



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 ИУ, Информатика и системы управления

КАФЕДРА ИУ7, Программное обеспечение ЭВМ и информационные технологии

ОТЧЕТ

к лабораторной работе №3

по курсу

“Моделирование”

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

Студент ИУ7-64Б
(Группа)

(Подпись, дата)

А.А. Андреев
(И.О.Фамилия)

Преподаватель

(Подпись, дата)

В.М. Градов
(И.О.Фамилия)

2022 г.

Оглавление

1. Аналитическая часть	3
Исходные данные.	3
Физическое содержание задачи	4
II. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ	6
Листинг	6
III. ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ	10

1. Аналитическая часть

Цель работы: Получение навыков разработки алгоритмов решения краевой задачи при реализации моделей, построенных на ОДУ второго порядка.

1.1. Исходные данные.

1. Задана математическая модель.

Квазилинейное уравнение для функции $T(x, t)$

$$\frac{\partial}{\partial x} (\lambda(T) \frac{\partial T}{\partial x}) - 4 \cdot k(T) \cdot n_p^2 \cdot \sigma \cdot (T^4 - T_0^4) = 0 \quad (1)$$

Краевые условия

$$\{ x = 0, -\lambda(T(0)) \frac{\partial T}{\partial x} = F_0, x = l, -\lambda(T(l)) \frac{\partial T}{\partial x} = \alpha(T(l) - T_0) \}$$

2. Функция $\lambda(T)$, $k(T)$ заданы таблицей

T, K	$\lambda, \text{Вт/(см К)}$	T, K	$k, \text{см}^{-1}$
300	$1.36 \cdot 10^{-2}$	293	$2.0 \cdot 10^{-2}$
500	$1.63 \cdot 10^{-2}$	1278	$5.0 \cdot 10^{-2}$
800	$1.81 \cdot 10^{-2}$	1528	$7.8 \cdot 10^{-2}$
1100	$1.98 \cdot 10^{-2}$	1677	$1.0 \cdot 10^{-1}$
2000	$2.50 \cdot 10^{-2}$	2000	$1.3 \cdot 10^{-1}$
2400	$2.74 \cdot 10^{-2}$	2400	$2.0 \cdot 10^{-1}$

3. Разностная схема с разностным краевым условием при $x = 0$ получена в Лекции 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 применительно к краевому условию при $x = 0$ в указанной лекции. Для этого надо проинтегрировать на отрезке $[x_{N-1/2}, x_N]$ выписанное выше уравнение (1) и учесть, что поток $F_N = \alpha_N(\hat{y}_N - T_0)$, а $F_{N-1/2} = \hat{\chi}_{N-1/2} \frac{\hat{y}_{N-1} - \hat{y}_N}{h}$

4. Значения параметров для отладки (все размерности согласованы)

$n_p = 1.4$ – коэффициент преломления,

$l = 0.2$ см – толщина слоя,

$T_0 = 300$ К – температура окружающей среды,

$\sigma = 5.668 \cdot 10^{-12}$ Вт/(см²К⁴)- постоянная Стефана- Больцмана,

$F_0 = 100$ Вт/см² - поток тепла,

$\alpha = 0.05$ Вт/(см² К) – коэффициент теплоотдачи.

5. Выход из итераций организовать по температуре и по балансу энергии, т.е.

$$\max \left| \frac{y_n^s - y_n^{s-1}}{y_n^s} \right| \leq \epsilon_1, \text{ для всех } n = 0, 1, \dots, N.$$

и

$$\max \left| \frac{f_1^s - f_2^s}{f_1^s} \right| \leq \epsilon_2,$$

где

$$f_1 = F_0 - \alpha(T(l) - T_0) \text{ и } f_2 = 4n_p^2 \sigma \int_0^l k(T(x))(T^4(x) - T_0^4) dx.$$

