}{h_t^2} = \alpha^2 \left(\dfrac{\nu_{i-1}^k - 2\nu_i^k + \nu_{i+1}^k}{h_x^2} - \left(\dfrac{\pi}{l_y}\right)^2 \nu_i^k \right), i = \overline{1, I-1}, k = \overline{1, K-1} \\ \dfrac{\nu_i^1 - \nu_i^0}{h_t} = 0, i = \overline{1, I-1} \\ \nu_i^0 = \psi_i, i = \overline{1, I-1} \\ \nu_0^k = 0, k = \overline{1, K} \\ \nu_I^k = 0, k = \overline{1, K} \end{cases} \tag{12}$$

Система соотношений (12) представляет собой конечный вид разностной схемы, используемой в рамках данной курсовой работы.

2.2 Получение расчетных формул

Выразим из (12) явные расчетные формулы для численного решения задачи (6):

Расчетная формула для $i = \overline{1, I-1}, k = \overline{1, K-1}$:

$$\begin{aligned} &\gamma = \left(\frac{\alpha h_t}{h_x}\right)^2 \\ &\nu_i^{k+1} = \gamma \nu_{i-1}^k + \left(-2\gamma + 2 - \left(\frac{\alpha \pi h_t}{l_y}\right)^2\right) \nu_i^k + \gamma \nu_{i+1}^k - \nu_i^{k-1} \end{aligned} \tag{13}$$

Расчетная формула для $i = \overline{1, I-1}, k = 0$:

$$\nu_i^1 = -\frac{i h_x^2}{l_x} + i h_x \tag{14}$$

Расчетная формула для $i = 0, k = \overline{1, K-1}$:

$$\nu_0^k = 0 \tag{15}$$

Расчетная формула для $i = I, k = \overline{1, K-1}$:

$$\nu_I^k = 0 \tag{16}$$

2.3 Исследование аппроксимации простейшей явной схемы

Запишем разностную схему (12) в операторной форме:

$$L_h \nu_h = f_h$$

$$L_h \nu_h = \begin{pmatrix} L_h^1 \nu_h \\ L_h^2 \nu_h \\ L_h^3 \nu_h \\ L_h^4 \nu_h \\ L_h^5 \nu_h \end{pmatrix} = \begin{pmatrix} \dfrac{\nu_i^{k-1} - 2\nu_i^k + \nu_i^{k+1}}{h_t^2} - \alpha^2 \left(\dfrac{\nu_{i-1}^k - 2\nu_i^k + \nu_{i+1}^k}{h_x^2} - \left(\dfrac{\pi}{l_y}\right)^2 \nu_i^k\right) \\ \dfrac{\nu_i^1 - \nu_i^0}{h_t} \\ \nu_i^0 \\ \nu_0^k \\ \nu_I^k \end{pmatrix}$$

$$f_h = \begin{pmatrix} f_h^1 \\ f_h^2 \\ f_h^3 \\ f_h^4 \\ f_h^5 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \psi_i \\ 0 \\ 0 \end{pmatrix}$$

$$(17)$$

Невязка запишется следующим образом:

$$\delta f_h = \begin{pmatrix} \delta f_h^1 \\ \delta f_h^2 \\ \delta f_h^3 \\ \delta f_h^4 \\ \delta f_h^5 \end{pmatrix} = \begin{pmatrix} [L_h^1[\nu]_h - f_h^1] \\ [L_h^2[\nu]_h - f_h^2] \\ [L_h^3[\nu]_h - f_h^3] \\ [L_h^4[\nu]_h - f_h^4] \\ [L_h^5[\nu]_h - f_h^5] \end{pmatrix} \tag{18}$$

Определим порядок невязки, выбрав максимальный из всех элементов вектора. Зафиксируем узел сетки и разложим все функции, входящие в состав каждого оператора (17) в ряд Тейлора в окрестности выбранного узла.

