

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»
(Самарский университет)

Институт информатики, математики и электроники
Факультет информатики
Кафедра технической кибернетики

Отчет по курсовой работе
Дисциплина: «Уравнения математической физики»

**Аналитическое решение краевых задач
математической физики**

Вариант №8

Студенты группы 6407	Анурин А.С. Григина В.С. Попов Д.С.
Проверил	Дегтярев А.А.

ЗАДАНИЕ К КУРСОВОЙ РАБОТЕ

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

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.

Исследовать сходимость ряда Фурье, получить оценку остатка ряда.

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

Используя разработанную программу, провести экспериментальное исследование качества полученной аналитической оценки остатка ряда Фурье.

Оформить отчет о выполненной курсовой работе.

ВАРИАНТ 8

Разработать программу расчета динамики поля концентрации вещества в цилиндре на временном промежутке $0 < t \leq T$. Для численных расчетов использовать представление решения описанной задачи математической физики в виде ряда Фурье по собственным функциям оператора Лапласа, удовлетворяющим краевым условиям задачи.

В цилиндре конечной длины l находится диффундирующее вещество, концентрация частиц которого в момент времени $t = 0$ описывается функцией:

$$u|_{t=0} = \psi(x), \quad 0 \leq x \leq l,$$

где x — числовая ось, направленная вдоль оси цилиндра.

Коэффициент диффузии вещества является постоянным и равен D . Концы цилиндра закрыты полунепроницаемыми мембранами, через которые происходит диффузия вещества. Предполагается, что поток вещества через мембраны пропорционален разности концентраций вещества во внешней среде и в цилиндре, т.е.:

$$\left[\frac{\partial u}{\partial x} - H(u - u_c) \right]_{x=0} = 0,$$
$$\left[\frac{\partial u}{\partial x} + H(u - u_c) \right]_{x=l} = 0,$$

где u_c — концентрация вещества во внешней по отношению к цилиндру среде, H — мембранный коэффициент диффузии.

Боковая поверхность цилиндра является непроницаемой.

Процесс диффузии вещества сопровождается явлением размножения его частиц, причем скорость размножения в каждой точке пропорциональна их концентрации в этой точке. Коэффициент пропорциональности равен β

Концентрацию вещества считать одинаковой во всех точках поперечного сечения цилиндра в любой момент времени.

При проведении расчетов использовать значения параметров и выражение функции, указанные преподавателем:

$$l = 20\text{м},$$

$$D = 0.6\text{м}^2/\text{с},$$

$$u_c = 0\text{кг}/\text{м}^3,$$

$$H = 4\text{м}^2/\text{с},$$

$$T = 100\text{с},$$

$$\beta = 0,$$

$$\psi(x) = 0.4 \sin^2\left(\frac{\pi x}{l}\right) \text{кг}/\text{м}^3.$$

РЕФЕРАТ

Отчет по курсовой работе: 35 с., 5 рисунков, 2 таблицы, 5 источников, 3 приложения.

УРАВНЕНИЯ МАТЕМАТИЧЕСКОЙ ФИЗИКИ, КРАЕВАЯ ЗАДАЧА, УРАВНЕНИЕ ДИФФУЗИИ, МЕТОД РАЗДЕЛЕНИЯ ПЕРЕМЕННЫХ, СОБСТВЕННЫЕ ФУНКЦИИ ОПЕРАТОРА ЛАПЛАСА, РЯД ФУРЬЕ, ОЦЕНКА ОСТАТКА РЯДА.

Целью курсовой работы является получение решения краевой задачи диффузии для цилиндра конечной длины в виде разложения в ряд Фурье по собственным функциям оператора Лапласа и создание компьютерной программы для расчета функции-решения с заданной точностью.

Для получения аналитического решения краевой задачи использован метод разделения переменных (метод Фурье). Решение задачи получено в виде бесконечного ряда Фурье-Бесселя. Получена оценка остатка ряда.

Разработана компьютерная программа, обеспечивающая расчет и графическую визуализацию концентрации вещества в цилиндре на временном промежутке. Для контроля погрешности усечения бесконечного ряда использована оценка остатка ряда.

Приведены графические результаты численного решения задачи диффузии, а также результаты экспериментального исследования практической пригодности полученной оценки остатка ряда.

Программа написана на языке Python 3.7 в среде разработки Pycharm 2018, операционная система macOS Mojave.

СОДЕРЖАНИЕ

Введение	7
1 Математическая постановка задачи	8
2 Аналитическое решение краевой задачи диффузии методом разделения переменных	9
3 Расчет собственных чисел оператора Лапласа	12
4 Расчет интегральных коэффициентов ряда Фурье.....	15
5 Исследование сходимости ряда Фурье. Оценка остатка ряда	16
6 Исследование практической пригодности оценки остатка ряда с помощью вычислительных экспериментов	19
7 Сценарий использования программы.....	21
Заключение.....	24
Список использованных источников.....	25
Приложение А Код интерфейса оконного приложения	26
Приложение Б Код вычислительного модуля	30
Приложение В Код модуля символьного исчисления	33

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.

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 краевой задачи третьего вида диффузии. На основе этого результата разработан алгоритм и компьютерная программа численного моделирования процесса диффузии в цилиндре.

1 Математическая постановка задачи

Рассмотрим цилиндр конечной длины, концентрация вещества которого в точке пространства (x, y, z) в момент времени t определяется функцией $u(x, y, z, t)$.

Математическая модель будет выглядеть следующим образом:

$$\begin{cases} u_t(x, t) = Du_{xx}(x, t), & 0 < x < l, \quad 0 < t \leq T; \\ u(x, 0) = \psi(x), & 0 \leq x \leq l; \\ u_x(0, t) = H(u(0, t) - u_c), & 0 < t \leq T; \\ u_x(l, t) = -H(u(l, t) - u_c), & 0 < t \leq T. \end{cases} \quad (1.1)$$

Сделаем замену $u(x, t) = v(x, t) + w(x, t)$ [1] так, чтобы граничные условия задачи (1.1) стали однородными относительно $v(x, t)$.

$$\begin{cases} v(x, 0) + w(x, 0) = \psi(x), & 0 \leq x \leq l; \\ v_x(0, t) + w_x(0, t) = H(v(0, t) + w(0, t) - u_c), & 0 < t \leq T; \\ v_x(l, t) + w_x(l, t) = -H(v(l, t) + w(l, t) - u_c), & 0 < t \leq T. \end{cases} \quad (1.2)$$

Очевидно, мы можем взять $w(x, t) = u_c$.

Тогда после замены итоговая математическая модель будет иметь вид:

$$\begin{cases} v_t(x, t) = Dv_{xx}(x, t), & 0 < x < l, \quad 0 < t \leq T; \\ v(x, 0) = \psi(x) - u_c = \varphi(x), & 0 \leq x \leq l; \\ v_x(0, t) - Hv(0, t) = 0, & 0 < t \leq T; \\ v_x(l, t) + Hv(l, t) = 0, & 0 < t \leq T. \end{cases} \quad (1.3)$$

2 Аналитическое решение краевой задачи диффузии методом разделения переменных

Поиском аналитического решения занимался студент Гринина В.С.

