



**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 ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

Студент _____ Соколов Ефим _____
Группа _____ ИУ7-63Б _____
Дисциплина _____ Моделирование _____

Преподаватель:

_____ Градов В.М.
подпись, дата Фамилия, И.О.

Оценка _____

Москва — 2021 г.

Введение

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

Задача

Задана математическая модель. Уравнение для функции $T(x)$

$$\frac{d}{dx} \left(\gamma(x) \frac{dT}{dx} \right) - 4k(T) n_p^2 * \sigma * (T^4 - T_0^4) = 0 \quad (1)$$

Краевые условия

$$\begin{cases} x=0, -\gamma(T(0)) \frac{dT}{dx} = F_0, \\ x=l, -\gamma(T(l)) \frac{dT}{dx} = \alpha(T(l) - T_0) \end{cases} \quad (2)$$

2. Функции $\gamma(T)$, $k(T)$ заданы таблицей (3,4)

T, K	$\lambda, \text{Вт}/(\text{см K})$		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	$2.0 \cdot 10^{-1}$

Разностная схема с разностным краевым условием при $x = 0$ была получена в лекции №7:

$$A_n y_{n+1} - B_n y_n + C_n y_{n+1} = -D_n, 1 \leq n \leq N-1, \quad (5)$$

где

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

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

$$B_n = A_n + C_n + p_n h$$

$$D_n = f_n h$$

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

Для величин $X_{n+\frac{1}{2}}$ можно получить различные приближенные выражения, численно вычисляя интеграл методом трапеций или методом средних.

Для вычислений будет использоваться метод средних:

$$X_{n\pm\frac{1}{2}} = \frac{k_n + k_{n\pm 1}}{2} \quad (6)$$

Разностный аналог краевого условия при $x=0$:

$$\left(X_{\frac{1}{2}} + \frac{h^2}{8} p_{\frac{1}{2}} + \frac{h^2}{4} p_0\right) y_0 - \left(X_{\frac{1}{2}} - \frac{h^2}{8} p_{\frac{1}{2}}\right) y_1 = h F_0 + \frac{h^2}{4} (f_{\frac{1}{2}} + f_0) \quad (7)$$

Простая аппроксимация:

$$p_{\frac{1}{2}} = \frac{p_0 + p_1}{2}$$

$$f_{\frac{1}{2}} = \frac{f_0 + f_1}{2} \quad (8,9)$$

При выполнении лабораторной необходимо учесть, что

$$F_N = a_N (y_N - T_0)$$

$$F_{N-\frac{1}{2}} = X_{N-\frac{1}{2}} \frac{y_{N-1} - y_N}{h} \quad (10,11)$$

4. Значения параметров для отладки (все размерности согласованы)

$n_p = 1.4$ – коэффициент преломления,

$l = 0.2$ см – толщина слоя,

$T_0 = 300\text{K}$ – 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.

$$\max \left| \frac{y_n^s - y_n^{s-1}}{y_n^s} \right| \leq \varepsilon_1, \text{ для всех } n = 0, 1, \dots, N.$$

и

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

где

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

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 $\gamma(T)$, $k(T)$ являются, соответственно, коэффициентами теплопроводности и оптического поглощения материала стенки.