Невязка для первого оператора имеет следующий вид:

$$\begin{aligned}
&\delta f_h^1|_{(x_i,t_k)} = [L_h^1[\nu]_h - f_h^1]_{(x_i,t_k)} = \frac{\nu(x_i,t_{k-1}) - 2\nu(x_i,t_k) + \nu(x_i,t_{k+1})}{h_t^2} \\
&- \alpha^2 \left(\frac{\nu(x_{i-1},t_k) - 2\nu(x_i,t_k) + \nu(x_{i+1},t_k)}{h_x^2} - \left(\frac{\pi}{ly}\right)^2 \nu(x_i,t_k) \right) = \\
&(\nu_t - h_t\nu_t' + \frac{h_t^2\nu_{tt}''}{2} - \frac{h_t^3\nu_{ttt}'''}{6} + \frac{h_t^4\nu_{tttt}^{IV}}{24} - 2\nu_t \\
&+ \nu_t + h_t\nu_t' + \frac{h_t^2\nu_{tt}''}{2} + \frac{h_t^3\nu_{ttt}'''}{6} + \frac{h_t^4\nu_{tttt}^{IV}}{24} + O(h_t^5))\frac{1}{h_t^2} \\
&- \alpha((\nu_x - h_x\nu_x' + \frac{h_x^2\nu_{xx}''}{2} - \frac{h_x^3\nu_{xxx}'''}{6} + \frac{h_x^4\nu_{xxxx}^{IV}}{24} - 2\nu_x \\
&+ \nu_x + h_x\nu_x' + \frac{h_x^2\nu_{xx}''}{2} + \frac{h_x^3\nu_{xxx}'''}{6} + \frac{h_t^4\nu_{xxxx}^{IV}}{24} + O(h_x^5)))\frac{1}{h_x^2} - \left(\frac{\pi}{l_y}\right)^2 \nu) \\
&= \nu_t'' + \frac{h_t^2\nu_{tttt}^{IV}}{12} + O(h_t^3) - \alpha^2 \left(\nu_x'' + \frac{h_x^2\nu_{xxxx}^{IV}}{12} + O(h_x^3) - \left(\frac{\pi}{l_y}\right)^2 \nu \right) \\
&= O(h_x^2, h_t^2)
\end{aligned} \tag{19}$$

Для второго оператора невязка равна:

$$\begin{aligned}
&\delta f_h^2|_{(x_i,t_0)} = [L^2[u]_h - f_h^2]_{(x_i,t_0)} = \\
&\frac{\nu(x_i,1) - \nu(x_i,0)}{h_t} - 0 = \left(\nu_t(0) + h_t\nu_t'(0) + \frac{h_t^2\nu_{tt}''(0)}{2} + O(h_t^3) \right) \frac{1}{h_t} = \\
&\frac{\nu_t''(0)}{2h_t} + O(h_t^2) = O(h_t)
\end{aligned}$$

Невязка для третьего оператора:

$$\delta f_h^3|_{(x_i,t_0)} = [L^3[u]_h - f_h^3]_{(x_i,t_0)} = \nu(x_i,0) - \psi_i = \psi_i - \psi_i = 0.$$

Невязка для четвертого оператора:

$$\delta f_h^4|_{(x_0,t_k)} = [L^4[u]_h - f_h^4]_{(x_0,t_k)} = \nu(0,t_k) - 0 = 0.$$

Невязка для пятого оператора:

$$\delta f_h^5|_{(x_I,t_k)} = [L^5[u]_h - f_h^5]_{(x_I,t_k)} = \nu(l_x,t_k) - 0 = 0.$$

После нахождения всех необходимых величин, определим порядок аппроксимации неявной схемы, применив равномерную норму к порядкам аппроксимации всех выражений, входящих в состав схемы:

$$||\delta f_h||_{F_h} = \max_{\substack{i=\overline{1,I-1} \\ k=\overline{0,K-1}}} |\delta f_h^1| + \max_{i=\overline{0,I}} |\delta f_h^2| + \max_{k=\overline{1,K}} |\delta f_h^3| + \max_{k=\overline{1,K}} |\delta f_h^4| + \max_{k=\overline{1,K}} |\delta f_h^5| = O(h_x^2, h_t). \tag{20}$$

В ходе исследования было установлено, что простейшая явная схема аппроксимирует исходную задачу линейно относительно шага по времени h_t и квадратично относительно шага по пространственной переменной h_x.

3 Результаты вычислительного эксперимента

В ходе курсовой работы была реализована программа, осуществляющая численное решение задачи колебний прямоугольной мембраны в промежутке времени от 0 до T, на основе простейшей явной разностной схемы.

На рисунках 1-3 приведены графики зависимости положения точек поверхности мембраны от пространственной переменной x в момент времени $t = 0.75$, полученные с помощью простейшей явной схемы.

