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: Платонова О. С.

Группа: ИУ7-65Б

Оценка(баллы) ____________

Преподаватель: Градов В. М.

Москва, 2021 г.

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

Входные данные:

1. Квазилинейное уравнение для функции *T(x)*.

$$\frac{d}{dx}\left(\lambda(T)\frac{dT}{dx}\right) - 4\,k(T)\,n_p^2\,\sigma\,(T^4 - T_0^4) = 0 \qquad (1)$$

Краевые условия:

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

2. Функции $\lambda(T)$, *k(T)* заданы таблицей.
3. Разностная схема с разностным краевым условием при $x = 0$.
4. Значения параметров для отладки.
5. Выход из итераций организован по температуре и балансу энергии.

Выходные данные:

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

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

$$f_1 = F_0 - \alpha(T(l) - T_0) \text{ и } f_2 = 4\, n_p^2\, \sigma \int_0^1 k\big(T(x)\big)(T^4(x) - T_0^4)dx.$$

Каковы использованные в работе значения точности выхода из итераций ε_1 (по температуре) и ε_2 (по балансу энергии)?

Решение

1. *Разностный аналог краевого условия при x = l и его краткий вывод интегро-интерполяционным методом.*

Рассмотрим уравнение

$$\frac{d}{dx}\left(k(x,u)\frac{du}{dx}\right) - p(x,u)u + f(x,u) = 0 \qquad (2)$$

В условии поставленной задачи, уравнение (2) примет вид:

k(x, u) = $\lambda(u)$

p(x, u) = 0

f(x, u) = $-4\, k(u)\, n_p^2\, \sigma\, (u^4 - T_0^4)$

$$\frac{d}{dx}\left(k(x,u)\frac{du}{dx}\right) + f(x,u) = 0 \qquad (3)$$

Введем сетку в области интегрирования уравнения *[0, l]*:

$\omega_h = \{x_n : x_n = nh, n = 0, 1, \dots N, h = l/n\}$.

Обозначим

$$F = -k(x)\,\frac{du}{dx} \qquad (4)$$

Проинтегрируем уравнение (3) с учетом (4) на отрезке $[x_{N-1/2}, x_N]$ и учетом того, что поток $F_N = \alpha_N(y_N - T_0)$, а $F_{N-1/2} = \chi_{N-1/2}\frac{y_{N-1}-y_N}{h}$.

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

Второй интеграл вычисляется методом трапеций.

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

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

$$(\chi_{N-1/2}\frac{y_{N-1}-y_N}{h} - \alpha_N(y_N - T_0)) + \frac{h}{4}(f_N + f_{N-\frac{1}{2}}) = 0$$

$$\chi_{N-1/2}\, y_{N-1} - \left(\chi_{N-\frac{1}{2}} + h\alpha_N\right) y_N + h(\alpha_N T_0 + \frac{h}{4}(f_N + f_{N-\frac{1}{2}})) = 0$$

$$\frac{\chi_{N-1/2}}{h} y_{N-1} - \frac{1}{h}\left(\chi_{N-\frac{1}{2}} + h\alpha_N\right) y_N + (\alpha_N T_0 + \frac{h}{4}(f_N + f_{N-\frac{1}{2}})) = 0$$

