Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

Лабораторная работа № 3
Дисциплина: «Моделирование»

Тема: «Программно-алгоритмическая реализация моделей на основе ОДУ второго порядка с краевыми условиями II и III рода»

Студент Овчинникова А. П.
Группа ИУ7-65Б
Оценка (баллы)
Преподаватель Градов В.М.

Москва, 2020 г.

Цель работы

Целью данной работы является получение навыков разработки алгоритмов решения краевой задачи при реализации моделей, построенных на ОДУ второго порядка.

Исходные данные

1. Задана математическая модель.

Уравнение для функции $T(x)$:

$$\frac{d}{dx}\left(k(x)\frac{dT}{dx}\right) - \frac{2}{R}\alpha(x)T + \frac{2T_0}{R}\alpha(x) = 0 \tag{1}$$

Краевые условия

$$\begin{cases} x = 0, -k(0)\frac{dT}{dx} = F_0 \\ x = l, -k(l)\frac{dT}{dx} = \alpha_N\left(T(l) - T_0\right) \end{cases} \tag{2}$$

2. Функции $k(x), \alpha(x)$ заданы своими константами:

$$k(x) = \frac{a}{x-b} \tag{3}$$

$$\alpha(x) = \frac{c}{x-d} \tag{4}$$

Константы a, b следует найти из условий $k(0) = k_0, k(l) = k_N$, а константы c, d из условий $\alpha(0) = a_0, \alpha(l) = \alpha_N$. Величины $k_0, k_N, \alpha_0, \alpha_N$ задает пользователь, их надо вынести в интерфейс.

3. Разностная схема с разностным краевым условием при $x = 0$.

$$A_n y_{n+1} - B_n y_n + C_n y_{n-1} = -D_n, 1 \le n \le N-1 \tag{5}$$

где

$A_n = \frac{\chi_{n+\frac{1}{2}}}{h}$

$C_n = \frac{\chi_{n-\frac{1}{2}}}{h}$

$B_n = A_n + C_n + p_n h$

$D_n = f_n h$

Система (5) совместно с краевыми условиями решается методом прогонки.

Для величин $\chi_{N\pm1/2}$ можно получить различные приближенные выражения, численно вычисляя интеграл методом трапеций или методом средних. Будем использовать метод средних:

$$\chi_{N\pm1/2} = \frac{k_n + k_{n\pm1}}{2}. \tag{6}$$

Разностный аналог краевого условия при $x = 0$:

$$y_0 = \frac{\chi_{\frac{1}{2}} - \frac{h^2}{8}p_{\frac{1}{2}}}{\chi_{\frac{1}{2}} + \frac{h^2}{8}p_{\frac{1}{2}} + \frac{h^2}{4}p_0} + \frac{hF_0 + \frac{h^2}{4}\left(f_{\frac{1}{2}} + f_0\right)}{\chi_{\frac{1}{2}} + \frac{h^2}{8}p_{\frac{1}{2}} + \frac{h^2}{4}p_0} \tag{7}$$

Примем простую аппроксимацию:
$p_{\frac{1}{2}} = \frac{p_0+p_1}{2}$
$f_{\frac{1}{2}} = \frac{f_0+f_1}{2}$

При получении разностного аналога краевого условия при $x = l$ интегро-интерполяционным методом необходимо учесть, что

$$F_N = \alpha_N \left(y_N - T_0\right) \tag{8}$$

$$F_{N-\frac{1}{2}} = \chi_{N-\frac{1}{2}} \frac{y_{N-1} - y_N}{h}. \tag{9}$$

4. Значения параметров для отладки (все размерности согласованы) приведены в таблице 1.

Таблица 1: Значения параматров.

$k_0 =$	0.4 Вт/см К
$k_N =$	0.1 Вт/см К
$\alpha_0 =$	0.05 Вт/см^2 К
$\alpha_N =$	0.01 Вт/см^2 К
$l =$	10 см
$T_0 =$	300 К
$R =$	0.5 см
$F_0 =$	50 Вт/см^2

Физическое содержание задачи

Сформулированная математическая модель описывает температурное поле $T(x)$ вдоль цилиндрического стержня радиуса R и длины l, причем $R << l$ и температуру можно принять постоянной по радиусу цилиндра. Ось X направлена вдоль оси цилиндра и начало координат совпадает с левым торцем стержня. Слева при $x = 0$ цилиндр нагружается тепловым потоком F_0. тержень обдувается воздухом, температура которого равна T_0. В результате происходит съем тепла с цилиндрической поверхности и поверхности правого торца при $x = l$. Функции $k(x), \alpha(x)$ являются, соответственно, коэффициентами теплопроводности материала стержня и теплоотдачи при обдуве.

Задание

1. Представить разностный аналог краевого условия при $x = l$ и его краткий вывод интегро-интерполяционным методом.

2. График зависимости температуры $T(x)$ от координаты X при заданных выше параметрах.