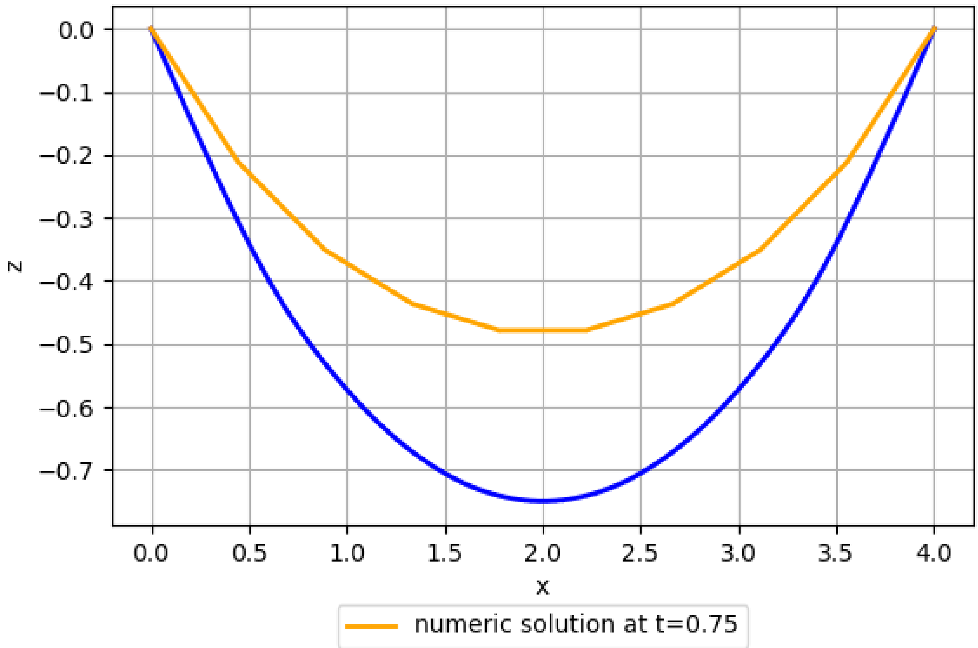

Рисунок 1 – График численного и аналитического решений при параметрах

$K = 10, I = 10, t = 0.75$

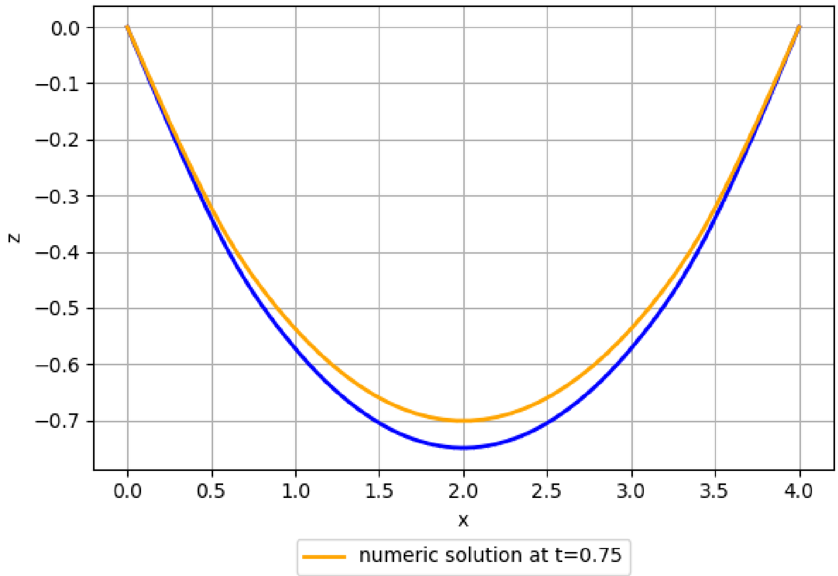

Рисунок 2 – График численного и аналитического решений при параметрах

$$K = 50, I = 50, t = 0.75$$

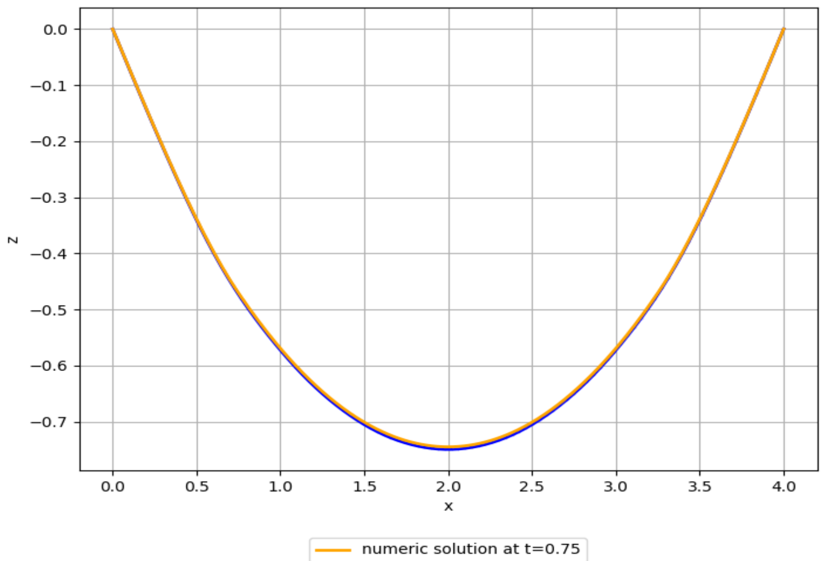

Рисунок 3 – График численного и аналитического решений при параметрах

$$K = 500, I = 500, t = 0.75$$