Ищем решение задачи в виде ряда по собственным функциям задачи Штурма-Лиувилля [2]:

$$v(x, t) = \sum_{n=1}^{\infty} v_n(x, t) = \sum_{n=1}^{\infty} T_n(t) X_n(x). \quad (2.1)$$

Рассмотрим один член ряда отдельно, что нам позволяет сделать ортогональность собственных функций.

$$v_n(x, t) = T_n(t) X_n(x). \quad (2.2)$$

Тогда первое уравнение модели примет вид:

$$T'(t) X(x) = DX''(x) T(t). \quad (2.3)$$

Разделяя переменные, получим:

$$\frac{T'(t)}{T(t)} = \frac{DX''(x)}{X(x)} = -\lambda^2, \quad (2.4)$$

где $\lambda^2 = \text{const}$, так как левая часть зависит только от t , а правая — от x .

Заметим, что решение нашей задачи относительно $X(x)$ сводится к решению задачи Штурма-Лиувилля:

$$\begin{cases} DX''(x) = -\lambda^2(x); \\ X'(0) - HX(0) = 0; \\ X'(l) + HX(l) = 0. \end{cases} \quad (2.5)$$

Так как для данных краевых условий аналитическое выражение решения задачи может оказаться очень сложным, введем дополнительное условие симметричности функции $\psi(x)$ относительно значения $x = \frac{l}{2}$.

Тогда задача будет иметь следующий вид:

$$\begin{cases} DX''(x) = -\lambda^2 X(x); \\ X'(0) - HX(0) = 0; \\ X'(\frac{l}{2}) = 0. \end{cases} \quad (2.6)$$

Из характеристического уравнения задачи получим:

$$Dk^2 + \lambda^2 = 0. \quad (2.7)$$

Найдем общий вид решения и его первой производной:

$$X(x) = C_1 \cos\left(\frac{\lambda}{\sqrt{D}}x\right) + C_2 \sin\left(\frac{\lambda}{\sqrt{D}}x\right), \quad (2.8)$$

$$X'(x) = -C_1 \frac{\lambda}{\sqrt{D}} \sin\left(\frac{\lambda}{\sqrt{D}}x\right) + C_2 \frac{\lambda}{\sqrt{D}} \cos\left(\frac{\lambda}{\sqrt{D}}x\right). \quad (2.9)$$

Подставим уравнения (2.8) и (2.9) в краевые условия и получим:

$$\begin{cases} C_2 \frac{\lambda}{\sqrt{D}} = HC_1; \\ \left(-C_1 \frac{\lambda}{\sqrt{D}} + HC_2\right) \sin\left(\frac{\lambda}{\sqrt{D}}l\right) + (C_2 + HC_1) \cos\left(\frac{\lambda}{\sqrt{D}}l\right) = 0; \\ -C_1 \frac{\lambda}{\sqrt{D}} \sin\left(\frac{\lambda}{2\sqrt{D}}l\right) + C_2 \frac{\lambda}{\sqrt{D}} \cos\left(\frac{\lambda}{2\sqrt{D}}l\right) = 0. \end{cases} \quad (2.10)$$

Решив данную систему, получим уравнение:

$$\frac{\lambda}{H\sqrt{D}} \sin\left(\frac{\lambda l}{2\sqrt{D}}\right) = \cos\left(\frac{\lambda l}{2\sqrt{D}}\right). \quad (2.11)$$

Введем две замены:

$$\alpha = \frac{\lambda l}{2\sqrt{D}}, \quad (2.12)$$

$$C = \frac{Hl}{2}. \quad (2.13)$$

Упростив уравнение (2.11), получим:

$$\operatorname{tg} \alpha = \frac{C}{\alpha}. \quad (2.14)$$

Уравнение (2.14) не решается аналитически. Его численное решение представлено в разделе 3.

Подставим в (2.8) найденные численно собственные числа α_n и получим выражение для собственных функций:

$$X_n(x) = \frac{2\alpha_n}{Hl} \cos\left(\frac{2\alpha_n}{l}x\right) + \sin\left(\frac{2\alpha_n}{l}x\right). \quad (2.15)$$

Теперь рассмотрим уравнение (2.4) относительно функции $T(t)$. Получим следующее уравнение:

$$\frac{T'(t)}{T(t)} = -\lambda^2. \quad (2.16)$$

Решим это уравнение методом разделения переменных:

$$T_n(t) = A_n e^{-\lambda^2 t}, \quad (2.17)$$

где A_n — интегральный коэффициент ряда Фурье. Нахождение этих коэффициентов представлено в разделе 4.

Итоговая формула расчета концентрации вещества:

$$\begin{aligned} v(x, t) &= \sum_{n=1}^{\infty} X_n(x) \cdot T_n(t) = \\ &= \sum_{n=1}^{\infty} A_n e^{-\lambda^2 t} \cdot \left(\frac{2\alpha_n}{Hl} \cos\left(\frac{2\alpha_n}{l}x\right) + \sin\left(\frac{2\alpha_n}{l}x\right) \right). \end{aligned} \quad (2.18)$$

3 Расчет собственных чисел оператора Лапласа

Расчетом собственных чисел оператора Лапласа занимался студент Гринина В.С.

Из исходной краевой задачи методом разделения переменных было получено, что собственные числа удовлетворяют уравнению (2.14), которое является трансцендентным.

Рисунок 1 показывает решения этого уравнения с заданными параметрами.

