Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Самарский национальный исследовательский университет
имени академика С.П.Королёва»

Факультет информатики

Кафедра технической кибернетики

Отчет по курсовой работе

Дисциплина «Уравнения математической физики»

Тема: «**АНАЛИТИЧЕСКОЕ РЕШЕНИЕ КРАЕВЫХ ЗАДАЧ МАТЕМАТИЧЕСКОЙ ФИЗИКИ**»

Вариант № 40

Выполнил студент: Белоусов А. А.

Группа: 6409

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. Используя метод разделения переменных (метод Фурье), получить решение краевой задачи в виде разложения в ряд Фурье по собственным функциям оператора Лапласа, соответствующим краевым условиям задачи.

3. Провести анализ погрешности решения.

4. Разработать компьютерную программу расчета функции-решения краевой задачи (суммирования ряда Фурье). При расчете коэффициентов ряда использовать метод численного интегрирования, если это необходимо. Если необходимо, то разработать специальный программный модуль для вычисления используемых собственных чисел оператора Лапласа. Компьютерная программа должна обеспечивать возможность диалогового режима ввода физических, геометрических параметров задачи, числа суммируемых элементов ряда, графическую визуализацию рассчитанного решения задачи.

5. Оформить отчет о выполненной курсовой 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.

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

29 страниц, 8 рисунков, 2 таблиц, 3 источников, 1 приложение

УРАВНЕНИЯ МАТЕМАТИЧЕСКОЙ ФИЗИКИ, КРАЕВАЯ ЗАДАЧА, УРАВНЕНИЕ ПОПЕРЕЧНЫХ КОЛЕБАНИЙ ПРЯМОУГОЛЬНОЙ МЕМБРАНЫ, МЕТОД РАЗДЕЛЕНИЯ ПЕРЕМЕННЫХ, СОБСТВЕННЫЕ ФУНКЦИИ ОПЕРАТОРА ЛАПЛАСА, РЯД ФУРЬЕ.

Целью курсовой работы является получение решения краевой задачи колебаний прямоугольной мембраны в виде разложения в ряд Фурье по собственным функциям оператора Лапласа и создание компьютерной программы для расчета функции-решения.

Для получения аналитического решения краевой задачи использован метод разделения переменных. Решение задачи получено в виде конечного ряда Фурье.

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

Приведены графические результаты численного решения задачи колебаний мембраны, а также анализ погрешности решения.

Программа написана на языке Python в среде разработки PyCharm на операционной системе Windows.

СОДЕРЖАНИЕ

Введение 6
1 Постановка краевой задачи 7
2 Аналитическое решение краевой задачи и получение решения задачи в виде ряда Фурье 9
3 Получение расчетной формулы 12
4 Оценка остатка ряда 13
5 Экспериментальное исследование качества оценки остатка ряда 14
6 Результаты вычислительных экспериментов 16
Заключение 21
Список литературы 22
Приложение А Код программы 23

ВВЕДЕНИЕ

Метод разделения переменных относится к классу аналитических методов решения краевых задач математической физики. Характеризуя этот метод необходимо выделить его достоинства и недостатки в сравнении с другими методами.

К достоинствам метода разделения переменных следует отнести возможность получения точного решения краевой задачи в виде ряда Фурье. Такая форма решения задачи часто и весьма успешно используется для теоретического исследования свойств этого решения. В случае достаточно быстрой сходимости ряда Фурье она может с успехом использоваться для численного моделирования физического процесса (явления).

К числу недостатков метода следует отнести его невысокую универсальность. Этот метод весьма проблематично использовать для решения нелинейных уравнений математической физики, уравнений с переменными операторными коэффициентами, а также для решения краевых задач в областях со сложными границами.

Суть метода разделения переменных состоит в факторизации по каждой независимой переменной функции, определяющей решение уравнения математической физики. Далее осуществляется переход к так называемой задаче Штурма-Лиувилля, решение которой приводит к получению собственных функций и соответствующих им собственных чисел оператора Лапласа. Затем решение исходной задачи ищется в виде ряда Фурье по этим собственным функциям.

В настоящей работе метод разделения переменных применен для получения аналитической формы решения задачи описания процесса колебаний мембраны. На основе этого результата разработан алгоритм и компьютерная программа численного моделирования процесса колебаний мембраны.

