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 № 4

Дисциплина: Моделирование

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

Студент Юмаев А. Р.

Группа ИУ7-65Б

Оценка (баллы) ________________

Преподаватель Градов В.М.

Москва, 2020 г.

1. Теоретический раздел

Задана математическая модель

Уравнение для функции $T(x,t)$ (1)

$$c(T)\frac{\partial T}{\partial t}=\frac{\partial}{\partial x}\left(k(T)\frac{\partial T}{\partial x}\right)-\frac{2}{R}a(x)T+\frac{2T_0}{R}a(x)$$

Краевые условия

$$\begin{cases} t=0, T(x,0)=T_0 \\ x=0, -k\big(T(0)\big)\dfrac{\partial T}{\partial x}=F_0 \\ x=l, -k(l)\dfrac{\partial T}{\partial x}=a_N(T(l)-T_0) \end{cases}$$

В обозначениях уравнения 14.1 лекции №14

$$p(x)=\frac{2}{R}\,a(x)$$

$$f(x)=\frac{2T_0}{R}a(x)$$

Функция $a(x)$ задана уравнением:

$$a(x)=\frac{c}{x-d}$$

Константы c и d из условий $a(0)=a_0, a(l)=a_N$,

$$c=-a_0 d=\frac{a_0 a_N l}{a_0-a_N}$$

$$d=\frac{a_N l}{a_N-a_0}$$

Разностная схема

$$\hat{A}_n\hat{y}_{n-1}-\hat{B}_n\hat{y}_n+\hat{D}_n\hat{y}_{n+1}=-\hat{F}_n, 1\le n\le N-1$$

$$\hat{A}_n=\hat{X}_{n-\frac{1}{2}}\frac{\tau}{h},$$

$$\hat{B}_n=\hat{A}_n+\hat{D}_n+\hat{c}_n h+p_n h\tau,$$

$$\hat{D}_n=\hat{X}_{n+\frac{1}{2}}\frac{\tau}{h},$$

$$\hat{F}_n=f_n h\tau+\hat{c}_n y_n h$$

Разностные аналоги краевых условий при $x = 0$ (получены в Лекции №14)

$$\left(\frac{h}{8}\hat{c}_{\frac{1}{2}} + \frac{h}{4}\hat{c}_0 + \hat{X}_{\frac{1}{2}}\frac{\tau}{h} + \frac{\tau h}{8}p_{\frac{1}{2}} + \frac{\tau h}{4}p_0\right)\hat{y}_0 + \left(\frac{h}{8}\hat{c}_{\frac{1}{2}} - \hat{X}_{\frac{1}{2}}\frac{\tau}{h} + \frac{\tau h}{8}p_{\frac{1}{2}}\right)\hat{y}_1$$
$$= \frac{h}{8}\hat{c}_{\frac{1}{2}}(y_0 + y_1) + \frac{h}{4}\hat{c}_0 y_0 + \hat{F}\tau + \frac{\tau h}{4}\left(\hat{f}_{\frac{1}{2}} + \hat{c}_0\right)$$

Получим интегро-интерполяционным методом разносный аналог краевого условия при $x = l$.

Для этого обозначим $F = -k(T)\frac{\partial T}{\partial x}$ и учтем то, что поток

$$\hat{F}_N = a_N(\hat{y}_N - T_0), \hat{F}_{N-\frac{1}{2}} = \hat{X}_{N-\frac{1}{2}}\frac{\hat{y}_{N-1} - \hat{y}_N}{h}$$

Проинтегрировав уравнение (1) на отрезке $[x_{N-\frac{1}{2}}, x_N]$ получим:

$$\hat{c}_{N-\frac{1}{2}}\left(y_N\frac{h}{8} + y_{N-1}\frac{h}{8}\right) + \hat{c}_N y_N\frac{h}{4} + \tau a_N T_0 + \left(\hat{f}_N + \hat{f}_{N-\frac{1}{2}}\right)\tau\frac{h}{4}$$

Из полученных краевых условий находим коэффициенты $K_0, K_N, M_0, M_N, P_0, P_N$.

$$\begin{cases} \hat{K}_0\hat{y}_0 + \hat{M}_0\hat{y}_1 = \hat{P}_0 \\ \hat{A}_n\hat{y}_{n-1} - \hat{B}_n\hat{y}_n + \hat{D}_n\hat{y}_{n+1} = -\hat{F}_n, 1 \le n \le N-1 \\ \hat{K}_N\hat{y}_N + \hat{M}_{N-1}\hat{y}_{N-1} = \hat{P}_N \end{cases}$$

