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

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Дисциплина: «Моделирование»

Лабораторная работа №4

Тема работы: «Программно- алгоритмическая реализация моделей на основе дифференциальных уравнений в частных производных с краевыми условиями II и III рода.»

Студент: Левушкин И. К.
Группа: ИУ7-62Б
Преподаватель: Градов В. М.

Москва, 2020 г.

Цель работы

Получение навыков разработки алгоритмов решения смешанной краевой задачи при реализации моделей, построенных на квазилинейном уравнении параболического типа.

Исходные данные

1. Задана математическая модель.

   Уравнение для функции $T(x,t)$

$$c(T)\frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k(T)\frac{\partial T}{\partial x}\right) - \frac{2}{R}\alpha(x)T + \frac{2T_0}{R}\alpha(x) \qquad (1)$$

   Краевые условия

$$\begin{cases} t = 0, T(x,0) = T_0, x = 0, -k(T(0))\frac{\partial T}{\partial x} = F_0, \\ x = l, -k(T(l))\frac{\partial T}{\partial x} = \alpha_N(T(l) - T_0) \end{cases}$$

   В обозначениях уравнения (14.1) лекции №14

$$p(x) = \frac{2}{R}\alpha(x), f(u) \equiv f(x) = \frac{2T_0}{R}\alpha(x).$$

2. Разностная схема с разностным краевым условием при $x = 0$ получена в Лекции №14 (14.6),(14.7) и может быть использована в данной работе. Самостоятельно надо получить интегро-интерполяционным методом разностный аналог краевого условия при $x = l$, точно так же, как это сделано при $x = 0$ (формула (14.7)). Для этого надо проинтегрировать на отрезке $[x_{N-\frac{1}{2}}, x_N]$ выписанное выше уравнение (1) и учесть, что поток $\widehat{F_N} = \alpha_N(\widehat{y_N} - T_0)$, а $\widehat{F_{N-\frac{1}{2}}} = \widehat{\chi_{N-\frac{1}{2}}}\frac{\widehat{y_{N-1}} - \widehat{y_N}}{h}$.

3. Значения параметров для отладки (все размерности согласованы)

   $k(T) = a_1(b_1 + c_1T^{m_1})$, Вт/см К,

   $c(T) = a_2 + b_2T^{m_2} - \frac{c_2}{T^2}$, Дж/см^3 К.

   $a_1 = 0.0134, b_1 = 1, c_1 = 4.3510^{-4}, m_1 = 1,$

   $a_2 = 2.049, b_2 = 0.56310^{-3}, c_2 = 0.52810^5, m_2 = 1.$

   $\alpha_0 = 0.05$ Вт/см^2 К,

   $\alpha_N = 0.01$ Вт/см^2 К,

   $l = 10$ см,

$T_0 = 300$ K,

$R = 0.5$ см,

$F(t) = 50$ Вт/см^2 (для отладки принять постоянным)

Физическое содержание задачи (для понимания получаемых результатов при отладке программы).

Постановки задач в данной лабораторной работе и работе №3 во многом совпадают. Отличия заключаются в следующем:

1. Сформулированная в данной работе математическая модель описывает **нестационарное** температурное поле $T(x,t)$, зависящее от координаты x и меняющееся во времени.

2. Свойства материала стержня привязаны к температуре, т.е. теплоемкость и коэффициент теплопроводности $c(T), k(T)$ зависят от T, тогда как в работе №3 $k(x)$ зависит от координаты, а $c = 0$.

3. При $x = 0$ цилиндр нагружается тепловым потоком $F(t)$, в общем случае зависящим от времени, а в работе №3 поток был постоянный.

   Если в настоящей работе задать поток постоянным, т.е. $F(t) = const$, то будет происходить формирование температурного поля от начальной температуры T_0 до некоторого установившегося (стационарного) распределения $T(x,t)$. Это поле в дальнейшем с течением времени меняться не будет и должно совпасть с температурным распределением $T(x)$, получаемым в лаб. работе №3, если все параметры задач совпадают, в частности, вместо $k(T)$ надо использовать $k(x)$ из лаб. работы №3. Это полезный факт для тестирования программы.

   Если после разогрева стержня положить поток $F(t) = 0$, то будет происходить остывание, пока температура не выровняется по всей длине и не станет равной T_0.

   При произвольной зависимости потока $F(t)$ от времени температурное поле будет как-то сложным образом отслеживать поток.

   Замечание. Варьируя параметры задачи, следует обращать внимание на то, что решения, в которых температура превышает примерно 2000K, физического смысла не имеют и практического интереса не представляют.

Результаты работы

Задание 1.

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

Проинтегрируем уравнение (14.3) с учетом (14.2) из Лекции №14 на отрезке $[x_{N-\frac{1}{2}}, x_N]$ и на временном интервале $[t_m, t_{m+1}]$

$$\int_{x_{N-\frac{1}{2}}}^{x_N} dx \int_{t_m}^{t_{m+1}} c(u)\frac{\partial u}{\partial t}dt =$$
$$-\int_{t_m}^{t_{m+1}} dt \int_{x_{N-\frac{1}{2}}}^{x_N} \frac{\partial F}{\partial x}dx - \int_{x_{N-\frac{1}{2}}}^{x_N} dx \int_{t_m}^{t_{m+1}} p(x)udt + \int_{x_{N-\frac{1}{2}}}^{x_N} \int_{t_m}^{t_{m+1}} f(u)dt$$

Приближенно вычисляя итегралы по времени, как и выше получим

$$\int_{x_{N-\frac{1}{2}}}^{x_N} \widehat{c}(\widehat{u} - u)dx = -\int_{t_m}^{t_{m+1}} (F_N - F_{N-\frac{1}{2}})dt - \int_{x_{N-\frac{1}{2}}}^{x_N} p\widehat{u}\tau dx + \int_{x_{N-\frac{1}{2}}}^{x_N} \widehat{f}\tau dx$$