1.2. Физическое содержание задачи

Сформулированная математическая модель описывает температурное поле $T(x)$ в плоском слое с внутренними стоками тепловой энергии. Можно представить, что это стенка из полупрозрачного материала, например, кварца или сапфира, нагружаемая тепловым потоком на одной из поверхностей (у нас - слева). Другая поверхность (справа) охлаждается потоком воздуха, температура которого равна T_0 . Например, данной схеме удовлетворяет цилиндрическая оболочка, ограничивающая разряд в газе, т.к. при больших диаметрах цилиндра стенку можно считать плоской. При высоких температурах раскаленный слой начинает объемно излучать, что описывает второе слагаемое в (1) (закон Кирхгофа). Зависимость от температуры излучательной способности материала очень резкая. При низких температурах стенка излучает очень слабо, второе слагаемое в уравнении (1) практически отсутствует. Функции $\lambda(T)$, $k(T)$ являются, соответственно, коэффициентами теплопроводности и оптического поглощения материала стенки.

2. Технологическая часть

ЯП был выбран Python 3 из-за простоты работы с графиками и библиотеки matplotlib. Ниже на листингах будет представлена реализация программы:

```
1. import numpy as np
2. import matplotlib.pyplot as plt
3. import pandas as pd
4. import math
5. from scipy import integrate
6. from scipy.interpolate import interp1d
7.
8. plt.rcParams['figure.figsize'] = 12, 8
9. SAVE_DIR = './'
10.
11.
12. def plot_helper(x, y, xlabel, ylabel, title=False, pic='pic.png', save=True):
13.     plt.scatter(x, y, s=8, c='g')
14.     plt.plot(x, y, '-', c='r')
15.     plt.grid()
16.     plt.ylabel(ylabel)
17.     plt.xlabel(xlabel)
18.     if title: plt.title(title)
19.     if save: plt.savefig(SAVE_DIR + pic, bbox_inches='tight')
20.     fig = plt.figure()
21.
22.
23. n_p = 1.4
24. l = 0.2
25. T_0 = 300
26. sigma = 5.668e-12
27. F_0 = 100
28. alpha = 0.05
29.
30. data1 = pd.read_csv('k_T.csv')
31. data2 = pd.read_csv('lambda_T.csv')
32. k_T = interp1d(data1['T'], data1['k'], fill_value="extrapolate")
33. lambda_T = interp1d(data2['T'], data2['lambda'], fill_value="extrapolate")
34.
35. f_helper = 4 * n_p ** 2 * sigma
36. T_04 = T_0 ** 4
37.
38.
39. def T(n):
40.     if (n * 2) % 2 == 1:
41.         n = int(n - 1 / 2)
42.         return (T_list[n] + T_list[n + 1]) / 2
43.     return T_list[int(n)]
44.
45.
46. def chi_plus_half(n):
47.     return (k(n) + k(n + 1)) / 2
48.
49.
50. def chi_minus_half(n):
51.     return (k(n) + k(n - 1)) / 2
52.
53.
54. def k(n):
55.     return lambda_T(T(n))
56.
57.
58. def f_inner(n):
59.     return k(n) * (T(n) ** 4 - T_04)
```

Листинг 1: Код программы, Часть 1

```

60.
61. def f(n):
62.     return -f_helper * f_inner(n)
63.
64.
65. def A(n): return chi_minus_half(n) / h
66.
67.
68. def C(n): return chi_plus_half(n) / h
69.
70.
71. def B(n): return A(n) + C(n)
72.
73.
74. def D(n): return f(n) * h
75.
76.
77. def trapezoidal_intergrate(func, a, b, step):
78.     S = (func(a) + func(b)) / 2
79.     for x in range(a + step, b, step):
80.         S += func(x)
81.     return S * step
82.
83.
84. def boundary_condition():
85.     K0 = chi_plus_half(0)
86.     M0 = -K0
87.     P0 = h * F_0 + h ** 2 / 4 * (f(1 / 2) + f(0))
88.     KN = -chi_minus_half(N)
89.     MN = alpha * h - KN
90.     PN = alpha * h * T_0 + h ** 2 / 4 * (f(N - 1 / 2) + f(N))
91.     return K0, M0, P0, KN, MN, PN
92.
93.
94. def err_temperature(t_old, t_new):
95.     return max([abs(1 - t_old[i] / t_new[i]) for i in range(len(t_old))])
96.
97.
98. def err_energy_balance(t_new):
99.     f1 = F_0 - alpha * (t_new[N] - T_0)
100.    f2 = f_helper * trapezoidal_intergrate(f_inner, 0, N, 1) * h
101.    return abs(1 - f2 / f1)
102.
103.
104.    def thomas_algorithm():
105.        K0, M0, P0, KN, MN, PN = boundary_condition()
106.
107.        # forward
108.        xi = [None, - M0 / K0]
109.        eta = [None, P0 / K0]
110.
111.        for i in range(1, N):
112.            denominator = (B(i) - A(i) * xi[i])
113.            x = C(i) / denominator
114.            e = (D(i) + A(i) * eta[i]) / denominator
115.            xi.append(x)
116.            eta.append(e)
117.
118.        # backward
119.        y = [(PN - KN * eta[-1]) / (MN + KN * xi[-1])]
120.
121.        for i in range(N - 1, -1, -1):
122.            yi = xi[i + 1] * y[0] + eta[i + 1]
123.            y.insert(0, yi)
124.        return y