В результате, разностное краевое условие при $x = l$ приводится к виду

$K_N y_{N-1} + M_N y_N = P_N$, где

$$K_N = \frac{\chi_{N-1/2}}{h}$$

$$M_N = -\left(\frac{\chi_{N-1/2}}{h} + \alpha_N\right)$$

$$P_N = -\,(\alpha_N T_0 + \frac{h}{4}(f_N + f_{N-\frac{1}{2}}))$$

Листинг

```
#include <iostream>
#include <cmath>
#include <iomanip>

#include "getTables.h"
#include "interpolation.h"

using namespace std;

struct Params
{
    double K; double M; double P;
};

struct Coefs
{
    vector<double> A; vector<double> B;
    vector<double> C; vector<double> D;
};

static vector<double> TLTable, LTable, TKTable, KTable;
static vector<double> TFPlot, TAlphaPlot, TF0Plot;
static double eps1 = 1e-3, eps2 = 1e-3;

double p(const double kf, const double T)
{
    return kf * interpolate(T, TKTable, KTable) * pow(T, 3);
}

double f(const double kf, const double T, const double T0)
{
    return kf * interpolate(T, TKTable, KTable) * pow(T0, 4);
}




int main()
{
    double Np = 0, l = 0, T0 = 0, sigma = 0, F0 = 0, alpha = 0;
    double h = 1e-3;

    getParams(Np, l, T0, sigma, F0, alpha);
    getTable1(TLTable, LTable);
    getTable2(TKTable, KTable);

    int N = static_cast<int>(l / h);
    double kf = 4 * Np * Np * sigma;

    vector<double> T;
    for (int i = 0; i < N; i++) {
        T.push_back(T0);
    }

    Params p0, pN;
    Coefs c;

    getParams0(p0, T, kf, T0, h, F0);
    getParamsN(pN, T, N, kf, T0, h, alpha);
    getCoefs(T, N, c, kf, T0, h);

    vector<double> Tres = getY(c, p0, pN);
```

```
    int num = 0;
    for (int i = 0; i < 100 && isTemperature(T, Tres, N) && isBalanceEnergy(T, N, kf, F0, alpha, T0, h); i++) {
        T = Tres;

        getParams0(p0, T, kf, T0, h, F0);
        getParamsN(pN, T, N, kf, T0, h, alpha);
        getCoefs(T, N, c, kf, T0, h);

        for (int i = N - 1; i >= 0; i--) {
            Tres[i] = T[i] + alpha * ((getY(c, p0, pN))[i] - T[i]);
        }

        num = i;
    }

    plot(Tres, num);

    return 0;
}
```

```
void getParams0(Params &p0, const vector<double> T,
                const double kf, const double T0, const double h, const double F0)
{
    double int0 = interpolate(T[0], TLTable, LTable);
    double int1 = interpolate(T[1], TLTable, LTable);
    double hi12 = (int0 + int1) / 2;

    double f0 = f(kf, T[0], T0);
    double f1 = f(kf, T[1], T0);
    double f12 = (f0 + f1) / 2;

    p0.K = hi12 / h;
    p0.M = -hi12 / h;
    p0.P = F0 + (h / 4) * (f0 + f12);
}

void getParamsN(Params &pN, const vector<double> T, const int N,
                const double kf, const double T0, const double h, const double alpha)
{
    double intN1 = interpolate(T[N - 1], TLTable, LTable);
    double intN2 = interpolate(T[N - 2], TLTable, LTable);
    double hiN12 = (intN1 + intN2) / 2;

    double fN1 = f(kf, T[N - 1], T0);
    double fN2 = f(kf, T[N - 2], T0);
    double fN12 = (fN1 + fN2) / 2;

    pN.K = hiN12 / h;
    pN.M = -alpha - hiN12 / h;
    pN.P = -alpha * T0 - (h / 4) * (fN12 + fN1);
}

void getCoefs(vector<double> &T, const int N, Coefs &c,
              const double kf, const double T0, const double h)
{
    for (int i = 1; i < N; i++) {
        double lPrev = interpolate(T[i - 1], TLTable, LTable);
        double lCur = interpolate(T[i], TLTable, LTable);
        double lNext = interpolate(T[i + 1], TLTable, LTable);

        c.A.push_back((lPrev + lCur) / (2 * h));
        c.C.push_back((lCur + lNext) / (2 * h));
        c.B.push_back(c.A[i - 1] + c.C[i - 1] + p(kf, T[i]) * h);
        c.D.push_back(f(kf, T[i], T0) * h);
    }
}
```

```cpp
double f1(vector<double> T, const int N, const double F0, const double alpha, const double T0)
{
    return F0 - alpha * (T[N - 1] - T0);
}

double f2(const double kf, vector<double> T, const int N, const double h, const double T0)
{
    return kf * Simpson(T, N, h, T0);
}
```

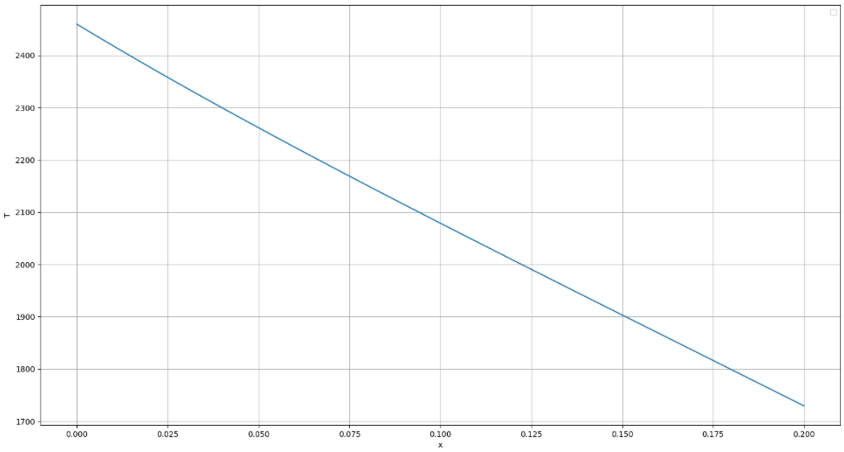
```cpp
bool isTemperature(const vector<double> T1, const vector<double> T2, const int N)
{
    bool res = true;
    for (int i = 0; i < N && res; i++) {
        double x = (T1[i] - T2[i]) / T1[i];
        if (abs(x) > eps1) {
            res = false;
        }
    }
    return res;
}

bool isBalanceEnergy(const vector<double> T, const int N,
                     const double kf, const double F0, const double alpha, const double T0, const double h)
{
    bool res = true;
    for (int i = 0; i < N && res; i++) {
        double ff = f1(T, N, F0, alpha, T0);
        double fs = f2(kf, T, N, h, T0);

        double x = (ff - fs) / ff;
        if (abs(x) > eps2) {
            res = false;
        }
    }
    return res;
}
```

```cpp
vector<double> getY(Coefs c, Params p0, Params pN)
{
    int n = c.A.size();
    vector<double> xi, eta;
    xi.push_back(-p0.M / p0.K);
    eta.push_back(p0.P / p0.K);

    for (int i = 0; i < n; i++) {
        double x = c.C[i] / (c.B[i] - c.A[i] * xi[i]);
        double e = (c.D[i] + c.A[i] * eta[i]) / (c.B[i] - c.A[i] * xi[i]);

        xi.push_back(x);
        eta.push_back(e);
    }

    vector<double> y;
    y.push_back((pN.P - pN.K * eta[n]) / (pN.M + pN.K * xi[n]));
    for (int i = n - 1; i >= 0; i--) {
        double yi = xi[i] * y[0] + eta[i];
        y.insert(y.begin(), yi);
    }

    return y;
}
```