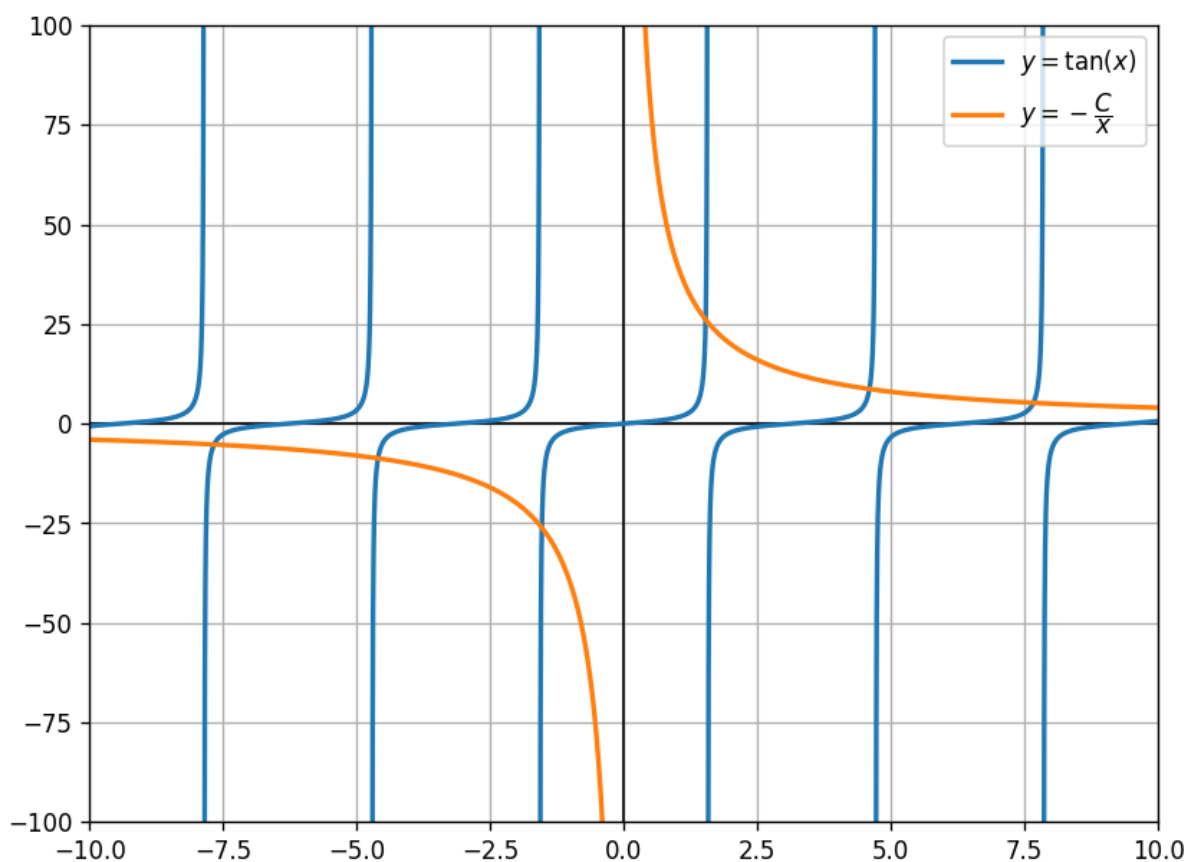


Рисунок 1 — Решения уравнения (2.14)

Найдем численные решения уравнения (2.14) с помощью метода Ньютона, так как он обладает квадратичной скоростью сходимости и прост в реализации. Таким образом, сведем задачу к задаче численного нахождения корней функции:

$$f(x) = \operatorname{tg} x - \frac{C}{x}. \quad (3.1)$$

Проверим условия сходимости метода Ньютона [3], если мы задаем начальное приближение x_0 на отрезке (a, b) :

1. $f''(x)$ не меняет своего знака на (a, b) ;
2. $f'(x) \neq 0$;
3. $\forall x_0 \in (a, b), \quad f(x_0)f''(x_0) > 0$.

Для уравнения (3.1) все эти условия выполняются. Решение уравнения будем искать следующим итеративным методом:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \quad (3.2)$$

Начальное приближение $x_0^{(k)}$ для поиска k -го значения будем выбирать слева от полюсов с учетом погрешности ε следующим образом:

$$x_0^{(k)} = \frac{\pi}{2} + \pi k - \varepsilon. \quad (3.3)$$

Рисунок 2 показывает график функции (3.1) с отмеченными корнями и асимптотами.

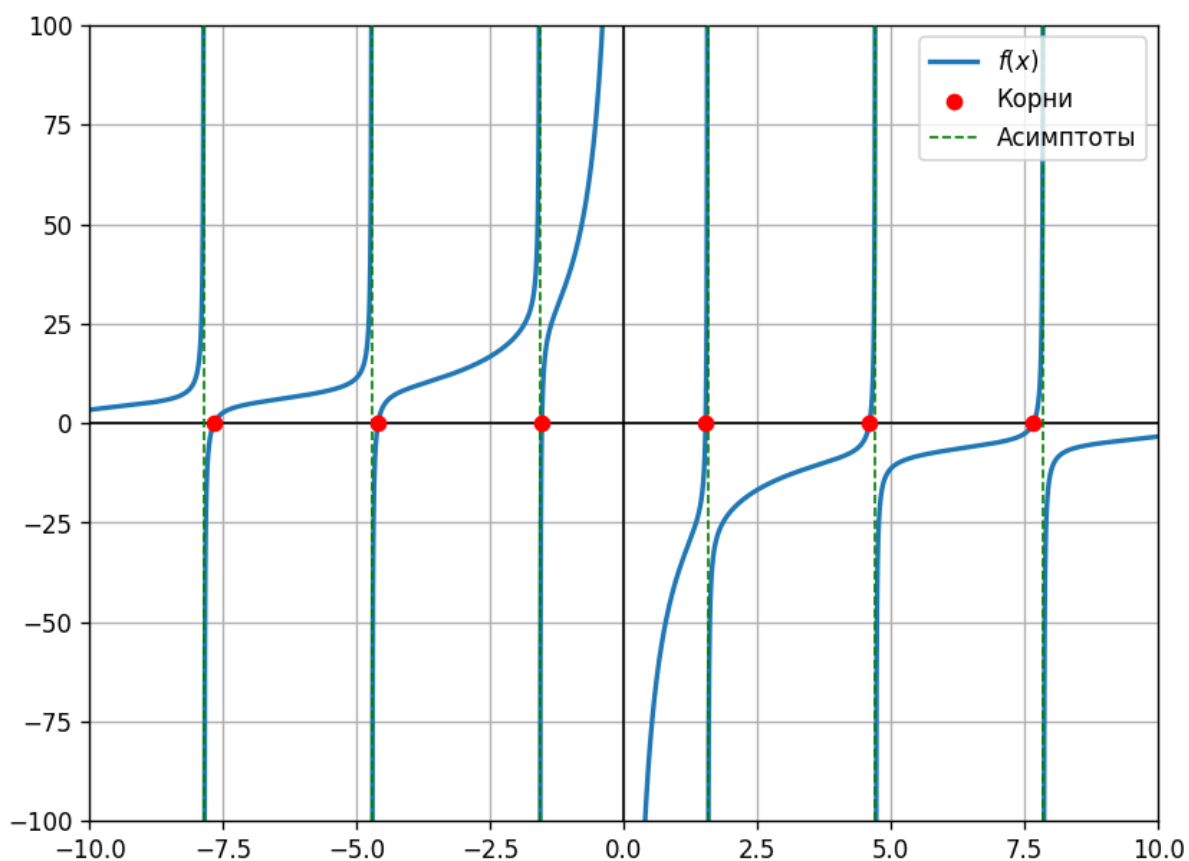


Рисунок 2 — График функции (3.1)

Таблица 1 показывает результаты использования метода Ньютона для нахождения коэффициентов α_n .

