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»

ОТЧЕТ

к лабораторной работе №3

По курсу: «Моделирование»

Студент Жигалкин Д.Р.
Группа ИУ7-65Б
Преподаватель Градов В.М.

Москва, 2021 г.

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

Исходные данные.

1. Задана математическая модель. Квазилинейное уравнение для функции T(x):

$$\frac{d}{dx}\left(\lambda(T)\frac{dT}{dx}\right) - 4k(T) * n_p^2 * \sigma * (T^4 - T_0^4) = 0$$

Краевые условия

$$\begin{cases} x = 0, & -\lambda\big(T(0)\big)\dfrac{dT}{dx} = F_0, \\ x = l, & -\lambda\big(T(l)\big)\dfrac{dT}{dx} = \alpha(T(l) - T_0) \end{cases}$$

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

T, K	λ, Вт/(см К)		T, K	k, см-1
300	1.36*10-2		293	2.0*10-2
500	1.63*10-2		1278	5.0*10-2
800	1.81*10-2		1528	7.8*10-2
1100	1.98*10-2		1677	1.0*10-1
2000	2.50*10-2		2000	1.3*10-1
2400	2.74*10-2		2400	2.0*10-1

3. Разностная схема с разностным краевым условием при x = 0 . Получена в Лекции №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 было сделано применительно к краевому условию при x = 0 в указанной лекции. Для этого надо проинтегрировать на отрезке $[x_{N-\frac{1}{2}}, x_N]$записанное выше уравнение (1) и учесть, что поток

$$F_N = \alpha(y_N - T_0), \qquad F_{N-1/2} = \varkappa_{N-1/2}\frac{y_{N-1} - y_N}{h}$$

4. Значения параметров для отладки (все размерности согласованы)
n_p= 1.4 – коэффициент преломления,
l= 0.2 см – толщина слоя,

T_0= 300К – температура окружающей среды,
σ =5.668 10-12 Вт/(см2К 4)- постоянная Стефана- Больцмана,
F_0 =100 Вт/см2 - поток тепла,
α= 0.05 Вт/(см2 К) – коэффициент теплоотдачи.

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

$$\max\left|\frac{y_n^s - y_n^{s-1}}{y_n^s}\right| < \varepsilon_1, \forall n \in \{0,..,N\}$$

и

$$\max\left|\frac{f_1^s - f_2^s}{f_1^s}\right| < \varepsilon_2,$$

Где

$$f_1 = F_0 - \alpha(T(l) - T_0), \qquad f_2 = 4n_p^2\sigma\int_0^1 k\big(T(x)\big)(T^4(x) - T_0^4)dx$$

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

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 λ(T), k(T) являются, соответственно, коэффициентами теплопроводности и оптического поглощения материала стенки.

Результат работы.

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

Пусть $F = -\lambda(T)\frac{dT}{dx}$.

Тогда

$\frac{dF}{dx} + f(T) = 0$, где

$f(T) = -4k(T)n_p^2\sigma(T^4 - T_0^4)$

Проитегрируем на отрезке $[x_{N-1/2}, x_N]$

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

Применим метод трапеций.

$$-\left(F_N - F_{N-\frac{1}{2}}\right) + \frac{h}{4}\left(f_N + f_{N-\frac{1}{2}}\right) = 0$$

С учетом $F_N = \alpha(y_N - \beta)$, $F_{N-1/2} = \varkappa_{N-1/2}\frac{y_{N-1}-y_N}{h}$, получим

$$-\left(\alpha(y_N - T_0) - \chi_{N-\frac{1}{2}}\frac{y_{N-1}-y_N}{h}\right) + \frac{h}{4}\left(f_N + f_{N-\frac{1}{2}}\right) = 0$$

$$-h\alpha(y_N - T_0) + \chi_{N-\frac{1}{2}}(y_{N-1} - y_N) + \frac{h^2}{4}\left(f_N + f_{N-\frac{1}{2}}\right) = 0$$