На рисунках 4-6 приведены графики зависимости положения точек поверхности мембраны от пространственной переменной x в момент времени $t = 1.5$, полученные с помощью простейшей явной схемы. Можно видеть, что точность численного решения понижается с увеличением промежутка времени и, соответственно, шага сетки по временной переменной.

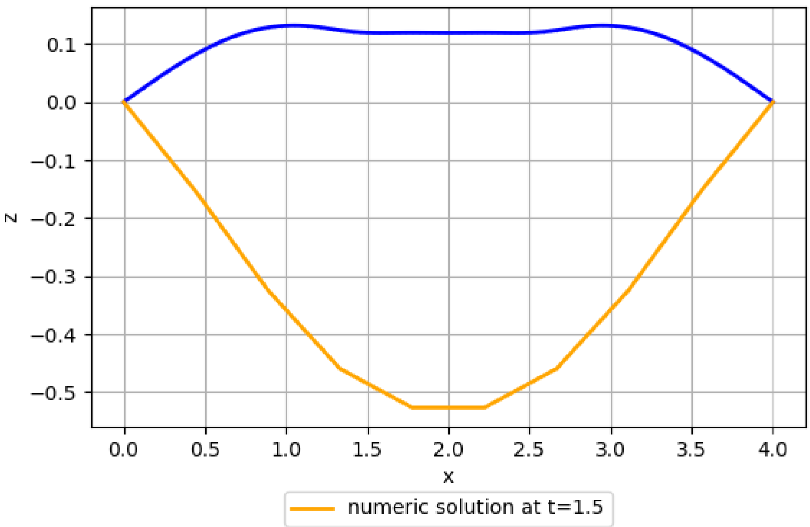

Рисунок 4 – График численного и аналитического решений при параметрах $K = 10, I = 10, t = 1.5$