3. График зависимости $T(x)$ при $F_0 = -10$ Вт/см^2. Справка. При отрицательном тепловом потоке слева идет съем тепла, поэтому производная $T'(x)$ должна быть положительной.

4. График зависимости $T(x)$ при увеличенных значениях $\alpha(x)$ (например, в 3 раза). Сравнить с п.2. Справка. При увеличении теплосъема и неизменном потоке F_0 уровень температур $T(x)$ должен снижаться, а градиент увеличиваться.

5. График зависимости $T(x)$ при $F_0 = 0$. Справка. В данных условиях тепловое нагружение отсутствует, причин для нагрева нет, температура стержня должна быть равна температуре окружающей среды T_0 (разумеется с некоторой погрешностью, определяемой приближенным характером вычислений).

Предварительные вычисления

1. Константы a, b, c, d:

$k_0 = -\frac{a}{b} \Rightarrow a = -k_o b$

$k_N = -\frac{a}{l-b} \Rightarrow b = \frac{k_N l}{k_N - k_0}$

$\alpha_0 = -\frac{c}{d} \Rightarrow c = -\alpha_0 d$

$\alpha_N = \frac{c}{l-d} \Rightarrow d = \frac{\alpha_N l}{\alpha_N - \alpha_0}$.

2. Разностный аналог краевого условия при $x = l$:

Пусть $p(x) = \frac{2}{R}\alpha(x)$, $f(x) = \frac{2T_0}{R}\alpha(x)$. Тогда имеем квазилинейное уравнение:

$$\frac{d}{dx}\left(k(x)\frac{dT}{dx}\right) - p(x)T + f(x) = 0 \tag{10}$$

Краевое условие справа III рода:

$x = l, -k(l)\frac{dT}{dx} = \alpha_N(T(l) - T_0)$

Получим разностный аналог краевого условия при $x = l$.

Примем $F = -k(x)\frac{dT}{dx}$

Проинтегрируем уравнение 9 на отрезке $[x_{N-1/2}; x_N]$:

$$-\int\limits_{x_{N-1/2}}^{x_N} \frac{dF}{dx}dx - \int\limits_{x_{N-1/2}}^{x_N} p(x)T dx + \int\limits_{x_{N-1/2}}^{x_N} f(x)dx = 0 \tag{11}$$

Второй и третий интегралы вычислим методом трапеций:

$$-(F_N - F_{N-1/2}) - \frac{h}{4}(p_N y_N + p_{N-1/2}y_{N-1/2}) + \frac{h}{4}(f_N + f_{N-1/2}) = 0 \tag{12}$$

Примем простую аппроксимацию $y_{N-1/2} = \frac{y_{N-1}+y_N}{2}$ и применим (7), (8) к (11):

$$\begin{aligned} -\left[\alpha_N(y_N - T_0) - \chi_{N-1/2}\frac{y_{N-1} - y_N}{h}\right] - \\ -\frac{h}{4}\left(p_N y_N + p_{N-1/2}\frac{y_N + t_{N-1}}{2}\right) + \\ +\frac{h}{4}\left(f_N + f_{N-1/2}\right) = 0 \end{aligned} \tag{13}$$

Из (12) получаем:

$$
\begin{aligned}
y_{N-1}\left(\frac{\chi_{N-1/2}}{h} - \frac{hp_{N-1/2}}{8}\right) + \\
+y_N\left(-\alpha_N - \frac{\chi_{N-1/2}}{h} - \frac{hp_N}{4} - \frac{hp_{N-1/2}}{8}\right) = \\
= -\alpha_N T_0 - \frac{h}{4}\left(f_N + f_{N-1/2}\right)
\end{aligned}
\tag{14}
$$

Можно принять простую аппроксимацию:
$p_{N-1/2} = \frac{p_N+p_{N-1}}{2}$
$f_{N-1/2} = \frac{f_N+f_{N-1}}{2}$.
Тогда:

$$
\begin{aligned}
y_{N-1}\left(\frac{\chi_{N-1/2}}{h} - \frac{h}{8}\cdot\frac{p_N+p_{N-1}}{2}\right) + \\
y_N\left(-\alpha_N - \frac{\chi_{N-1/2}}{h} - \frac{hp_N}{4} - \frac{h}{8}\cdot\frac{p_N+p_{N-1}}{2}\right) = \\
= -\alpha_N T_0 - \frac{h}{4}\left(f_N + \frac{f_N+f_{N-1}}{2}\right)
\end{aligned}
\tag{15}
$$

Код программы

Код программы представлен в листингах 1-2.

