



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

По курсу: "Моделирование"

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

Группа ИУ7-63Б

Студент Сукочева А.

Преподаватель Градов В.М.

0.1 Постановка задачи

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

0.1.1 Исходные данные

Задана математическая модель:

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

Краевые условия:

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

Функции $\lambda(T)$ и $k(T)$ заданы таблицей.

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

Рис. 1: Таблица функций $\lambda(T)$ и $k(T)$

Заданы начальные параметры:

$n_p = 1.4$ - коэффициент преломления

$l = 0.2$ см - толщина слоя

$T_0 = 300\text{К}$ - температура окружающей среды

$\sigma = 5.668 \cdot 10^{-12}$ Вт / ($\text{см}^2 \cdot \text{К}^4$) - постоянная Стефана - Больцмана

$F_0 = 100$ Вт / см^2 - поток тепла

$\alpha = 0.05$ Вт / ($\text{см}^2 \cdot \text{К}$) - коэффициент теплоотдачи

Выход из итераций организовать по температуре и по балансу энергии:

$$\max \left| \frac{y_n^s - y_n^{s-1}}{y_n^s} \right| \leq \varepsilon_1$$

для всех $n = 0, 1, \dots, N$. и

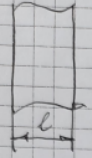
$$\max \left| \frac{f_1^s - f_2^s}{f_1^s} \right| \leq \varepsilon_1$$

где

$$f_1 = F_0 - \alpha(T(l) - T_0)$$

$$f_2 = 4n_p^2 \sigma \int_0^1 k(T(x))(T^4(x) - T_0^4) dx$$

Лабораторная работа №3 2021г.
Моделирование



$$\begin{cases} \frac{d}{dx} (\lambda(T) \frac{dT}{dx}) - 4k(T) \cdot n_p^2 \cdot \sigma (T^4 - T_0^4) = 0 \\ x=0, -\lambda(0) \frac{dT}{dx} = F_0, \\ x=l, -\lambda(l) \frac{dT}{dx} = \lambda(T(l) - T_0). \end{cases}$$

Вспомогательное к уравнению (1) из лекции №8.
Видим, что $p(x, u) = 0$, $f(x, u) = f(u) = -4k(T) n_p^2 \sigma (T^4 - T_0^4)$

Разностная схема:

$$\begin{cases} A_n y_{n-1} - B_n y_n + C_n y_{n+1} = -G_n \\ K_0 y_0 + M_0 y_1 = P_0 \\ K_N y_{N-1} + M_N y_N = P_N \end{cases}$$

$$A_n = \frac{\lambda_{n-1/2}}{h} = \frac{\lambda_{n-1} + \lambda_n}{2h},$$

$$C_n = \frac{\lambda_{n+1/2}}{h} = \frac{\lambda_n + \lambda_{n+1}}{2h},$$

$$B_n = A_n + C_n,$$

$$G_n = -4k_n \cdot n_p^2 \sigma (y_n^4 - T_0^4) h$$

$$K_0 = \lambda_{1/2} = \frac{\lambda_0 + \lambda_1}{2},$$

$$M_0 = -K_0,$$

$$P_0 = h F_0 + \frac{h^2}{2} (f_{1/2} + f_0),$$

$$K_N = -\lambda_N,$$

$$M_N = \lambda_N + \lambda_N$$

$$P_N = \lambda_N T_0 h$$

Рис. 2: Правильное выделение функции $p(x)$ и $f(x)$.