1 Постановка краевой задачи

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

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

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:

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

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 нашу задачу:

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

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

2 Аналитическое решение краевой задачи и получение решения задачи в виде ряда Фурье

Для решения поставленной задачи в уравнении (6) воспользуемся методом разделения переменных:

$$\nu(x,t) = X(x)T(t). \tag{7}$$

Тогда уравнение (6) примет вид:

$$T''X = \alpha^2 \left(X''T - \frac{\pi^2}{l_y^2} XT \right);$$

Разделим обе части на a^2XT и получим следующее уравнение:

$$\frac{T'}{\alpha^2 T} = \frac{X'' - \frac{\pi^2}{l_y^2}X}{X}. \tag{8}$$

В равенстве (9) можно заметить, что левая и правая части зависят только от t и x соответственно. Тем самым, общая величина выражений - константа, которую мы возьмем равной $-\lambda^2$. Константа берется с отрицательным знаком, чтобы удовлетворять граничным условиям. Таким образом равенство (9) примет вид:

$$\frac{T'}{\alpha^2 T} = \frac{X'' - \frac{\pi^2}{l_y^2}X}{X} = -\lambda^2. \tag{9}$$

Откуда получим два уравнения:

$$T'' + \lambda^2\alpha^2 T = 0;$$
$$X'' + (\lambda^2 - \frac{\pi^2}{l_y^2})X = 0.$$

Общие решения этих уравнений будут иметь следующий вид:

$$X(x) = A\cos(\sqrt{\frac{\pi^2}{l_y^2} - \lambda^2}x) + B\sin(\sqrt{\frac{\pi^2}{l_y^2} - \lambda^2}x), \tag{10}$$

$$T(t) = C\cos(\alpha\lambda t) + D\sin(\lambda t), \tag{11}$$

где A, B, C, D - произвольные постоянные.

Подставляя полученные выражения $X(t), T(t)$ в равенство (7), получаем:

$$\nu(x,t) = (A\cos(\sqrt{\frac{\pi^2}{l_y^2} - \lambda^2}x) + B\sin(\sqrt{\frac{\pi^2}{l_y^2} - \lambda^2}x))(C\cos(\alpha\lambda t) + D\sin(\lambda t)). \tag{12}$$

Произведем подбор постоянных A, B, так, чтобы полученное выражение удовлетворяло граничным условиям (6).

Для удовлетворяющей указанным условиям функции $X(x)$ должно выполняться $X(0) = 0, X(l_x) = 0$. Подставляя значения $x = 0, x = l_x$ в (10), получаем:

$$0 = A\cos(0) + B\sin(0), \qquad 0 = A\cos(\sqrt{\frac{\pi^2}{l_y^2} - \lambda^2}l_x) + B\sin(\sqrt{\frac{\pi^2}{l_y^2} - \lambda^2}l_x).$$

Из первого уравнения очевидно следует $A = 0$. Из второго уравнения находим:

$$B\sin(\sqrt{\frac{\pi^2}{l_y^2} - \lambda^2}l_x) = 0. \tag{13}$$

$B \neq 0$, так как в противном случае было бы $X = 0$ и $u = 0$, что противоречит условию. Следовательно,

$$\sin(\sqrt{\frac{\pi^2}{l_y^2} - \lambda^2}l_x) = 0,$$

Откуда находим

$$\sqrt{\frac{\pi^2}{l_y^2} - \lambda^2} = \frac{n\pi}{l_x}, n = 1, 2, ...,$$
$$\lambda = \sqrt{\frac{n\pi^2}{l_x^2} + \frac{\pi^2}{l_y^2}}.$$

Итак, получаем подстановкой λ в (13) вид функции $X(x)$:

$$X(x) = B\sin(\sqrt{\frac{n\pi^2}{l_x^2}}x) = B\sin(\frac{n\pi^2}{l_x^2}x).$$

Теперь, зная значение λ, можем записать для уравнения (11):

$$T(t) = C\cos(\alpha\sqrt{\frac{n\pi^2}{l_x^2} + \frac{\pi^2}{l_y^2}}t) + D\sin(\alpha\sqrt{\frac{n\pi^2}{l_x^2} + \frac{\pi^2}{l_y^2}}t), n = 1, 2, ...,$$

Подставляя для каждого n полученные выражения в уравнение (7), получаем решение исходного уравнения (6), удовлетворяющее требуемым условиям. Обозначим это решение как $\nu_n(x,t)$:

$$\nu_n(x,t) = (C\cos(\alpha\sqrt{\frac{n\pi^2}{l_x^2} + \frac{\pi^2}{l_y^2}}t) + D\sin(\alpha\sqrt{\frac{n\pi^2}{l_x^2} + \frac{\pi^2}{l_y^2}}t))\sin(\frac{n\pi}{l_x}x) \tag{14}$$

Так как уравнение (6) является линейным и однородным, решение можно представить в виде ряда:

$$\nu(x,t) = \sum_{n=1}^{\infty}(C\cos(\alpha\sqrt{\frac{n\pi^2}{l_x^2} + \frac{\pi^2}{l_y^2}}t) + D\sin(\alpha\sqrt{\frac{n\pi^2}{l_x^2} + \frac{\pi^2}{l_y^2}}t))\sin(\frac{n\pi}{l_x}x). \tag{15}$$

Для удовлетворения начальным условиям требуется подобрать значения C_n и D_n. По условию $\nu|_{t=0} = p(x,y)$. Соответственно, подставив в (15) значение $t = 0$, получим:

$$p(x,y) = \sum_{n=1}^{\infty}\sin(\frac{n\pi}{l_x}x)C_n. \tag{16}$$

Если функция $p(x,y)$ такова, что в интервале $(0, l_x)$ ее можно разложить в ряд Фурье, условие (16) будет выполняться при:

$$C_n = \frac{2}{l_x}\int_0^{l_x} p(x)\sin(\frac{n\pi}{l_x}x)dx. \tag{17}$$

По условию $\frac{\partial\nu}{\partial t}|_{t=0} = 0$. Дифференцируем члены уравнения (15) по t и учитываем это условие:

$$0 = \sum_{n=1}^{\infty} D_n\alpha\sqrt{\frac{n\pi^2}{l_x^2} + \frac{\pi^2}{l_y^2}}\sin(\frac{n\pi}{l_x}x). \tag{18}$$

Очевидно, что $\alpha \neq 0$, следовательно, из (18) получаем $D_n = 0$.