Вычисляем интегралы. Первый интеграл справа, как и ранее, находим методом правых прямоугольников, а остальные - методом трапеций

$$\frac{h}{4}\left[\widehat{c_N}(\widehat{y_N} - y_N) + \widehat{c_{N-\frac{1}{2}}}(\widehat{y_{N-\frac{1}{2}}} - y_{N-\frac{1}{2}})\right] =$$
$$-(\widehat{F_N} - \widehat{F_{N-\frac{1}{2}}})\tau - (p_N\widehat{y_N} + p_{N-\frac{1}{2}}\widehat{y_{N-\frac{1}{2}}})\tau\frac{h}{4} + (\widehat{f_{N-\frac{1}{2}}} + \widehat{f_N})\tau\frac{h}{4}$$

Подставляя в данное уравнение выражение для потока $\widehat{F_{N-\frac{1}{2}}}$, учитывая, что $\widehat{F_N} = F(t_{m+1}) = \alpha(\widehat{y_N} - \beta)$ (где $\alpha = \alpha_N, \beta = T_0$), и заменяя $\widehat{y_{N-\frac{1}{2}}} = \frac{\widehat{y_N} + \widehat{y_{N-1}}}{2}, y_{N-\frac{1}{2}} = \frac{y_N + y_{N-1}}{2}$, найдем разностный аналог краевого условия

$$\left[\alpha\tau + \frac{h}{4}\widehat{c_N} + \frac{h}{8}\widehat{c_{N-\frac{1}{2}}} + \frac{\widehat{\chi_{N-\frac{1}{2}}}\tau}{h} + \frac{p_N\tau h}{4} + \frac{p_{N-\frac{1}{2}}\tau h}{8}\right]\widehat{y_N} +$$
$$\left[\frac{h}{8}\widehat{c_{N-\frac{1}{2}}} - \frac{\widehat{\chi_{N-\frac{1}{2}}}\tau}{h} + \frac{p_{N-\frac{1}{2}}\tau h}{8}\right]\widehat{y_{N-1}} =$$
$$\alpha\beta\tau + (\widehat{f_{N-\frac{1}{2}}} + \widehat{f_N})\tau\frac{h}{4} + \frac{h}{4}\widehat{c_N}y_N + \frac{h}{4}\widehat{c_{N-\frac{1}{2}}}\frac{y_N + y_{N-1}}{2}$$

Задание 2.

График зависимости температуры $T(x, t_m)$ от координаты x при нескольких фиксированных значениях времени t_m (аналогично рисунку в лекции №14) при заданных выше параметрах. Обязательно представить распределение $T(x, t)$ в момент времени, соответствующий установившемуся режиму, когда поле перестает меняться с некоторой точностью (например, $\left[\frac{T(t+\tau)-T(t)}{T(t+\tau)}\right] < 10^{-4}$), т.е. имеет место выход на стационарный режим. На этой стадии левая часть дифференциального уравнения близка к нулю, и на самом деле решается уравнение из лабораторной работы №3 (отличие только в том, что там было линейное уравнение).

Ниже приведены графики зависимости температуры стержня T, зависящее от координаты x в разные моменты времени t_m - от начального момента $t = 0$ (синий график) до момента, когда поле перестает меняться с точностью (серый график): $\left[\frac{T(t+\tau)-T(t)}{T(t+\tau)}\right] < 10^{-4}$. Поток постоянный: $F(x) = const = F_0$. Шаг по x: $h_x = 0.01$ см. Шаг по t: $h_t = 1$ сек.. Все остальные параметры взяты из значений параметров для отладки.

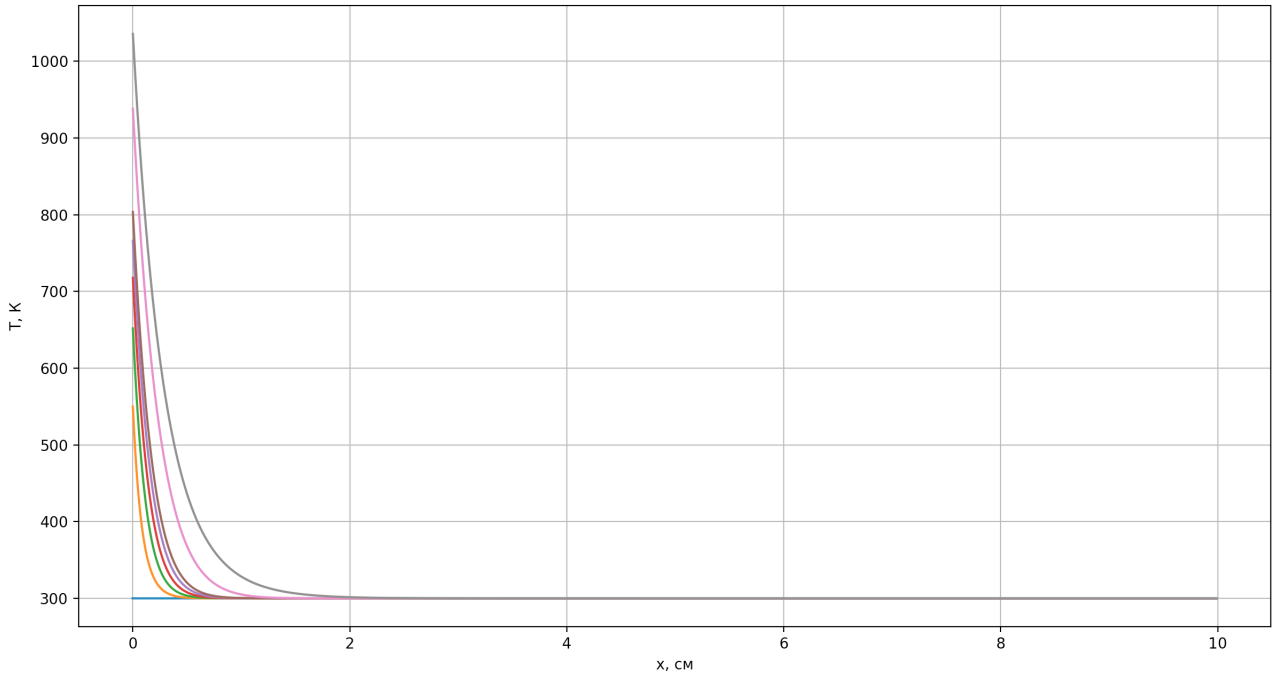

Рис. 1: Графики зависимости температуры $T(x, t_m)$ при $h_x = 0.01$ см; $h_t = 1$ сек.; $F_0 = 50.0$ Вт/см^2.

