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»

Лабораторная работа № 3

Дисциплина: Моделирование

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

Студент: Турсунов Ж.Р.

Группа ИУ7-66Б

Оценка (баллы) _______________

Преподаватель : Градов В.М.

Москва. 2021г

I. АНАЛИТИЧЕСКАЯ ЧАСТЬ

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 $T(x,t)$

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

Краевые условия

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

2. Функция $\lambda(T)$, $k(T)$ 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 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 $\hat{F}_N = \alpha_N(\hat{y}_N - T_0)$, а $\hat{F}_{N-1/2} = \hat{\chi}_{N-1/2}\frac{\hat{y}_{N-1}-\hat{y}_N}{h}$

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,

σ=5.668 10^{-12} Вт/(см^{2}К^4)- постоянная Стефана- Больцмана,

F_0=100 Вт/см^2 - поток тепла,

$\alpha = 0.05$ Вт/(см^2 К) – коэффициент теплоотдачи.

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

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

и

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

где

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

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 равнаT0. 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.

II. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

ЯП был выбран Python3 из-за простоты работы с графиками и библиотеки matplotlib. Ниже на листингах будет представлена реализация программы:

Листинг

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math
from scipy import integrate
from scipy.interpolate import interp1d

plt.rcParams['figure.figsize']=12,8
SAVE_DIR = './'

def plot_helper(x, y, xlabel, ylabel, title=False, pic='result.png', save=True):
    plt.scatter(x, y, s=8, c='g')
    plt.plot(x,y,'-', c='r')
    plt.grid()
    plt.ylabel(ylabel)
    plt.xlabel(xlabel)
    if title: plt.title(title)
    if save: plt.savefig(SAVE_DIR + pic, bbox_inches = 'tight')
    fig = plt.figure()
```

```
n_p   = 1.4
l     = 0.2
T_0   = 300
sigma = 5.668e-12
F_0   = 100
alpha = 0.05

def k_plus_avg(n):
   return (k(n) + k(n+1))/2

def k_minus_avg(n):
   return (k(n) + k(n-1))/2

def k(n):
   return lambda_T(T(n))

def f_inner(n):
   return k(n) * (T(n)**4 - T_04)

def f(n):
   return -f_helper * f_inner(n)
```

```python
def A(n): return k_minus_avg(n) / h

def C(n): return k_plus_avg(n) / h

def B(n): return A(n) + C(n)

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


def intergrate(func, a, b, step):
    S = (func(a) + func(b)) / 2
    for x in range(a+step, b, step):
        S += func(x)
    return S * step

def edge_condition():
    K0 = k_plus_avg(0)
    M0 = -K0
    P0 = h * F_0 + h**2 / 4 * (f(1/2) + f(0))
    KN = - k_minus_avg(N)
    MN = alpha * h - KN
    PN = alpha * h * T_0 + h**2 / 4 * (f(N-1/2) + f(N))
    return K0, M0, P0, KN, MN, PN

def err_temperature(t_old, t_new):
    return max([abs(1 - t_old[i] / t_new[i]) for i in range(len(t_old))])

def err_energy_balance(t_new):
    f1 = F_0 - alpha * (t_new[N] - T_0)
    f2 = f_helper * intergrate(f_inner, 0, N, 1) * h
    return abs(1 - f2/f1)


def thomas_algorithm():
    K0, M0, P0, KN, MN, PN = edge_condition()

    # forward
    xi = [None, - M0 / K0]
    eta = [None, P0 / K0]

    for i in range(1, N):
        denominator = (B(i) - A(i) * xi[i])
        x = C(i) / denominator
        e = (D(i) + A(i) * eta[i]) / denominator
        xi.append(x)
        eta.append(e)

    # backward
    y = [(PN - KN * eta[-1]) / (MN + KN * xi[-1])]
```

```
    for i in range(N - 1, -1, -1):
        yi = xi[i + 1] *  y[0] + eta[i + 1]
        y.insert(0, yi)
    return y

def fixed_iteration(esp1, esp2, max_loop=20):
    global T_list
    err1 = err2 = 1

    for i in range(max_loop):
        if err1 <= esp1 and err2 <= esp2:
            break
        t = thomas_algorithm()
        err1 = err_temperature(T_list, t)
        T_list = t
        err2 = err_energy_balance(T_list)
        print(i, err1, err2)


h = 0.01
N = int(l // h)
esp1 = 0.01
esp2 = 0.01

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

# F = 100, alpha = 0.05
T_list = [T_0] * (N + 1)
F_0   = 100
alpha = 0.05

fixed_iteration(esp1, esp2)
plot_helper(x, T_list, 'x, cm', 'T, K', pic=1.png')