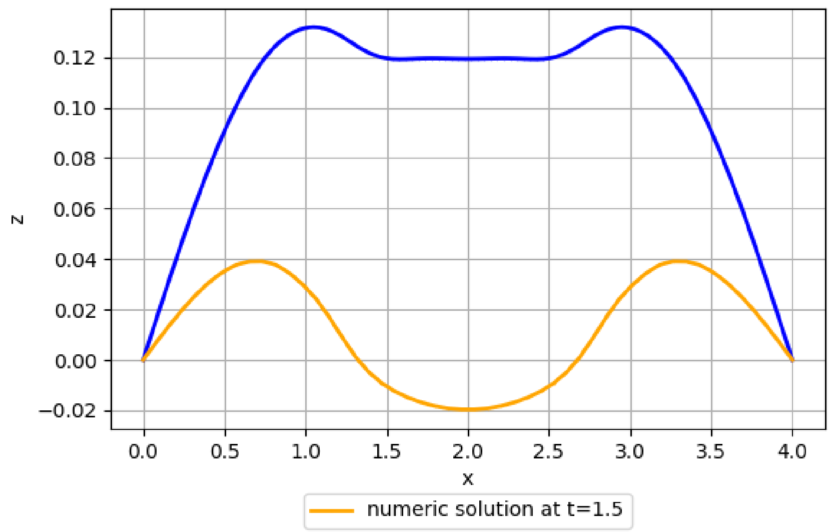

Рисунок 5 – График численного и аналитического решений при параметрах

$$K = 50, I = 50, t = 1.5$$

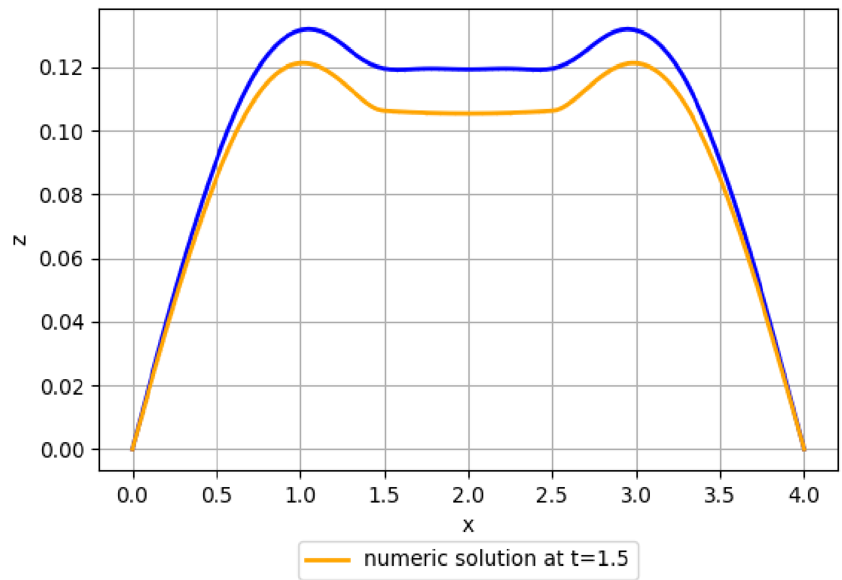

Рисунок 6 – График численного и аналитического решений при параметрах

$$K = 500, I = 500, t = 1.5$$

Как видно из рисунков 1-6, с увеличением числа узлов сетки наблюдается визуальная сходимость, иными словами, численное решение приближается к точному решению исходной задачи.

3.1 Экспериментальное исследование сходимости схемы

Исследуем экспериментально скорость сходимости простейшей явной схемы . Для этого выберем некоторую достаточно крупную сетку и будем ее последовательно измельчать, каждый раз определяя величину абсолютной погрешности.

При этом шаг h_x на каждой итерации будем измельчать в 2 раза, а h_t - в 4 раза. Согласно результатам теоретического исследования погрешность сеточного решения для явной схемы характеризуется величиной $O(h_x^2, h_t)$. Таким образом, следует ожидать, что при каждом измельчении сетки погрешность численного решения будет уменьшаться приблизительно в 4 раза.

Результаты экспериментального исследования сходимости разностной схемы представлены в таблице 1:

Таблица 1 – Погрешность простейшей явной схемы при $x = 1, t = 2$

K	I	h_t	h_x	$\epsilon_{(h,\tau)}$	$\delta_{(h,\tau)}$
4	4	0.5	1.0	0.5120	-
16	8	0.25	0.5	0.0770	6.65
64	16	0.0125	0.25	0.0540	1.43
256	32	0.0625	0.125	0.0211	2.55
1024	64	0.03125	0.0625	8.65×10^{-3}	2.45
4096	128	0.015625	0.03125	3.87×10^{-3}	2.23

Из приведенных данных можно видеть, что сходимость реализации разностной схемы не вполне соответствует результатам теоретического исследования.

ЗАКЛЮЧЕНИЕ

В результате выполнения работы осуществлена постановка краевой задачи для уравнения поперечных колебаний тонкой прямоуголдьной мембраны. Для численного решения краевой задачи построена простейшая явная разностная схема.

Проведено теоретическое исследование разностной схемы, в результате которого установлено, что явная схема имеет линейный порядок аппроксимации для временной переменной и квадратичный для пространственной переменной.

