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»

Дисциплина: «Моделирование»

Лабораторная работа №3

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

Студент: Левушкин И. К.
Группа: ИУ7-62Б
Преподаватель: Градов В. М.

Москва, 2020 г.

Цель работы

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

Исходные данные

1. Задана математическая модель.

   Уравнение для функции $T(x)$

   $$\frac{d}{dx}\left(k(x)\frac{dT}{dx}\right) - \frac{2}{R}\alpha(x)T + \frac{2T_0}{R}\alpha(x) = 0 \tag{1}$$

   Краевые условия

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

2. Функции $k(x), \alpha(x)$ заданы своими константами

   $$k(x) = \frac{a}{x-b},$$
   $$\alpha(x) = \frac{c}{x-d}$$

   Константы a, b следует найти из условий $k(0) = k_0, k(l) = k_N$, а константы c, d из условий $\alpha(0) = \alpha_0, \alpha(l) = \alpha_N$. Величины $k_0, k_N, \alpha_0, \alpha_N$ задает пользователь, их надо вынести в интерфейс.

3. Разностная схема с разностным краевым условием при $x = 0$. Получено в Лекции №7 (7.14), (7.15), и может быть использовано 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 Лекции №7 (формула (7.15)). Для этого надо проинтегрировать на отрезке $[x_{N-\frac{1}{2}}, x_N]$ выписанное выше уравнение (1) с учетом (7.9) из Лекции №7 и учесть, что поток

   $$F_n = \alpha_N(y_N - T_0), F_{N-\frac{1}{2}} = \chi_{N-\frac{1}{2}}\frac{y_{N-1} - y_N}{h}.$$

4. Значения параметров для отладки (все размерности согласованы)

   $k_0 = 0.4$ Вт/см К,

   $k_N = 0.1$ Вт/см К,

$\alpha_0 = 0.05$ Вт/см^2 К,

$\alpha_N = 0.01$ Вт/см^2 К,

$l = 10$ см,

$T_0 = 300$ К,

$R = 0.5$ см,

$F_0 = 50$ Вт/см^2.

Физическое содержание задачи

Сформулированная математическая модель описывает температурное поле $T(x)$ вдоль цилиндрического стержня радиуса R и длиной l , причем $R << l$ и температуру можно принять постоянной по радиусу цилиндра. Ось x направлена вдоль оси цилиндра и начало координат совпадает с левым торцем стержня. Слева при $x = 0$ цилиндр нагружается тепловым потоком F_0 . Стержень обдувается воздухом, температура которого равна T_0 . В результате происходит съем тепла с цилиндрической поверхности и поверхности правого торца при $x = l$. Функции $k(x), \alpha(x)$ являются, соответственно, коэффициентами теплопроводности материала стержня и теплоотдачи при обдуве.

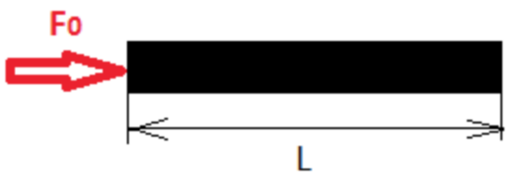

Рис. 1: Стержень нагревают с одного из торцов

Результаты работы.

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

Проинтегрируем уравнение (7.8) с учетом (7.9) из Лекции №7 на отрезке $[x_{N-\frac{1}{2}}, x_N]$

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

Второй и третий интегралы вычислим методом трапеций

$$-(F_{x_N} - F_{x_{N-\frac{1}{2}}}) - \frac{h}{4}(p_{x_{N-\frac{1}{2}}} y_{x_{N-\frac{1}{2}}} + p_{x_N} y_{x_N}) + \frac{h}{4}(f_{x_{N-\frac{1}{2}}} + f_{x_N}) = 0.$$

Подставляя выражение для $F_{x_{N-\frac{1}{2}}}$ согласно (7.12) из Лекции №7 при $n = N$ и краевое условие при $x = l$ (x_N), придем к формуле

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

Где $\alpha = \alpha_N, \beta = T_0$

Можно принять простую аппроксимацию

$$p_{N-\frac{1}{2}} = \frac{p_{N-1} + p_N}{2}, f_{N-\frac{1}{2}} = \frac{f_{N-1} + f_N}{2}$$

При уменьшении шага (в пределе при $h \to 0$), в (2) членами, содержащими h^2 можно пренебречь, тогда (2) преобразуется к виду

$$y_N = \frac{\chi_{N-\frac{1}{2}}}{\alpha h + \chi_{N-\frac{1}{2}}} y_{N-1} + \frac{h\alpha\beta}{\alpha h + \chi_{N-\frac{1}{2}}}$$

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

Ниже представлен график с шагом 0.01 см; $F_0 = 50$ Вт/см^2.