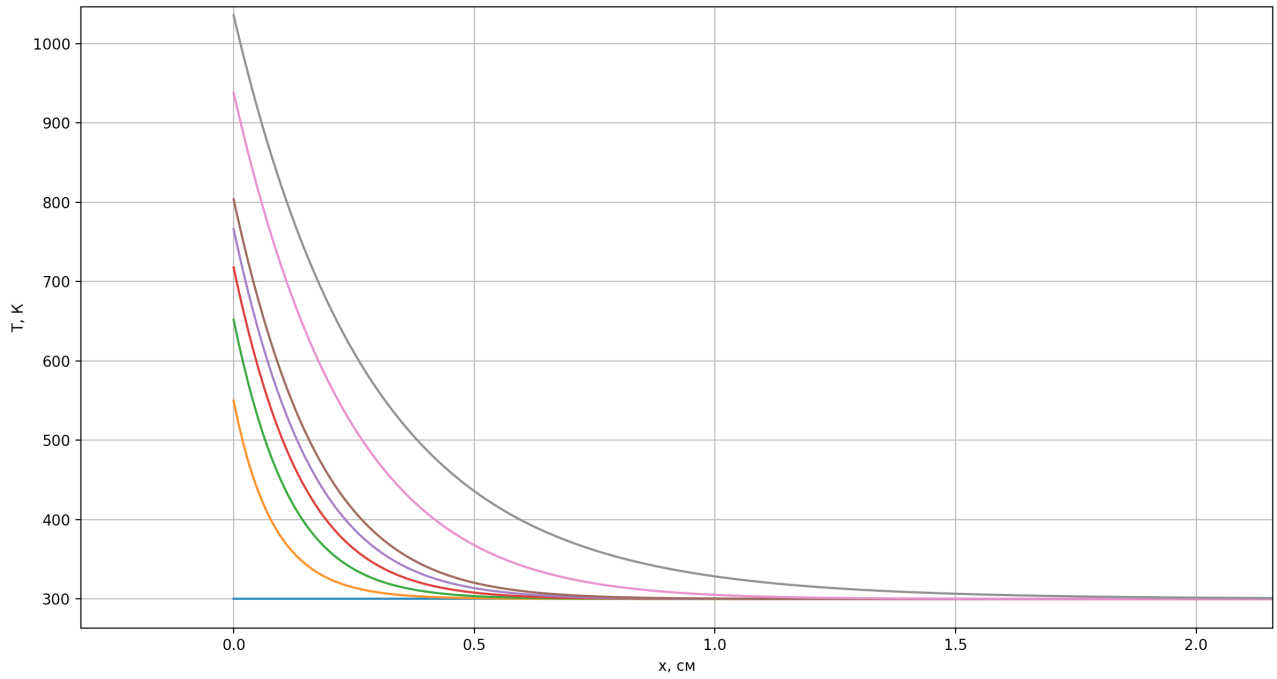

Рис. 2: Графики зависимости температуры $T(x, t_m)$ при $h_x = 0.01$ см; $h_t = 1$ сек.; $F_0 = 50.0$ Вт/см^2.

Где

- Синий цвет соответствует графику при $t = 0$ сек..
- Оранжевый цвет соответствует графику при $t = 1$ сек..
- Зеленый цвет соответствует графику при $t = 2$ сек..
- Красный цвет соответствует графику при $t = 3$ сек..
- Фиолетовый цвет соответствует графику при $t = 4$ сек..
- Коричневый цвет соответствует графику при $t = 5$ сек..
- Розовый цвет соответствует графику при $t = 12$ сек..
- Серый цвет соответствует графику при $t = 71$ сек. (стационарный режим).

Задание 3.

График зависимости $T(x_n, t)$ при нескольких фиксированных значениях координаты x_n. Обязательно представить случай $n = 0$, т.е. $x = x_0 = 0$. Ниже приведены графики зависимости температуры стержня T от координаты t в разные моменты времени x_n - от начала стержня $x = 0.0$ см (синий график) до конца стержня $x = 10.0$ (серый график). Поток постоянный: $F(x) = const = F_0$. Шаг по x: $h_x = 0.01$ см. Шаг по t: $h_t = 1$ сек.. Все остальные параметры взяты из значений параметров для отладки.