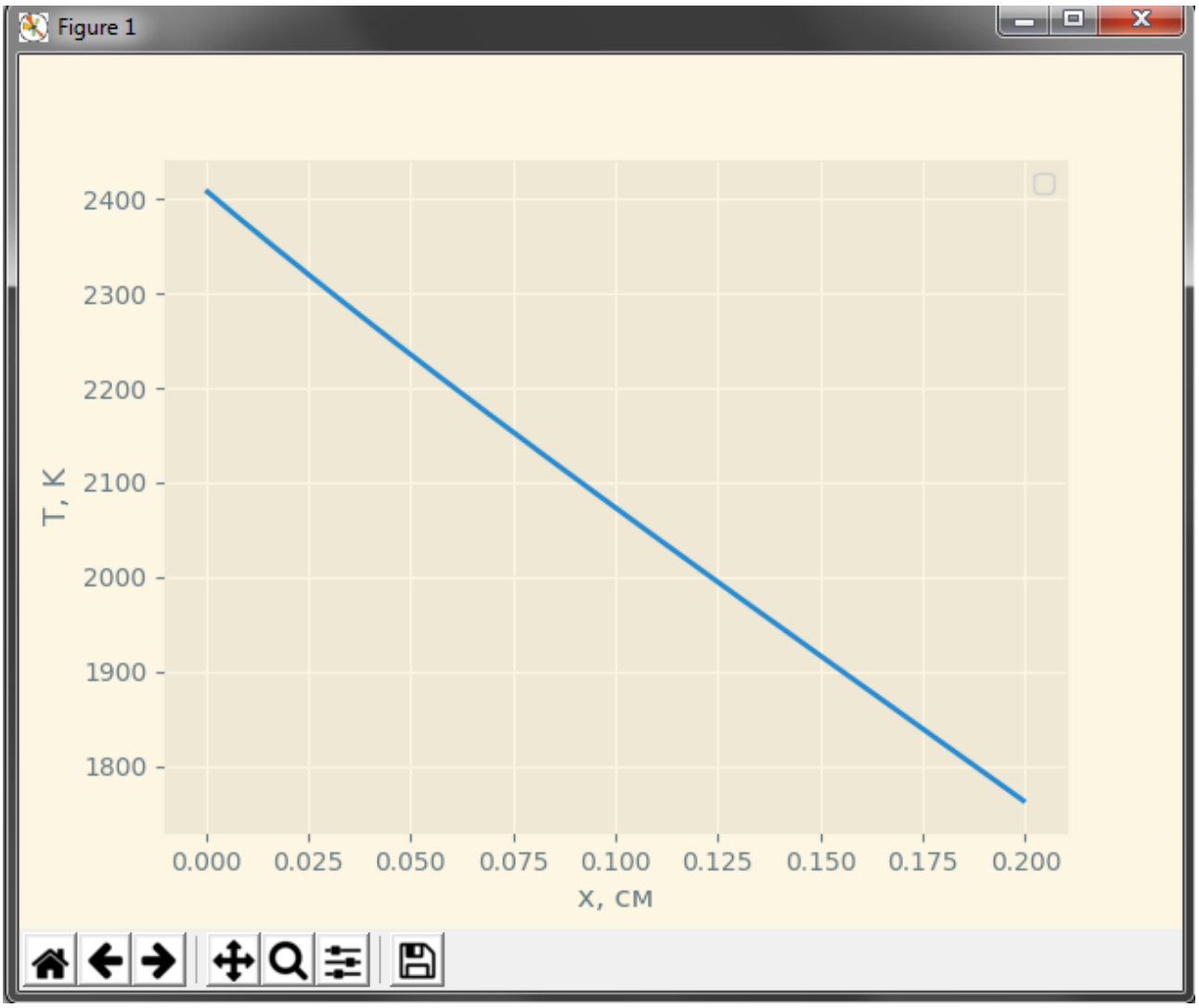
$$-\chi_{N-\frac{1}{2}}y_{N-1} + \left(h\alpha + \chi_{N-\frac{1}{2}}\right)y_N = \frac{h^2}{4}\left(f_N + f_{N-\frac{1}{2}}\right) + h\alpha T_0$$

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.