Таблица 1 — Значения корней функции (3.1)

n	α_n	n	α_n	n	α_n	n	α_n
0	1.532502	10	32.307315	20	63.394732	30	94.647633
1	4.597943	11	35.403797	21	66.514871	31	97.777691
2	7.664660	12	38.503570	22	69.636431	32	100.908363
3	10.733415	13	41.606420	23	72.759288	33	104.039604
4	13.804845	14	44.712127	24	75.883335	34	107.171372
5	16.879445	15	47.820472	25	79.008471	35	110.303630
6	19.957554	16	50.931240	26	82.134607	36	113.436343
7	23.039371	17	54.044232	27	85.261661	37	116.569481
8	26.124969	18	57.159259	28	88.389560	38	119.703016
9	29.214319	19	60.276146	29	91.518238	39	122.836920

4 Расчет интегральных коэффициентов ряда Фурье

Расчетом интегральных коэффициентов ряда Фурье занимался студент Анурийн А.С.

Для множества ортогональных базисных функций $X_n(x)$ найдем норму n -го члена как:

$$\begin{aligned} L_n &= \int_0^l X_n^2(x) dx = \\ &= \frac{l}{2} - \frac{l \sin(4\alpha_n)}{8\alpha_n} + \frac{\sin^2(2\alpha_n)}{H} + \frac{2\alpha_n^2}{H^2 l} + \frac{\alpha_n \sin(4\alpha_n)}{2H^2 l}. \end{aligned} \quad (4.1)$$

Используя этот результат, выразим интегральный коэффициент ряда Фурье A_n через $X_n(x)$ и краевое условие:

$$\begin{aligned} A_n &= \frac{1}{L_n} \int_0^l \varphi(x) X_n(x) dx = \\ &= \frac{1}{L_n} \frac{\pi^2 (Hl \cos(2\alpha_n) - Hl - 2\alpha_n \sin(2\alpha_n))}{10H\alpha_n(\alpha_n^2 - \pi^2)}. \end{aligned} \quad (4.2)$$

5 Исследование сходимости ряда Фурье. Оценка остатка ряда

Сходимость ряда Фурье исследовал студент Попов Д.С.

Попробуем найти верхнюю оценку на сумму ряда (2.18). Вспомним найденное значение последовательности функций $T_n(t)$ из (2.17):

$$T_n(t) = A_n e^{-\lambda^2 t}. \quad (5.1)$$

Так как $e^{-\lambda^2 t} \leq 1$ при $t \geq 0$, оценим:

$$T_n(t) \leq A_n. \quad (5.2)$$

Из уравнения (2.15):

$$X_n(x) = \frac{2\alpha_n}{Hl} \cos\left(\frac{2\alpha_n}{l}x\right) + \sin\left(\frac{2\alpha_n}{l}x\right). \quad (5.3)$$

Оценим $|X_n(x)|$ сверху:

$$|X_n(x)| \leq \frac{2\alpha_n}{Hl}. \quad (5.4)$$

Из уравнения (4.1):

$$L_n = \frac{1}{2} - \frac{l \sin(4\alpha_n)}{8\alpha_n} + \frac{1 - \cos(4\alpha_n)}{2H} + \frac{4\alpha_n^2 + \alpha_n \sin(4\alpha_n)}{2Hl}. \quad (5.5)$$

Заметим, что:

$$\frac{1}{2} - \frac{l \sin(4\alpha_n)}{8\alpha_n} + \frac{1 - \cos(4\alpha_n)}{2H} \geq 0, \quad \alpha_n > \frac{l}{8}. \quad (5.6)$$

Тогда:

$$\left| \frac{1}{L_n} \right| \leq \frac{2Hl}{4\alpha_n^2 + \alpha_n \sin(4\alpha_n)}. \quad (5.7)$$

Учитывая, что из (2.14):

$$\begin{aligned} \sin(\alpha_n) &= \frac{Hl \cos(\alpha_n)}{\alpha_n}, \\ \cos^2(\alpha_n) &\leq 1, \end{aligned} \quad (5.8)$$

и используя тригонометрические тождества:

$$\begin{aligned}\sin(2x) &= 2\sin(x)\cos(x), \\ \cos(2x) &= 2\cos^2(x) - 1,\end{aligned}\tag{5.9}$$

воспользуемся тождествами (5.8) и (5.9) и распишем $\sin(4\alpha_n)$:

$$\begin{aligned}\sin(4\alpha_n) &= 2\sin(2\alpha_n)\cos(2\alpha_n) = \\ &= 4\sin(\alpha_n)\cos(\alpha_n)\cos(2\alpha_n) = \\ &= \frac{4Hl}{\alpha_n}\cos^2(\alpha_n)(2\cos^2(\alpha_n) - 1).\end{aligned}\tag{5.10}$$

Оценим получившееся выражение сверху:

$$\frac{4Hl}{\alpha_n}\cos^2(\alpha_n)(2\cos^2(\alpha_n) - 1) \leq \frac{4Hl}{\alpha_n},\tag{5.11}$$

то есть:

$$\left| \frac{1}{L_n} \right| \leq \frac{H^2 l}{2\alpha_n^2 + 2Hl}.\tag{5.12}$$

Пользуясь (5.8):

$$\begin{aligned}\left| \sum -a \right| &= \left| \sum a \right|, \\ \cos(2\alpha_n) &\leq 1, \\ \sin(2\alpha_n) &\leq \frac{2Hl}{\alpha_n},\end{aligned}\tag{5.13}$$

оценим:

$$\left| L_n A_n \right| \leq \frac{4\pi^2}{5\alpha_n(\alpha_n^2 - \pi^2)}.\tag{5.14}$$

Отсюда получается, что:

$$\begin{aligned}
\left| \frac{1}{L_n} A_n X_n \right| &\leq \frac{H^2 l}{2\alpha_n^2 + 2Hl} \frac{2\alpha_n}{Hl} \frac{4\pi^2 l}{5\alpha_n(\alpha_n^2 - \pi^2)} \leq \\
&\leq \frac{H^2 l}{\alpha_n^2 + Hl} \frac{2\alpha_n}{Hl} \frac{2\pi^2 l}{\alpha_n(\alpha_n^2 - \pi^2)} \leq \\
&\leq \frac{4\pi^2 H}{5} \frac{1}{(\alpha_n^2 + Hl)(\alpha_n^2 - \pi^2)}.
\end{aligned} \tag{5.15}$$

Так как в пределе $\lambda_n - \lambda_{n-1} = \pi$, мы можем оценить $\pi n - \pi \leq \alpha_n \leq \pi n$ и:

$$\left| \frac{1}{L_n} X_n A_n \right| \leq \frac{4\pi^2 H}{5} \frac{1}{\left(2(\pi n - \pi)^2 + Hl\right)\left((\pi n - \pi)^2 - \pi^2\right)}. \tag{5.16}$$

Подставив исходные значения H и l , оценим:

$$\left| \frac{1}{L_n} X_n A_n \right| \leq \frac{16}{5} \frac{1}{(n-2)n(\pi^2(n-1)^2 + 40)}. \tag{5.17}$$

В итоге получаем:

$$\begin{aligned}
\left| \sum_{n=N+1}^{\infty} T_n(t) X_n(x) \right| &\leq \int_{N+1}^{\infty} \frac{16}{5} \frac{1}{(n-2)n(\pi^2(n-1)^2 + 40)} \leq \\
&\leq \frac{40 \ln\left(\frac{N+1}{N}\right) - 2\sqrt{10}\pi \left(\pi - 2 \arctg\left(\frac{N}{2\sqrt{10}}\right) \right)}{25(40 + \pi^2)}.
\end{aligned} \tag{5.18}$$

6 Исследование практической пригодности оценки остатка ряда с помощью вычислительных экспериментов

Исследованием практической пригодности оценки ряда занимался студент Анурин А.С.