# F = -10
T_list = [T_0] * (N + 1)
F_0 = -10

fixed_iteration(esp1, esp2)
plot_helper(x, T_list, 'x, cm', 'T, K', 'F0 = -10', pic='2.png')

# alpha x3
T_list = [T_0] * (N + 1)
F_0 = 100
alpha = 0.15

fixed_iteration(esp1, esp2)
plot_helper(x, T_list, 'x, cm', 'T, K', 'alpha = 0.15 (x3)', pic='3.png')

# F = 0
T_list = [T_0] * (N + 1)
```

```
F_0 = 0
alpha = 0.05

fixed_iteration(esp1, esp2)
plot_helper(x, T_list, 'x, cm', 'T, K', 'F0 = 0', pic='4.png')
```

III. 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 $F = -k(x)\dfrac{du}{dx}$.

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

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

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

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

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

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

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

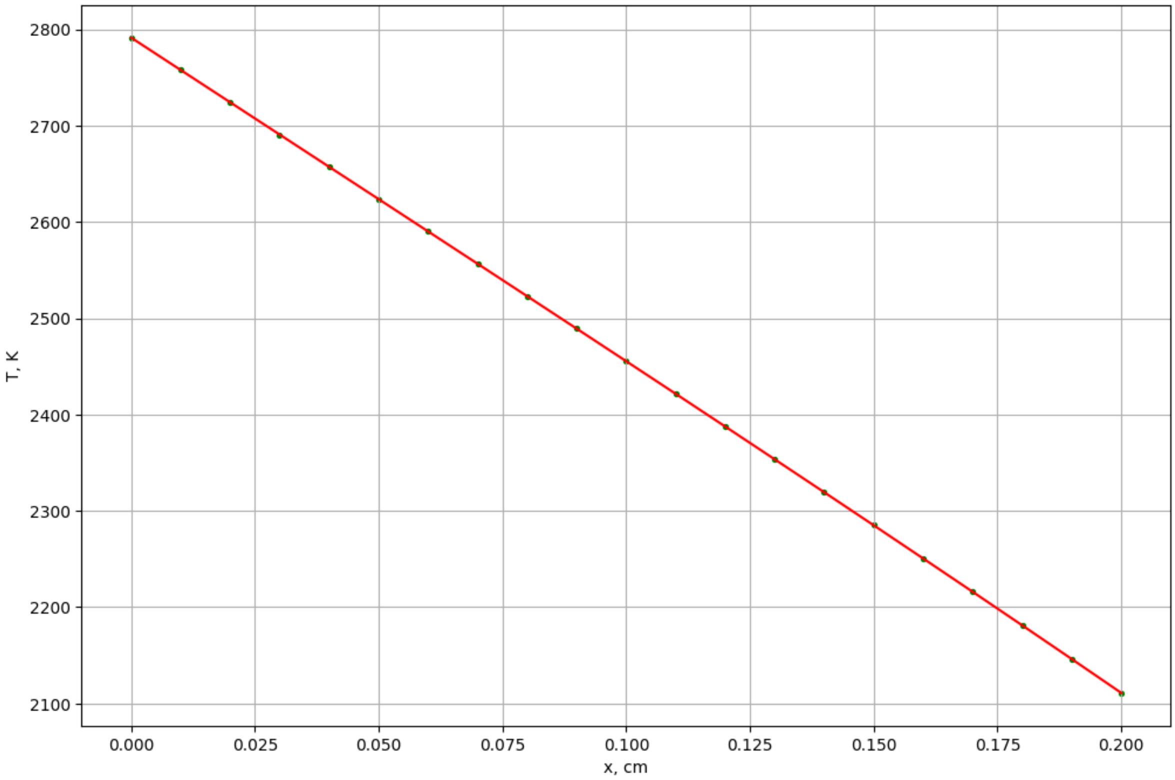