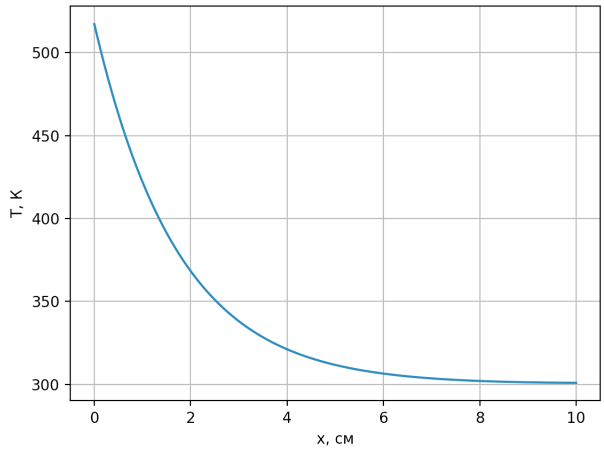

Рис. 2: График зависимости температуры $T(x)$ от координаты х при h = 0.01 см; $F_0 = 50$ Вт/см^2.

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

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

Ниже представлен график с шагом 0.01 см; $F_0 = -10$ Вт/см^2.

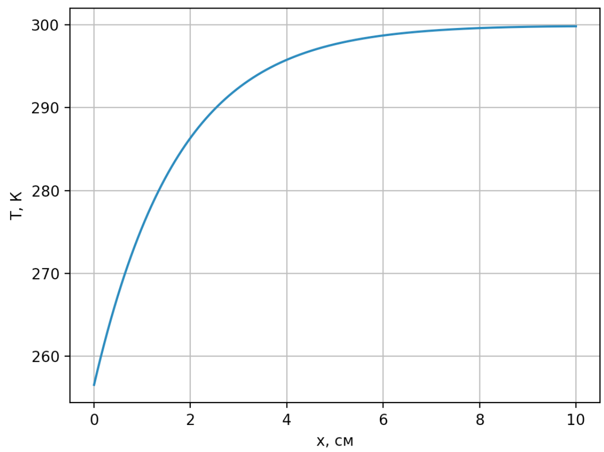

Рис. 3: График зависимости температуры $T(x)$ от координаты х при h = 0.01 см; $F_0 = -10$ Вт/см^2.

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

Сравнить с п.2.

Справка. При увеличении теплосъема и неизменном потоке F_0 уровень температур $T(x)$ должен снижаться, а градиент увеличиваться.

Ниже представлен график с шагом 0.01 см; $F_0 = 50$ Вт/см^2; $\alpha_0 = 0.15$ Вт/см^2 К, $\alpha_N = 0.03$ Вт/см^2 К (увеличено в 3 раза).