Данная система решаемся методом итераций. Обозначим текущую итерацию s, а предыдущую $s - 1$.

$$A_n^{s-1}y_{n+1}^s - B_n^{s-1}y_n^s + D_n^{s-1}y_{n-1}^s = -F_n^{s-1}$$

Значения параметров

$k(T) = a_1(b_1 + c_1T^{m_1}), \frac{\text{Вт}}{\text{см К}}$,

$c(T) = a_2 + b_2T^{m_2} - \frac{c_2}{T^2}, \frac{\text{Дж}}{\text{см}^3\ K}$,

$a_1 = 0.0134$,

$b_1 = 1$,

$c_1 = 4.35 * 10^{-4}$,

$m_1 = 1$,

$a_2 = 2.049$,

$b_2 = 0.563 * 10^{-3}$,

$c_2 = 0.528 * 10^5$,

$m_2 = 1$,

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

$a_0 = 0.05 \frac{\text{Вт}}{\text{см}^2\ \text{К}}$,

$a_N = 0.01 \frac{\text{Вт}}{\text{см}^2\text{К}}$,

$l = 10$ см,

$T_0 = 300$ К,

$R = 0.5$ см,

$F(t) = 50 \frac{\text{Вт}}{\text{см}^2}$.

1. Сформулированная в данной работе математическая модель описывает нестационарное температурное поле $T(x,\ t)$, зависящее от координаты x и меняющееся во времени.
2. Свойства материала стержня привязаны к температуре, т.е. теплоемкость и коэффициент теплопроводности $c(T)$, $k(T)$ зависят от T тогда как в работе №3 $k(x)$ зависит от координаты, а $c = 0$.
3. При $x = 0$ цилиндр нагружается тепловым потоком $F(t)$, в общем случае зависящим от времени, а в работе №3 поток был постоянный. Если в настоящей работе задать поток постоянным, т.е. $F(t) = const$, то будет происходить формирование температурного поля от начальной температуры $T0$ до некоторого установившегося (стационарного) распределения оте математическая модель описывает нестационарное температурное поле $T(x,\ t)$. Это поле в дальнейшем с течением времени меняться не будет и должно совпасть с температурным распределением $T(x)$ математическая модель описывает нестационарное температурное поле $T(x)$, получаемым в лаб. работе №3, если все параметры задач совпадают, в частности, вместо $k(T)$ надо использовать $k(x)$ из лаб. работы №3. Если после разогрева стержня положить поток $F(t) = 0$, то будет происходить остывание, пока температура не выровняется по всей длине и не станет равной $T0$. При произвольной зависимости потока $F(t)$ от времени температурное поле будет отслеживать поток.