$$K_N = \frac{h^2}{8}p_{N-1/2} - \chi_{N-1/2}, \quad M_N = \alpha h + \chi_{N-1/2} + \frac{h^2}{8}p_{N-1/2} + \frac{h^2}{4}p_N, \quad P_N = \alpha\beta h + \frac{h^2}{4}(f_{N-1/2} + f_N)$$

$$\chi_{n\pm\frac{1}{2}} = \frac{\lambda_n + \lambda_{n\pm1}}{2}, \quad f(x) = -4\cdot k(T)\cdot n_p^2\cdot\sigma\cdot(T^4 - T_0^4)$$

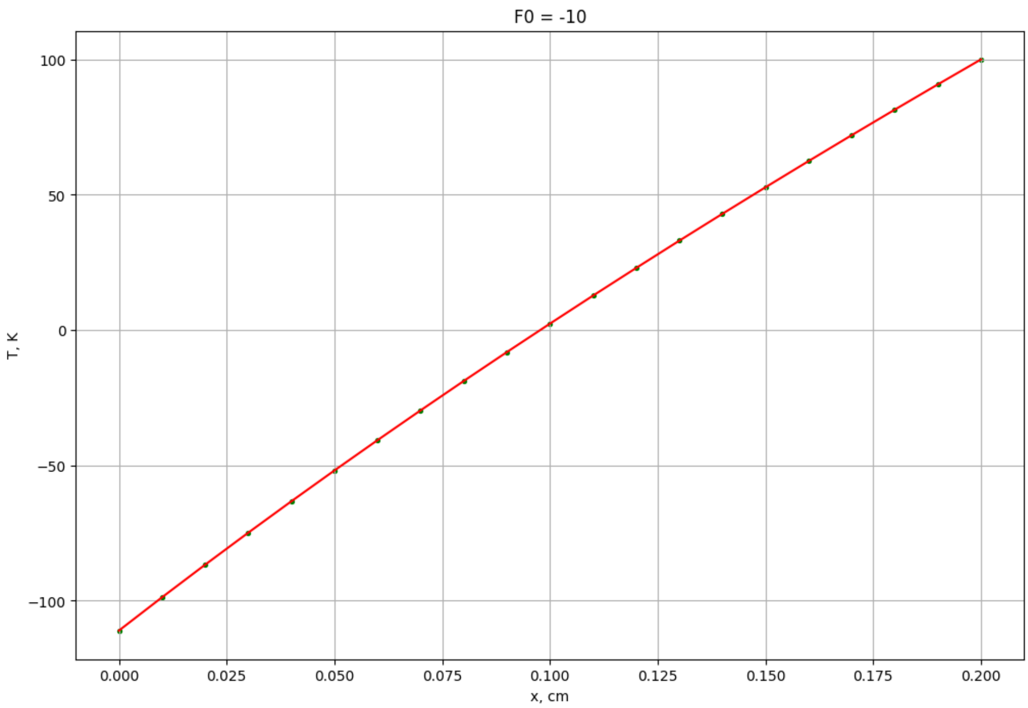
$$K_N = -\chi_{N-1/2}, \quad M_N = \alpha h + \chi_{N-1/2}, \quad P_N = \alpha T_0 h + \frac{h^2}{4}(f_{N-1/2} + f_N)$$

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= 0.01см и $\epsilon_1 = \epsilon_2$ = 0.01 требуется 10 итераций.

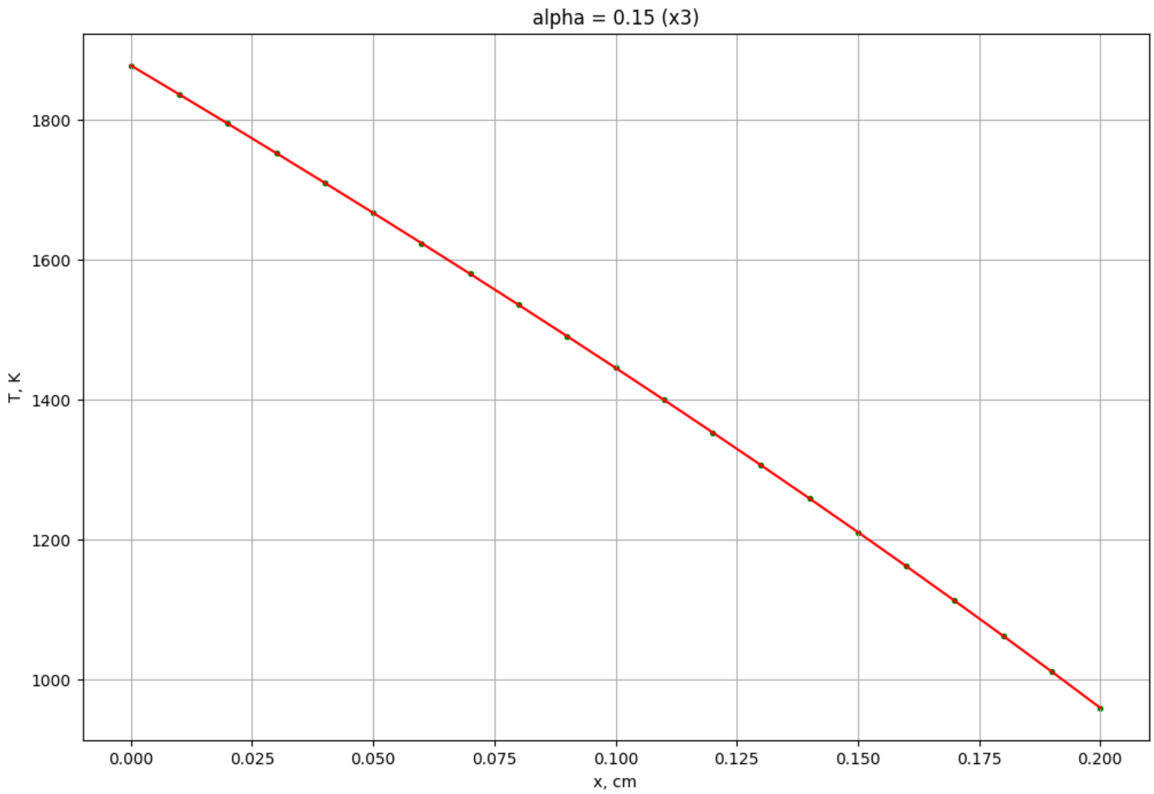


3. График зависимости T(x) при F0 = −10 Вт/см2.



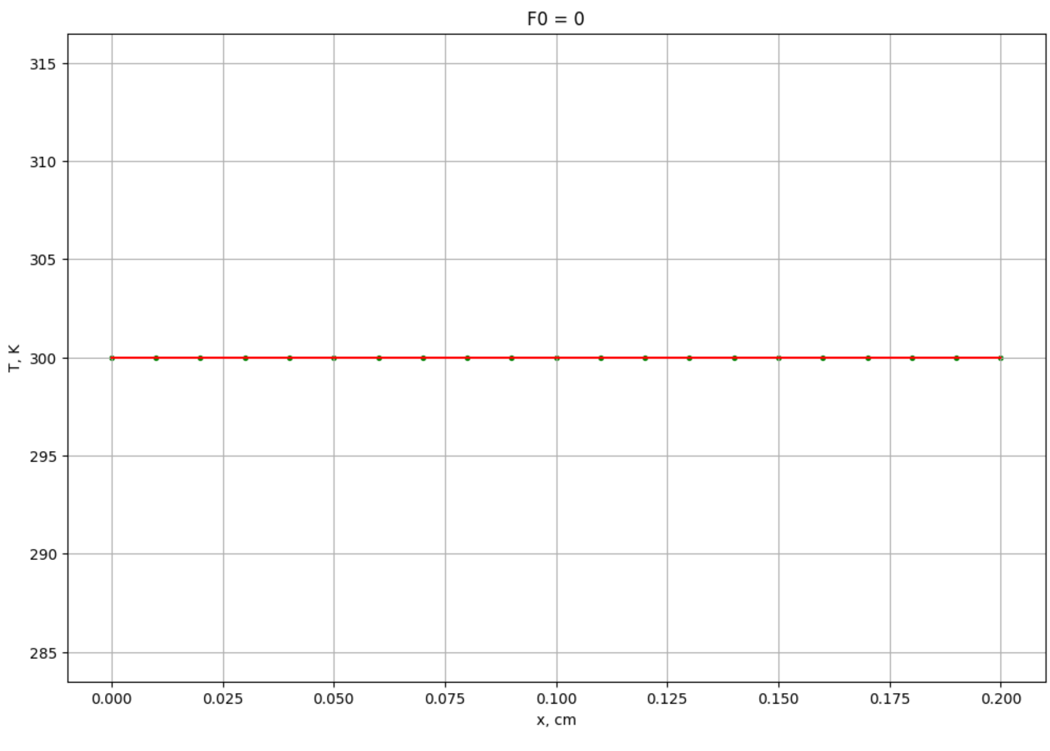
4. График зависимости T(x) при увеличенных значениях α (например, в 3 раза). Сравнить с п.2.

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 T0 (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

$f_1 = F_0 - \alpha(T(l) - T_0)$ 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, $max\left|\dfrac{y_n^s - y_n^{s-1}}{y_n^s}\right|$, и $\left|\dfrac{f_1^s - f_2^s}{f_1^s}\right|$,

```
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
```

IV. ОТВЕТЫ НА ВОПРОСЫ

1. Какие способы тестирования программы можно предложить?

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

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

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

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

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

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.

$$\begin{cases} x = 0, \quad -k(0)\dfrac{\mathrm{d}T}{\mathrm{d}x} = F_0, \\ x = l, \quad -k(l)\dfrac{\mathrm{d}T}{\mathrm{d}x} = \alpha_N(T(l) - T_0) + \varphi(T). \end{cases}$$

Будем использовать левую прогонку, основная прогоночная формула:

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

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

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

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

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