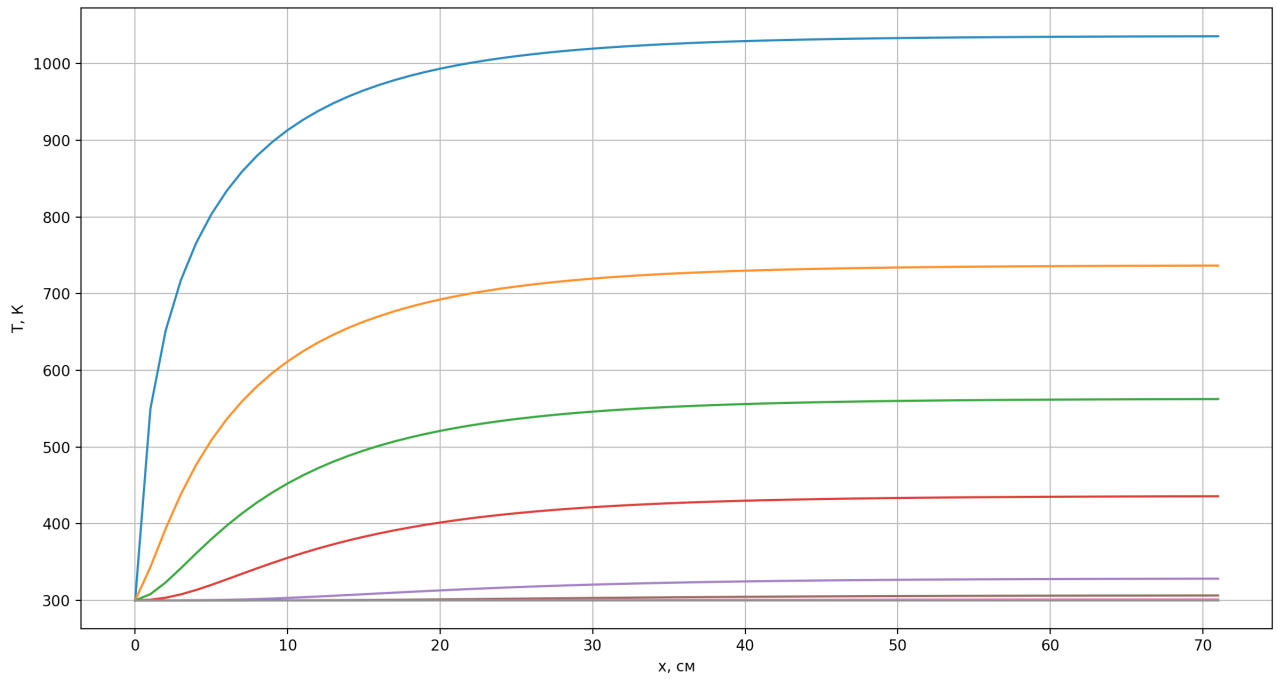

Рис. 3: Графики зависимости температуры $T(x_n, t)$ при $h_x = 0.01$ см; $h_t = 1$ сек.; $F_0 = 50.0$ Вт/см^2.

Где

- Синий цвет соответствует графику при $x = 0$ см.
- Оранжевый цвет соответствует графику при $x = 0.15$ см.
- Зеленый цвет соответствует графику при $x = 0.3$ см.
- Красный цвет соответствует графику при $x = 0.5$ см.
- Фиолетовый цвет соответствует графику при $x = 1.0$ см.
- Коричневый цвет соответствует графику при $x = 1.5$ см.

- Розовый цвет соответствует графику при $x = 2.0$ см.
- Серый цвет соответствует графику при $x = 10.0$ см.

Вопросы при защите лабораторной работы.

Ответы на вопросы дать письменно в Отчете о лабораторной работе.

1. Приведите результаты тестирования программы (графики, общие соображения, качественный анализ). Учесть опыт выполнения лабораторной работы №3.

Пример 1.

Ниже приведены графики зависимости температуры стержня T, зависящее от координаты x в разные моменты времени t_m - от начального момента $t = 0$ (синий график) до момента, когда поле перестает меняться с точностью (серый график): $\left[\frac{T(t+\tau)-T(t)}{T(t+\tau)}\right] < 10^{-4}$. Поток постоянный: $F(x) = const = F_0$. Шаг по x: $h_x = 0.01$ см. Шаг по t: $h_t = 1$ сек.. Все остальные параметры взяты из значений параметров для отладки за исключением $k(T)$ (заменен на $k(x)$ из лаб. работы №3).

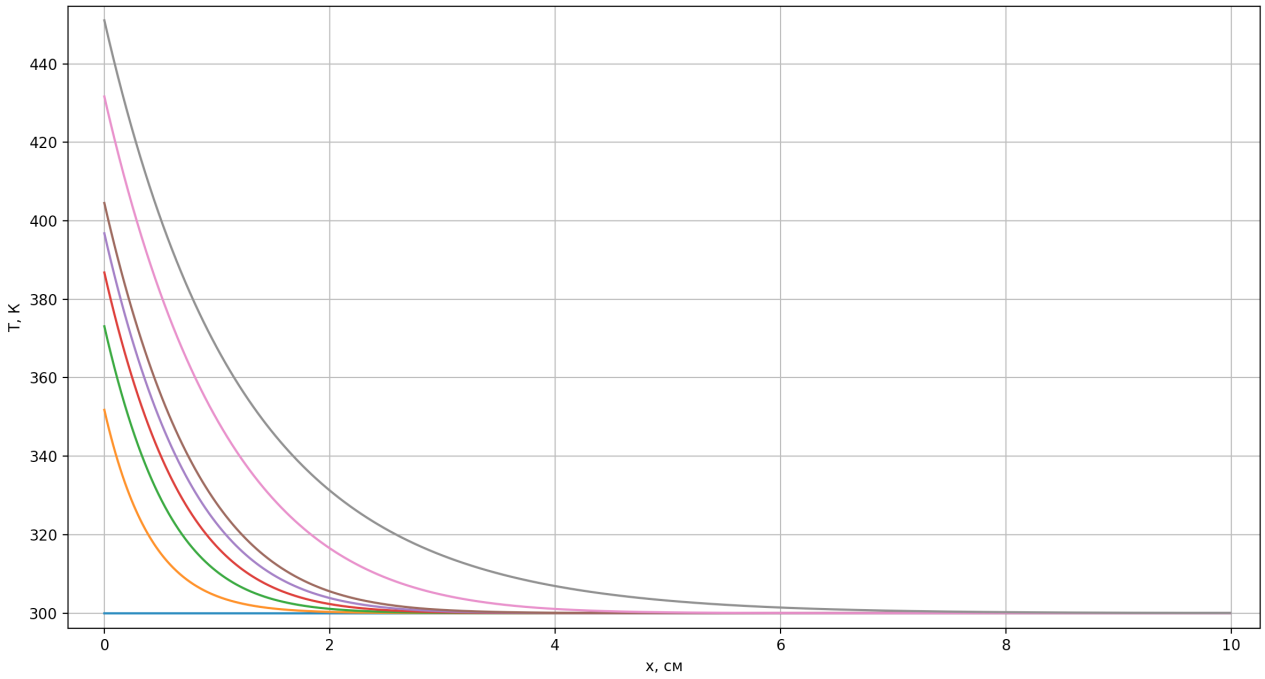

Рис. 4: Графики зависимости температуры $T(x, t_m)$ при $h_x = 0.01$ см; $h_t = 1$ сек.; $F_0 = 50.0$ Вт/см^2.

Где график серого цвета отвечает за состояние температуры стержня в момент установления стационарного режима. Этот график полностью совпадает с графиком, полученным в 3 лаб. работе при тех же параметрах, что иллюстрирует правильную работу программы.

Пример 2.

Ниже приведены графики с теми же параметрами, что и в задании 2 из раздела «Результаты работы» за исключением $F(x) = const - 0.0$ Вт/см^2 и начальной температуры стержня. Она задается такой, какой была температура стержня из примера 1 в момент установления стационарного режима. Таким образом модулируется процесс остывания стержня после его разогрева.