В результате серии вычислительных экспериментов заметна визуальная сходимость разностного решения, вычисляемого с помощью схем, при увеличении количества шагов, т.е. с измельчением сетки решение дискретной задачи приближается к решению исходной задачи. Однако, теоретическая скорость сходимости решения не была достигнута в ходе вычислительного эксперимента.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Дегтярев А.А. Примеры построения и исследования разностных схем. – Электронное учебное пособие [Электронный ресурс] / А.А Дегтярев, 2011.- 54с.

2 Тихонов А.Н., Самарский А.А. Уравнения математической физики (5-е изд.) [Текст] / Тихонов А.Н., Самарский А.А. М.: Наука, 1977.- 742 с.

3 Numpy and Scipy Documentation [Электронный ресурс]: Официальный сайт документации библиотек Numpy и Scipy. - URL: https://docs.scipy.org/doc/

4 Свешников А. Г., Боголюбов А. Н., Кравцов В. В. Лекции по математической физике [Текст] / Свешников А. Г., Боголюбов А. Н., Кравцов В. В. М.: МГУ, 1993. - 352 с.

ПРИЛОЖЕНИЕ А

Код программы

```
import numpy as np
import timeit
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import pyplot as plt
from matplotlib import animation

# Constants
LX = 4
LY = 1
A = 1

# Number of grid nodes per time and space variables
K_big = 100
I_big = 100

ANIMATION_TIME_STEPS = 100


def __plot_2d(x_sp, y_sp, t, figname, vline=0, y=0, savefig=False):
    fig = plt.figure(figname)
    ax = plt.subplot(111)

    plt.rc('lines', linewidth=1)

    graph, = ax.plot(x_sp, y_sp, color='orange', marker='o',
                    linestyle='-', linewidth=2, markersize=0.1)

    plt.xlabel('x')
    plt.ylabel('z')

    box = ax.get_position()
    ax.set_position([box.x0, box.y0 + box.height * 0.1,
                    box.width, box.height * 0.9])

    if vline != 0:
        line = plt.axvline(x=vline, color='r')
```

```
        ax.legend([line, graph], ['x={0}'.format(vline), 'u(x,y,t) at t={0}'.format(t)],
                  loc='upper center', bbox_to_anchor=(0.5, -0.13), ncol=3, fancybox=True)
    else:
        ax.legend([graph], ['numeric solution at t={0}'.format(t)],
                  loc='upper center', bbox_to_anchor=(0.5, -0.13), ncol=3, fancybox=True)

    plt.grid(True)

    if savefig:
        name = '{0}_{1}_{2}'.format(vline, y, t).replace('.', '')
        plt.savefig(name)
    # plt.show()


def __anim_plot_2d(x_vals, y_per_time, h_t):
    fig = plt.figure("Numerical solution animated")
    ax = plt.axes(xlim=(0, LX), ylim=(- 1.5, 1.5))
    line, = ax.plot([], [], lw=2, color="orange")
    time_text = ax.text(.2, 1.5, '', fontsize=15)

    plt.xlabel('x')
    plt.ylabel('y')

    # initialization function: plot the background of each frame
    def init():
        time_text.set_text('')
        line.set_data([], [])
        return line, time_text

    # animation function. This is called sequentially
    def animate(i):
        index = i % len(y_per_time)
        x = np.linspace(0, LX, I_big)
        y = y_per_time[index]
        line.set_data(x, y)
        time_text.set_text('T={0}'.format(round(index * h_t, 2)))
        return line, time_text

    # call the animator. blit=True means only re-draw the parts that have changed.
    anim = animation.FuncAnimation(fig, animate,
```

```
                                   frames=200, interval=30, blit=False)
    plt.show()


def get_value_at(x, time):
    h_x = LX / I_big
    h_t = time / K_big
    res = differential_scheme(h_x, h_t)

    for i in range(len(res)-1):
        if i * h_x == x:
            return res[i]

        if i*h_x < x < (i+1)*h_x:
            left_x = h_x * i
            t = (x-left_x) / h_x
            value = left_x + t * h_x
            return value


def static_2d(time, figname="Numerical solution"):
    start = timeit.default_timer()