Листинг

```
import matplotlib.pyplot as plt
import numpy as np
from scipy import integrate
from scipy.interpolate import InterpolatedUnivariateSpline
from decimal import *

def my_alpha(x):
    return interpolate(x, T_lambda_table[0], T_lambda_table[1])

def my_k(x):
    return interpolate(x, T_k_table[1], T_k_table[0])

def p(x):
    return (4 * (my_k(x)**4) * n_p * n_p * sigma)*x

def f(x):
    return 4 * x * n_p * n_p * sigma * T0

def A(n):
    return approc_plus_half(my_alpha, n) / h

def C(n):
    return approc_minus_half(my_alpha, n) / h
```

```
def B(n):
    return A(n) + C(n) + p(n) * h
```

```
def D(n):
    return f(n) * h
```

```
def approc_plus_half(func, n):
    return (func(n) + func(n + h)) / 2
```

```
def approc_minus_half(func, n):
    return (func(n - h) + func(n)) / 2
```

```
def left():
    k0 = approc_plus_half(my_k, 0) + (h * h * approc_plus_half(p, 0) / 8) + ((h * h *
p(0)) / 4)
    M0 = -approc_plus_half(my_k, 0) + (h * h * approc_plus_half(p, 0) / 8)
    P0 = h * F0 + (h * h / 4) * (approc_plus_half(f, 0) + f(0))
    return k0, M0, P0
```

```
def right():
    kN = (approc_minus_half(my_k, N) / h) - (h * approc_minus_half(p, N) / 8)
    MN = -my_alpha(N) - (approc_minus_half(my_k, N) / h) - (h * p(N) / 4) - (h *
approc_minus_half(p, N) / 8)
    PN = -(h / 4) * (f(N) + approc_minus_half(f, N)) - T0 * my_alpha(N)
    return kN, MN, PN
```

```
def interpolate(x, first_ar, second_ar):
    order = 1
    s = InterpolatedUnivariateSpline(first_ar, second_ar, k=order)
    return float(s(float(x)))
```

```

def find_abs_max_y(y_n, len):
    cur_max = -1e10
    for i in range(len-1):
        if (abs((y_n[i+1]-y_n[i])/y_n[i+1]) > cur_max):
            cur_max = abs((y_n[i+1]-y_n[i])/y_n[i+1])
    if ((cur_max <= eps_1) and (cur_max != -1e10)):
        return False
    print(cur_max)
    return True

```

```

def simpson(func, left, right, n):
    h = (right - left) / n
    ans = h / 3
    even = 0.0
    odd = 0.0
    for i in range(1, n):
        if i % 2 == 0:
            even += func(left + h * i)
        else:
            odd += func(left + h * i)
    ans *= (2 * even + 4 * odd + func(left) + func(right))
    return ans

```

```

def f_2():
    func = lambda x: x*(my_k(x)**4 - T0**4)
    integral = simpson(func, 0, 1, 41)
    return 4 * n_p * n_p * sigma * integral

```

```

def f_1():
    return F0 - alpha * (2400 - T0)

```

```

def find_abs_max_f(s):
    cur_max = -1e10
    for i in range(1, s):
        if (abs((f_1() ** i - f_2() ** i) / f_1() ** i) > cur_max):
            cur_max = abs((f_1() ** i - f_2() ** i) / f_1() ** i)
    if ((cur_max <= eps_2) and (cur_max != -1e10)):
        return False
    return True

```



```

if __name__ == "__main__":

    h = 1e-4
    N = 0.2
    eps_1 = 1e-2
    eps_2 = 2e-2

    n_p = 1.4
    l = 0.2
    T0 = 300
    sigma = 5.668 * 1e-12
    F0 = 100
    alpha = 0.05
    T_lambda_table = [[300, 500, 800, 1100, 2000, 2400],
                       [1.36 * 1e-2, 1.63 * 1e-2, 1.81 * 1e-2, 1.98 * 1e-2, 2.5 * 1e-2,
2.74 * 1e-2]]
    T_k_table = [[293, 1278, 1528, 1677, 2000, 2400], [2 * 1e-2, 5 * 1e-2, 7.8 * 1e-2,
1 * 1e-1, 1.3 * 1e-1, 2 * 1e-1]]

    cnt = 0
    y_n = [0]*200
    x = h
    n = 1
    t = [0] * (n + 1)
    while ((cnt < 200) and (find_abs_max_y(y_n, cnt)) and (find_abs_max_f(cnt))):
        k0, M0, P0 = left()
        kN, MN, PN = right()

        eps = [0, -M0 / k0]
        eta = [0, P0 / k0]

        x = h
        n = 1
        while x + h < N:

```

```

    eps.append((C(x) / (B(x) - A(x) * eps[n])))
    eta.append((A(x) * eta[n] + D(x)) / (B(x) - A(x) * eps[n]))
    n += 1
    x += h

```

```

t = [0] * (n + 1)

```

```

t[n] = (PN - MN * eta[n]) / (kN + MN * eps[n])

```

```

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

```

```

y_n[cnt] = t[n]
cnt += 1
h = h / 2

```

```

x = [i for i in np.arange(0, N, h)]