Параметры, заданные выше:

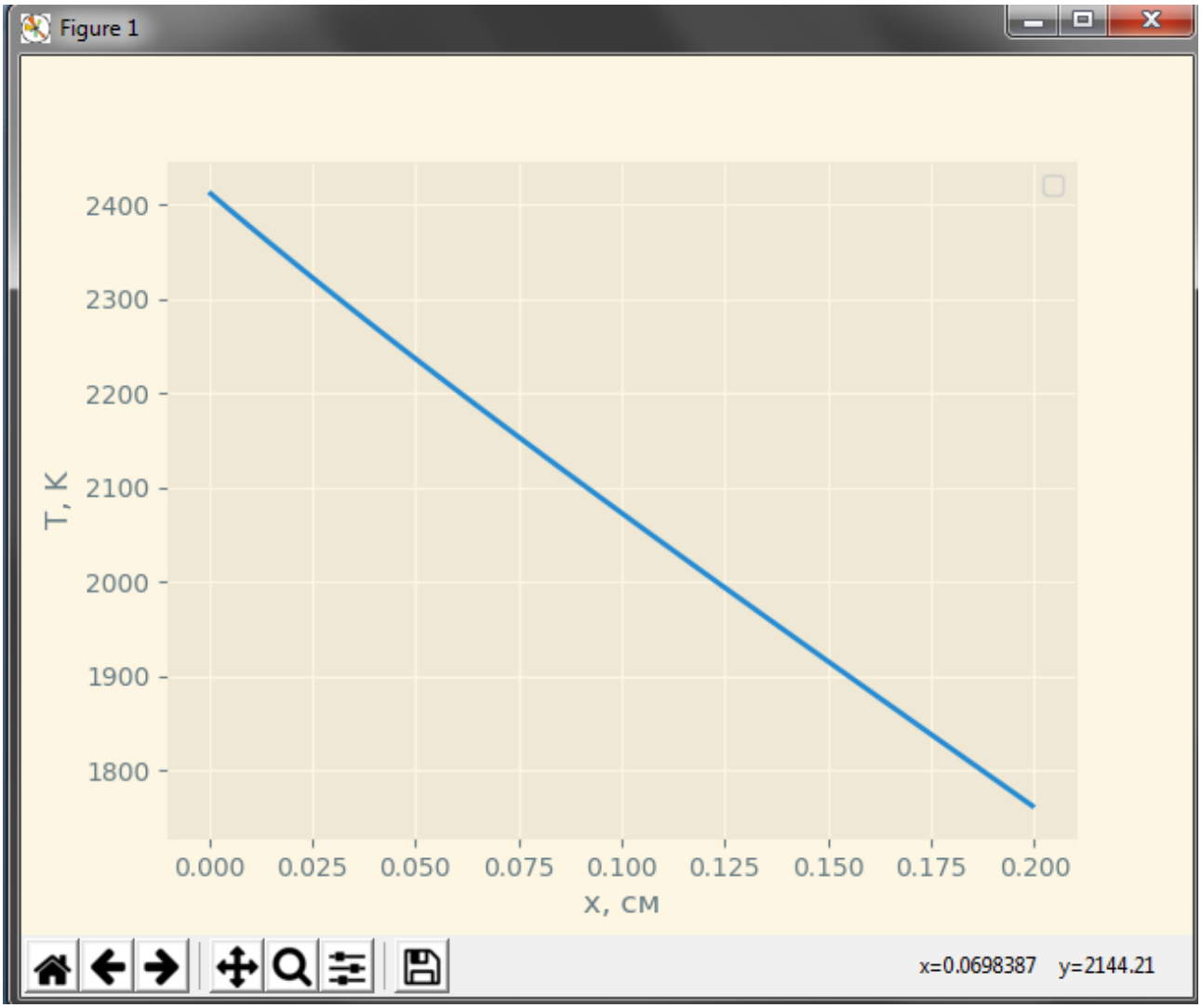
$F_0 = 100, T_0 = 300, n_p = 1.4, \alpha = 0.05$, начальное приближение: 300K