Листинг 1: Класс MyApp.

```
import sys

from PyQt5 import QtWidgets
from PyQt5.QtWidgets import QMessageBox

from Ui_MainWindow import Ui_MainWindow

from Modeller import Modeller

class MyApp(QtWidgets.QMainWindow):
    def __init__(self):
        super(MyApp, self).__init__()
        self.ui = Ui_MainWindow()
```

```
        self.ui.setupUi(self)

        self.ui.set_def_button.clicked.connect(self.set_defaults)
        self.ui.run_button.clicked.connect(self.run)

        self.defaults = {
            "k0" : 0.4,
            "kN" : 0.1,
            "a0" : 0.05,
            "aN" : 0.01,
            "I"  : 10,
            "T0" : 300,
            "R"  : 0.5,
            "F0" : 50,
            "h"  : 0.1
        }

        self.data = {
            "k0" : None,
            "kN" : None,
            "a0" : None,
            "aN" : None,
            "I"  : None,
            "T0" : None,
            "R"  : None,
            "F0" : None,
            "h"  : None
        }

        self.set_defaults()

    def set_defaults(self):
        self.ui.lineEdit_k0.setText(str(self.defaults.get("k0")))
        self.ui.lineEdit_kN.setText(str(self.defaults.get("kN")))
        self.ui.lineEdit_alpha0.setText(str(self.defaults.get("a0")))
        self.ui.lineEdit_alphaN.setText(str(self.defaults.get("aN")))
```

```python
        self.ui.lineEdit_l.setText(str(self.defaults.get("l")))
        self.ui.lineEdit_T0.setText(str(self.defaults.get("T0")))
        self.ui.lineEdit_R.setText(str(self.defaults.get("R")))
        self.ui.lineEdit_F0.setText(str(self.defaults.get("F0")))
        self.ui.lineEdit_h.setText(str(self.defaults.get("h")))

    def get_data(self):
        try:
            self.data["k0"] = float(self.ui.lineEdit_k0.text())
            self.data["kN"] = float(self.ui.lineEdit_kN.text())
            self.data["a0"] = float(self.ui.lineEdit_alpha0.text())
            self.data["aN"] = float(self.ui.lineEdit_alphaN.text())
            self.data["l"] = float(self.ui.lineEdit_l.text())
            self.data["T0"] = float(self.ui.lineEdit_T0.text())
            self.data["R"] = float(self.ui.lineEdit_R.text())
            self.data["F0"] = float(self.ui.lineEdit_F0.text())
            self.data["h"] = float(self.ui.lineEdit_h.text())
        except ValueError:
            return False
        return True

    def run(self):
        if self.get_data():
            print("Computing...")
            mdlr = Modeller(self.data)
            mdlr.compute()
            print("Finish.")
        else:
            self.msg_box("Error!", "Error! Incorrect input!", QMessageBox.Critical)
```

```

    def msg_box(self, title, message, type):
        msg = QMessageBox(self)
        msg.setIcon(type)
        msg.setWindowTitle(title)
        msg.setText(message)
        msg.addButton('Ok', QMessageBox.AcceptRole)
        msg.exec()


def main():
    app = QtWidgets.QApplication(sys.argv)
    window = MyApp()
    window.show()
    app.exec_()


if __name__ == '__main__':
    main()
```

Листинг 2: Класс Modeller.

```
import matplotlib.pyplot as plt

class Modeller():
    def __init__(self, data):
        self.data = data

        self.k0 = self.data.get("k0")
        self.kN = self.data.get("kN")
        self.a0 = self.data.get("a0")
        self.aN = self.data.get("aN")
        self.l = self.data.get("l")
        self.T0 = self.data.get("T0")
        self.R = self.data.get("R")
        self.F0 = self.data.get("F0")
        self.h = self.data.get("h")

        self.start_l = 0

        self.b = (self.kN * self.l) / (self.kN - self.k0)
        self.a = - self.k0 * self.b
```

```
        self.d = (self.aN * self.l) / (self.aN - self.a0)
        self.c = - self.a0 * self.d

        self.M0, self.K0, self.P0 = self.left_boundary_condition()
        self.MN, self.KN, self.PN = self.right_boundary_condition()

    # приближенно вычислим интеграл методом средних
    def chi_plus_half(self, func, x):
        return (func(x) + func(x + self.h)) / 2

    # приближенно вычислим интеграл методом средних
    def chi_minus_half(self, func, x):
        return (func(x) + func(x - self.h)) / 2

    def right_boundary_condition(self):
        chi = self.chi_minus_half(self.k, self.l)

        pn = self.p(self.l)
        fn = self.f(self.l)

        pn12 = (pn + self.p(self.start_l - self.h)) / 2
        fn12 = (fn + self.f(self.start_l - self.h)) / 2

        MN = chi / self.h - (self.h * pn12) / 8
        KN = -self.aN - chi / self.h - (self.h * pn) / 4 - (self.h * pn12) / 8
        PN = - self.aN * self.T0 - self.h / 4 * (fn + fn12)

        return MN, KN, PN

    def left_boundary_condition(self):
        chi = self.chi_minus_half(self.k, self.start_l)
        h_2 = self.h ** 2
        p0 = self.p(self.start_l)
        f0 = self.f(self.start_l)
        p12 = (p0 + self.p(self.start_l + self.h)) / 2
        f12 = (f0 + self.f(self.start_l + self.h)) / 2
        M0 = - chi + h_2 / 8 * p12
        K0 = chi + h_2 / 8 * p12 + h_2 / 4 * p0
```

```
            P0 = self.h * self.F0 + h_2 / 4 * (f12 + f0)

            return M0, K0, P0

    def compute(self):
        epsilon = [0, - self.M0 / self.K0]
        eta = [0, self.P0 / self.K0]
        xes = [0, self.h]

        # вычисляем значения прогоночных коэффицентов
        x = self.h
        n = 1
        while x + self.h < self.l:
            newEps = self.C(x) / (self.B(x) - self.A(x) *
                epsilon[n])
            epsilon.append(newEps)

            newEta = (self.D(x) + self.A(x) * eta[n]) / (self.
                B(x) - self.A(x) * epsilon[n])
            eta.append(newEta)

            x += self.h
            n += 1
            xes.append(x)

        # обратный ход

        T = [0] * (n + 1)
        T[n] = (self.PN - self.MN * eta[n]) / (self.KN + self.
            MN * epsilon[n])

        for i in range(n-1, -1, -1):
            T[i] = epsilon[i + 1] *T[i + 1] + eta[i+1]

        self.plot1(xes, T)

    def plot1(self, x, y):
        print(y)
        plt.plot(x, y)
        plt.xlabel("x, cm")
        plt.ylabel("t, K")
```

```
            plt.grid()
            plt.show()

    def p(self, x):
        return (2 * self.alpha(x)) / self.R

    def f(self, x):
        return (2 * self.T0 * self.alpha(x)) / self.R

    def k(self, x):
        return self.a / (x - self.b)

    def alpha(self, x):
        return self.c / (x - self.d)

    def A(self, n):
        return self.chi_plus_half(self.k, n) / self.h

    def C(self, n):
        return self.chi_minus_half(self.k, n) / self.h

    def B(self, n):
        return self.A(n) + self.C(n) + self.p(n) * self.h

    def D(self, n):
        return self.f(n) * self.h
```