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. Зависимость от температуры излучательной способности материала очень резкая. При низких температурах стенка излучает очень слабо. Функции $\lambda(T), k(T)$ являются, соответственно, коэффициентами теплопроводности и оптического поглощения материала стенки

0.2 Реализация

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

class Params:
    F0 = 100
    T0 = 300
    l = 0.2
    h = 1e-3
    np_coef = 1.4
    alpha = 0.05
    sigma = 5.668e-12
    tl_tab = ((300, 500, 800, 1100, 2000, 2400),
              (1.36e-2, 1.63e-2, 1.81e-2, 1.98e-2, 2.50e-2, 2.74e-2))
    tk_tab = ((293, 1278, 1528, 1677, 2000, 2400),
              (2.0e-2, 5.0e-2, 7.8e-2, 1.0e-1, 1.3e-1, 2.0e-1))

params = Params()
get_lambda = InterpolatedUnivariateSpline(params.tl_tab[0], params.tl_tab[1],
                                           k=1)
get_k = InterpolatedUnivariateSpline(params.tk_tab[0], params.tk_tab[1], k=1)

def p(T):
    return 0

def f(T):
    return -4 * params.np_coef**2 * params.sigma * get_k(T) * \
           (pow(T, 4)-pow(params.T0, 4))

def LeftConditions(ts):
    h2 = params.h * params.h
    lm = (get_lambda(ts[0]) + get_lambda(ts[1])) / 2
    fm = (f(ts[0]) + f(ts[1])) / 2
    K0 = lm
    M0 = -lm
    P0 = params.h * params.F0 + (h2 / 4) * (f(ts[0]) + fm)
    return K0, M0, P0

def RightConditions(ts):
    KN = get_lambda(ts[-1])
    MN = params.alpha * params.h + get_lambda(ts[-1])
    PN = params.alpha * params.T0 * params.h
    return KN, MN, PN

def RunMethod(A, B, C, D, K0, M0, P0, KN, MN, PN):
    n = len(A)
    xi = [-M0/K0]
    eta = [P0/K0]
    for i in range(len(A)):
        x = C[i]/(B[i]-A[i] * xi[i])
        e = (D[i]+A[i] * eta[i])/(B[i]-A[i] * xi[i])
        xi.append(x)
        eta.append(e)
    y = [(PN-KN * eta[n])/(MN+KN * xi[n])]
    for i in range(len(A), -1, -1):
        y_i = xi[i] * y[0]+eta[i]
        y.insert(0, y_i)
    return np.array(y)
```

```

def CalcCoefficients(ts):
    K0, M0, P0 = LeftConditions(ts)
    KN, MN, PN = RightConditions(ts)
    N = int(params.l/params.h)
    A, B, C, D = [0]*(N-1), [0]*(N-1), [0]*(N-1), [0]*(N-1)
    for n in range(1, N):
        lam_n_m_1 = get_lambda(ts[n-1])
        lam_n = get_lambda(ts[n])
        lam_n_p_1 = get_lambda(ts[n+1])
        A[n-1] = (lam_n_m_1+lam_n)/2/params.h
        C[n-1] = (lam_n+lam_n_p_1)/2/params.h
        B[n-1] = A[n-1]+C[n-1]+p(ts[n]) * params.h
        D[n-1] = f(ts[n]) * params.h
    return A, B, C, D, K0, M0, P0, KN, MN, PN

def f1(T):
    return params.F0-params.alpha * (T[-1]-params.T0)

def f2 (ts):
    return 4 * params.np_coef**2 * params.sigma *
        integrate.trapezoid([get_k(ti) * \
                               (pow(ti, 4) - pow(params.T0, 4)) for ti in
                               ts], np.arange(0, params.l + params.h,
                               params.h))

def SimpleIterations(alpha, eps1, eps2, maxIter):
    N = int(params.l/params.h)
    cnt = 0
    T_sp = np.array([params.T0] * (N+1), dtype=float)
    A, B, C, D, K0, M0, P0, KN, MN, PN = CalcCoefficients(T_sp)
    T_s = RunMethod(A, B, C, D, K0, M0, P0, KN, MN, PN)
    while np.max(abs((f1(T_s)-f2(T_s))/f1(T_s))) > eps2 and
           np.max(abs((T_s-T_sp)/T_s)) > eps1 and cnt < maxIter:
        T_sp = T_s
        A, B, C, D, K0, M0, P0, KN, MN, PN = CalcCoefficients(T_sp)
        new_Ts_norelax = RunMethod(A, B, C, D, K0, M0, P0, KN, MN, PN)
        T_s = T_sp+params.alpha * (new_Ts_norelax-T_sp)
        cnt += 1
    return T_s

def SaveImg(xs, ys, name_x, name_y, file_name):
    fig, ax = plt.subplots()
    ax.plot(xs, ys)
    ax.grid()
    ax.set_xlabel(name_x)
    ax.set_ylabel(name_y)