```

Листинг 2: Код программы, Часть 2

```

125.
126. def fixed_point_iteration(esp1, esp2, max_loop=20):
127.     global T_list
128.     err1 = err2 = 1
129.
130.     for i in range(max_loop):
131.         if err1 <= esp1 and err2 <= esp2:
132.             break
133.         t = thomas_algorithm()
134.         err1 = err_temperature(T_list, t)
135.         T_list = t
136.         err2 = err_energy_balance(T_list)
137.         print(i, err1, err2)
138.
139.
140. h = 0.01
141. N = int(1 // h)
142. esp1 = 0.01
143. esp2 = 0.01
144.
145. x = [i for i in np.arange(0, 1 + h, h)]
146.
147. # F = 100, alpha = 0.05
148. T_list = [T_0] * (N + 1)
149. F_0 = 100
150. alpha = 0.05
151.
152. fixed_point_iteration(esp1, esp2)
153. plot_helper(x, T_list, 'x, cm', 'T, K', pic='F0=100,a=0.05.png')
154.
155. # F = -10
156. T_list = [T_0] * (N + 1)
157. F_0 = -10
158.
159. fixed_point_iteration(esp1, esp2)
160. plot_helper(x, T_list, 'x, cm', 'T, K', 'F0 = -10', pic='F0=-10,a=0.05.png')
161.
162. # alpha x3
163. T_list = [T_0] * (N + 1)
164. F_0 = 100
165. alpha = 0.15
166.
167. fixed_point_iteration(esp1, esp2)
168. plot_helper(x, T_list, 'x, cm', 'T, K', 'alpha = 0.15 (x3)',
169.             pic='F0=100,a=0.15.png')
170.
171. # F = 0
172. T_list = [T_0] * (N + 1)
173. F_0 = 0
174. alpha = 0.05
175. fixed_point_iteration(esp1, esp2)
176. plot_helper(x, T_list, 'x, cm', 'T, K', 'F0 = 0', pic='F0=0.png')

```

Листинг 3: Код программы, Часть 3

3. Экспериментальная часть

В данном разделе будет рассмотрен вывод программы и представлены графики зависимостей.

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

Краевая условия $x = l$

Обозначим $F = -k(x) \frac{du}{dx}$.

Проинтегрируем на отрезке $[x_{N-1/2}, x_{N+1/2}]$:

$$-\int_{x_{N-1/2}}^{x_N} \frac{dF}{dx} dx - \int_{x_{N-1/2}}^{x_N} p(x)u dx + \int_{x_{N-1/2}}^{x_N} f(x) dx = 0.$$