Результаты работы программы

1. Краткий вывод разностного аналога краевого условия при $x = l$ представлен в разделе "Предварительные вычисления".

2. График зависимости температуры $T(x)$ от координаты X при заданных выше параметрах (рисунок 1).

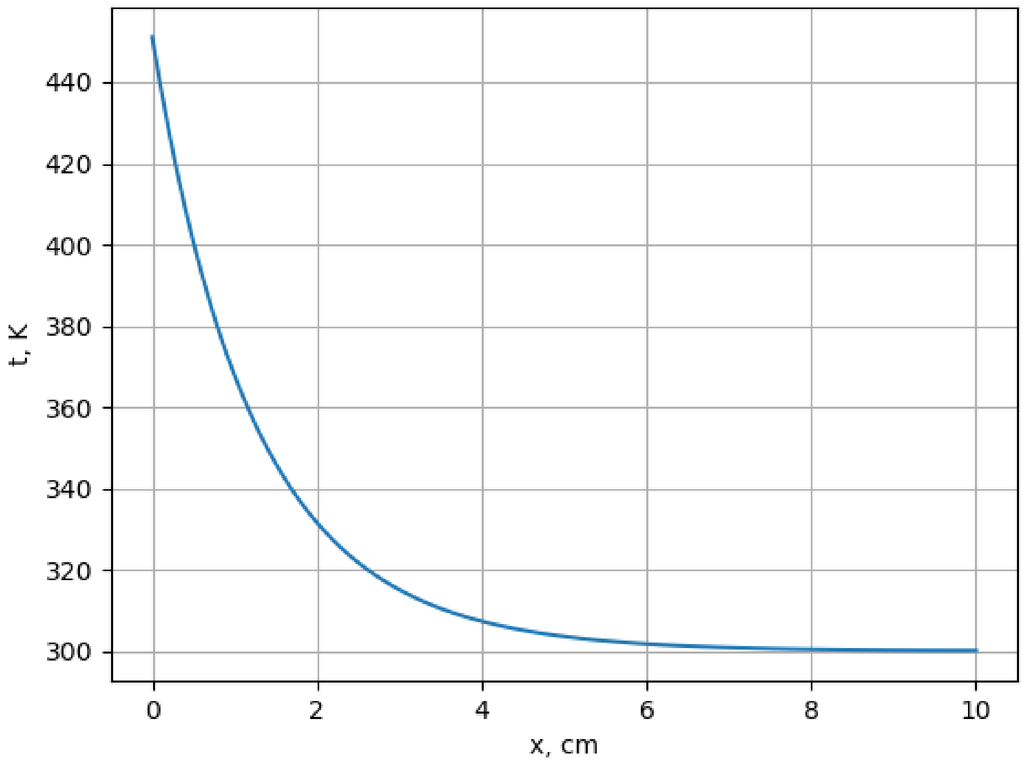

Рис. 1: График зависимости температуры $T(x)$ от координаты X при параметрах по умолчанию.