Получив оценку ряда, проведем исследование ее качества, сравнив теоретически предсказанное количество членов ряда (2.18) N_T , необходимое для достижения необходимой точности ε , с практически найденным количеством членов N_p . Так как оценка ряда не зависит от координат x и t , нам придется посчитать ее только один раз. Напротив, практическое количество членов мы посчитаем в нескольких точках диапазона $0 \leq x \leq l$, $0 \leq t \leq T$. Таблица 2 показывает найденные значения.

Таблица 2 — Количество членов в различных оценках остатка ряда

ε	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
N_T	1	7	85	848	8476	84753
$N_p, x=10, t=10^1$	1	2	2	3	3	4
$N_p, x=5, t=10^1$	1	2	2	3	3	4
$N_p, x=0, t=10^1$	1	1	2	2	3	3
$N_p, x=10, t=10^0$	1	2	4	6	7	9
$N_p, x=5, t=10^0$	1	2	3	5	7	9
$N_p, x=0, t=10^0$	1	1	3	5	7	9
$N_p, x=10, t=10^{-1}$	1	2	4	8	13	19
$N_p, x=5, t=10^{-1}$	1	2	3	5	7	7
$N_p, x=0, t=10^{-1}$	1	2	5	10	16	22
$N_p, x=10, t=10^{-2}$	1	2	4	9	16	27
$N_p, x=5, t=10^{-2}$	1	2	3	5	7	25
$N_p, x=0, t=10^{-2}$	1	2	6	15	28	44

Из таблицы 2 видно, что медленнее всего ряд сходится в точках, расположенных на концах цилиндра, в моменты времени близкие к

начальному. Возможно, это связано с маленькими значениями концентрации в этих точках и, соответственно, медленными темпами диффузии.

В целом, теоретическая оценка оказалась мало эффективна на практике из-за своего быстрого роста. Однако она всё равно позволяет судить о сходимости самого ряда.

7 Сценарий использования программы

Реализацией программного решения занимался студент Анурин А.С.

В данной работе активно использовались программные инструменты на основе языка программирования Python 3. С помощью библиотеки символьного исчисления SymPy [4] был верифицирован процесс нахождения аналитического решения задачи. С помощью библиотеки Matplotlib [5] были созданы графики и иллюстрации. Была реализована программа численного моделирования процесса диффузии вещества в цилиндре с полунепроницаемыми мембранами.

Также было создано оконное приложение, которое интегрируется с системой символьного исчисления. Приложение производит поиск собственных чисел α_n и коэффициентов интегрального разложения A_n для нахождения суммы ряда и последующей визуализации. Программное решение позволяет рассчитать поле концентрации вещества в цилиндре в любой момент времени для любого разумного набора входных параметров.

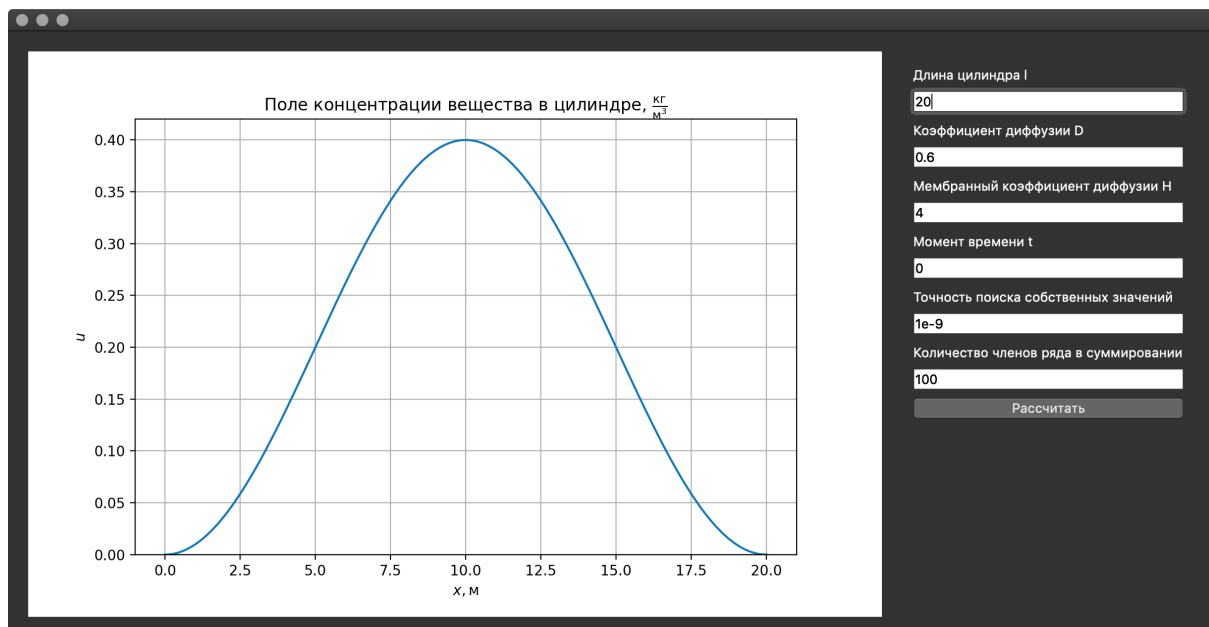


Рисунок 3 — Интерфейс оконного приложения

Рисунок 3 иллюстрирует оконное приложение. Видно, что концентрация в нулевой момент времени совпадает с начальным условием

из (1.3). Программа позволяет пользователю ввести желаемые начальные параметры и получить график поля концентрации вещества. Присутствует обработчик ошибок на случай, если пользователь введет некорректные данные. Рисунок 4 иллюстрирует такую ситуацию.