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

$$\text{Подставим } F_N = \alpha(y_N - \beta), \quad F_{N-1/2} = \chi_{N-1/2} \frac{y_{N-1} - y_N}{h}$$

$$-(\alpha h(y_N - \beta) - \chi_{N-1/2}(y_{N-1} - y_N)) - \frac{h^2}{4} \cdot (p_{N-1/2} \frac{y_{N-1} + y_N}{2} + p_N y_N) + \frac{h^2}{4} (f_{N-1/2} + f_N) = 0$$

$$\alpha h y_N - \chi_{N-1/2}(y_{N-1} - y_N) + \frac{h^2}{4} \cdot (p_{N-1/2} \frac{y_{N-1} + y_N}{2} + p_N y_N) = \alpha \beta h + \frac{h^2}{4} (f_{N-1/2} + f_N)$$

$$(\frac{h^2}{8} p_{N-1/2} - \chi_{N-1/2}) y_{N-1} + (\alpha h + \chi_{N-1/2} + \frac{h^2}{8} p_{N-1/2} + \frac{h^2}{4} p_N) y_N = \alpha \beta h + \frac{h^2}{4} (f_{N-1/2} + f_N)$$

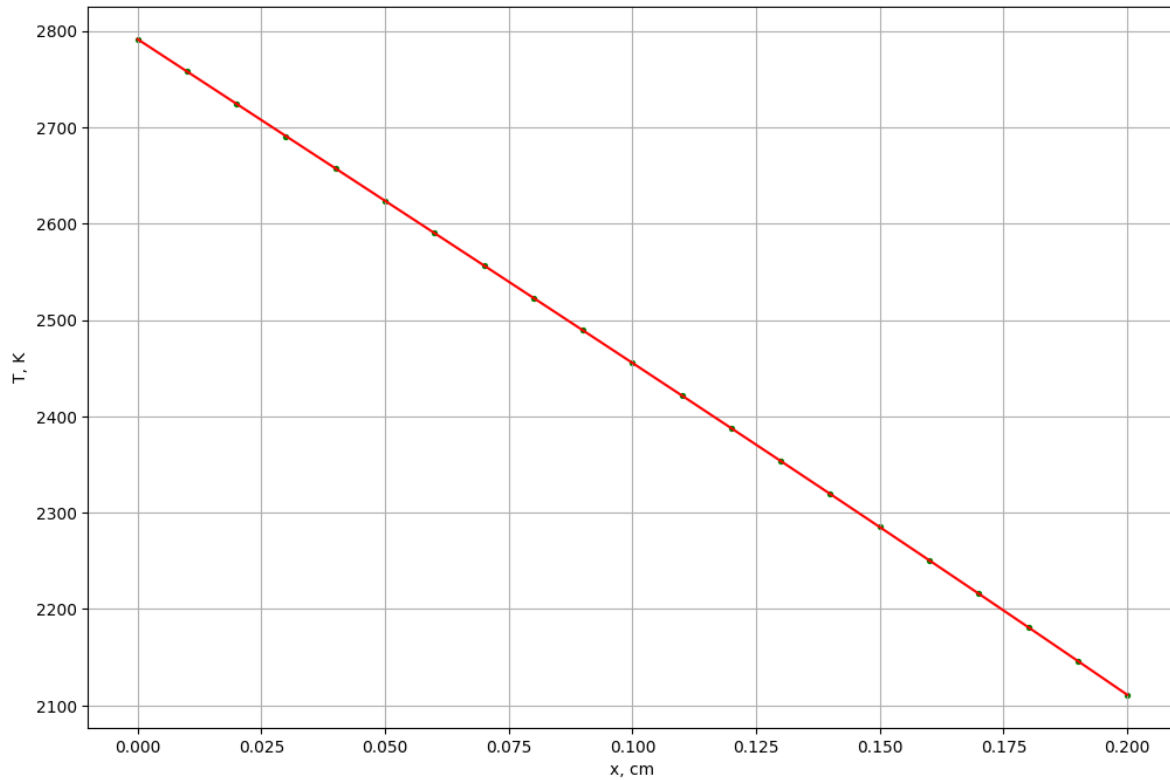
$$K_N = \frac{h^2}{8} p_{N-1/2} - \chi_{N-1/2}, \quad M_N = \alpha h + \chi_{N-1/2} + \frac{h^2}{8} p_{N-1/2} + \frac{h^2}{4} p_N, \quad P_N = \alpha \beta h + \frac{h^2}{4} (f_{N-1/2} + f_N)$$

$$\chi_{n \pm \frac{1}{2}} = \frac{\lambda_n + \lambda_{n \pm 1}}{2}, \quad f(x) = -4 \cdot k(T) \cdot n_p^2 \cdot \sigma \cdot (T^4 - T_0^4)$$