Таким образом, приходим к аналитическому решению поставленной задачи (6) в следующем виде:

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

$$\nu(x,t) = \sum_{n=1}^{\infty}(C_n\cos(\alpha\sqrt{\frac{n\pi^2}{l_x^2} + \frac{\pi^2}{l_y^2}}t))\sin(\frac{n\pi}{l_x}x), \tag{19}$$

$$C_n = \frac{2}{l_x}\int_0^{l_x} p(x)\sin(\frac{n\pi}{l_x}x)dx.$$

3 Получение расчетной формулы

Преподавателем была выдана функция $p(x, y)$, задающая начальное отклонение мембраны и значение поверхностной плотности ρ:

$$p(x, y) = \frac{x^2}{4} + x, \quad \rho = 1.$$

Размеры мембраны были заданы как $l_x = 4, \quad l_y = 1$.

Из вывода уравнения колебаний мембраны следует, что коэффициент α^2 равен $\frac{1}{\rho}$. Следовательно, в нашем случае $\alpha = 1$ и в дальнейших расчетах не оказывает влияния.

Вычислим значение коэффициента C_n в сумме ряда (19) с учетом выданных значений.

$$C_n = \frac{2}{l_x} \int_0^{l_x} p(x) \sin(\frac{n\pi}{l_x} x) dx = \frac{1}{2} \int_0^4 p(x) \sin(\frac{n\pi}{4} x) dx = 8 \frac{n\pi \sin(n\pi) + 2\cos(n\pi) - 2}{n^3 \pi^3}.$$

Перепишем решение задачи с учетом вычисленного коэффициента C_n и выданных значений, получая итоговые расчетные формулы:

$$u(x, y, t) = \nu(x, t) \sin(\pi y),$$

$$\nu(x, t) = \sum_{n=1}^{\infty} 8 \frac{n\pi \sin(n\pi) + 2\cos(n\pi) - 2}{n^3 \pi^3} \cos(\sqrt{\frac{n\pi^2}{16} + \pi^2} t) \sin(\frac{n\pi}{4} x). \tag{20}$$

4 Оценка остатка ряда

Оценим остаток полученного ранее ряда Фурье (20). Далее обозначаем n-ый элемент ряда как $D_n(x,t)$. Остаток оценивается как сумма элементов ряда, начиная с $N+1$-го:

$$|R_n| = \left| \beta \sum_{n=N+1}^{\infty} D_n(x,t) \right|.$$

Тригонометрические функции не принимают значения по модулю больше единицы, поэтому могут быть заменены на 1:

$$|\sin(n\pi)| < 1,$$
$$|\cos(n\pi)| < 1,$$
$$|\cos(\sqrt{\frac{n\pi^2}{16} + \pi^2 t})| < 1.$$

Тогда оценка остатка ряда приобретает следующий вид:

$$|R_n| \leq \sum_{n=N+1}^{\infty} \left| 8\frac{n\pi + 4}{n^3\pi^3} \right| \leq \frac{8}{\pi^2} \int_N^{\infty} \frac{dk}{k^2} + \frac{32}{\pi^3} \int_N^{\infty} \frac{dk}{k^3} \leq \left| 8\frac{\pi N + 16}{\pi^3 N^2} \right| = \Phi(N). \tag{21}$$

Для контроля точности принимаем $\Phi(N) \leq E$.

Отсюда получаем формулу для расчета числа членов ряда для достижения заданной точности:

$$N \geq \frac{4(\sqrt{\pi E + 1} + 1)}{\pi^2 E}. \tag{22}$$

5 Экспериментальное исследование качества оценки остатка ряда

Проведём экспериментальное исследование качества оценки остатка ряда следующим образом:

1. Будем менять требуемую точность вычислений от 10^{-1} до 10^{-8}.

2. Для каждой требуемой точности рассчитаем $N_{\text{избыточное}}$ с помощью формулы (22).

3. Рассчитаем $N_{\text{достаточное}}$, уменьшая количество слагаемых в сумме, начиная с $N_{\text{избыточное}}$ до тех пор, пока результат сохраняет требуемую точность.

4. Шаги 1-3 проделаем для нескольких точек поверхности мембраны и моментов времени $t = 0$, $t = 10$ и $t = 10000$. Это необходимо для того, чтобы проследить за изменением $N_{\text{достаточное}}$ на разных временных промежутках.