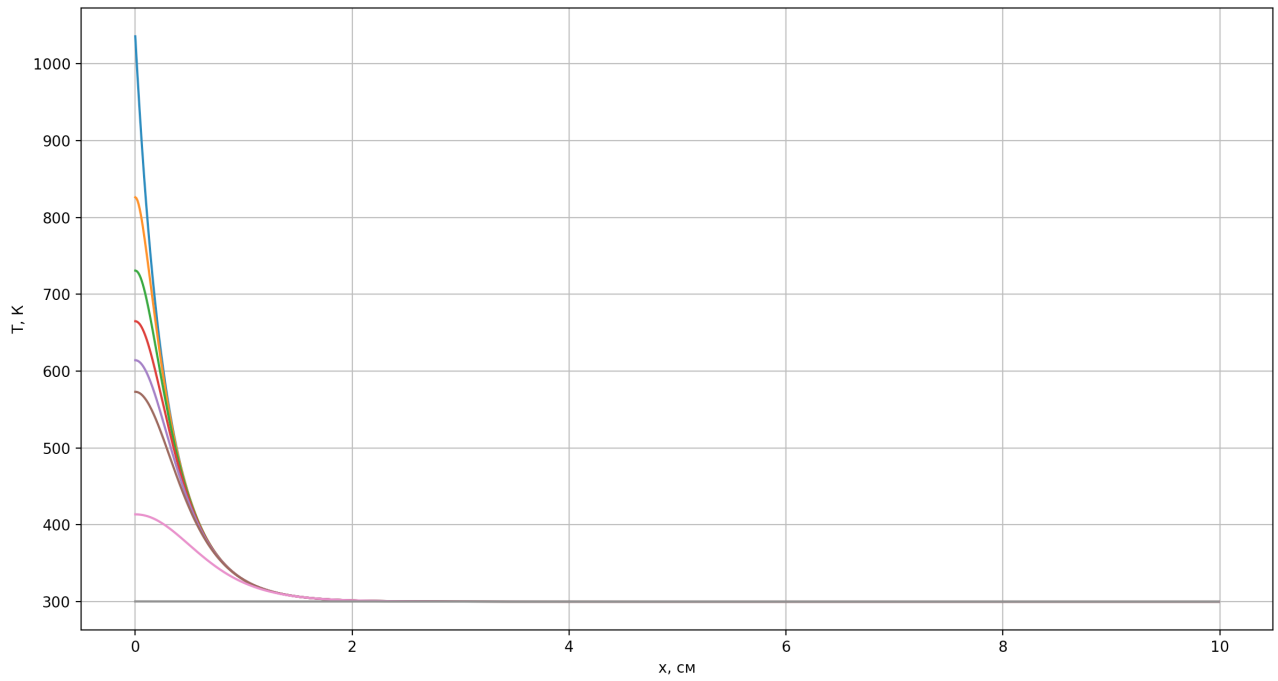

Рис. 5: Графики зависимости температуры $T(x, t_m)$ при $h_x = 0.01$ см; $h_t = 1$ сек.; $F_0 = 0.0$ Вт/см^2.

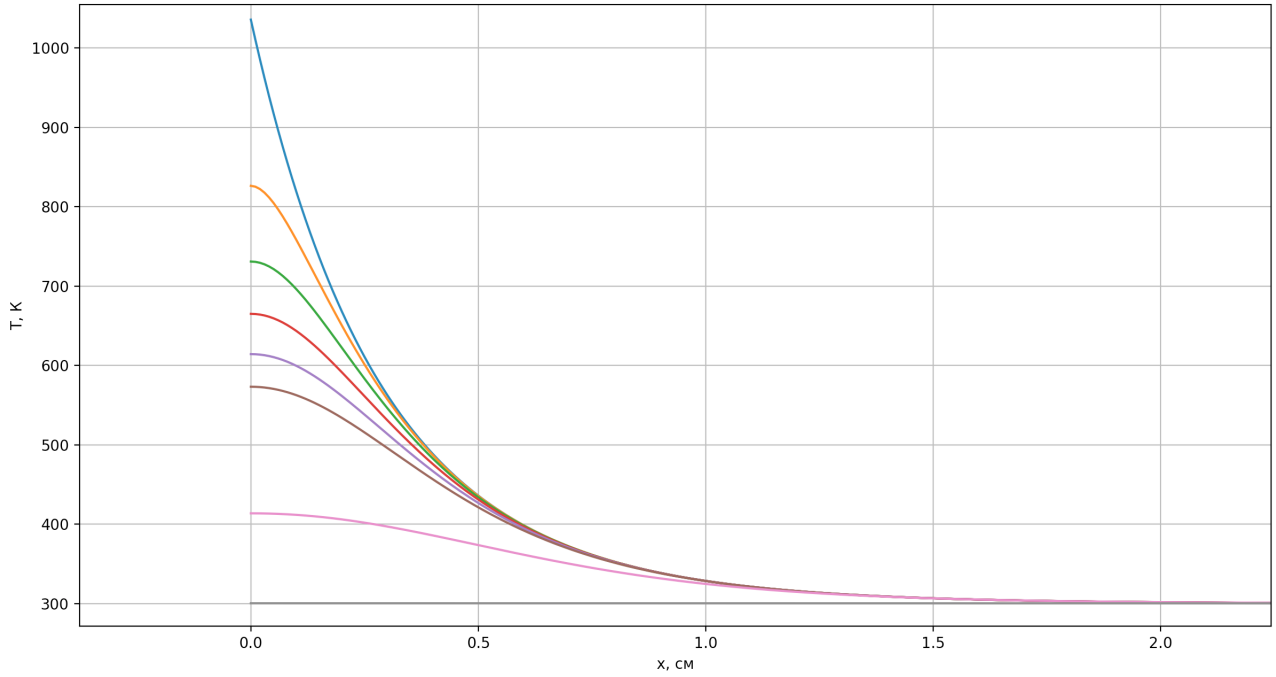

Рис. 6: Графики зависимости температуры $T(x, t_m)$ при $h_x = 0.01$ см; $h_t = 1$ сек.; $F_0 = 0.0$ Вт/см^2.