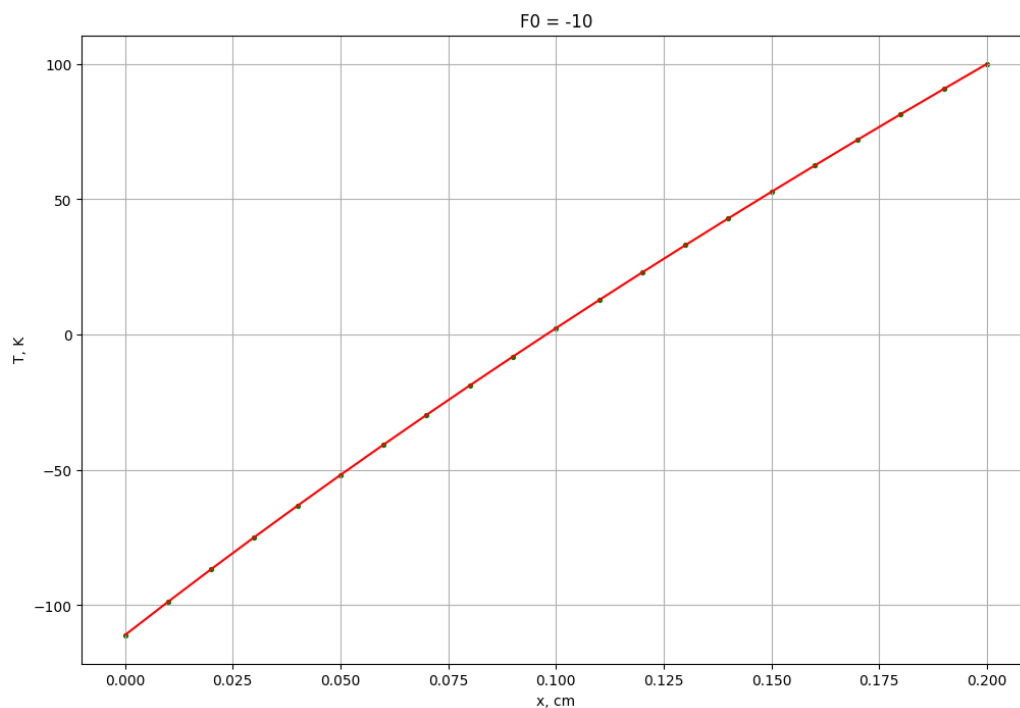
$$K_N = -\chi_{N-1/2}, \quad M_N = \alpha h + \chi_{N-1/2}, \quad P_N = \alpha T_0 h + \frac{h^2}{4} (f_{N-1/2} + f_N)$$

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

Выяснить, как сильно зависят результаты расчета $T(x)$ и необходимое для этого количество итераций от начального распределения температуры и шага сетки. Количество необходимых повторений не сильно зависит от начального распределения температуры и шага сетки. В этом случае при $h = 0.01\text{ см}$ и $\epsilon_1 = \epsilon_2 = 0.01$ требуется 10 итераций.