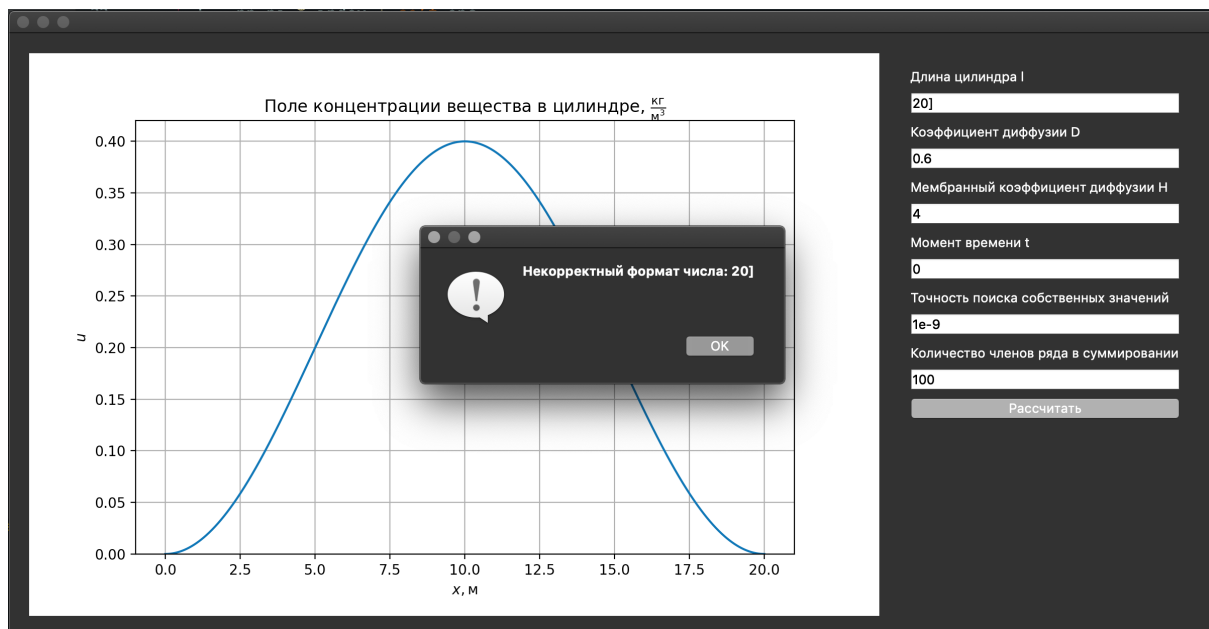


Рисунок 4 — Ввод некорректного значения

В частности, программа позволяет рассмотреть интересный частный случай, когда коэффициент диффузии D относительно велик по сравнению с мембранным коэффициентом диффузии H . В такой ситуации вещество должно практически равномерно распределиться по длине цилиндра, лишь в небольшой степени диффундируя через стенки. Рисунок 5 иллюстрирует такой частный случай.

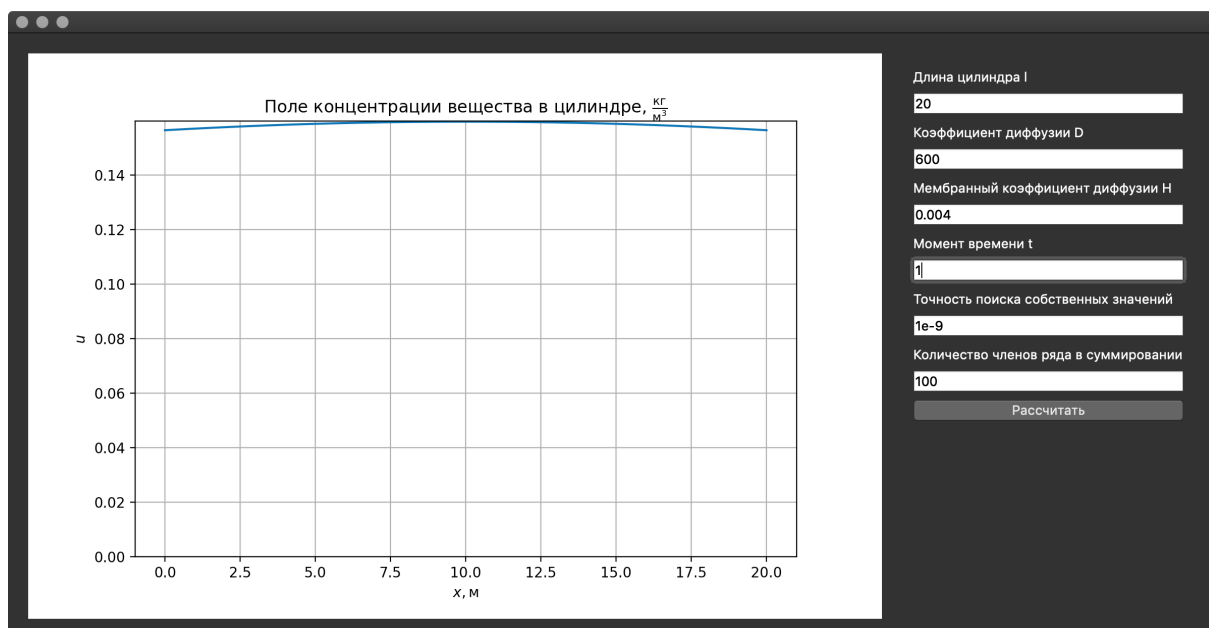


Рисунок 5 — Иллюстрация частного случая

Использование векторных оптимизаций в реализации программы позволяет достичь высокой степени быстродействия, сохраняя точность. При задании стандартных входных данных программа вычисляет значения функции $u(x, t)$ с точностью до $\varepsilon = 10^{-7}$, затрачивая не более 800нс на одну сумму ряда. Весь программный код приложения приведен в приложениях.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была осуществлена постановка краевой задачи математической физики. Было получено разложение решения задачи в ряд Фурье по последовательности ортогональных базисных функций. Аналитическое выражение собственных чисел и коэффициентов ряда оказалось невозможным, однако они были найдены приближенно численными методами.

Была установлена сходимость ряда и найдена аналитическая оценка остатка ряда. Был произведен анализ соответствия этой оценки численным результатам.

Был реализован программный комплекс для символьного исчисления решения задачи и численного поиска собственных чисел и коэффициентов ряда. Также было создано оконное приложение, способное визуализировать поле концентрации вещества в цилиндре для различных наборов входных параметров. Пользователю предоставляется возможность вводить желаемые входные параметры, после чего проводится проверка этих параметров на корректность и расчет поля концентрации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Владимиров В.С. Уравнения математической физики: учебник для вузов [Текст] / Владимиров В.С., Жаринов В.В. – М.: Физико-математическая литература, 2000. – 399 с.

2 Тихонов А.Н. Уравнения математической физики: учебное пособие для университетов [Текст] / Тихонов А.Н., Самарский А.А. – М.: Наука, 1972. – 736 с.

3 Бахвалов Н.С. Численные методы: учебное пособие для физ.-мат. специальностей вузов [Текст] / Бахвалов Н.С., Жидков Н.П., Кобельков Н.М. – М.: Бином. Лаб. знаний, 2007. – 636 с.

4 SymPy: a Python library for symbolic mathematics. [Электронный ресурс] – 2018. – URL: <https://docs.sympy.org/latest/index.html> (дата обращения: 22.12.2018).

5 Matplotlib: a plotting library for the Python programming language [Электронный ресурс] – 2018. – URL: <https://matplotlib.org/> (дата обращения: 22.12.2018).