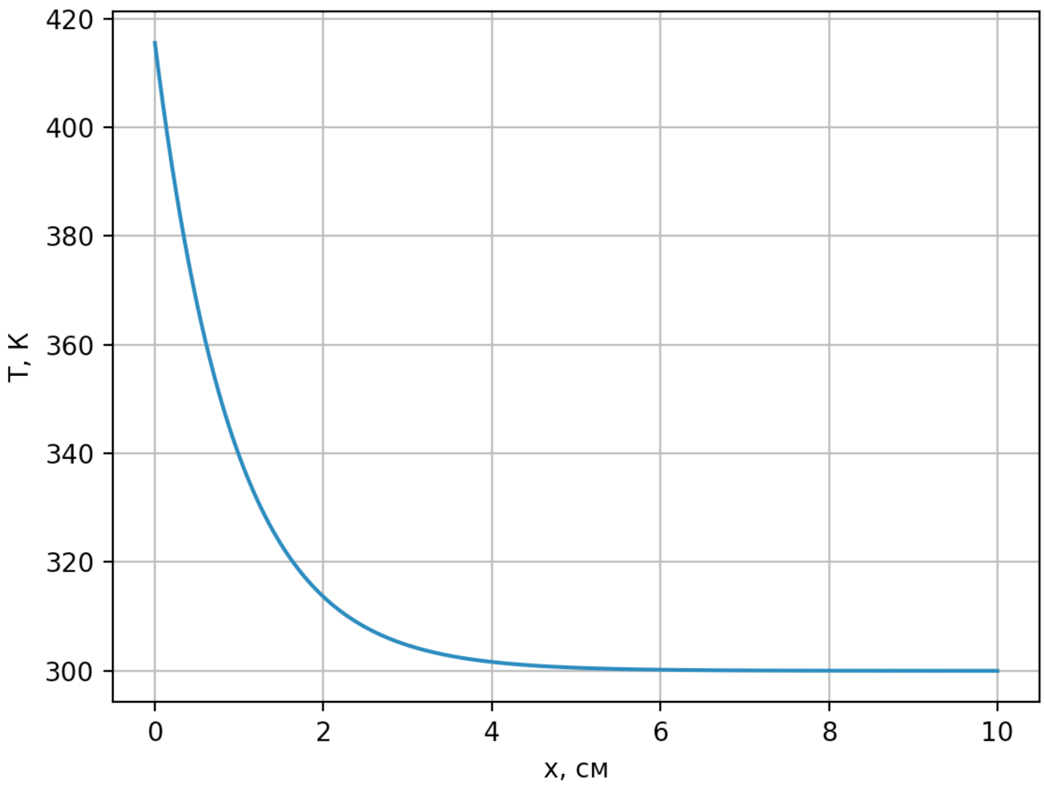

Рис. 4: График зависимости температуры $T(x)$ от координаты x при h $= 0.01$ см; $F_0 = 50$ Вт/см^2; $\alpha_0 = 0.15$ Вт/см^2 К, $\alpha_N = 0.03$ Вт/см^2 К.

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).

Ниже представлен график с шагом 0.01 см; $F_0 = 0.0$ Вт/см^2.