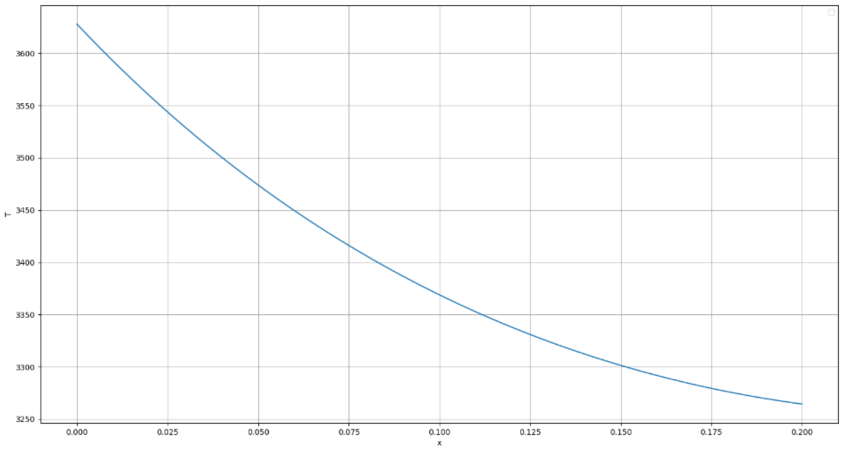
<u>Результаты работы</u>

1. 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.

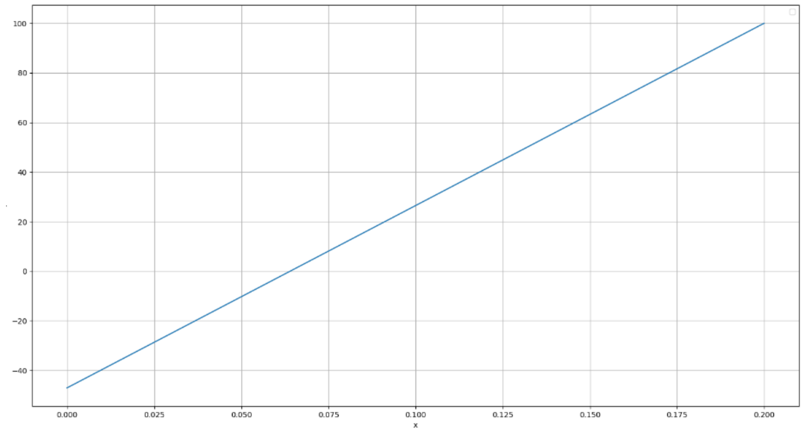
   a. График зависимости *T(x)* от координаты *x* при начальных значениях.



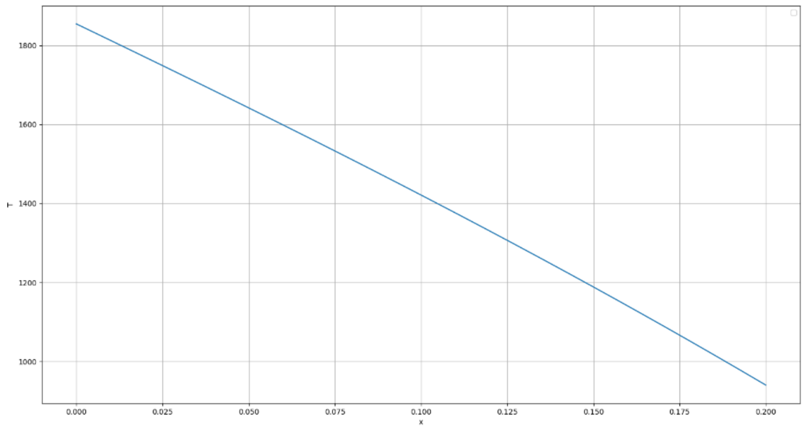
   b. График зависимости T(x) от координаты x при увеличении T_0 в 10 раз.