Таблица 1 – Значения $N_{\text{избыточное}}$ и $N_{\text{достаточное}}$ при различных ϵ и t для точки (1;0.5).

ϵ	$t = 0$		$t = 10$		$t = 10000$	
	$N_{\text{избыточное}}$	$N_{\text{достаточное}}$	$N_{\text{избыточное}}$	$N_{\text{достаточное}}$	$N_{\text{избыточное}}$	$N_{\text{достаточное}}$
10^{-1}	5	1	5	1	5	1
10^{-2}	51	3	51	1	51	3
10^{-3}	507	9	507	7	507	5
10^{-4}	5066	19	5066	15	5066	13
10^{-5}	50661	41	50661	41	50661	37
10^{-6}	506606	89	506606	79	506606	85
10^{-7}	5066059	193	5066059	145	5066059	187
10^{-8}	50660592	417	50660592	259	50660592	409

Таблица 2 – Значения $N_{избыточное}$ и $N_{достаточное}$ при различных ϵ и t для точки (2;0.5).

ϵ	$t = 0$		$t = 10$		$t = 10000$	
	$N_{\text{избыточное}}$	$N_{\text{достаточное}}$	$N_{\text{избыточное}}$	$N_{\text{достаточное}}$	$N_{\text{избыточное}}$	$N_{\text{достаточное}}$
10^{-1}	5	1	5	1	5	1
10^{-2}	51	3	51	2	51	3
10^{-3}	507	9	507	7	507	9
10^{-4}	5066	21	5066	17	5066	21
10^{-5}	50661	45	50661	45	50661	43
10^{-6}	506606	101	506606	87	506606	99
10^{-7}	5066059	217	5066059	157	5066059	215
10^{-8}	50660592	469	50660592	283	50660592	465

Из полученных данных видно, что теоретическая оценка отличается на много порядков от практической. С математической точки зрения это объясняется тем, что в выражении (22) для расчета числа членов ряда значение требуемой точности E находится в знаменателе дроби с большим порядком, чем в числителе и, принимая малые значения, приводит к быстрому росту и низкой точности результата. Можно видеть, что $N_{\text{избыточное}}$ не зависит от времени, а $N_{\text{достаточное}}$ зависит от времени достаточно слабо. Также отметим слабую независимость $N_{\text{избыточное}}$ не зависит от точки поверхности мембраны, и слабую зависимость $N_{\text{достаточное}}$. В целом можно сделать вывод, что найденная оценка точности ряда при выполнении условия $\Phi(N) \leq E$ недостаточно корректна и плохо подходит для использования на практике из-за на порядки завышенного числа итераций по сравнению с практически необходимой.

6 Результаты вычислительных экспериментов

Ниже показан пример использования программы для моделирования процесса колебаний мембраны. Пользователю доступен ряд команд, которые вызываются путем набора в терминале программы. Команда ’-man’ в любой момент времени вызовет краткую справку по остальным командам и формату их параметров.

Вычисление значения отклонения мембраны в точке выполняется командой ’-calc x y t eps’, где x, y, t - значения координат x, y и момента времени t, eps - требуемая точность. При выполнении данной команды полученное значение выводится в терминал, в отдельном окне выводится двумерный график среза мембраны в плоскости xz в выбранный момент времени. Команды ’-g2 t’, ’-g3 t’ строят двумерный график среза мембраны и трехмерую картину ее поверхности в момент времени t соответственно. Команды ’-a2 t’, ’-a3 t’ строят аналогичные анимированные графики, показывающие процесс колебаний с 0 до момента времени t.

Ниже, на рисунках 1, 2, 3 и 4 представлены выводы программы зависимости положения точки мембраны от времени t.