```

```

plt.plot(x, t[:-1])
plt.xlabel("x, cm")
plt.ylabel("temperature, K")
plt.grid()
plt.show()

```

Результаты работы

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

$$F = -\gamma(T) \frac{dT}{dx}$$

$$p(x) = 4k(T)n_p^2 \sigma$$

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

Тогда (11) можно записать как

$$-\frac{d}{dx}(F) - p(x) + f(x) = 0 \quad (12)$$

Проинтегрируем (12) на отрезке $[x_{N-\frac{1}{2}}; x_N]$:

$$-\int_{x_{N-\frac{1}{2}}}^{x_N} \frac{dF}{dx} dx - \int_{x_{N-\frac{1}{2}}}^{x_N} p(x) dx + \int_{x_{N-\frac{1}{2}}}^{x_N} f(x) dx = 0 \quad (13)$$

Второй и третий интегралы вычислим методом трапеций:

$$-(F_N - F_{N-\frac{1}{2}}) - \frac{h}{4} \left(p_N y_N^4 + p_{N-\frac{1}{2}} y_{N-\frac{1}{2}}^4 \right) + \frac{h}{4} (f_N + f_{N-\frac{1}{2}}) = 0 \quad (14)$$

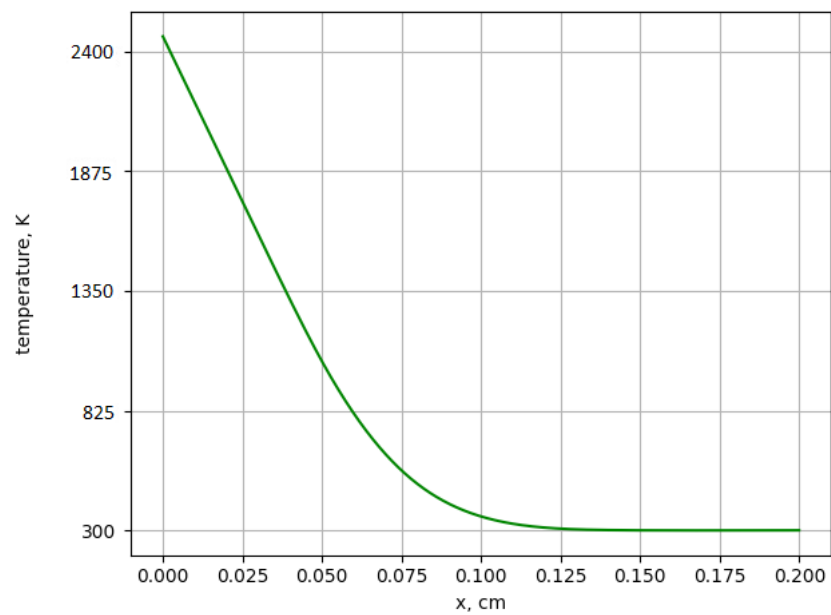
Применив (10, 11) к (14) получим:

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

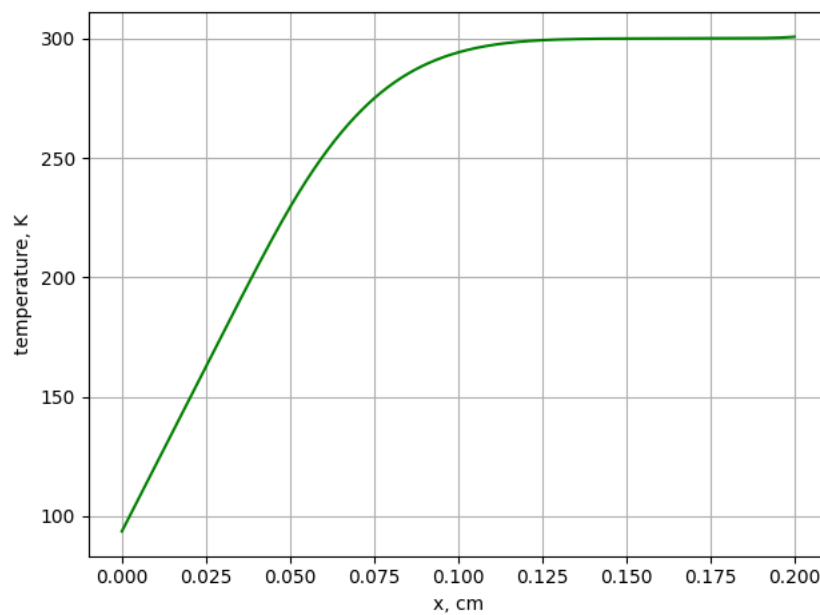
$$X_{N-\frac{1}{2}} \frac{y_{N-1} - y_N}{h} - \frac{h}{4} p_N y_N - \frac{h}{4} p_{N-\frac{1}{2}} \frac{y_N^4 + y_{N-1}^4}{2} = \frac{-h}{4} (f_N + f_{N-\frac{1}{2}}) + a_N (y_N - T_0)$$