h = 1e-3



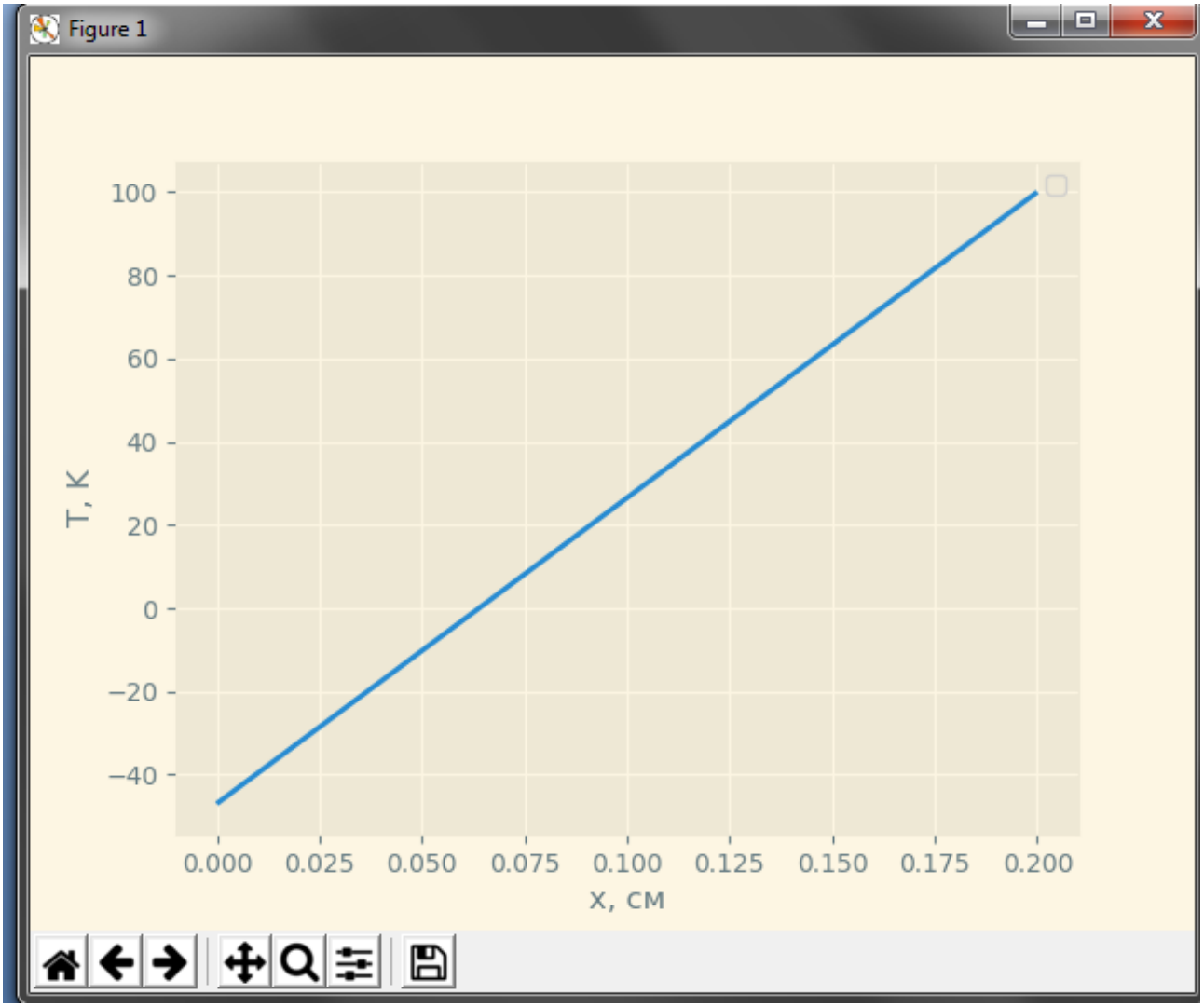
$F_0 = 100, T_0 = 300, n_p = 1.4, \alpha = 0.05$, начальное приближение: 1600K