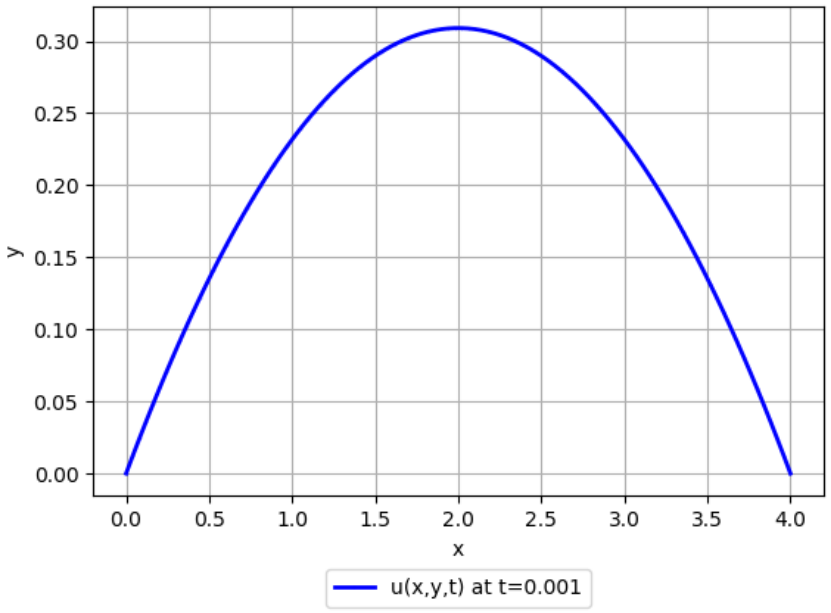

Рисунок 1 – Положения точек мембраны при $y = 0.1$, $t = 0.001$

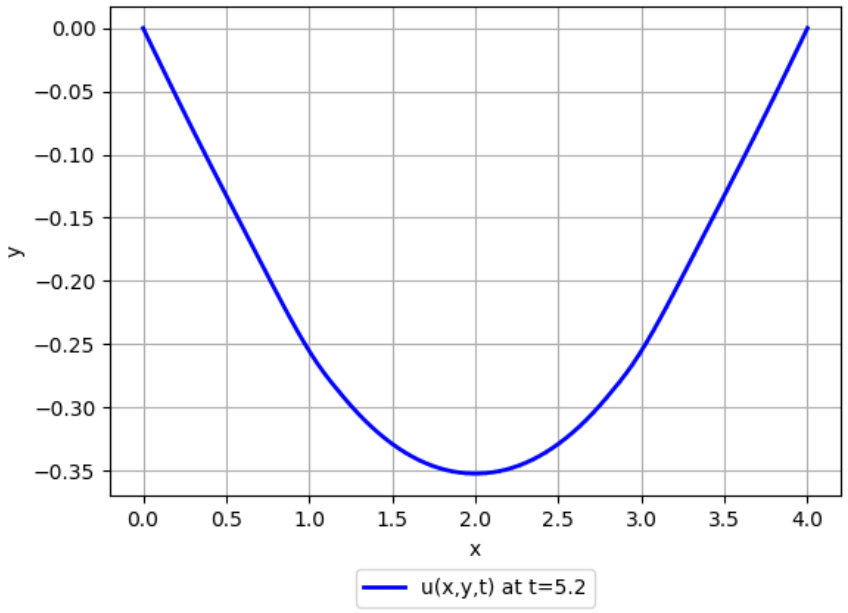

Рисунок 2 – Положения точек мембраны при $y = 0.7$, $t = 5.2$

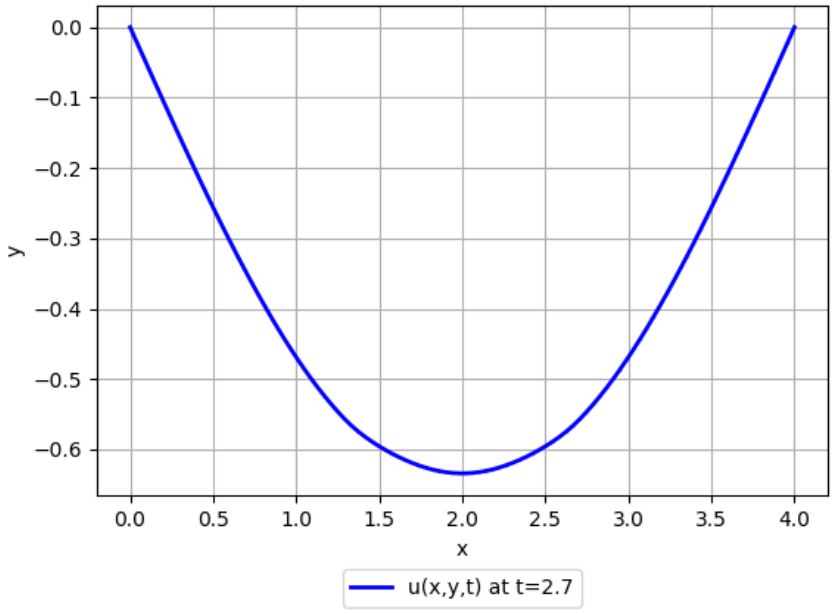

Рисунок 3 – Положения точек мембраны при $y = 0.7$, $t = 2.7$

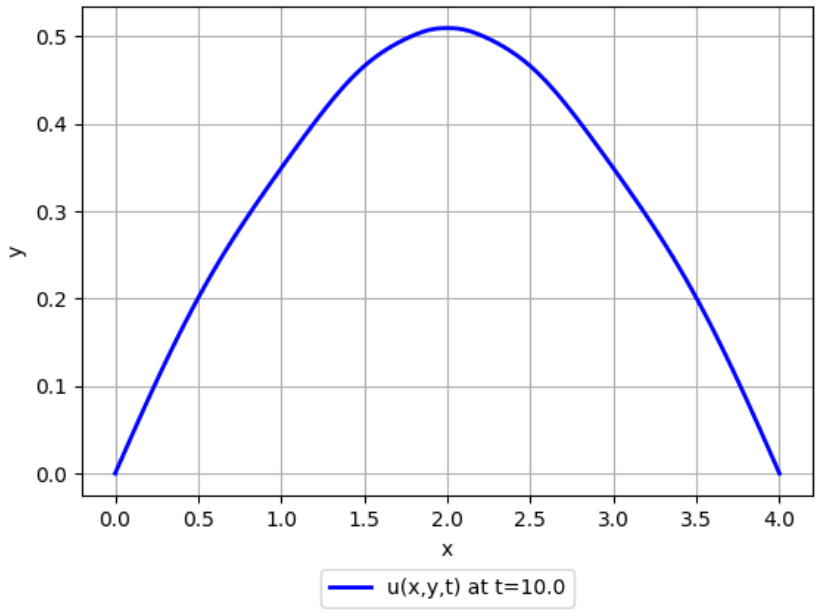

Рисунок 4 – Положения точек мембраны при $y = 0.33$, $t = 10$

На рисунках 5, 6, 7, 8 представлена трехмерная форма поверхности мембраны в моменты времени t.