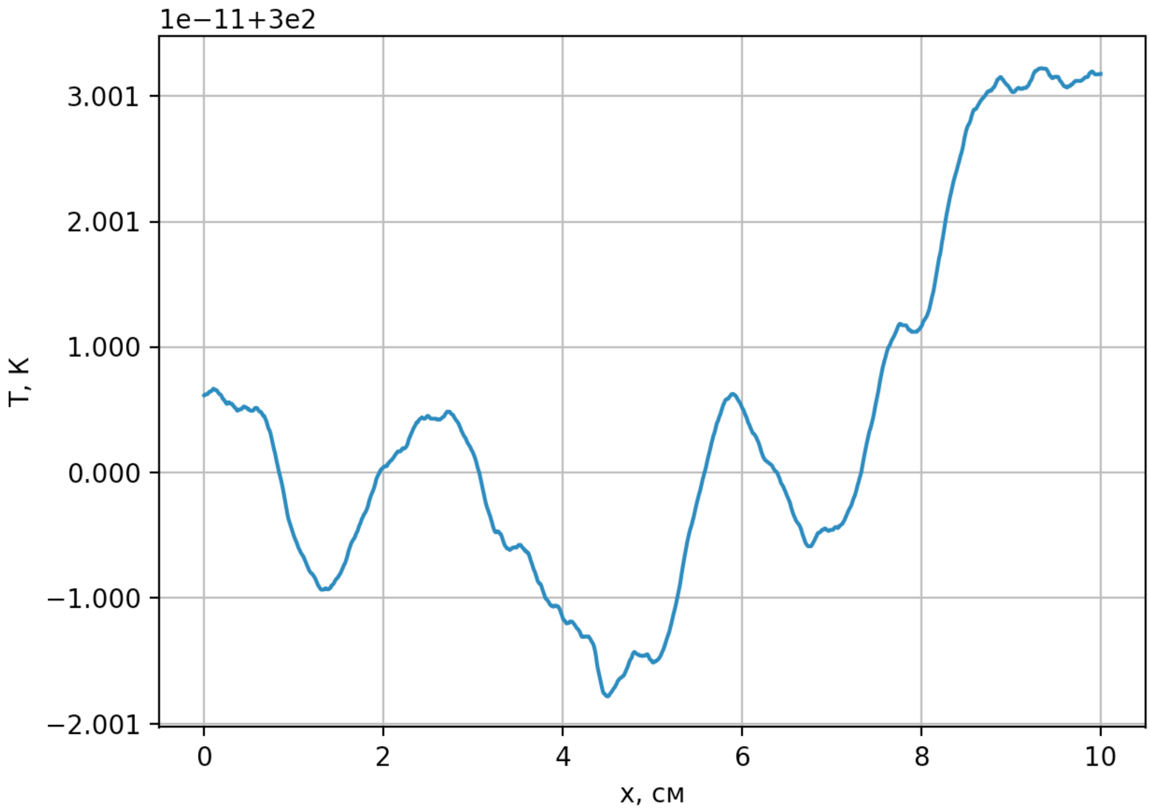

Рис. 5: График зависимости температуры $T(x)$ от координаты x при h = 0.01 см; $F_0 = 0.0$ Вт/см^2.

Вопросы при защите лабораторной работы.

Ответы на вопросы дать письменно в отчете о лабораторной работе.

1. Какие способы тестирования программы можно предложить?

1. Первым делом, необходимо проверить, как будет вести себя график при положительно заданном F_0. Физический смысл этого тестирования такой: F_0 - тепловой поток, которым нагревают стержень с левого края. Если он больше 0, то температура стержня слева должна быть выше окружающей температуры T_0, и чем ближе мы приближаемся к правому краю, с которого нагрева нет (рис. 1), тем сильнее график $T(x)$ должен приближаться к значению T_0 (см. пункт 2 в Результатах работы).

2. Аналогично при отрицательных значениях F_0 происходит охлаждение стержня с левой стороны, значит значение функции $T(x)$ при $x = 0$ должно быть меньше температуры окружающей среды T_0, и затем постепенно (по мере увеличения x) приближаться к T_0(см. пункт 3 в Результатах работы).

3. Следующий способ тестирования - увеличение/уменьшение начальных значений $\alpha(x)$: α_0, α_N, которые являются коэффициентами теплоотдачи при обдуве, и, соответственно, чем выше эти коэффициенты, тем меньше должен быть уровень температур при неизменном потоке F_0 (см. пункт 4 в Результатах работы).

4. Последний способ: задать $F_0 = 0$. То есть стержень не охлаждается и не нагревается, в следствие чего должна получиться "прямая"линия (с учетом некоторой погрешности) на всем интервале $x \in [0; l]$, принимающая значение $T = 300$K (температура окружающей среды) (см. пункт 5 в Результатах работы).

2. Получите простейший разностный аналог нелинейного краевого условия при

$$x = l, -k(l)\frac{dT}{dx} = \alpha_N(T(L) - T_0) + \varphi(T), \tag{3}$$

где $\varphi(T)$ - заданная функция.

Решение:

Аппроксимируя производную односторонней разностью имеем:

$$-k_N\frac{y_N - y_{N-1}}{h} = \alpha_N(y_N - T_0) + \varphi(y_N) \tag{4}$$

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

Необходимо использовать правую прогонку. В этом случае основаная прогоночная формула записывается в виде

$$y_n = \xi_{n+1} y_{n+1} + \eta_{n+1}, \tag{5}$$

а рекуррентные соотношения для определения прогоночных коэффициентов

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

Принимая простейшую (первого порядка точности) аппроксимацию краевого условия $x = 0$, получим его разностный аналог

$$-k_0 \frac{y_1 - y_0}{h} = F_0$$

Откуда, учитывая, что $y_0 = \xi_1 y_1 + \eta_1$, найдем

$$\xi_1 = 1, \eta_1 = \frac{hF_0}{k_0}$$

Аналогичная разностная аппроксимация правого краевого условия имеет вид (4).

Откуда получаем уравнение для определения y_N

$$\varphi(y_N) + y_N(\alpha_N + \frac{k_N}{h}(1 - \xi_N)) - \alpha_N T_0 - \eta_N = 0 \tag{6}$$

Решение данного уравнения удобно искать, например, методом половинного деления. К этому моменту прогоночные коэффициенты ξ_N, η_N уже определены.

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). Краевые условия линейные.