h = 5e-3



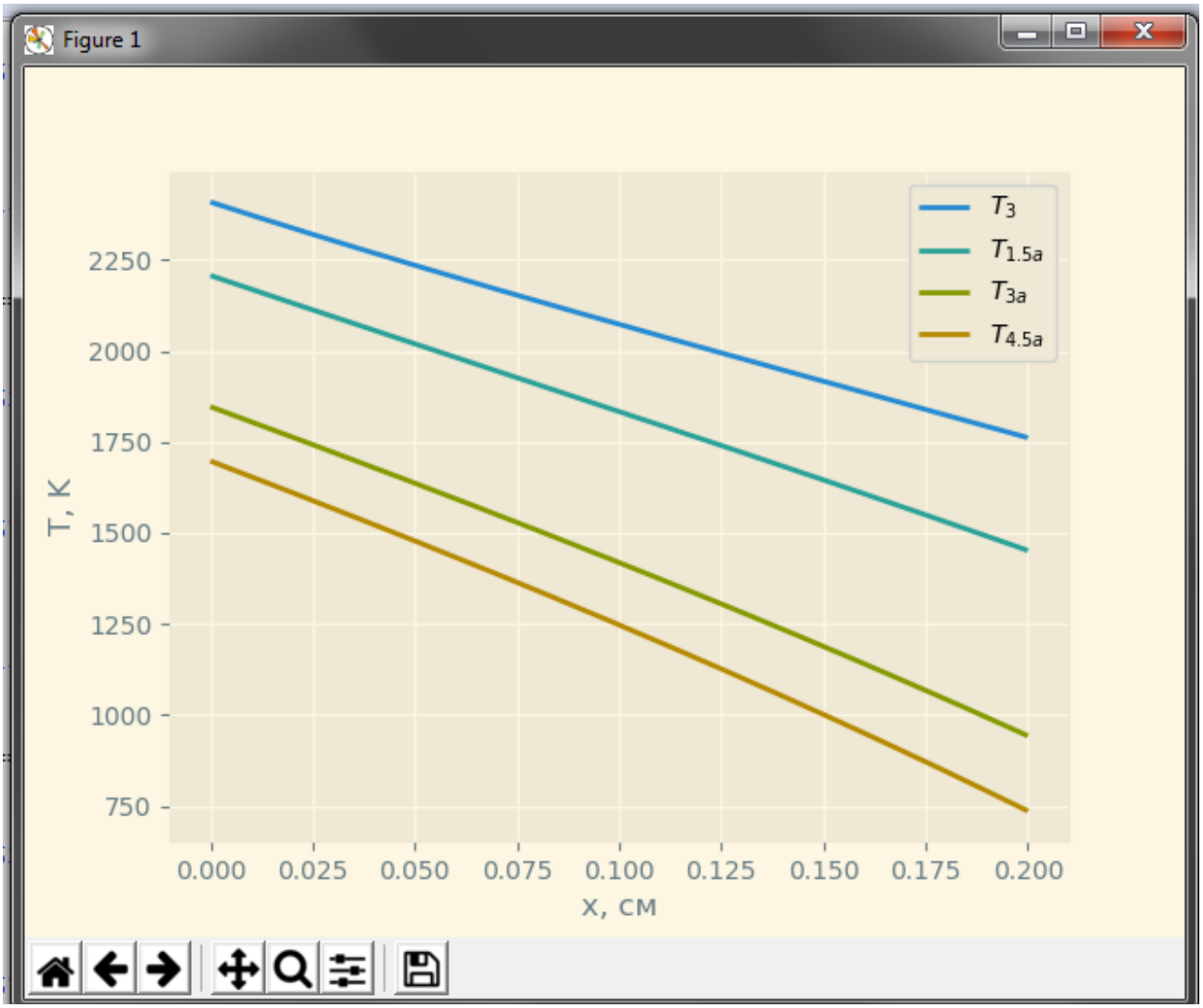
3) График зависимости T(x) при F_0= -10 Вт/см2.

Справка. При отрицательном тепловом потоке слева идет съем тепла, поэтому производная $T'(x)$должна быть положительной.