    # params.alpha *= 3
    # x = np.arange(0, params.l+params.h, params.h)
    # T = SimpleIterations(0.25, 1e-5, 1e-3, 100)
    # ax.plot(x, T)

    plt.savefig(file_name, bbox_inches="tight") # color = 'green'

def main():
    x = np.arange(0, params.l+params.h, params.h)
    T = SimpleIterations(0.25, 1e-5, 1e-3, 100)
    SaveImg(x, T, '1', '2', '1.png')

if __name__ == "__main__":
    main()

```

0.3 Экспериментальная часть

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

№ 3.

$$\begin{cases} \frac{d}{dx}(\lambda(x) \frac{dT}{dx}) - 4k(T)h_p^2 \delta(T^4 - T_0^4) = 0 \\ x=0, -\lambda(T(0)) \frac{dT}{dx} = F_0 \\ x=l, -\lambda(T(l)) \frac{dT}{dx} = \alpha(T(l) - T_0) \end{cases}$$

Приведем к виду:

$$\frac{d}{dx}(k(x, u) \frac{du}{dx}) - p(x, u)u + f(x, u) = 0$$

где $p(x, u) = 0$
 $f(x, u) = f(u) = -4k(T)h_p^2 \delta(T^4 - T_0^4)$

Обозначим: $F = -\lambda(x) \frac{dT}{dx}$

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

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

Умножив:

$$F_N = \alpha u_N (u_N - T_0) \quad F_{N-1/2} = X_{N-1/2} \frac{u_{N-1} - u_N}{h}$$

Подставим:

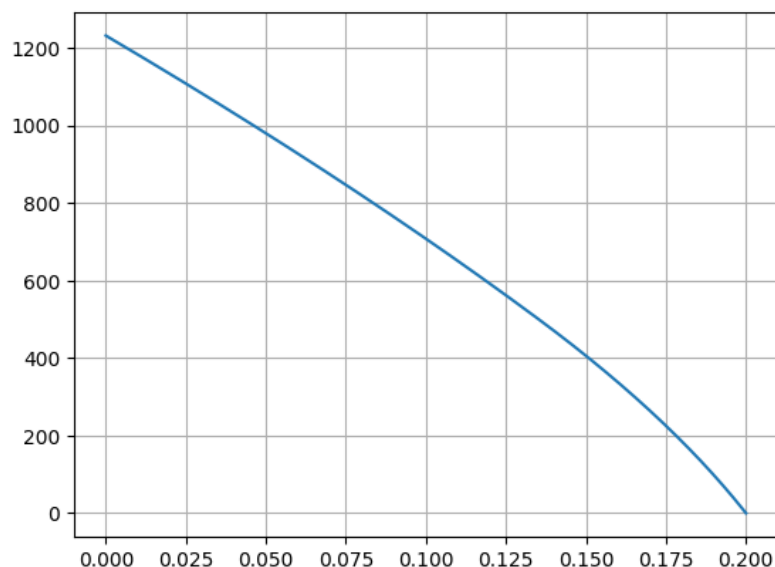
$$\alpha u_N (u_N - T_0) - \frac{h}{4} (f_N + f_{N-1/2}) = X_{N-1/2} \frac{u_{N-1} - u_N}{h}$$

$$-T_0 \alpha u_N - \frac{h}{4} (f_N + f_{N-1/2}) = \frac{u_{N-1}}{h} (X_{N-1/2} - \alpha u_N) + \alpha u_N \frac{X_{N-1/2}}{h}$$

Полученное краевое условие:

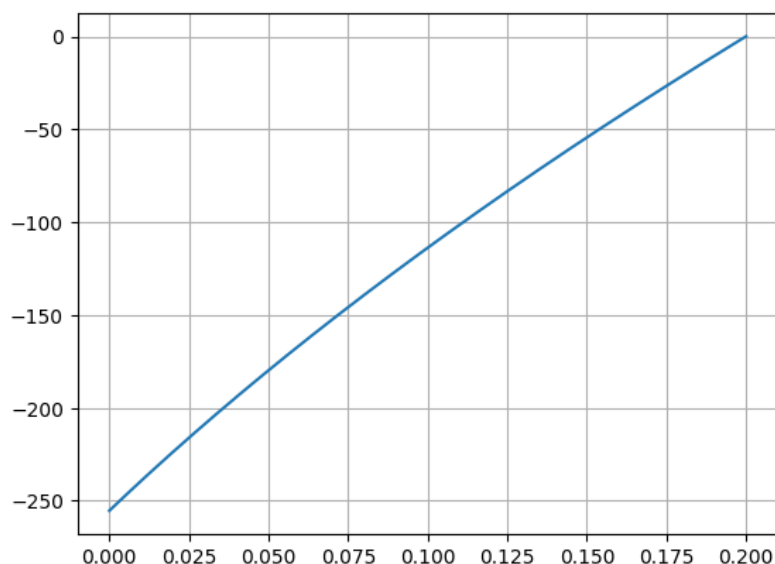
$$K_N u_{N-1} + M_N u_N = P_N$$