2. Листинг

```
# -*- coding: utf-8 -*-

import numpy
import matplotlib.pyplot as plt
from math import fabs

def k(T_num):
    return a1 * (b1 + c1 * (T_num**m1))

def c(T_num):
    return a2 + b2 * T_num**m2 - c2/(T_num**2)

def alpha(x):
    return c_coef / (x - d)

def p(x):
    return 2/R * alpha(x)

def f(x):
    return 2*T0/R * alpha(x)


def An(T_num, h):
    k_minus = (k(T_num - tao) + k(T_num)) / 2
    return tao/h * k_minus

def Bn(T_num, Ai, Di, xi, h):
    return Ai + Di + c(T_num)*h + p(xi)*h*tao

def Dn(T_num, h):
    k_plus = (k(T_num + tao) + k(T_num)) / 2
    return tao/h * k_plus

def Fn(T_num, xi, h):
    return f(xi) * h * tao + c(T_num) * T_num * h


def getK0(T_start, h):
    k_plus = (k(T_start + tao) + k(T_start)) / 2
    c0_half = (c(T_start) + c(T_start + tao)) / 2
    pn_half = p(h/2)
    return h/8 * c0_half + h/4 * c(T_start) + tao/h * k_plus + tao*h/8 *
pn_half + tao*h/4 * p(x0)

def getM0(T_start, h):
    k_plus = (k(T_start + tao) + k(T_start)) / 2
    c0_half = (c(T_start) + c(T_start + tao)) / 2
    pn_half = p(h/2)
    return h/8 * c0_half - tao/h * k_plus + tao * h/8 * pn_half
```

```
def getP0(T_start, h):
    c0_half = (c(T_start) + c(T_start + tao)) / 2
    return  h/8  *  c0_half  *  (T_old[0]  +  T_old[1])  +  h/4  *  c(T_start)  *
T_start \
        + F0 * tao \
        + tao * h/8 * (3 * f(x0) + f(h))

def getKN(T_end, h):
    k_minus = (k(T_end - tao) + k(T_end)) / 2
    cn_half = (c(T_end) + c(T_end - tao)) / 2
    pn_half = p(l - h/2)
    return h/4 * c(T_end) \
        + h/8 * cn_half \
        + tao*h/4 * p(l) \
        + tao*h/8 * pn_half \
        + tao * alphaN \
        + tao/h * k_minus

def getMN(T_end, h):
    k_minus = (k(T_end - tao) + k(T_end)) / 2
    cn_half = (c(T_end) + c(T_end - tao)) / 2
    pn_half = p(l - h/2)
    return h/8 * cn_half \
        + tao*h/8 * pn_half \
        - tao/h * k_minus

def getPN(T_end, h):
    cn_half = (c(T_end) + c(T_end - tao)) / 2
    fn_half = f(l - h / 2)
    return h/4 * c(T_end) * T_old[-1] \
        + alphaN * T0 * tao \
        + h/4 * tao * (f(l) + fn_half) \
        + h/8 * cn_half * (T_old[-1] + T_old[-2])

def progonka(A, B, C, D, K0, M0, P0, KN, MN, PN):
    xi = [0]
    eta = [0]
    xi.append(-M0 / K0)
    eta.append(P0 / K0)

    for i in range(1, len(A)):
        xi.append(C[i] / (B[i] - A[i] * xi[-1]))
        eta.append((D[i] + A[i] * eta[-1]) / (B[i] - A[i] * xi[-2]))

    y = [(PN - MN * eta[-1]) / (KN + MN * xi[-1])]
    for i in range(len(A) - 2, -1, -1):
        y.append(xi[i] * y[-1] + eta[i])

    y.reverse()

    return y
```

```
def do_plot(masx, masy, xlabel, ylabel):
    plt.plot(masx, masy)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.grid(True)

def calc_changes(a, b):
    lb = len(b)
    la = len(a)

    if lb > la:
        a, b = b, a

    la = len(a)

    diff = []
    for i in range(la):
        if i < lb:
            diff.append(fabs(b[i] - a[i]))
        else:
            diff.append(a[i])

    return diff


if __name__ == "__main__":
    a1 = 0.0134
    b1 = 1
    c1 = 4.35 * 1e-4
    m1 = 1
    a2 = 2.049
    b2 = 0.563 * 1e-3
    c2 = 0.528 * 1e+5
    m2 = 1

    alpha0 = 0.05
    alphaN = 0.01

    l = 10
    T0 = 300
    R = 0.5
    F0 = 50

    x0 = 0
    h = 1e-2
    tao = 1

    d = alphaN * l / (alphaN - alpha0)
    c_coef = - alpha0 * d

    T = [T0 for x in numpy.arange(x0, l + h, h)]
    time = 0
```

```
    mas_x = [x for x in numpy.arange(x0, l + h, h)]
    mas_t = [0]
    do_plot(mas_x[1:], T[1:], 'Length (cm)', 'Temperature (K)')
    T_global = [T[:]]

    A = []
    D = []
    B = []
    F = []

    k_i = 0
    while True:
        T_old = T[:]
        A.clear()
        B.clear()
        D.clear()
        F.clear()
        i = 0
        for x in numpy.arange(x0, l + h, h):
            Ai = An(T_old[i], h)
            Di = Dn(T_old[i], h)
            Bi = Bn(T_old[i], Ai, Di, x, h)
            Fi = Fn(T_old[i], x, h)

            A.append(Ai)
            D.append(Di)
            B.append(Bi)
            F.append(Fi)
            i += 1

        K0 = getK0(T_old[0], h)
        M0 = getM0(T_old[0], h)
        P0 = getP0(T_old[0], h)

        KN = getKN(T_old[-1], h)
        MN = getMN(T_old[-1], h)
        PN = getPN(T_old[-1], h)

        T = progonka(A, B, D, F, K0, M0, P0, KN, MN, PN)
        T[0] = T[1]

        diff = calc_changes(T, T_old)
        maxDiff = max(diff)
        if fabs(maxDiff / T[diff.index(maxDiff)]) < 1e-4:
            T_global.append(T[:])
            time += tao
            mas_t.append(time)
            do_plot(mas_x[1:], T[1:], 'Length (cm)', 'Temperature (K)')
            break

        if k_i % 10 == 0:
            do_plot(mas_x[1:], T[1:], 'Length (cm)', 'Temperature (K)')
```

```
        T_global.append(T[:])
        time += tao
        mas_t.append(time)
        k_i += 1

    plt.show()


    matrix = []
    for i in range(len(T_global[0])):
        matrix.append([])

    for i in range(len(T_global)):
        for j in range(len(T_global[i])):
            matrix[j].append(T_global[i][j])

    for i in range(len(matrix)):
        if (i % 10 == 0):
            do_plot(mas_t, matrix[i], 'Time (sec)', 'Temperature (K)')
    plt.show()
```