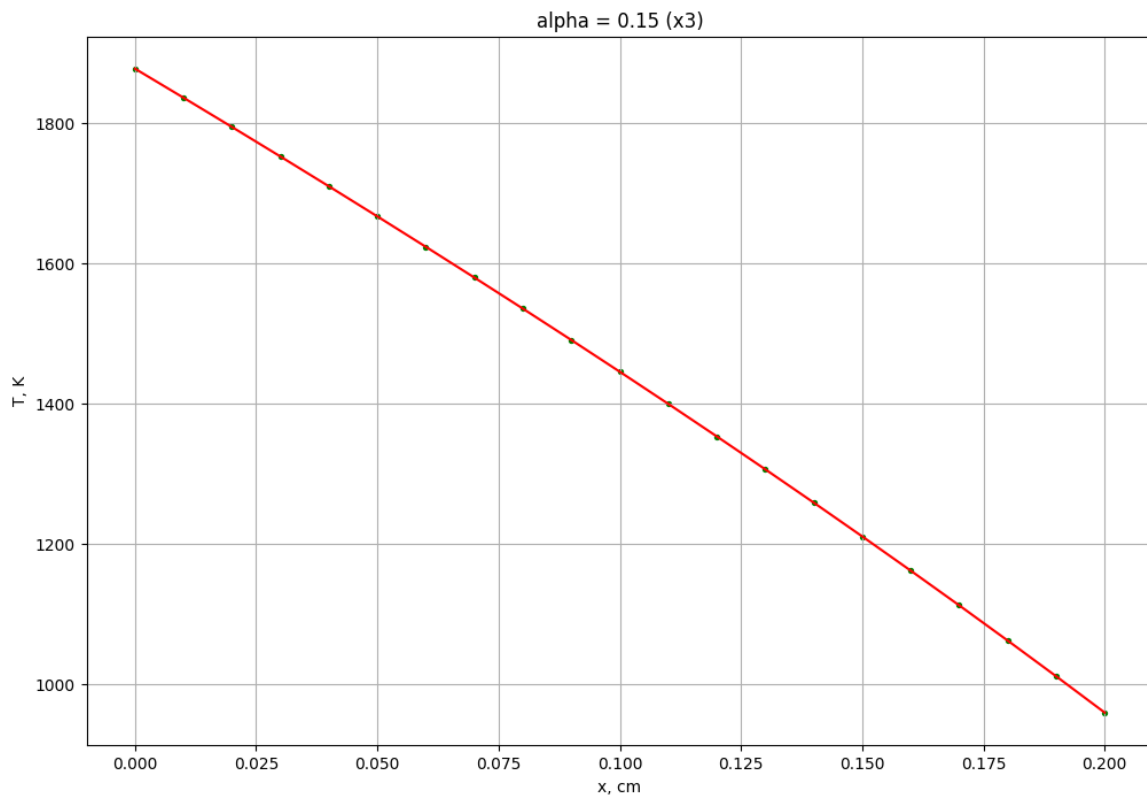


3. График зависимости $T(x)$ при $F_0 = -10\text{ Вт/см}^2$.



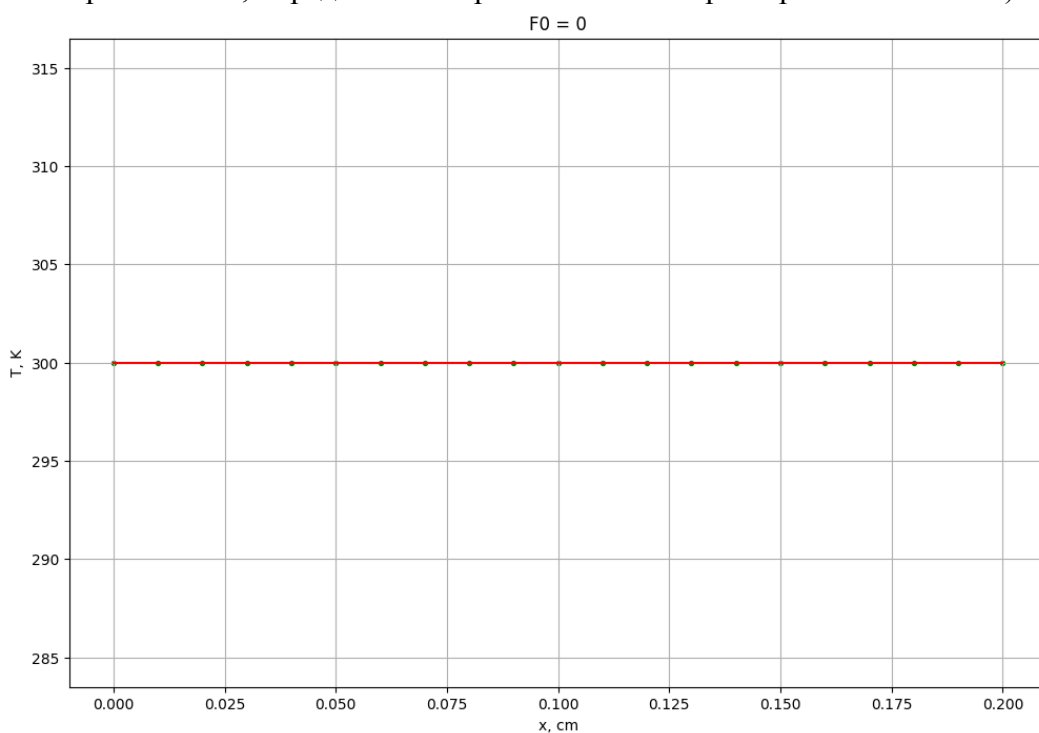
4. График зависимости $T(x)$ при увеличенных значениях α (например, в 3 раза). Сравнить с п.2.