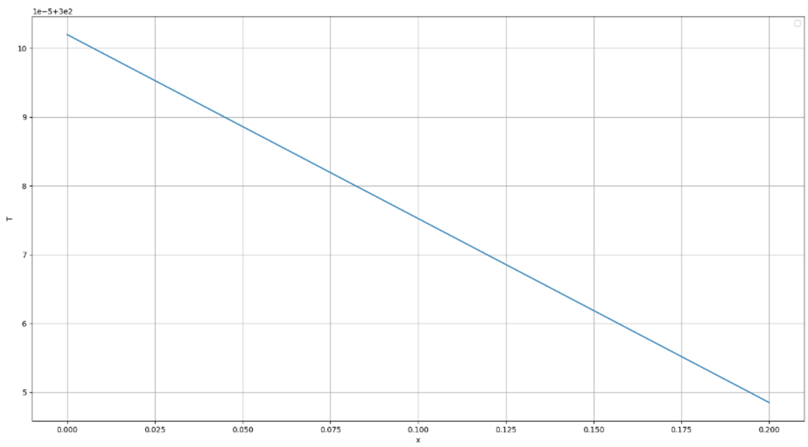
2. График зависимости $T(x)$ при F_0 = -10 Вт/см2.



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



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



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

$$f_1 = F_0 - \alpha(T(l) - T_0) \text{ и } f_2 = 4\, n_p^2\, \sigma \int_0^1 k\big(T(x)\big)(T^4(x) - T_0^4)dx.$$

$f_1 = 28.536$

$f_2 = 28.535$

Каковы использованные в работе значения точности выхода из итераций ε_1 (по температуре) и ε_2 (по балансу энергии)?

$\varepsilon_1 = 0.001$

$\varepsilon_2 = 0.001$

<u>Вопросы</u>

1. Какие способы тестирования программы можно предложить?
   - Тестирование с изменением шага: уменьшение его до тех пор, пока изменение значений не станет пренебрежимо мало.
   - Тестирование на основе физического смысла задачи. Так, при $F_0 = 0$, температура должна равняться температуре окружающей среды, т.е. $T = T_0$, график – прямая. А при $F_0 < 0$, температура должна увеличиваться, $T^I(x) > 0$.

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

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

   где $\varphi(T)$ – заданная функция.

   Производную аппроксимируйте односторонней разностью.

$$\frac{dT}{dx} = \frac{T_{i+1} - T_i}{h}$$

В результате подстановки:

$$-k_N \frac{T_N - T_{N-1}}{h} = \alpha_n (T_N - T_0) + \varphi(T_N)$$

В результате домножения на h:

$$-k_N T_N + k_N T_{N-1} = h\alpha_n T_N - h\alpha_n T_0 + h\varphi(T_N)$$

$$k_N T_{N-1} - (k_N + h\alpha_n) T_N = h\varphi(T_N) - h\alpha_n T_0$$

3. Опишите алгоритм применения метода прогонки, если при x = 0 краевое условие квазилинейное, а при *x* = *l*, как в п.2.

Основная прогоночная формула:

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

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

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

В результате подстановки y_{n-1} в уравнение из п.2

$$k_N (\xi_N T_N + \eta_N) - (k_N + h\alpha_n) T_N = h\varphi(T_N) - h\alpha_n T_0$$

$$(k_N \xi_N - k_N - h\alpha_n) T_N = h\varphi(T_N) - h\alpha_n T_0 - k_N \eta_N$$

4. Опишите алгоритм определения единственного значения сеточной функции y_p в одной заданной точке р. Оба краевых условия линейные.

Основная прогоночная формула для левой прогонки:

$$y_n = \xi_{n-1} y_{n-1} + \eta_{n-1}, \text{ где}$$

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

Объединение левой и правой прогонок:

$$\begin{cases} y_{n-1} = a_n y_n + b_n \\ y_n = c_{n-1} y_{n-1} + d_{n-1} \end{cases}$$

$y_n = c_{n-1}(a_n y_n + b_n) + d_{n-1}$

$(1 - c_{n-1} a_n) y_n = c_{n-1} b_n + d_{n-1}$

$$y_p = \frac{c_{p-1} b_p + d_{p-1}}{1 - c_{p-1} a_p}$$