0.3.2 2. График зависимости температуры $T(x)$ 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.



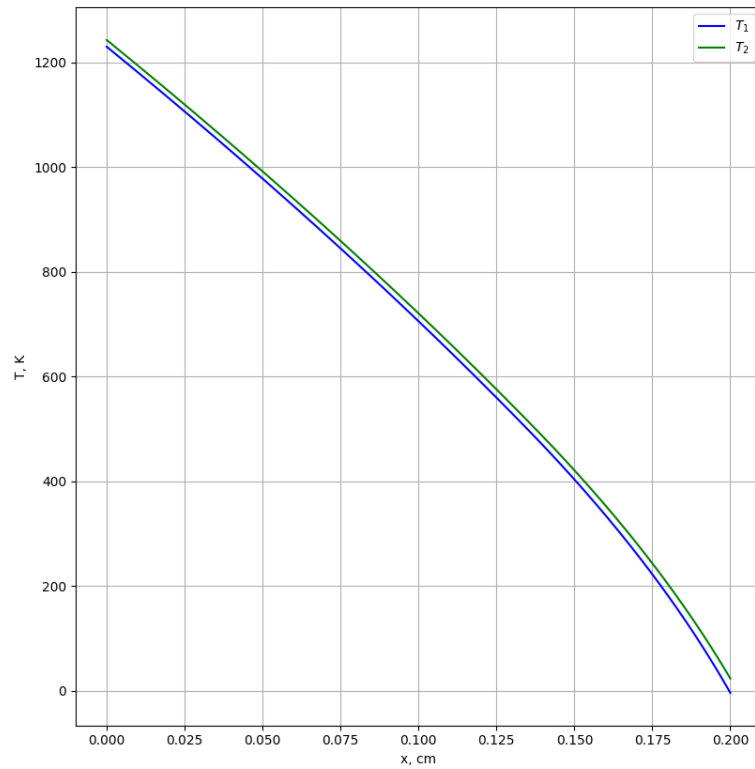
При увеличении T_0 общий вид графика не меняется. Увеличивается количество итераций.

0.3.3 3. График зависимости $T(x)$ при $F_0 = -10 \frac{B}{cm^2}$.

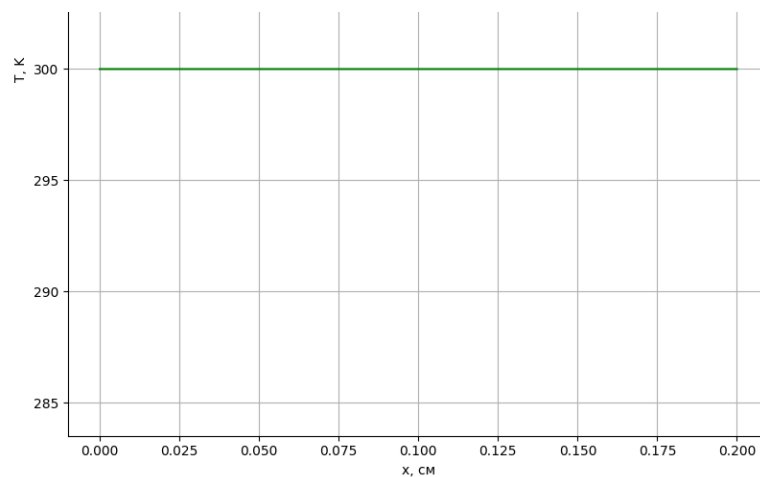


0.3.4 4. График зависимости $T(x)$ при увеличенных значениях α (например, в 3 раза). Сравнить с п. 2.

Синяя кривая при увеличенном в 3 раза коэффициенте теплоотдачи.



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



0.3.6 6. Для указанного в задании исходного набора параметров привести данные по балансу энергии.

- Точность выхода $\varepsilon_1 = 1e-5$ (по температуре)
- Точность выхода $\varepsilon_2 = 1e-3$ (по балансу)

0.4 Ответы на вопросы

1. Какие способы тестирования программы можно предложить?

В данной программе можно предложить тестирование при помощи изменения значения параметра F_0 . Также при $F_0 > 0$ происходит охлаждение пластины, при $F_0 < 0$ нагревание. При увеличении теплосъема и неизменном потоке F_0 уровень температур $T(x)$ должен снижаться, а градиент увеличиваться.

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

№2
Аппроксимируем производную $\left[-k(l) \frac{dT}{dx} = \alpha_0 (T(l) - T_0) + f(T)(l) \right]$

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

 Подставим в (3):

$$-k_N \frac{y_N - y_{N-1}}{h} = \alpha_0 (y_N - T_0) + \varphi(y_N)$$