Где график серового цвета отвечает за состояние температуры стержня в момент установления стационарного режима. На нем, как и ожидалось, температура выровнилась по всей длине стержня и стала равной T_0, что также иллюстрирует правильную работу программы.

Пример 3.

Ниже приведены графики с теми же параметрами, что и в задании 2 из раздела «Результаты работы» за исключением того, что $F(t) \neq const$. Было решено задать $F(t) = 10 + 20sin(t)$ Вт/см^2.

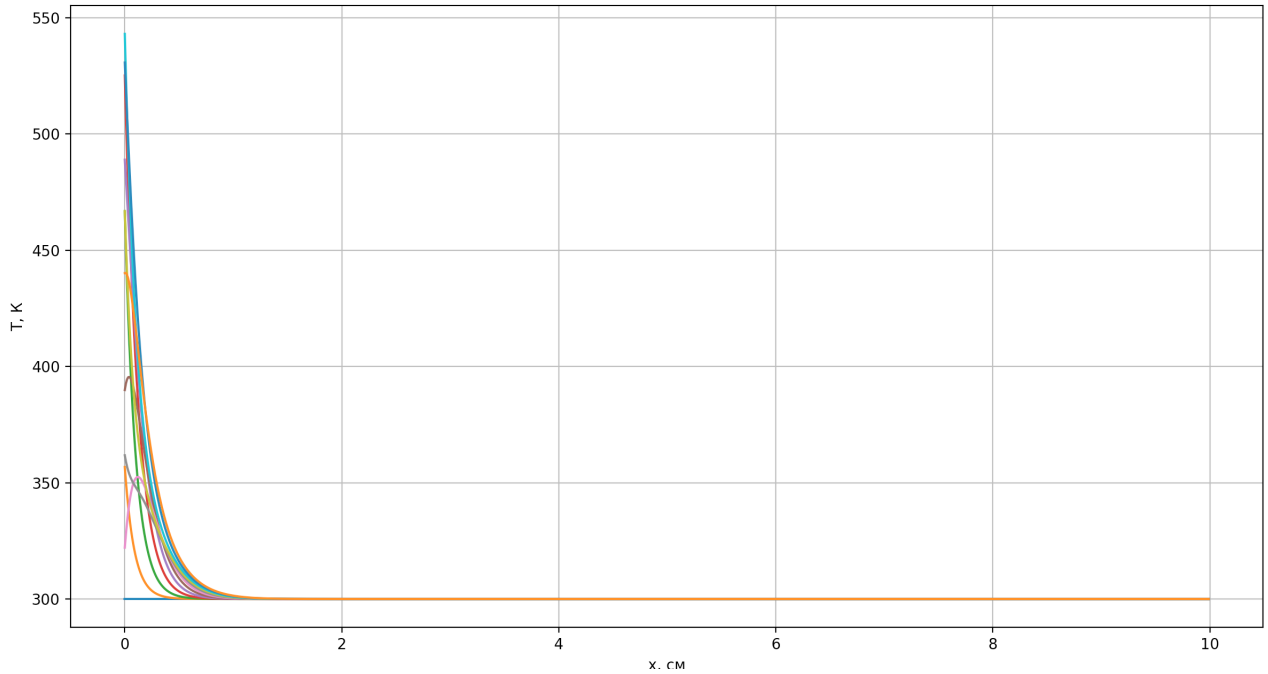

Рис. 7: Графики зависимости температуры $T(x, t_m)$ при $h_x = 0.01$ см; $h_t = 1$ сек.; $F(t) = 10 + 20sin(t)$ Вт/см^2.

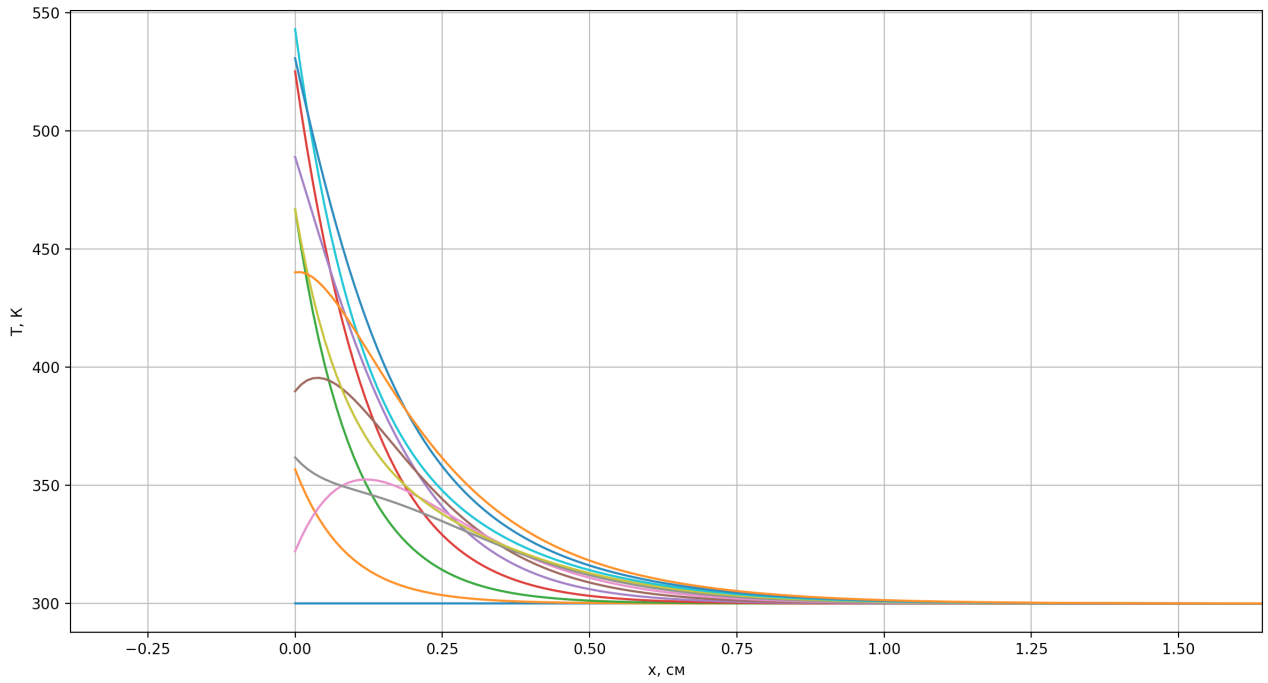

Рис. 8: Графики зависимости температуры $T(x, t_m)$ при $h_x = 0.01$ см; $h_t = 1$ сек.; $F(t) = 10 + 20sin(t)$ Вт/см^2.