Коэффициент теплоотдачи больше, тепло быстрее передается наружу, более низкая температура



5. График зависимости $T(x)$ при $F0 = 0$.

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)



6. Для указанного в задании исходного набора параметров привести данные по балансу энергии, т.е. значения величин

$$f_1 = F_0 - \alpha(T(l) - T_0) \text{ и } f_2 = 4n_p^2 \sigma \int_0^l k(T(x))(T^4(x) - T_0^4)dx.$$

Каковы использованные в работе значения точности выхода из итераций ϵ_1 (по температуре) и ϵ_2 (по балансу энергии)?

В работе использовал $\epsilon_1 = 0.01, \epsilon_2 = 0.01$

Столбцы соответственно: № итерация, $\max \left| \frac{y_n^s - y_n^{s-1}}{y_n^s} \right|$, и $\left| \frac{f_1^s - f_2^s}{f_1^s} \right|$,

```
0 0.92044 241498676052815.93750
1 0.65778 0.86107
2 0.24279 2.34494
3 0.11745 0.37317
4 0.05376 0.26953
5 0.02578 0.10286
6 0.01217 0.05438
7 0.00579 0.02412
8 0.00275 0.01223
9 0.00131 0.00523
```

4. Ответы на контрольные вопросы

1. Какие способы тестирования программы можно предложить?

- $F_0 = 0 \Rightarrow T(x) = T_0$
- $F_0 > 0 \Rightarrow T'(x) < 0$, $F_0 < 0 \Rightarrow T'(x) > 0$
- α увеличивается $\Rightarrow T(x)$ уменьшается

2. Получите простейший разностный аналог нелинейного краевого условия при $x = l$

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

где $\phi(T)$ - заданная функция. Производную аппроксимируйте односторонней разностью.

$$\begin{aligned} -k_l \frac{T_l - T_{l-1}}{h} &= \alpha_N(T_l - T_0) + \varphi(T_l) \\ -(k_l + \alpha_N h)T_l + k_l T_{l-1} &= \varphi(T_l)h - \alpha_N h T_0 \end{aligned}$$

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

$$\begin{cases} x = 0, & -k(0) \frac{dT}{dx} = F_0, \\ x = l, & -k(l) \frac{dT}{dx} = \alpha_N(T(l) - T_0) + \varphi(T). \end{cases}$$

Будем использовать левую прогонку, основная прогоночная формула:

$$y_n = \xi_{n+1} y_{n+1} + \eta_{n+1}$$

$$-k_0 \frac{T_1 - T_0}{h} = F_0 \quad \Rightarrow \quad T_0 = T_1 + \frac{F_0 h}{k_0}.$$

$$-(k_l + \alpha_N h)T_l + k_l T_{l-1} = \varphi(T_l)h - \alpha_N h T_0,$$

$$-(k_l + \alpha_N h)T_l + k_l(\xi_l T_l + \eta_l) = \varphi(T_l)h - \alpha_N h T_0.$$

$$T_l = \frac{\alpha_N h T_0 + k_l \eta_l - \varphi(T_l)h}{k_l(1 - \xi_l) + \alpha_N h}$$