3. График зависимости $T(x)$ при $F_0 = -10$ Вт/см^2. Справка. При отрицательном тепловом потоке слева идет съем тепла, поэтому производная $T'(x)$ должна быть положительной (рисунок 2).

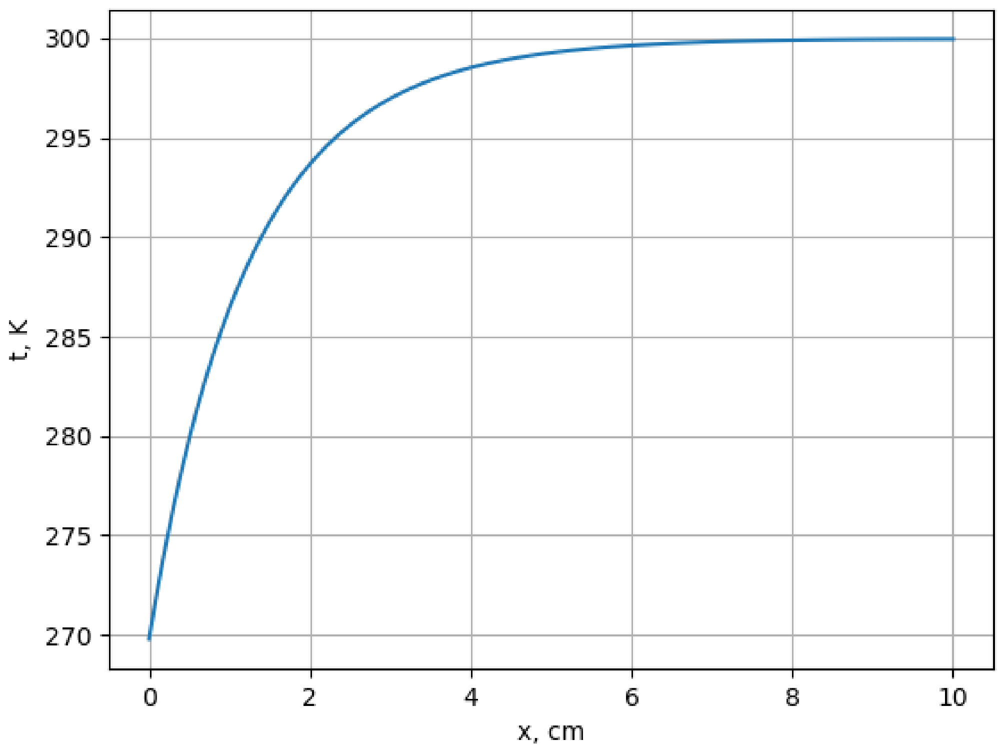

Рис. 2: График зависимости $T(x)$ при $F_0 = -10$ Вт/см^2.

4. График зависимости $T(x)$ при увеличенных значениях $\alpha(x)$ (например, в 3 раза). Сравнить с п.2. Справка. При увеличении теплосъема и неизменном потоке F_0 уровень температур $T(x)$ должен снижаться, а градиент увеличиваться (рисунки 3-4).

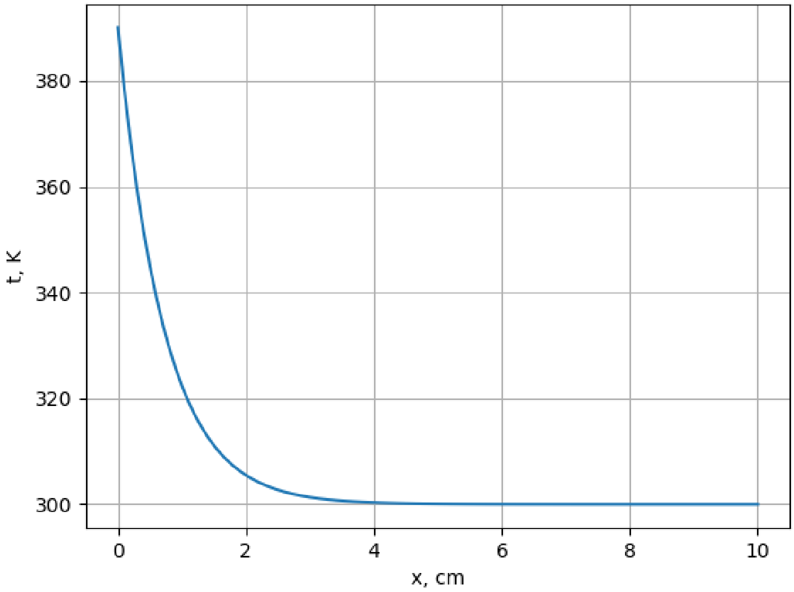

Рис. 3: График зависимости $T(x)$ при увеличенных значениях $\alpha(x)$ (в 3 раза).