Дано:

Коэффициенты A_n, B_n, C_n, F_n уравнения $A_n y_{n-1} - B_n y_n + C_n y_{n+1} = -F_n$.

Краевые условия линейны, то есть:

1. $K_0 y_0 + M_0 y_1 = P_0$
2. $K_N y_N + M_N y_{N-1} = P_N$

Алгоритм действий:

1. Вычисляем прогоночные коэффициенты правой и левой прогонок по следующим формулам:
   - Левая прогонка:
     (a) $\xi_1 = -\frac{M_0}{K_0}, \xi_{n+1} = \frac{C_n}{B_n - A_n \xi_n}$
     (b) $\eta_1 = -\frac{P_0}{K_0}, \eta_{n+1} = \frac{F_n + A_n \eta_n}{B_n - A_n \xi_n}$
   - Правая прогонка:
     (a) $\alpha_N = -\frac{K_N}{M_N}, \alpha_{n-1} = \frac{A_n}{B_n - C_n \alpha_n}$
     (b) $\beta_N = -\frac{P_N}{M_N}, \beta_{n-1} = \frac{F_n + C_n \beta_n}{B_n - C_n \alpha_n}$
2. Используя рекуррентные соотношения для определения неизвестных y_i, составим систему из двух уравнений при $i = p - 1$:

$$\begin{cases} y_{p-1} = \xi_p y_p + \eta_p \\ y_p = \alpha_p y_{p-1} + \beta_p \end{cases}$$

Решая ее, получим:

$$y_p = \frac{\alpha_p \eta_p + \beta_p}{1 - \alpha_p \xi_p}$$

Где коэффициенты $\alpha_p, \eta_p, \beta_p, \xi_p$ уже найдены.

Листинг кода программы

Метод прогоночных коэффициентов:

```python
def calc_n_plus_1_ksi(ksi_n, C_n, B_n, A_n, x, a, b, c, d, R, h):
    return C_n(a, b, h, x) / (B_n(a, b, h, c, d, R, x) - A_n(a, b, h, x) * ksi_n)

def calc_n_plus_1_etta(ksi_n, etta_n, B_n, A_n, x, F_n, a, b, c, d, T_0, R, h):
    return (F_n(T_0, R, c, d, h, x) + A_n(a, b, h, x) * etta_n) / (B_n(a, b, h, c, d, R, x) - A_n(a, b, h, x) * ksi_n)


def calc_y_n(ksi_n_plus_1, etta_plus_1, y_plus_1):
    return ksi_n_plus_1 * y_plus_1 + etta_plus_1

def progonka(A_n, B_n, C_n, F_n, K_0, M_0, P_0, K_N, M_N, P_N, x_0, h, x_N, a, b, c, d, T_0, R):

    ksi = []
    etta = []
    y_result = []

    #начальные кси и этта
    ksi_1 = -M_0 / K_0
    etta_1 = P_0 / K_0

    ksi.append(ksi_1)
    etta.append(etta_1)

    #вычисление всех кси и этта
    i = 0
    while (x_0 < (x_N - h)):
        ksi.append(calc_n_plus_1_ksi(ksi[i], C_n, B_n, A_n, x_0, a, b, c, d, R, h))
        etta.append(calc_n_plus_1_etta(ksi[i], etta[i], B_n, A_n, x_0, F_n, a, b, c, d, T_0, R, h))
        x_0 += h
        i += 1
```

```python
    #вычисление y_N
    y_N = (P_N - M_N * etta[i]) / (K_N + M_N * ksi[i])

    y_result.append(y_N)

    #вычислене всех y_n
    for l in range(len(ksi) - 1):
        y_result.append(calc_y_n(ksi[i], etta[i], y_result[-1]))
        i -= 1

    y_result.reverse()

    return y_result
```

Рис. 6: Реализация метода прогоночных коэффициентов.