3. Результаты работы программы

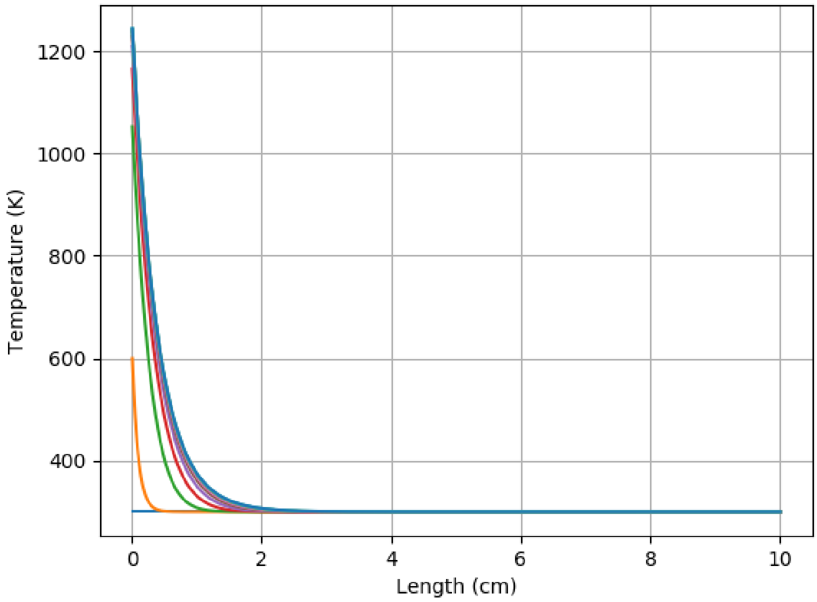

Рисунок 1. График зависимости T(x, t_m) от координаты x при нескольких фиксированных значениях времени t_m (верхний график построен при установившемся распределении T(x, t)).

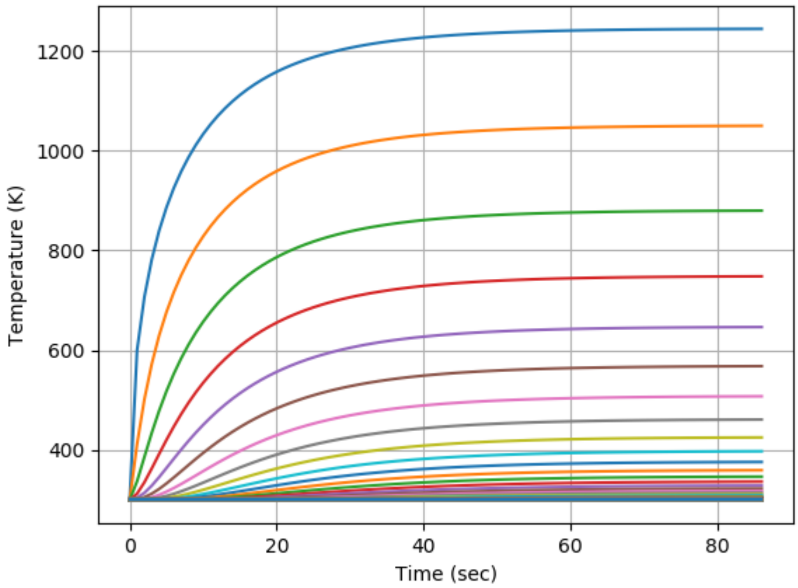

Рисунок 2. График зависимости T(x_n, t) при нескольких фиксированных значениях координаты x_n.