ПРИЛОЖЕНИЕ А

КОД ИНТЕРФЕЙСА ОКОННОГО ПРИЛОЖЕНИЯ

```
import sys

from PyQt5.QtWidgets import QApplication, QGridLayout,
QGroupBox, QLabel, QLineEdit, QMessageBox, QPushButton, \
    QSizePolicy, QVBoxLayout, QWidget
from matplotlib.backends.backend_qt5agg import
FigureCanvasQTAgg as FigureCanvas
from matplotlib.figure import Figure

import solver

class App(QWidget):

    def __init__(self):
        super().__init__()

        self.solver = solver.Solver()

        self.grid = QGridLayout()
        self.setLayout(self.grid)

        self.resize(1200, 600)

        self.plot = PlotCanvas(width=10, height=4)
        self.grid.addWidget(self.plot, 1, 1, 1, 1)

        param_box = QGroupBox()

        param_vbox = QVBoxLayout()

        param_vbox.addWidget(QLabel('Длина цилиндра l'))
        self.l_field = QLineEdit('20')
        self.l_field.setStyleSheet("color: black;")
        param_vbox.addWidget(self.l_field)

        param_vbox.addWidget(QLabel('Коэффициент диффузии D'))
        self.D_field = QLineEdit('0.6')
        self.D_field.setStyleSheet("color: black;")
        param_vbox.addWidget(self.D_field)

        param_vbox.addWidget(QLabel('Мембранный коэффициент
диффузии H'))
        self.H_field = QLineEdit('4')
        self.H_field.setStyleSheet("color: black;")
```

```

param_vbox.addWidget(self.H_field)

param_vbox.addWidget(QLabel('Момент времени t'))
self.t_field = QLineEdit('0')
self.t_field.setStyleSheet("color: black;")
param_vbox.addWidget(self.t_field)

param_vbox.addWidget(QLabel('Точность поиска собственных
значений'))
self.eps_field = QLineEdit('1e-9')
self.eps_field.setStyleSheet("color: black;")
param_vbox.addWidget(self.eps_field)

param_vbox.addWidget(QLabel('Количество членов ряда в
суммировании'))
self.sum_count_field = QLineEdit('100')
self.sum_count_field.setStyleSheet("color: black;")
param_vbox.addWidget(self.sum_count_field)

calc_button = QPushButton('Рассчитать')
calc_button.pressed.connect(self.calc)
param_vbox.addWidget(calc_button)

param_vbox.addStretch(1)
param_box.setLayout(param_vbox)

self.grid.addWidget(param_box, 1, 2, 1, 1)

self.calc()

def show_error(self, text):
    msg = QMessageBox()
    msg.setIcon(QMessageBox.Information)

    msg.setText(text)
    msg.setWindowTitle("Ошибка расчетов")
    msg.setStandardButtons(QMessageBox.Ok)

    msg.exec_()

def calc(self):
    try:
        l_str = self.l_field.text()
        l_val = float(l_str)
    except Exception as e:
        self.show_error(f'Некорректный формат числа: {l_str}')
        return

    try:

```

```

        D_str = self.D_field.text()
        D_val = float(D_str)
    except Exception as e:
        self.show_error(f'Некорректный формат числа: {D_str}')
        return

    try:
        H_str = self.H_field.text()
        H_val = float(H_str)
    except Exception as e:
        self.show_error(f'Некорректный формат числа: {H_str}')
        return

    try:
        t_str = self.t_field.text()
        t_val = float(t_str)
    except Exception as e:
        self.show_error(f'Некорректный формат числа: {t_str}')
        return

    try:
        eps_str = self.eps_field.text()
        eps_val = float(eps_str)
    except Exception as e:
        self.show_error(f'Некорректный формат числа: {eps_str}')
        return

    try:
        sum_count_str = self.sum_count_field.text()
        sum_count_val = int(sum_count_str)
    except Exception as e:
        self.show_error(f'Некорректный формат числа:
{sum_count_str}')
        return

    if not (
        1e-3 <= l_val <= 1e+3 and
        1e-3 <= D_val <= 1e+3 and
        1e-3 <= H_val <= 1e+3 and
        0 <= t_val <= 1e+3 and
        1e-20 <= eps_val <= 1e+3 and
        1 <= sum_count_val <= 1e+3
    ):
        self.show_error(f'Некорректные входные данные')
        return

    self.solver.set_params(l_val, D_val, H_val, t_val,
eps_val, sum_count_val)

```

```

    plot_x, plot_y = self.solver.calc()

    self.plot.plot(plot_x, plot_y)

class PlotCanvas(FigureCanvas):
    def __init__(self, parent=None, width=4, height=4, dpi=100):
        fig = Figure(figsize=(width, height), dpi=dpi)
        self.axes = fig.add_subplot(111)

        FigureCanvas.__init__(self, fig)
        self.setParent(parent)

        FigureCanvas.setSizePolicy(self, QSizePolicy.Expanding,
        QSizePolicy.Expanding)
        FigureCanvas.updateGeometry(self)

    def plot(self, plot_x, plot_y):
        self.axes.clear()

        self.axes.plot(plot_x, plot_y)
        self.axes.set_xlabel('$x, \mathrm{m}$')
        self.axes.set_ylabel('$u$')
        self.axes.set_title('Поле концентрации вещества в
цилиндре, $\frac{\mathrm{kg}}{\mathrm{m}^3}$')

        self.axes.set_ylim(0, None)

        self.axes.grid(True)
        self.draw()

if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = App()
    ex.show()
    sys.exit(app.exec_())

```

ПРИЛОЖЕНИЕ Б

КОД ВЫЧИСЛИТЕЛЬНОГО МОДУЛЯ

```
import datetime
import pickle

import numpy as np

class Solver:
    def __init__(self):
        self.l = None
        self.D = None
        self.H = None
        self.t = None

        self.eps = 1e-9
        self.sum_count = 100

        self.alphas = np.zeros(0)

    def find_root(self, index):
        C = self.H * self.l / 2

        l = np.pi * index + self.eps
        r = np.pi * index + np.pi / 2 - self.eps

        while True:
            m = (l + r) / 2
            val = np.tan(m) - C / m

            if abs(val) < self.eps or abs(r - l) < self.eps:
                return m

            if val > 0:
                r = m
            else:
                l = m

    def set_params(self, l, D, H, t, eps, sum_count):
        self.l = l
        self.D = D
        self.H = H
        self.t = t
        self.eps = eps
        self.sum_count = sum_count
```

```

        self.alphas = np.array([self.find_root(i) for i in
range(self.sum_count)])

    def calc(self):
        plot_x = np.linspace(0, self.l, 1000)

        start_dt = datetime.datetime.utcnow()

        plot_y = np.array([self.calc_single(x) for x in plot_x])

        end_dt = datetime.datetime.utcnow()

        print((end_dt - start_dt) / 1000)

        return plot_x, plot_y

# noinspection PyTypeChecker
def calc_single(self, x: float) -> float:
    if self.l is None:
        raise ValueError('Not initialized')

    # @formatter:off
    return np.sum(
        (
            -4 * np.pi ** 2 * (
                self.H * self.l * np.sin(2 * self.alphas * x /
self.l)
                + 2 * self.alphas * np.cos(2 * self.alphas * x /
self.l)
            ) * (
                -self.H * self.l * np.cos(2 * self.alphas)
                + self.H * self.l
                + 2 * self.alphas * np.sin(2 * self.alphas)
            ) * np.exp(
                -4 * self.D * self.alphas ** 2 * self.t / self.l **
2
            ) / (
                5 * (
                    self.alphas ** 2
                    - np.pi ** 2
                ) * (
                    4 * self.H ** 2 * self.alphas * self.l ** 2
                    - self.H ** 2 * self.l ** 2 * np.sin(4 *
self.alphas)
                    - 4 * self.H * self.alphas * self.l * (np.cos(4 *
self.alphas) - 1)
                    + 4 * self.alphas ** 2 * (
                        4 * self.alphas
                        + np.sin(4 * self.alphas)