Реализация задачи:

```
from progonka import *

def calc_const(k_0, k_N, x_N, x_0):
    b = (k_N * x_N - x_0 * k_0) / (k_N - k_0)
    a = k_0 * (x_0 - b)
    return a, b

def calc_a_b_c_d(k_0, k_N, alpha_0, alpha_N, x_N, x_0):
    a, b = calc_const(k_0, k_N, x_N, x_0)
    c, d = calc_const(alpha_0, alpha_N, x_N, x_0)
    return a, b, c, d


def alpha(c, d, x):
    return c / (x - d)

def k(a, b, x):
    return a / (x - b)

def f(T_0, R, c, d, x):
    return 2.0 * T_0 / R * alpha(c, d, x)

def p(c, d, R, x):
    return 2.0 / R * alpha(c, d, x)

#метод трапеций
def X_n_plus_half(k_n, k_n_plus_1):
    return 2.0 * k_n * k_n_plus_1 / (k_n + k_n_plus_1)

def A_n(a, b, h, x):
    return X_n_plus_half(k(a, b, x), k(a, b, x - h)) / h

def C_n(a, b, h, x):
    return X_n_plus_half(k(a, b, x), k(a, b, x + h)) / h
```

```
def B_n(a, b, h, c, d, R, x):
    return A_n(a, b, h, x) + C_n(a, b, h, x) + p(c, d, R, x) * h

def F_n(T_0, R, c, d, h, x):
    return f(T_0, R, c, d, x) * h

def solve_task(const, x_0, x_N, h):

    k_0 = const.get("k_0")
    k_N = const.get("k_N")
    alpha_0 = const.get("alpha_0")
    alpha_N = const.get("alpha_N")
    F_0 = const.get("F_0")
    T_0 = const.get("T_0")
    R = const.get("R")


    a, b, c, d = calc_a_b_c_d(k_0, k_N, alpha_0, alpha_N, x_N, x_0)

    #вариант с обнулением h^2
    K_0 = 1
    M_0 = -1
    X_half = X_n_plus_half(k_0, k(a, b, x_0 + h))
    P_0 = h * F_0 / X_half
```

```
    K_N = 1
    X_n_minus_half = X_n_plus_half(k_N, k(a, b, x_N - h))
    znam = alpha(c, d, x_N) * h + X_n_minus_half
    M_N = -1 * X_n_minus_half / znam
    P_N = h * alpha(c, d, x_N) * T_0 / znam

    #вариант без обнуления h^2
    '''K_0 = 1
    X_half = X_n_plus_half(k_0, k(a, b, x_0 + h))
    temp1 = 0.5 * (p(c, d, R, x_0) + p(c, d, R, x_0 + h))
    znam1 = X_half + h * h / 8 * temp1 + h * h / 4 * p(c, d, R, x_0)
    M_0 = (h * h / 8 * temp1 - X_half) / znam1
    temp3 = h * h / 4 * (1.5 * f(T_0, R, c, d, x_0) + 0.5 * f(T_0, R, c, d, x_0 + h))
    P_0 = (h * F_0 + temp3) / znam1

    K_N = 1
    X_N_minus_half = X_n_plus_half(k_N, k(a, b, x_N - h))
    temp2 = h * h / 8 * 0.5 * (p(c, d, R, x_N) + p(c, d, R, x_N - h))
    alp = alpha(c, d, x_N)
    znam2 = alp * h + temp2 + h * h / 4 * p(c, d, R, x_N) + X_N_minus_half
    M_N = (temp2 - X_N_minus_half) / znam2
    P_N = (h * alp * T_0 + temp3) / znam2'''

    return progonka(A_n, B_n, C_n, F_n,
                    K_0, M_0, P_0,
                    K_N, M_N, P_N,
                    x_0, h, x_N, a, b, c, d, const.get("T_0"), const.get("R"))
```

Рис. 7: Вычисление коэф. a, b, c, d, и коэф. линейных краевых условий

Интерфейс программы

```
k_0 : 0.4
k_N : 0.1
alpha_0 : 0.05
alpha_N : 0.01
l : 10.0
T_0 : 300.0
R : 0.5
F_0 : 50.0
Выберите действие:
1. Сменить k_0
2. Сменить k_N
3. Сменить alpha_0
4. Сменить alpha_N
0. Настройка завершена
1
```

```
Введите k_0: 0.3
k_0 : 0.3
k_N : 0.1
alpha_0 : 0.05
alpha_N : 0.01
l : 10.0
T_0 : 300.0
R : 0.5
F_0 : 50.0
Выберите действие:
1. Сменить k_0
2. Сменить k_N
3. Сменить alpha_0
4. Сменить alpha_N
0. Настройка завершена
0
Введите шаг: 0.01
```

Рис. 8: Интерфейс программы