4. Ответы на вопросы

1. **Результаты тестирования программы**

Если из полученных разностных аналогов краевых условий обнулить $c(u)$, принять τ равным 1 и заменить зависимость коэффициента теплопроводности от T на зависимость от координаты на стержне x, то можно получить график (Рис. 3), соответствующий полученному графику в 3 лабораторной работе (Рис. 4).

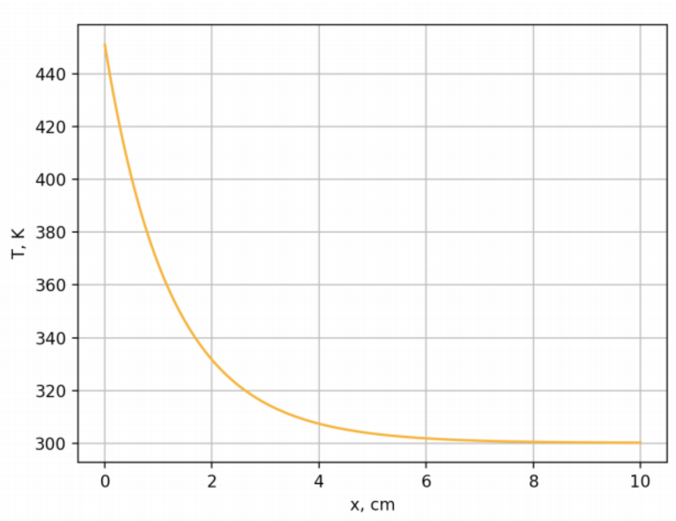

Рисунок 3. График зависимости T от x

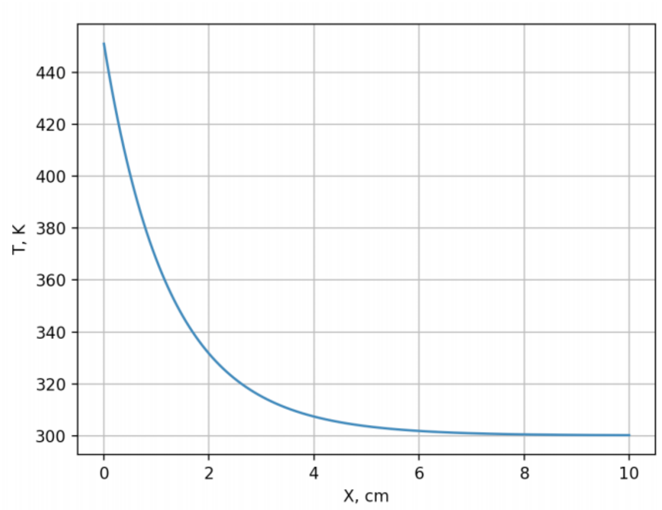

Рисунок 4. График зависимости T от x

2. Выполните линеаризацию уравнения (14.8) по Ньютону, полагая для простоты, что все коэффициенты зависят только от одной переменной $\hat{y}_n$.

Уравнение, которое необходимо линеаризовать

$$\begin{cases} \hat{K}_0 \hat{y}_0 + \hat{M}_0 \hat{y}_1 = P_0 \\ \hat{A}_n \hat{y}_{n-1} - \hat{B}_n \hat{y}_n + \hat{D}_n \hat{y}_{n+1} = -\hat{F}_n, \quad 1 \le n \le N-1 \\ \hat{K}_N \hat{y}_N + \hat{M}_{N-1} \hat{y}_{N-1} = \hat{P}_N \end{cases}$$

Выполним линеаризацию методом Ньютона по переменным $\hat{y}_{n-1}, \hat{y}_n, \hat{y}_{n+1}$. Текущая итерация – s, предыдущая: $s-1$. Начальное распределение: $\hat{y}_0$. Значения предыдущей итерации известны.