$$y_{N-1} \left(\frac{X_{N-\frac{1}{2}}}{h} - \frac{h}{8} p_{N-\frac{1}{2}} y_{N-1}^3 \right) + y_N \left(-a_N - \frac{X_{N-\frac{1}{2}}}{h} - \frac{h}{4} p_N - \frac{h}{8} p_{N-\frac{1}{2}} y_N^3 \right) = \frac{-h}{4} (f_N + f_{N-\frac{1}{2}}) - T_0 a_N$$

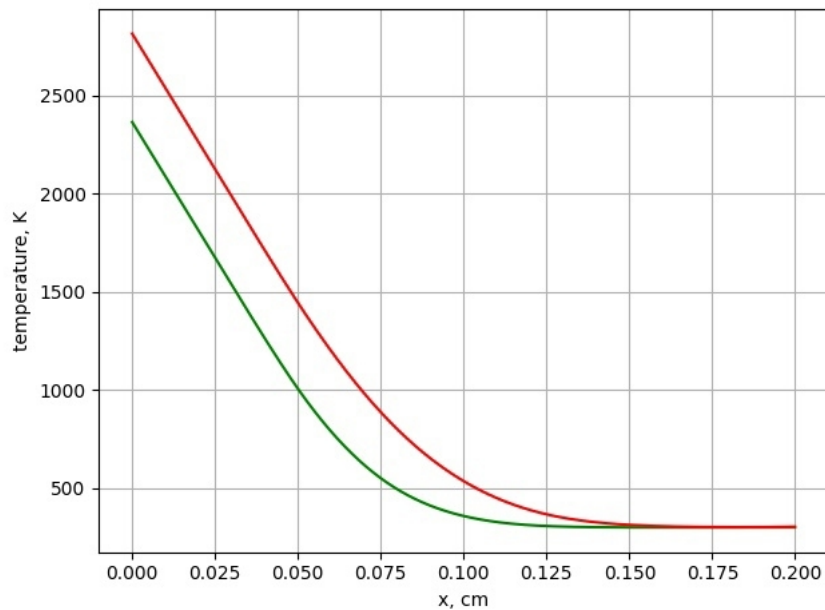
2. График зависимости температуры $T(x)$ от координаты x при заданных выше параметрах.



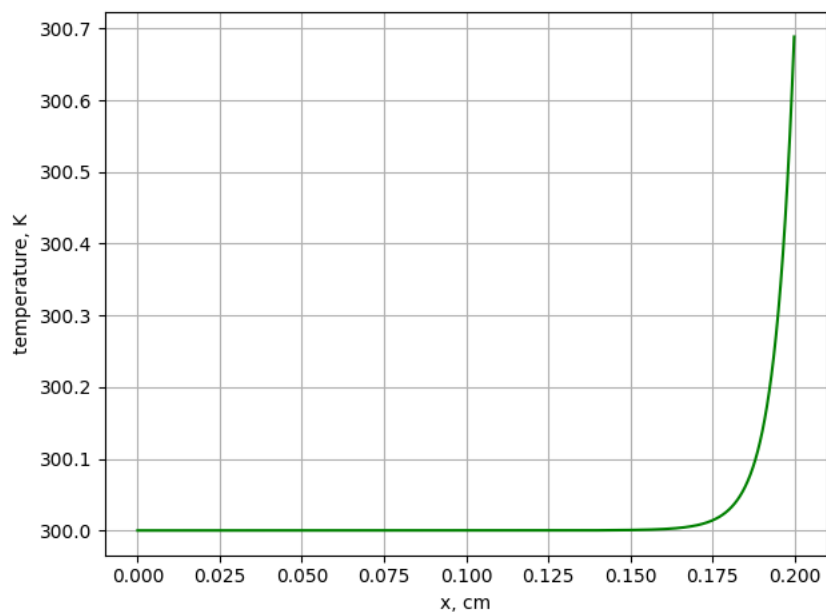
3. График зависимости $T(x)$ при $F_0 = -10$ Вт/см²