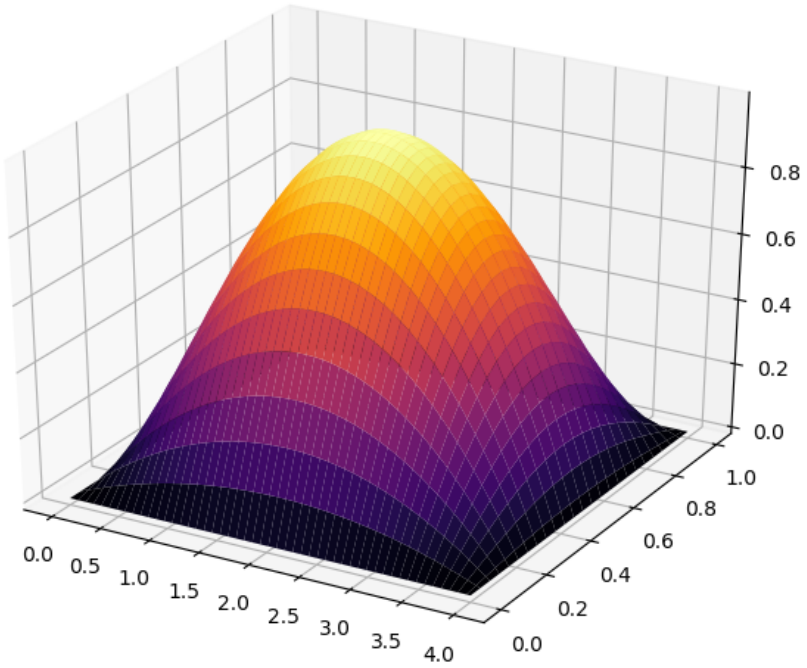

Рисунок 5 – Положения точек мембраны при $t = 0$

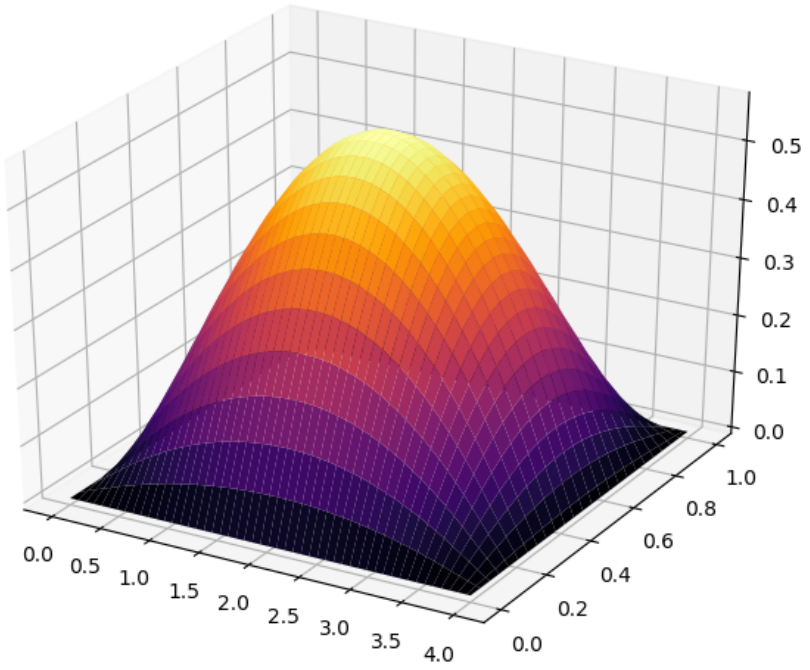

Рисунок 6 – Положения точек мембраны при $t = 0.3$

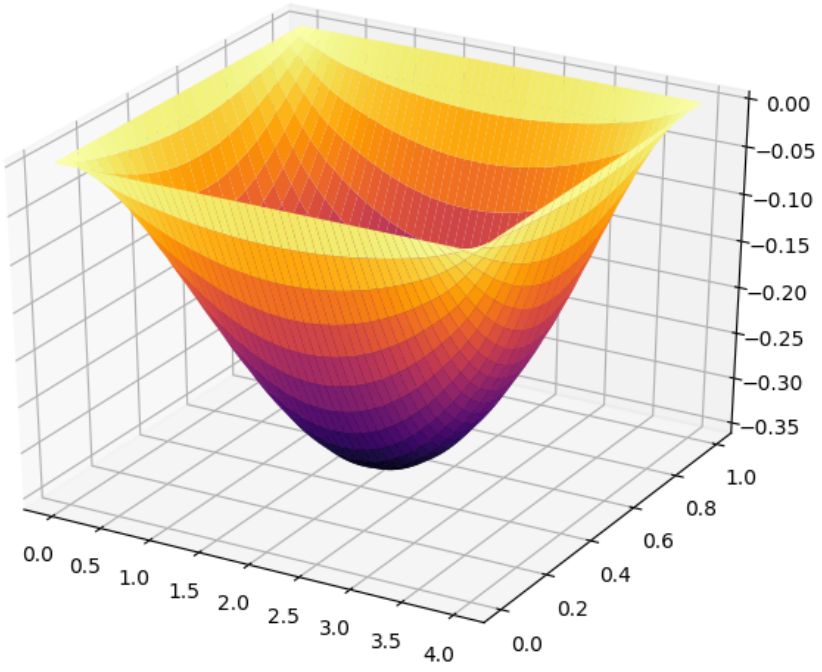

Рисунок 7 – Положения точек мембраны при $t = 0.6$

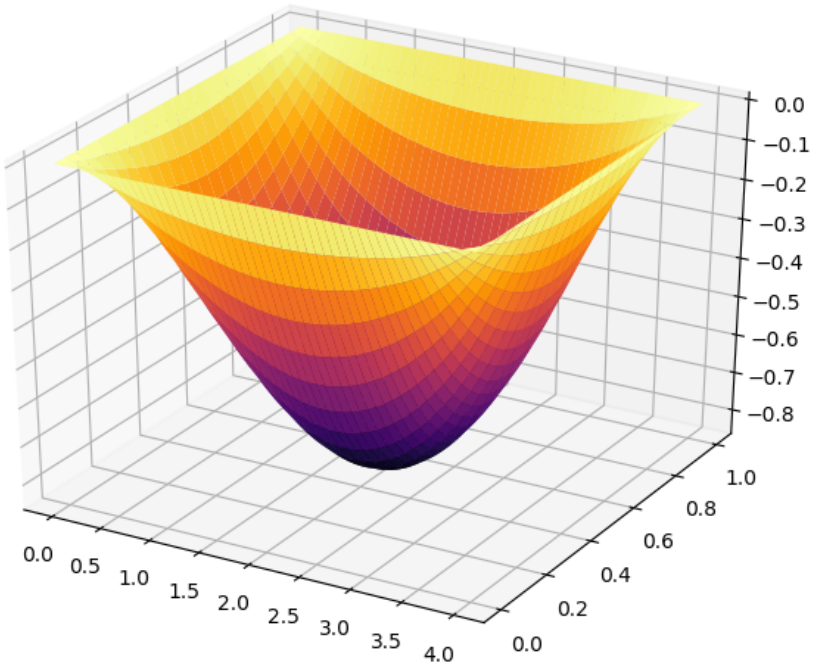

Рисунок 8 – Положения точек мембраны при $t = 0.8$

Заключение

При выполнении данной работы были изучены аналитические методы решения краевой задачи колебаний прямоугольной мембраны. Составлена постановка краевой задачи, а также было получено аналитическое решение в виде конечного ряда Фурье.

В ходе работы для решения задачи была написана программа численного моделирования процесса колебаний тонкой однородной прямоугольной мембраны. Данная программа выполняет расчеты положения точек мембраны с заданными параметрами и позволяет провести анализ погрешности решения для функции начального условия.

Анализ полученных результатов показал, что теоретическая оценка погрешности вычислений при помощи оценки остатка ряда нуждается в дальнейшем уточнении для успешного применения программы на практике.

Список литературы

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

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

3. MpMath Documentation [Электронный ресурс]: Официальный сайт документации библиотеки MpMath. - URL: http://mpmath.org/doc/current/

Приложение А

Код программы

Файл membrane.py:

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

T_MIN = 0
N_MAX = 20
GRID_STEP = 0.04
TIME_STEP = 0.1
LX = 4
LY = 1


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

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

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

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

    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], ['u(x,y,t) 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):
    fig = plt.figure()
    ax = plt.axes(xlim=(0, LX), ylim=(-LY * 2, LY * 2))
    line, = ax.plot([], [], lw=2)
    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 = x_vals
        y = y_per_time[index]
        line.set_data(x, y)
        time_text.set_text('T={0}'.format(round(index * TIME_STEP, 3)))
        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=60, blit=False)
    plt.show()


def __plot_3d(t, x, y, z):
    fig = plt.figure()
    axes = Axes3D(fig)

    axes.plot_surface(x, y, z, cmap='inferno')

    plt.show()
    name = '{0}'.format(t).replace('.', '')
    plt.savefig('{0}_3d'.format(name))

def __anim_plot_3d(x_vals, y_vals, z_per_time):
    fig = plt.figure()
    axes = Axes3D(fig)
    axes.plot_surface(x_vals, y_vals, z_per_time[0])