Из графиков видно, что стержень то остывает, то нагревается в разные промежутки времени. Это происходит из-за немонотонности функции $F(t)$.

2. Выполните линеаризацию уравнения (14.8) по Ньютону, полагая для простоты, что все коэффициенты зависят только от одной переменной $\widehat{y_n}$. Приведите линеаризованный вариант уравнения и опишите алгоритм его решения. Воспользуйтесь процедурой вывода, описанной в лекции №8.

В нащем случае:

$$\widehat{A_n} = \widehat{A_n}(\widehat{y_n}), \widehat{B_n} = \widehat{B_n}(\widehat{y_n}), \widehat{C_n} = \widehat{C_n}(\widehat{y_n}), \widehat{D_n} = \widehat{D_n}(\widehat{y_n}).$$

Выполняя линеаризацию по Ньютону по неизвестному $\widehat{y_n}$, получим

$$\begin{aligned}
&(\widehat{A_n}\widehat{y_{n-1}} - \widehat{B_n}\widehat{y_n} + \widehat{D_n}\widehat{y_{n+1}} + \widehat{F_n})|_{s-1} + \\
&\frac{\delta(\widehat{A_n}\widehat{y_{n-1}} - \widehat{B_n}\widehat{y_n}) + \widehat{D_n}\widehat{y_{n+1}} + \widehat{F_n}}{\delta\widehat{y_{n-1}}}|_{s-1}\Delta\widehat{y^s_{n-1}} + \\
&\frac{\delta(\widehat{A_n}\widehat{y_{n-1}} - \widehat{B_n}\widehat{y_n}) + \widehat{D_n}\widehat{y_{n+1}} + \widehat{F_n}}{\delta\widehat{y_n}}|_{s-1}\Delta\widehat{y^s_n} + \\
&\frac{\delta(\widehat{A_n}\widehat{y_{n-1}} - \widehat{B_n}\widehat{y_n}) + \widehat{D_n}\widehat{y_{n+1}} + \widehat{F_n}}{\delta\widehat{y_{n+1}}}|_{s-1}\Delta\widehat{y^s_{n+1}} = 0
\end{aligned} \tag{2}$$

$$\begin{aligned}
&(\widehat{A_n}\widehat{y_{n-1}} - \widehat{B_n}\widehat{y_n} + \widehat{D_n}\widehat{y_{n+1}} + \widehat{F_n})|_{s-1} + \widehat{A_n}|_{s-1}\Delta\widehat{y^s_{n-1}} + \\
&\left(\frac{\delta\widehat{A_n}}{\delta\widehat{y_n}}\widehat{y_{n-1}} - \frac{\delta\widehat{B_n}}{\delta\widehat{y_n}}\widehat{y_n} - \widehat{B_n} + \frac{\delta\widehat{D_n}}{\delta\widehat{y_n}}\widehat{y_{n+1}} + \frac{\delta\widehat{F_n}}{\delta\widehat{y_n}}\right)|_{s-1}\Delta\widehat{y^s_n} + \widehat{D_n}|_{s-1}\Delta\widehat{y^s_{n+1}} = 0
\end{aligned} \tag{3}$$

Уравнение (3) решается методом прогонки, в результате находитятся все $\Delta\widehat{y^s_n}$, после чего определяются значения искомой функции в узлах на s-итерации $\widehat{y^s_n} = \widehat{y^{s-1}_n}\Delta\widehat{y^s_n}$. Итерационный процесс заканчивается при выполнении условия $max\left|\frac{\Delta\widehat{y^s_n}}{\widehat{y^s_n}}\right| \leqslant \varepsilon$, для всех $n = 0, 1, ...N$

Листинг кода программы

Листинг 1: Реализация задачи

```
from progonka import *
from math import *
import numpy as np