 Размножим на h :

$$-k_N (y_N - y_{N-1}) = h \alpha_0 (y_N - T_0) + h \varphi(y_N)$$

 Произведем замену $y_{N-1} = \xi_N + \eta_N$

$$-k_N (y_N - \xi_N + \eta_N) = \alpha_0 (y_N - T_0) h + \varphi(y_N) h$$

 Приведем к виду:

$$f(y_N) h + (k_N + \alpha_0 h - k_N \xi_N - k_N \eta_N) y_N - h \alpha_0 T_0 = 0$$

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.

№3.
Для прямого хода найдем нач. проз. коэф. по формулам:

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

 Рекуррентные соотношения для коэф.:

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

 y_N можно получить из п.2
 По прогонной формуле можно найти все значения.
 Ищем y_n :

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

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.

4. Вычисление нач. прог. коэфф. Для правой прогонки:

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

Для левой прогонки:

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

Рек. соотнош. для коэфф. Для правой прогонки:

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

Для левой прогонки:

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

Рек. соотнош. для левой и правой прогонки:

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

$$y_n = \alpha_{n-1} y_{n-1} + \beta_{n-1}$$

Выразим y_p :

$$\begin{cases} y_{p-1} = \xi_p y_p + \eta_p \\ y_p = \alpha_{p-1} y_{p-1} + \beta_{p-1} \end{cases} \Rightarrow y_p = \frac{\xi_{n+1} \beta_n + \eta_{n+1}}{1 - \xi_{n+1} \alpha_n}$$