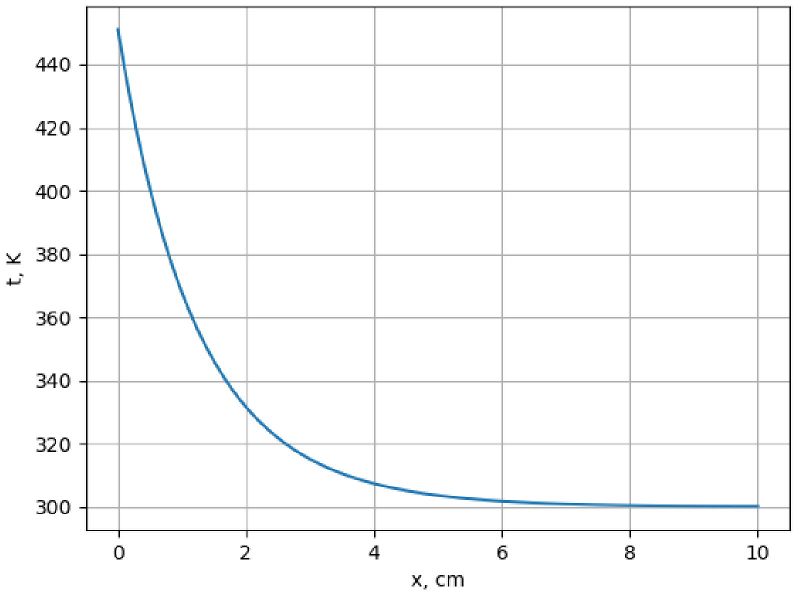

Рис. 4: График зависимости $T(x)$ при стандартных параметрах.

Если увеличить значение $\alpha(x)$ в 100 раз, то уровень температур будет снижаться быстрее (рисунок 5).

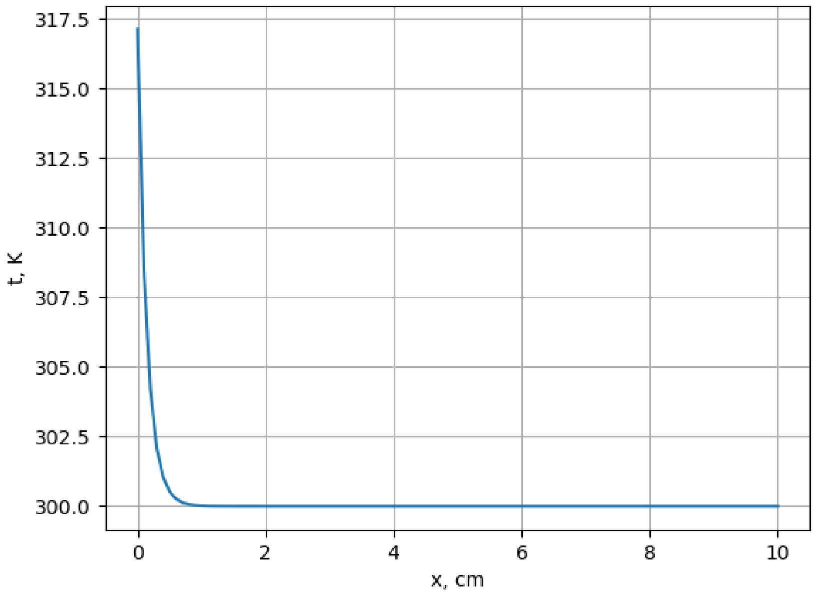

Рис. 5: График зависимости $T(x)$ при увеличенных значениях $\alpha(x)$ (в 100 раз).

5. График зависимости $T(x)$ при $F_0 = 0$. Справка. В данных условиях тепловое нагружение отсутствует, причин для нагрева нет, температура стержня должна быть равна температуре окружающей среды T_0 (разумеется с некоторой погрешностью, определяемой приближенным характером вычислений) (рисунки 6-7).