$$\left(\hat{A}_n \hat{y}_{n-1} - \hat{B}_n \hat{y}_n + \hat{D}_n \hat{y}_{n+1} + \hat{F}_n\right)\Big|_{(s-1)} + \hat{A}_n^{(s-1)} \Delta \hat{y}_{n-1}^{(s)} +$$

$$+ \left(\frac{\partial \hat{A}_n}{\partial \hat{y}_n} \hat{y}_{n-1} - \frac{\partial \hat{B}_n}{\partial \hat{y}_n} \hat{y}_n - \hat{B}_n \frac{\partial \hat{D}_n}{\hat{y}_n} \hat{y}_{n+1} + \frac{\partial \hat{F}_n}{\partial \hat{y}_n}\right)\Big|_{(s-1)} \Delta \hat{y}_n^{(s)} +$$

$$+ \hat{D}_n^{(s-1)} \Delta \hat{y}_{n+1}^{(s)} = 0$$

Введем обозначения

$$A_n = \hat{A}_n^{(s-1)}$$

$$B_n = \left(-\frac{\partial \hat{A}_n}{\partial \hat{y}_n} \hat{y}_{n-1} + \frac{\partial \hat{B}_n}{\partial \hat{y}_n} \hat{y}_n + \hat{B}_n - \frac{\partial \hat{D}_n}{\partial \hat{y}_n} \hat{y}_{n+1} + \frac{\partial \hat{F}_n}{\partial \hat{y}_n}\right)\Big|_{(s-1)}$$

$$D_n = \hat{D}_n^{(s-1)}$$

$$F_n = \left(\hat{A}_n \hat{y}_{n-1} - \hat{B}_n \hat{y}_n + \hat{D}_n \hat{y}_{n+1} + \hat{F}_n\right)\Big|_{(s-1)}$$

Тогда

$$A_n \Delta \hat{y}_{n-1}^{(s)} - B_n \Delta \hat{y}_n^{(s)} + D_n \Delta \hat{y}_{n+1}^{(s)} = -F_n$$

Выполним линеаризацию краевых условий для $\hat{y}_0$.

$$\left(\hat{k}_0 \hat{y}_0 + \hat{M}_0 \hat{y}_1 - P_0\right)\Big|_{(s-1)} + \hat{k}_0^{(s+1)} \Delta \hat{y}_0^{(s)} + \hat{M}_0^{(s-1)} \Delta \hat{y}_1^{(s)} = 0$$

$$k_0 = \hat{k}_0^{(s-1)}$$

$$M_0 = \hat{M}_0^{(s-1)}$$

$$P_0 = -\left(\hat{k}_0 \hat{y}_0 + \hat{M}_0 \hat{y}_1 - \hat{P}_0\right)\Big|_{(s-1)}$$

$$k_0 \Delta \hat{y}_0^{(s)} + M_0 \Delta \hat{y}_1^{(s)} = P_0$$

Для $\hat{y}_n$

$$\left(\hat{k}_N \hat{y}_N + \hat{M}_{N-1} \hat{y}_{N-1} - \hat{P}_N\right)\Big|_{(s-1)} + \bar{k}_n \Delta \hat{y}_N^{(s)} + \bar{M}_{N-1}^{(s-1)} \Delta \hat{y}_{N-1}^{(s)} = 0$$

$$k_N = \bar{k}_N^{(s-1)}$$

$$M_{N-1} = \hat{M}_{N-1}^{(s-1)}$$

$$P_N = -\left(\hat{k}_N \hat{y}_N + \hat{M}_{N-1} \hat{y}_{N-1} - \bar{P}_N\right)\Big|_{(s-1)}$$

Получаем систему:

$$A_n \Delta \hat{y}_{n-1}^{(s)} - B_n \Delta \bar{y}_n^{(s)} + D_n \Delta \bar{y}_{n+1}^{(s)} = -F_n$$

$$k_0 \Delta \bar{y}_0^{(s)} + M_0 \Delta \hat{y}_1^{(s)} = P_0$$

$$k_N \Delta \bar{y}_N^{(s)} + M_{N-1} \Delta \bar{y}_{N-1}^{(s)} = P_N$$

Система решается методом прогонки. Находим все $\Delta \bar{y}_n^{s}$. Зная приближение $s-1$, можно найти приближение s.

$$\bar{y}_n^{s} = \bar{y}_n^{(s-1)} + \Delta \bar{y}_n^{s}.$$

Процесс завершается, когда достигнуто условие

$$\max \left| \frac{\Delta \hat{y}_n^{s}}{\bar{y}_n^{s}} \right| \le \varepsilon.$$