```

```
)  
  )  
    )  
      )  
        )  
# @formatter:on
```

ПРИЛОЖЕНИЕ В

КОД МОДУЛЯ СИМВОЛЬНОГО ИСЧИСЛЕНИЯ

In [1]: `%pylab inline`

Populating the interactive namespace from numpy and matplotlib

```
In [2]: import pandas
import sympy as Ω
import matplotlib.pyplot as plt
import math
import pylab as p
import numpy as np
import decimal as d
from scipy.misc import derivative
import pickle
from tqdm import tqdm_notebook as tqdm
```

In [3]: `Ω.init_printing()`

```
In [4]: l_val = Ω.Integer(20)
D_val = Ω.Integer(6) / 10
H_val = Ω.Integer(4)
T_val = Ω.Integer(100)
```

```
In [5]: l, D, H, T, alpha_n, N = Ω.symbols('l D H T alpha_n N')
pi = Ω.pi
```

```
In [6]: x, t = Ω.symbols('x t')
```

```
In [7]: lambda_n = alpha_n * 2 * Ω.sqrt(D) / l
```

```
In [8]: with open('roots.pickle', 'rb') as file:
roots = pickle.load(file)
```

```
In [9]: alphas = roots
for alpha_val in alphas:
    y = (Ω.tan(alpha_val) - H * l / 2 / alpha_val).evalf(subs={
        H: H_val,
        l: l_val
    })
    assert y < 1e-9
print('roots ok')
```

roots ok

```
In [10]: phi = Ω.sin(pi * x / l) ** 2 * 4 / 10
phi
```

Out[10]:
$$\frac{2 \sin^2\left(\frac{\pi x}{l}\right)}{5}$$

```
In [11]: X_n = 2 * alpha_n / H / l * Omega.cos(2 * alpha_n / l * x) \
          + Omega.sin(2 * alpha_n / l * x)
          X_n
```

Out[11]:
$$\sin\left(\frac{2\alpha_n x}{l}\right) + \frac{2\alpha_n \cos\left(\frac{2\alpha_n x}{l}\right)}{Hl}$$

```
In [12]: norm = Omega.integrate(X_n * X_n, (x, 0, l))
          norm = norm.as_expr_set_pairs()[0][0]
          norm = norm.simplify()
          norm
```

Out[12]:
$$\frac{l}{2} - \frac{l \sin(4\alpha_n)}{8\alpha_n} + \frac{\sin^2(2\alpha_n)}{H} + \frac{2\alpha_n^2}{H^2 l} + \frac{\alpha_n \sin(4\alpha_n)}{2H^2 l}$$

```
In [13]: A_n = Omega.integrate(X_n * phi, (x, 0, l))
          A_n = A_n.as_expr_set_pairs()[-1][0] / norm
          A_n = A_n.simplify()
          A_n
```

Out[13]:
$$-\frac{4\pi^2 Hl (-Hl \cos(2\alpha_n) + Hl + 2\alpha_n \sin(2\alpha_n))}{5(\alpha_n^2 - \pi^2)(4H^2\alpha_n l^2 - H^2 l^2 \sin(4\alpha_n) + 8H\alpha_n l \sin^2(2\alpha_n) + 4\alpha_n^2(4\alpha_n + \sin(4\alpha_n)))}$$

```
In [14]: T_n = A_n * Omega.exp(-lambda_n**2 * t)
          T_n = T_n.simplify()
          T_n
```

Out[14]:
$$-\frac{4\pi^2 Hl (-Hl \cos(2\alpha_n) + Hl + 2\alpha_n \sin(2\alpha_n)) e^{-\frac{4D\alpha_n^2 t}{l^2}}}{5(\alpha_n^2 - \pi^2)(4H^2\alpha_n l^2 - H^2 l^2 \sin(4\alpha_n) + 8H\alpha_n l \sin^2(2\alpha_n) + 4\alpha_n^2(4\alpha_n + \sin(4\alpha_n)))}$$

```
In [15]: u_elem = (T_n * X_n).subs({
          H: H_val,
          l: l_val,
          D: D_val,
          }).simplify()
```

```
In [16]: u_elem
```

Out[16]:
$$-\frac{4\pi^2 \left(\alpha_n \cos\left(\frac{\alpha_n x}{10}\right) + 40 \sin\left(\frac{\alpha_n x}{10}\right)\right) (\alpha_n \sin(2\alpha_n) - 40 \cos(2\alpha_n) + 40) e^{-\frac{3\alpha_n^2 t}{500}}}{5(\alpha_n^2 - \pi^2)(\alpha_n^2(4\alpha_n + \sin(4\alpha_n)) + 160\alpha_n \sin^2(2\alpha_n) + 6400\alpha_n - 1600 \sin(4\alpha_n))}$$

```
In [17]: u = sum([
    u_elem.subs({
        alpha_n: alpha_val,
    })
    for alpha_val in tqdm(alphas)
])
```

100% 101/101 [00:01<00:00, 99.99it/s]

```
In [18]: (T_n * X_n).subs({
    H: H_val,
    l: l_val,
    D: D_val,
}).simplify()
```

```
Out[18]:
```

$$-\frac{4\pi^2 \left(\alpha_n \cos\left(\frac{\alpha_n x}{10}\right) + 40 \sin\left(\frac{\alpha_n x}{10}\right) \right) (\alpha_n \sin(2\alpha_n) - 40 \cos(2\alpha_n) + 40) e^{-\frac{3\alpha_n^2 t}{500}}}{5 (\alpha_n^2 - \pi^2) (\alpha_n^2 (4\alpha_n + \sin(4\alpha_n)) + 160\alpha_n \sin^2(2\alpha_n) + 6400\alpha_n - 1600 \sin(4\alpha_n))}$$

```
In [19]: f_u = Ω.lambdify((x, t), u)
f_u(0, 0)
```

```
Out[19]: 5.266645799609216e-07
```

```
In [23]: x = linspace(0, float(l_val), 1000)
y = [f_u(i, 0) for i in x]

plot(x, y)
title('Концентрация в момент t=0')
grid(True)
xlabel('x')
ylabel('u')
None
```