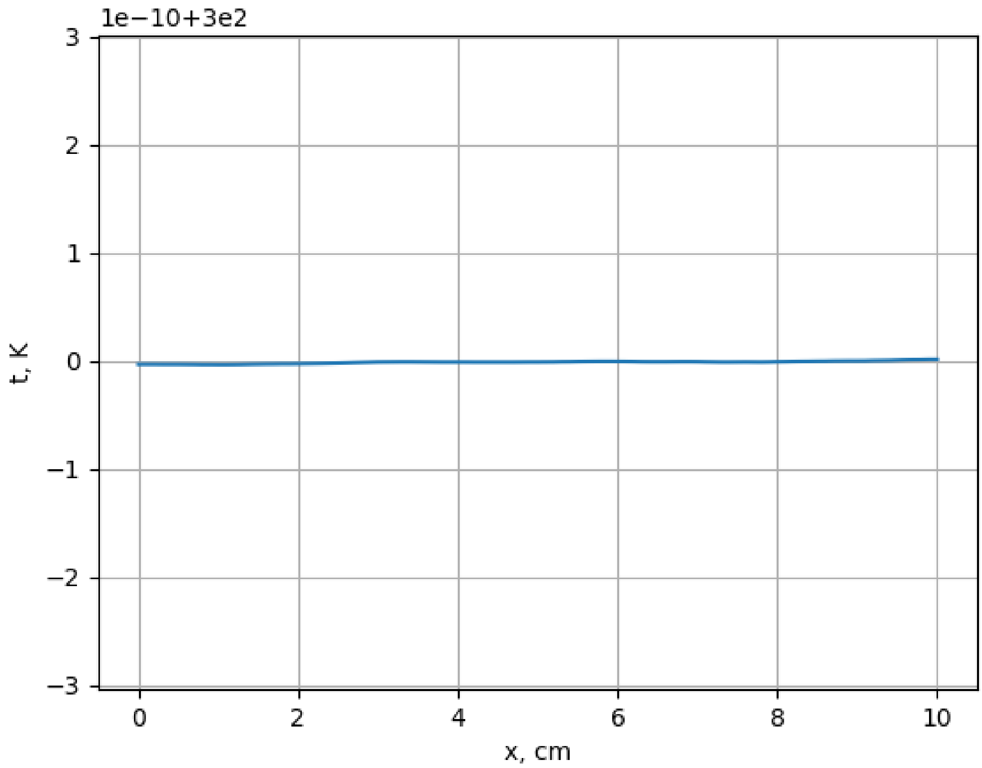

Рис. 6: График зависимости $T(x)$ при $F_0 = 0$ (шаг 0.01).

Если уменьшить шаг, то можно увидеть погрешность (рисунок 7).

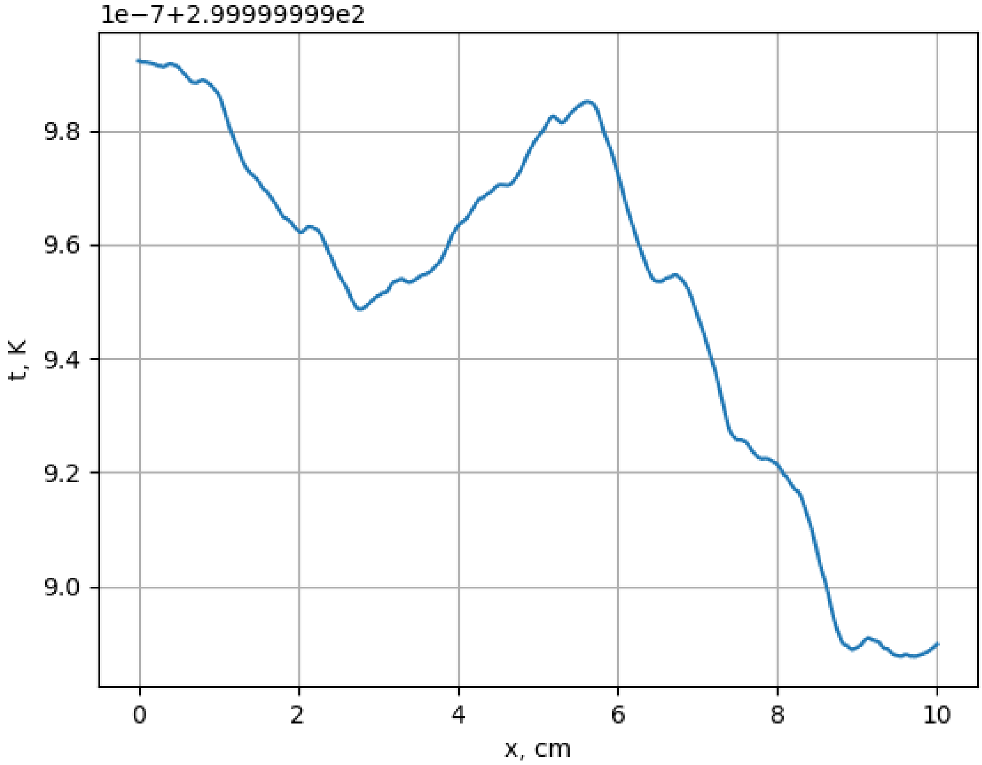

Рис. 7: График зависимости $T(x)$ при $F_0 = 0$ (шаш 0.0001).

Ответы на вопросы

1. Какие способы тестирования программы можно предложить?

Правильность работы программы можно определить по виду графиков. Графики должны соответствовать физическому смыслу задачи.

Например, в п. 5 задания к лабораторной работе (рисунки 6-7) программа тестировалась при $F_0 = 0$. Температура стержня равна температуре окружающей среды ($F_0 = T_0$), т.к. в данных условиях тепловое нагружение отсутствует и причин для нагревва нет. График на рисунках 6-7 соответствует физическому смыслу задачи.

Другие графики из задания к лабораторной работе также соответсвуют физическому смыслу задачи (см. справку к графикам).