```

```
5
6  def get_abs_dif(y_n_s_minus_1, y_n_s):
7    return fabs((y_n_s - y_n_s_minus_1) / y_n_s)
8
9  def get_max_dif_from_result(T_list, T_new_list):
10   max_dif = 0
11   for i in range(len(T_list)):
12     dif = get_abs_dif(T_list[i], T_new_list[i])
13     if (max_dif < dif):
14       max_dif = dif
15   return max_dif
16
17
18
19 def calc_A_n(T_n, T_n_plus_1, data, h_x, h_t):
20   return data.X_n_and_half(T_n, T_n_plus_1) * h_t / h_x
21
22
23 def calc_C_n(T_n, T_n_minus_1, data, h_x, h_t):
24   return data.X_n_and_half(T_n, T_n_minus_1) * h_t / h_x
25
26
27 def calc_B_n(T_n, data, A, C, h_x, h_t, cur_x):
28   return A + C + data.c_T(T_n) * h_x + data.p_x(cur_x) * h_x * h_t
29
30 def calc_F_n(T_n, data, h_x, h_t, cur_x, T_time_ago):
31   return data.f_x(cur_x) * h_x * h_t + data.c_T(T_n) * T_time_ago * h_x
32
33 def calc_coeff(data, T_list, h_x, h_t, T_time_ago_list):
34   A_list, B_list, C_list, F_list = [], [], [], []
35
36   for i in range(1, len(T_list) - 1):
37     cur_x = i * h_x
38
39     #A = calc_A_n(cur_x, cur_x + h_x, data, h_x, h_t)
40
41     #C = calc_C_n(cur_x, cur_x - h_x, data, h_x, h_t)
42
43     A = calc_A_n(T_list[i], T_list[i + 1], data, h_x, h_t)
44
45     C = calc_C_n(T_list[i], T_list[i - 1], data, h_x, h_t)
46
47     B = calc_B_n(T_list[i], data, A, C, h_x, h_t, cur_x)
48     F = calc_F_n(T_list[i], data, h_x, h_t, cur_x, T_time_ago_list[i])
49
50     A_list.append(A)
51     C_list.append(C)
52     B_list.append(B)
53     F_list.append(F)
54
55   return A_list, B_list, C_list, F_list
56
57
58 def calc_left_condition(data, T_list, h_x, h_t, T_old_list, t):
59   c_0 = data.c_T(T_list[0])
```

```
60    c_1 = data.c_T(T_list[1])
61    p_0 = data.p_x(0)
62    p_1 = data.p_x(h_x)
63    p_half = (p_0 + p_1) / 2
64    c_half = (c_0 + c_1) / 2
65    #X_half = data.X_n_and_half(0, h_x)
66    X_half = data.X_n_and_half(T_list[0], T_list[1])
67    y_0 = T_old_list[0]
68    y_1 = T_old_list[1]
69    f_0 = data.f_x(0)
70    f_1 = data.f_x(h_x)
71    f_half = (f_0 + f_1) / 2
72
73    K_0 = h_x * (c_half / 8 +
74    (c_0 / 4) +
75    (h_t * p_half / 8) +
76    (h_t * p_0 / 4)) + \
77    X_half * h_t / h_x
78
79    M_0 = h_x * c_half / 8 - \
80    X_half * h_t / h_x + \
81    h_t * h_x * p_half / 8
82
83    P_0 = h_x * (
84    c_half * (y_0 + y_1) / 8 +
85    c_0 * y_0 / 4 +
86    h_t * (f_half + f_0) / 4
87    ) + data.F_0 * h_t
88
89    return K_0, M_0, P_0
90
91  def calc_right_condition(data, T_list, h_x, h_t, T_old_list):
92    N = len(T_list)
93    c_N = data.c_T(T_list[N - 1])
94    c_N_minus_1 = data.c_T(T_list[N - 2])
95    p_N = data.p_x(data.l)
96    p_N_minus_1 = data.p_x(data.l - h_x)
97    p_N_minus_half = (p_N + p_N_minus_1) / 2
98    c_N_minus_half = (c_N + c_N_minus_1) / 2
99    #X_N_minus_half = data.X_n_and_half(data.l, data.l - h_x)
100   X_N_minus_half = data.X_n_and_half(T_list[N - 1], T_list[N - 2])
101   y_N = T_old_list[N - 1]
102   y_N_minus_1 = T_old_list[N - 2]
103   f_N = data.f_x(data.l)
104   f_N_minus_1 = data.f_x(data.l - h_x)
105
106   K_N = h_t * (X_N_minus_half / h_x + data.alpha_N + h_x / 4 * p_N + h_x / 8
          * p_N_minus_half) +\
107   h_x * c_N / 4 + h_x * c_N_minus_half / 8
108
109   M_N = - h_t * (X_N_minus_half / h_x - h_x * p_N_minus_half / 8) +\
110   h_x * c_N_minus_half / 8
111
112   P_N = data.alpha_N * data.T_0 * h_t +\
113   h_t * h_x * (3 * f_N + f_N_minus_1) / 8 +\
```

```
    h_x * c_N * y_N / 4 +\
    h_x * c_N_minus_half * (y_N + y_N_minus_1) / 8


    return K_N, M_N, P_N


def get_T_list_for_cur_time(data, T_old_list, h_x, h_t, t):

    T_list = T_old_list
    max_dif = 1

    while (max_dif > data.eps):

        A_list, B_list, C_list, F_list = calc_coeff(data, T_list, h_x, h_t,
            T_old_list)

        K_0, M_0, P_0 = calc_left_condition(data, T_list, h_x, h_t, T_old_list,
            t)

        K_N, M_N, P_N = calc_right_condition(data, T_list, h_x, h_t, T_old_list)

        T_new_list = progonka(A_list, B_list, C_list, F_list, K_0, M_0, P_0, K_N
            , M_N, P_N)

        max_dif = get_max_dif_from_result(T_list, T_new_list)

        T_list = T_new_list

    return T_list


def solve_task(data, h_x, h_t):
    T_list_list = []

    T_base_list = [data.T_0 for i in np.arange(0, data.l, h_x)]

    max_dif = 1

    T_list_list.append(T_base_list)

    t = 0
    while (max_dif > data.eps):
        T_new_list = get_T_list_for_cur_time(data, T_base_list, h_x, h_t, t)
        T_list_list.append(T_new_list)

        max_dif = get_max_dif_from_result(T_base_list, T_new_list)

        T_base_list = T_new_list
        t += h_t

    return T_list_list
```

Листинг 2: Класс данных передаваемых в программу

```
class Data:
```

```
  def __init__(self):
    self.a_1 = 0.0134
    self.b_1 = 1.0
    self.c_1 = 4.35e-4
    self.m_1 = 1.0
    self.a_2 = 2.049
    self.b_2 = 0.563e-3
    self.c_2 = 0.528e5
    self.m_2 = 1.0
    self.alpha_0 = 0.05
    self.alpha_N = 0.01
    self.l = 10.0
    self.T_0 = 300.0
    self.R = 0.5
    self.F_0 = 50.0
    self.eps = 1e-4
    self.k_0 = 0.4
    self.k_N = 0.1
    self.a, self.b = self.get_a_b()
    self.c, self.d = self.get_c_d()

  def get_c_d(self):
    d = (self.alpha_N * self.l) / (self.alpha_N - self.alpha_0)
    c = self.alpha_0 * (-d)
    return c, d

  def F_t(self, t):
    return 10 + 20 * sin(t)

  def get_a_b(self):
    b = (self.k_N * self.l) / (self.k_N - self.k_0)
    a = self.k_0 * (-b)
    return a, b

  def X_n_and_half(self, T_n, T_n_and_1):
    return 2.0 * self.k_T(T_n) * self.k_T(T_n_and_1) / (self.k_T(T_n) + self
        .k_T(T_n_and_1))

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

  def f_x(self, x):
    return 2 * self.T_0 * self.alpha_x(x) / self.R

  def p_x(self, x):
    return 2 * self.alpha_x(x) / self.R

  def k_T(self, T):
    return self.a_1 * (self.b_1 + self.c_1 * pow(T, self.m_1))

  def c_T(self, T):
    return self.a_2 + self.b_2 * pow(T, self.m_2) - self.c_2 / (T * T)
```