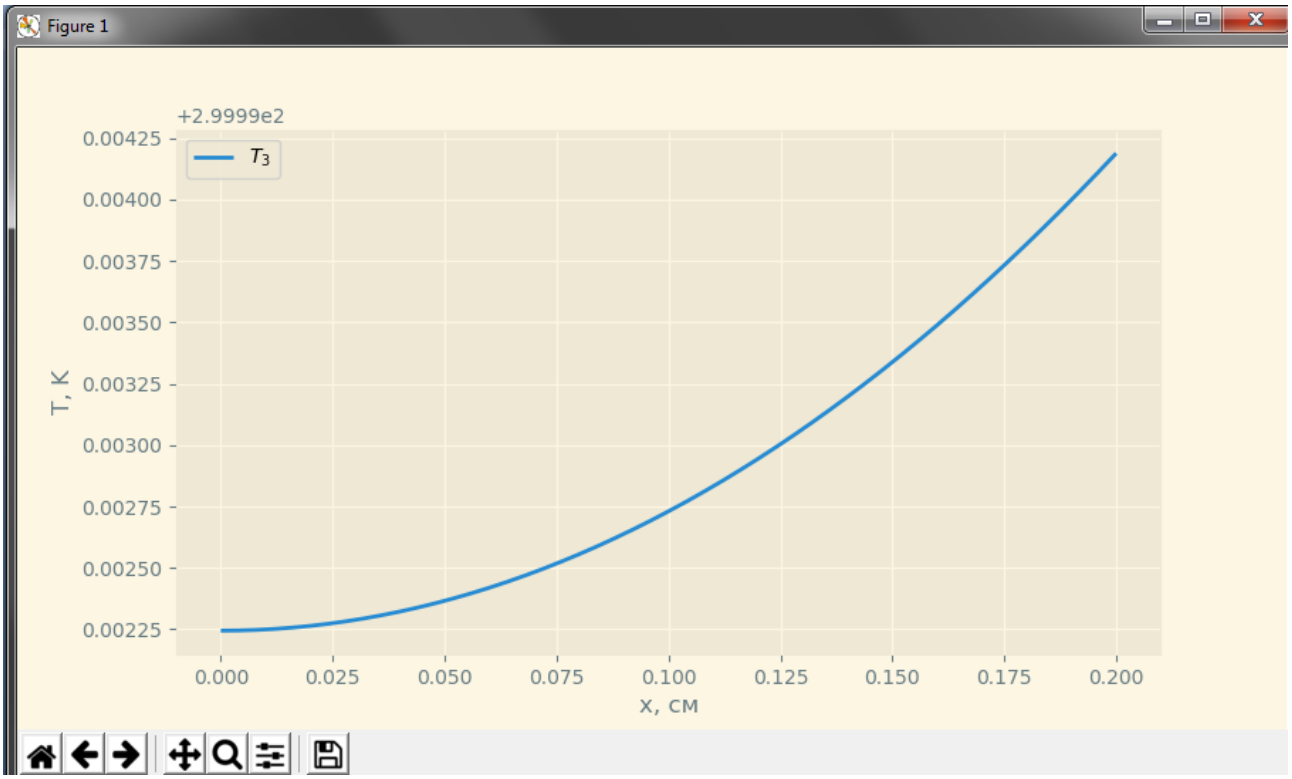
4) График зависимости T(x) при увеличенных значениях α (например, в 3 раза).

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).



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

$$f_1 = F_0 - \alpha(T(l) - T_0), \qquad f_2 = 4n_p^2\sigma \int_0^1 k\big(T(x)\big)(T^4(x) - T_0^4)dx$$

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

```
========================================= RESTART: C:\Users\Dmit
eps1: 8.557240659615842e-05
eps2: 0.0008425127105649767
```

Вопросы при защите лабораторной работы

1. Какие способы тестирования программы можно предложить?

   Исходя из физического смысла задачи, при $F_0 = 0$ температура должна быть равна температуре окружающей среды T_0. При отрицательном тепловом потоке слева идет съем тепла, поэтому производная $T'(x)$должна быть положительной.

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

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

где φ(T)- заданная функция. Производную аппроксимируйте односторонней разностью.

Аппроксимация производной:

$$\frac{dT}{dx} \approx \frac{T_{i+1} - T_i}{h}$$

Подстановка:

$$-k_N\frac{T_N - T_{N-1}}{h} = a_N(T_N - T_0) + \varphi(T_N)$$

Домножим на h:

$$-k_N T_N + k_N T_{N-1} = h a_N T_N - h a_N T_0 + h\varphi(T_N)$$

$$k_N T_{N-1} - (k_N + a_N h) T_N = \varphi(T_N) h - a_N h T_0$$

Листинг

```
def solve(Ts):
    A = [None] * (N - 1)
    B = [None] * (N - 1)
    C = [None] * (N - 1)
    D = [None] * (N - 1)

    X = np.arange(h, l, h)
    for j, x in enumerate(X):
        i = j + 1
        ln0 = getLambda(Ts[i - 1])
        ln1 = getLambda(Ts[i])
        ln2 = getLambda(Ts[i + 1])

        A[j] = cappa(ln0, ln1) / h
        C[j] = cappa(ln1, ln2) / h

        B[j] = A[j] + C[j] + p(Ts[i]) * h
        D[j] = f(Ts[i]) * h

    K0, M0, P0 = leftBoundary(Ts)
    KN, MN, PN = rightBoundary(Ts)

    return thomas_algorithm(A, B, C, D, K0, M0, P0, KN, MN, PN)

def f2(T):
    x = np.arange(0, l + h, h)
    yi = [getK(Ti) * (pow(Ti, 4) - pow(T0, 4)) for Ti in T]
    return 4 * koef_np**2 * sigma * integrate.simpson(yi, x)

def f1(T):
    return F0 - a * (T[-1] - T0)
def getLambda(T):
    return np.interp(T, table1[0], table1[1])

table2[1] = np.log(table2[1])
def getK(T):
    return np.exp(np.interp(T, table2[0], table2[1]))

def p(T):
    return 4 * koef_np**2 * sigma * getK(T) * pow(T, 3)

def f(T):
    return 4 * koef_np**2 * sigma * getK(T) * pow(T0, 4)

def cappa(ln, ln1):
    return (2 * ln * ln1) / (ln1 + ln)

def leftBoundary(Ts):
    h2 = h * h
```

```
    l0 = getLambda(Ts[0])
    l1 = getLambda(Ts[1])

    p1_2 = (p(Ts[0]) + p(Ts[1])) / 2

    K0 = cappa(l0, l1) + h2 / 8 * p1_2 + h2 / 4 * p(Ts[0])
    M0 = -cappa(l0, l1) + h2 / 8 * p1_2
    P0 = h * F0 + h2 / 4 * (3*f(Ts[0]) + f(Ts[1])) / 2

    return K0, M0, P0

def rightBoundary(Ts):
    h2 = h * h
    ln = getLambda(Ts[-1])
    ln1 = getLambda(Ts[-2])

    pn1_2 = (p(Ts[-1]) + p(Ts[-2])) / 2

    KN = -cappa(ln, ln1) + h2 / 8 * pn1_2
    MN = cappa(ln, ln1) + h2 / 8 * pn1_2 + h2 / 4 * p(Ts[-1]) + h * a
    PN = h * a * T0 + h2 / 4 * (3 * f(Ts[-1]) + f(Ts[-2])) / 2

    return KN, MN, PN

# алгоритм прогонки
def thomas_algorithm(A, B, C, D, K0, M0, P0, KN, MN, PN):
    n = len(A)
    xi = [- M0 / K0] + [None] * n
    eta = [P0 / K0] + [None] * n

    for i in range(n):
        znam = B[i] - A[i] * xi[i]
        xi[i + 1] = C[i] / znam
        eta[i + 1] = (D[i] + A[i] * eta[i]) / znam

    y = np.empty((n+2,), dtype=float)
    y[n + 1] = (PN - KN * eta[n]) / (MN + KN * xi[n])
    for i in range(n, -1, -1):
        y[i] = xi[i] * y[i + 1] + eta[i]

    return y

def solver(alpha, ys0, eps1, eps2, maxIter):
    i = 0
    ys = solve(ys0)
    while np.max(abs((f1(ys) - f2(ys)) / f1(ys))) > eps2 and np.max(abs((ys -
ys0) / ys)) > eps1 and i < maxIter:
        ys0 = ys
        ys = (1 - alpha) * ys0 + alpha * solve(ys0)
        i += 1
    print(f"eps1: {np.max(abs((ys - ys0) / ys))}")
    print(f"eps2: {np.max(abs((f1(ys) - f2(ys)) / f1(ys)))}")

    return ys

F0 = 100
T0 = 300
l = 0.2
h = 1e-3
N = int(l / h)

koef_np = 1.4
a = 0.05
sigma = 5.668e-12
```

```
T = np.array([400] * (N + 1), dtype=float)
T = solver(0.25, T, 1e-5, 1e-3, 100)

x = np.arange(0, l + h, h)

plt.plot(x, T, label='$T_{3}$')

plt.ylabel('T, K')
plt.xlabel('x, см')
plt.legend()
plt.show()
```