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



5. График зависимости $T(x)$ при $F_0 = 0$



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

Точность выхода ϵ_{rs_1} (по температуре) = 0.07

Точность выхода ϵ_{rs_2} (по балансу энергии) = 1.14

Ответы на контрольные вопросы

1. Какие способы тестирования программы можно предложить?

Для тестирования можно изменять значение параметра F_0 :

- $F_0 = 0$: отсутствует нагревание, температура стержня по всей его длине должна равняться температуре среды;
- $F_0 < 0$: происходит съём тепла, температура должна возрасти до температуры среды.

Также можно увеличить коэффициент теплоотдачи в N раз: стержень будет отдавать больше тепла, следовательно, он быстрее будет охлаждаться.

2. Получите простейший разностный аналог нелинейного краевого условия

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

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

1. Аппроксимируем производную:

$$\frac{dT}{dx} = \frac{y_N - y_{N-1}}{h}$$

.

2. Подставим в исходное уравнение:

$$-k(l) \frac{y_N - y_{N-1}}{h} = \alpha_N(y_N - T_0) + \varphi(y_N)$$

3. Домножим на h , произведём замену $y_{N-1} = \xi_N y_N + \eta_N$, получим:

$$-k(l)(y_N - \xi_N y_N + \eta_N) = \alpha_N(y_N - T_0)h + \varphi(y_N)h$$

4. Получим уравнение относительно y_N :

$$\varphi(y_N)h + y_N(k_N + \alpha_N h - k_N \xi_N - k_N \eta_N) - \alpha_N T_0 h = 0$$

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.

1. Используя рассчитанные на лекции коэффициенты M_0, P_0, K_0 , получим ξ_1 и η_1 :

$$\xi_1 = -\frac{M_0}{P_0}, \eta_1 = -\frac{K_0}{P_0}.$$

Найдем последующие прогоночные коэффициенты:

$$\xi_{n+1} = \frac{C_n}{B_n - A_n \xi_n}, \eta_{n+1} = \frac{F_n + A_n \eta_n}{B_n - A_n \xi_n}.$$

2. Решив полученное уравнение в п.2, получим значение y_N (решить можно, например, методом дихотомии).

3. Найдем все неизвестные значения y_n по прогоночной формуле:

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

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). Оба краевых условия линейные.

1. Начальные прогоночные коэффициенты для правой прогонки:

$$\xi_1 = -\frac{M_0}{P_0}, \eta_1 = -\frac{K_0}{P_0}.$$

2. Начальные прогоночные коэффициенты для левой прогонки:

$$\alpha_{N-1} = -\frac{M_N}{K_N}, \beta_{N-1} = -\frac{P_N}{K_N}.$$

3. Прогоночные коэффициенты правой прогонки (от 1 до $p-1$):

$$\xi_{n+1} = \frac{C_n}{B_n - A_n \xi_n}, \eta_{n+1} = \frac{F_n + A_n \eta_n}{B_n - A_n \xi_n}.$$

4. Прогоночные коэффициенты левой прогонки (от p до $N-1$):

$$\alpha_{n-1} = \frac{A_n}{B_n - C_n \alpha_n}, \beta_{n-1} = \frac{F_n + C_n \beta_n}{B_n - C_n \alpha_n}.$$

5. Составим систему уравнений из правой и левой прогоночной формулы и выразим y_p :

$$y_p = \frac{\xi_{n+1} \beta_n + \eta_{n+1}}{1 - \xi_{n+1} \alpha_n}.$$