2. Получите простейший разностный аналог нелинейного краевого условия при $x = l$: $x = l, -k(l)\frac{dT}{dx} = \alpha_N(T(l) - T_0) + \varphi(T)$, где $\varphi(T)$ – заданная функция. Производную аппроксимируйте односторонней разностью.

Примем простейшую аппроксимацию

$$\frac{dT}{dx} = \frac{T(x+\Delta x) - T(x)}{\Delta x} \tag{16}$$

Подставляя (16) в исходное уравнение при $x = l$ получим:

$$-k_l \frac{T_l - T_{l-1}}{\Delta x} = \alpha_N (T_l - T_0) + \varphi(T_l) \tag{17}$$

3. Опишите алгоритм применения метода прогонки, если при $x = 0$ краевое условие линейное (как в настоящей работе), а при $x = l$, как в п.2.

Имеем систему уравнений:

$$\begin{cases} A_n y_{n+1} - B_n y_n + C_n y_{n-1} = -D_n, 1 \leq n \leq N-1 \\ x = 0, -k(0)\frac{dT}{dx} = F_0 \\ x = l, -k(l)\frac{dT}{dx} = \alpha_N \left(T(l) - T_0\right) \end{cases}$$

Т. к. можно определить начальные значения прогоночиных коэффициентов, будем использовать правую прогонку.

Принимая простейшую (первого порядка точности) аппроксимацию краевого условия при $x = 0$, получим его разностный аналог

$$T_0 = T_1 + \frac{F_0 h}{k_0}. \tag{18}$$

Из (19) найдем начальные прогоночные коэффициенты:

$$\begin{aligned} \varepsilon_1 &= 1 \\ \eta_1 &= \frac{F_0 h}{k_0}. \end{aligned} \tag{19}$$

Рекуррентные соотношения для определения прогоночных коэффициентов:

$$\varepsilon_{n+1} = \frac{C_n}{B_n - A_n\varepsilon_n} \tag{20}$$

$$\eta_{n+1} = \frac{D_n + A_n\eta_n}{B_n - A_n\varepsilon_n} \tag{21}$$

Имея (20), по (21) и (22) находим прогоночные коэффициенты. После этого делаем обратный ход.

Чтобы в обратном ходе найти все значения T_n, надо знать T_N. Найдем T_N.

Основная прогоночная формула имеет вид:

$$T_{n-1} = \varepsilon_n T_n + \eta_n. \tag{22}$$

Из (23) и (18) находим:

$$-k_N\frac{T_N - T_{N-1}}{h} = \alpha_N(T_N - T_0) + \varphi(T_n) \tag{23}$$

$$-k_N\frac{T_n - (\varepsilon_N T_N + \eta_N)}{h} = \alpha_N(T_N - T_0) + \varphi(T_n) \tag{24}$$

$$T_N\left(-k_N + k_N\varepsilon_N - h\alpha_N\right) = -k_N\eta_N - h\alpha_N T_0 + \varphi(T_N)h \tag{25}$$

Решение уравнения (26) можно найти методом дихотомии.

Таким образом, метод прогонки содержит два этапа:

- Прямой ход. По формулам (20) определяем начальные значения прогоночных коэффициентов. По формулам (21) и (22) вычисляем массивы прогоночных коэффициентов.

- Обратный ход. По формуле (26) определяем T_N значение неизвестной функции в последней точке. Далее по формуле (23) находим все значения T_n.

4. Опишите алгоритм определения единственного значения сеточной функции y_p в одной заданной точке p. Использовать встречную прогонку, т.е. комбинацию правой и левой прогонок (лекция №8). Краевые условия линейные.

В правой прогонке начальные прогоночные коэффициенты находятся по формуле (19), рекурентные соотношения для определения остальных прогоночных коэффициентов – (21) и (22). Основная прогоночная формула – (23).

В левой прогонке рекурентные соотношения для определения прогоночных коэффициентов (обозначим их α и β):

$$\alpha_{n-1} = \frac{C_n}{B_n - A_n\alpha_n} \tag{26}$$

$$\beta_{n-1} = \frac{A_n\beta_n + D_n}{B_n - A_n\alpha_n}. \tag{27}$$

Основная прогоночная формула для левой прогонки:

$$T_n = \alpha_{n-1}T_{n-1} + \beta_{n-1}. \tag{28}$$

Объединив левую и правую прогонки, получим:

$$\begin{cases} T_p = \alpha_{p-1}T_{p-1} + \beta_{p-1} \\ T_{p-1} = \varepsilon_p T_p + \eta_p \end{cases} \tag{29}$$

Подставив второе уравнение в первое, получим:

$$T_p = \frac{\alpha_{p-1}\eta_p + \beta_{p-1}}{1 - \alpha_{p-1}\varepsilon_p} \tag{30}$$

Вывод

Таким образом, в ходе данной работы были получены навыки разработки алгоритмов решения краевой задачи при реализации моделей, построенных на ОДУ второго порядка.