    print("Starting calculation of numerical solution...")

    h_x = LX / I_big
    h_t = time / K_big

    print("h_x = ", h_x, "; h_t = ", h_t)
    res = differential_scheme(h_x, h_t)

    end = timeit.default_timer()
    print("Finished calculation in {0}s".format(end - start))

    __plot_2d(np.linspace(0, LX, I_big), res, time, figname)


def animated_2d(time):
    t_vals = np.linspace(0, time, ANIMATION_TIME_STEPS)

    x_vals = np.linspace(0, I_big, I_big)
```

```python
    values_per_time = []

    start = timeit.default_timer()
    print("Starting calculation for animated 2d...")
    time_step = time / ANIMATION_TIME_STEPS

    h_x = LX / I_big
    for t in t_vals:
        h_t = t / K_big
        res = differential_scheme(h_x, h_t)
        values_per_time.append(res)
    end = timeit.default_timer()
    print("Finished calculation in {0}s".format(end - start))

    __anim_plot_2d(x_vals, values_per_time, time_step)


# Initial shape
def psi(x):
    return -(x ** 2) / LX + x


# "gamma" factor equals (a*h_t/h_x)^2
def gamma_func(h_t, h_x):
    return (A * h_t / h_x) ** 2


# Solution of a differential scheme (v_i_k+1) for given indices
def solve_k_plus1(gamma, h_t: float, i: int, v_k: list, v_k_minus1: list):
    return gamma * v_k[i - 1] + (-2 * gamma + 2 - (A * np.pi * h_t / LY) ** 2) * v_k[i] +
    ↪  gamma \
        * v_k[i + 1] - v_k_minus1[i]


# Full solution of a differential scheme
def differential_scheme(h_x, h_t):
    # Grid of a differentiol scheme
    grid = []
    for k in range(K_big):
```

```python
        grid.append([0] * I_big)

    # ------------------------------------------------------
    # Setting the initial shape (v(k=0))
    x = np.linspace(0, LX, I_big)
    for i in range(0, I_big):
        grid[0][i] = psi(x[i])

    # Values at v(k=1) are equal to v(k=0)
    # v_i_k = v_i_k_minus1
    # ------------------------------------------------------

    # return grid[0]
    gamma = gamma_func(h_t, h_x)
    grid[1] = grid[0]
    # Computing full solution with given amount of time steps
    for k in range(1, K_big - 1):
        grid[k + 1][0] = 0
        # print(grid[k][int(I_big / 2)])
        for i in range(1, I_big - 1):
            grid[k + 1][i] = solve_k_plus1(gamma, h_t, i, grid[k], grid[k - 1])
        grid[k + 1][-1] = 0

    return grid[-1]


if __name__ == '__main__':
    pass
    # static_2d(2)
    # animated_2d(2)


import equations.numerical_solution as num
import equations.analytic_solution as an
import matplotlib.pyplot as plt


def to_file(filename, i_s: list, k_s: list, ht_s: list, hx_s: list, eps: list, deltas:
    ↪ list):
    f = open(filename, 'w+')
```

```
    f.write("I K ht hx eps d\n")
    for i in range(len(i_s)):
        f.write(f"{i_s[i]} {k_s[i]} {ht_s[i]} {hx_s[i]} {eps[i]} {deltas[i]}\n")


if __name__ == '__main__':
    time = 2
    x = 1
    num.I_big = 500
    num.K_big = 500
    # an.static_2d(time)
    # num.static_2d(time)
    # an.animated_2d(time)
    # num.animated_2d(time)

    # an.static_2d(time, 'solution')
    # num.static_2d(time, 'solution')

    #print(num.get_value_at(x, time))
    # print(an.get_value_at(x, time))
    #plt.show()

    i_big = 4
    k_big = 4
    eps = []
    i_s = []
    k_s = []
    ht_s = []
    hx_s = []
    deltas = []
    for i in range(6):
        print(f"--- i:{i_big}, k:{k_big}")

        hx = 4 / i_big
        ht = time / i_big

        num.I_big = i_big
        num.K_big = k_big

        n = num.get_value_at(x, time)
```

```
    a = an.get_value_at(x, time)
    epsilon = abs(n - a)

    d = 0
    if i > 0:
        d = eps[i-1] / epsilon

    eps.append(epsilon)
    i_s.append(i_big)
    k_s.append(k_big)
    ht_s.append(ht)
    hx_s.append(hx)
    deltas.append(d)

    i_big *= 2
    k_big *= 4

to_file('result.txt', i_s, k_s, ht_s, hx_s, eps, deltas)
```