    ##time_text = axes.text(.5, .5, '', fontsize=15, )

    # animation function. This is called sequentially
    def animate(i):
        axes.clear()

        axes.autoscale(False, axis='z', tight=None)
        axes.set_zlim(-1, 1)

        index = i % len(z_per_time)
        z = z_per_time[index]

        # time_text.set_text('T={0}'.format(round(index*TIME_STEP,3)))
        return axes.plot_surface(x_vals, y_vals, z, cmap='inferno'), # time_text

    # call the animator. blit=True means only re-draw the parts that have changed.
    anim = animation.FuncAnimation(fig, animate, frames=200, interval=60, blit=False)
```

```
    plt.show()


def __c_n(n):
    return 8 * (np.pi * n * np.sin(np.pi * n) + 2 * np.cos(np.pi * n) - 2) / (np.pi ** 3 *
        ↪  n ** 3)


def __series_element(n, x, t):
    return np.sin(n * np.pi * x / 4) * __c_n(n) * np.cos(np.sqrt((n * np.pi / 4) ** 2 + np
        ↪ .pi ** 2) * t)


def __series_sum_2d(N, x, t):
    res = 0
    for n in range(1, N):
        res += -__series_element(n, x, t)
    return res


def __series_sum_3d(N, x, y, t):
    res = 0
    for n in range(1, N):
        res += -__series_element(n, x, t)
    return res * np.sin(np.pi * y / LY)


def static_2d(time):
    x = np.linspace(0, LX, int(LX / GRID_STEP))

    start = timeit.default_timer()
    print("Starting calculation for 2d...")

    res = __series_sum_2d(N_MAX, x, time)

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

```
    __plot_2d(x, res, time)


def animated_2d(time, n=N_MAX):
    time_slices_count = int((time - T_MIN) / TIME_STEP)
    T = np.linspace(T_MIN, time, time_slices_count)

    grid_points_count = int(LX / GRID_STEP)
    x_vals = np.linspace(0, LX, grid_points_count)

    values_per_time = []

    start = timeit.default_timer()
    print("Starting calculation for animated 2d...")
    for t in T:
        res = []
        for x in x_vals:
            subres = __series_sum_2d(n, x, t)
            res.append(subres)
        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)


def animated_3d(time, n=N_MAX):
    time_slices_count = int((time - T_MIN) / TIME_STEP)
    T = np.linspace(T_MIN, time, time_slices_count)

    x = np.linspace(0, LX, int(LX / GRID_STEP))
    y = np.linspace(0, LY, int(LY / GRID_STEP))

    start = timeit.default_timer()
    print("Starting calculation for animated 3d...")

    xv, yv = np.meshgrid(x, y)
    values_per_time = []
```

```
    for t in T:
        res = __series_sum_3d(n, xv, yv, t)
        values_per_time.append(res)

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

    __anim_plot_3d(xv, yv, values_per_time)


def static_3d(time, n=N_MAX):
    x = np.linspace(0, LX, int(LX / GRID_STEP))
    y = np.linspace(0, LY, int(LY / GRID_STEP))

    start = timeit.default_timer()
    print("Starting calculation for 3d...")

    xv, yv = np.meshgrid(x, y)

    res = __series_sum_3d(n, xv, yv, time)

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

    __plot_3d(time, xv, yv, res)


def calculate_with_precision(x, y, t, eps):
    x_sp = np.linspace(0, LX, int(LX / GRID_STEP))
    n = int(find_n(eps))
    n = 500

    start = timeit.default_timer()
    print("Starting calculation with given precision {0}, N is {1}".format(eps, n))
    res = __series_sum_3d(n, x, y, t)
    end = timeit.default_timer()
    print("Finished calculation in {0}s".format(round(end - start, 5)))
```

```
    print("Result: {0}".format(res))

    y_sp = __series_sum_3d(n, x_sp, y, t)

    __plot_2d(x_sp, y_sp, t, x, y, True)


def find_n(eps):
    ##return (4 * np.sqrt(np.pi*eps+1)) / (np.pi**2 * eps)
    return (4 * np.sqrt(np.pi * eps + 1) + 1) / ((np.pi ** 2) * eps)


def test_precision(t, eps, n_step):
    x = 2
    y = 0.5
    x_sp = np.linspace(0, LX, int(LX / GRID_STEP))
    n = int(find_n(eps))

    print("Starting test for ", eps)
    print("Theor.n is ", n)
    print("max number of calcs is", n / n_step)
    n = 600
    prev_res = __series_sum_3d(n, x, y, t)
    while True:
        res = __series_sum_3d(n, x, y, t)
        if abs(res - prev_res) < eps and n >= 0:
            prev_res = res
            n -= n_step
            print('n: ', n, ' d: ', abs(res - prev_res))
            continue
        else:
            print("found n as ", n, " for eps=", eps)
            break


if __name__ == '__main__':
    test_precision(10000, 0.1, 1)
```