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 №4

По курсу: "Моделирование"

Тема Программно-алгоритмическая реализация моделей на основе дифференциальных уравнений в частных производных с краевыми условиями II и III рода.

Группа ИУ7-63Б

Студент Сукочева А.

Преподаватель Градов В.М.

Москва — 2021 г.

0.1 Постановка задачи

Цель работы. Получение навыков разработки алгоритмов решения смешанной краевой задачи при реализации моделей, построенных на квазилинейном уравнении параболического типа.

0.1.1 Исходные данные

Задана математическая модель.
Уравнение для функции $T(x,t)$

$$c(T)\frac{\partial T}{\partial t} = \frac{\partial}{\partial x}(k(T)\frac{\partial T}{\partial x}) - \frac{2}{R}\alpha(x)T + \frac{2T_0}{R}\alpha(x) \tag{1}$$

Краевые условия:

$$\begin{cases} t = 0, T(x,0) = T_0 \\ x = 0, -k(T(0))\frac{\partial T}{\partial x} = F_0 \\ x = l, -k(T(l))\frac{\partial T}{\partial x} = \alpha_N(T(l) - T_0) \end{cases} \tag{2}$$

В обозначениях уравнения лекции:

$$p(x) = \frac{2}{R}\alpha(x) \tag{3}$$

$$f(u) = f(x) = \frac{2T_0}{R}\alpha(x) \tag{4}$$

Разностная схема с разностным краевым условие при $x = 0$:

$$(\frac{h}{8}\,\widehat{c_{\frac{1}{2}}} + \frac{h}{4}\,\widehat{c_0} + \widehat{X_{\frac{1}{2}}}\,\frac{\tau}{h} + \frac{\tau h}{8}p_{\frac{1}{2}} + \frac{\tau h}{4}p_0)\,\widehat{y_0} + (\frac{h}{8}\,\widehat{c_{\frac{1}{2}}} - \widehat{X_{\frac{1}{2}}}\,\frac{\tau}{h} + \frac{\tau h}{8}p_{\frac{1}{2}})\,\widehat{y_1} =$$

$$= \frac{h}{8}\,\widehat{c_{\frac{1}{2}}}\,(y_0 + y_1) + \frac{h}{4}\,\widehat{c_0}\,y_0 + \widehat{F}\,\tau + \frac{\tau h}{4}(\widehat{f_{\frac{1}{2}}} + \widehat{c_0}) \tag{5}$$

При получении разностного аналога краевого условия при $x = l$ учесть, что поток:

$$F_N = \alpha N(y_N - T_0), F_{N-\frac{1}{2}} = X_{N-\frac{1}{2}}\frac{y_{N-1} - y_N}{h} \tag{6}$$

Заданы начальные параметры:

- $k(T) = a_1(b_1 + c_1T^{m_1})$, Вт/см К
- $c(T) = a_2 + b_2T^{m_2} - \frac{c_2}{T^2}$, Дж/$cm^3$ К
- $a_1 = 0.0134, b_1 = 1, c_1 = 4.35 \cdot 10^{-4}, m_1 = 1$
- $a_2 = 2.049, b_2 = 0.563 \cdot 10^{-3}, c_2 = 0.528 \cdot 10^5, m_2 = 1$
- $\alpha(x) = \frac{c}{x-d}, \alpha_0 = 0.05$ Вт/cm^2 К, $\alpha_N = 0.01$ Вт/cm^2 К

- $l = 10$ см
- $T_0 = 300K$
- $R = 0.5$ см
- $F(t) = 50$ Вт/cm^2

0.2 Реализация

```
using System;
using System.Collections.Generic;

namespace src
{
        class Conditions
        {
                static public double[] GetLeftConditions(List<double> T)
                {
                        double c_plus =
                           Functions.ApproximationPlus(Functions.c, T[0],
                           Constants.t);
                        double k_plus =
                           Functions.ApproximationPlus(Functions.k, T[0],
                           Constants.t);

                        double K0 = Constants.h / 8 * c_plus + Constants.h / 4
                           * Functions.c(T[0]) + Constants.t / Constants.h *
                           k_plus +
                                        Constants.t * Constants.h / 8 *
                                           Functions.p(Constants.h / 2) +
                                           Constants.t * Constants.h / 4 *
                                           Functions.p(0);

                        double M0 = Constants.h / 8 * c_plus - Constants.t /
                           Constants.h * k_plus + Constants.t * Constants.h /
                           8 * Functions.p(Constants.h / 2);

                        double P0 = Constants.h / 8 * c_plus * (T[0] + T[1]) +
                           Constants.h / 4 * Functions.c(T[0]) * T[0] +
                                        Constants.F0 * Constants.t +
                                           Constants.t * Constants.h / 8 * (3
                                           * Functions.f(0) +
                                           Functions.f(Constants.h));

                        double[] result = { K0, M0, P0 };
                        return result;
                }

                static public double[] GetRightConditions(List<double> T)
                {
                        double c_minus =
                           Functions.ApproximationMinus(Functions.c,
                           T[T.Count - 1], Constants.t);
                        double k_minus =
                           Functions.ApproximationMinus(Functions.k,
                           T[T.Count - 1], Constants.t);

                        double KN = Constants.h / 8 * c_minus + Constants.h /
                           4 * Functions.c(T[T.Count - 1]) + Constants.t /
                           Constants.h * k_minus +
                                        Constants.t * Constants.alphaN +
                                           Constants.t * Constants.h / 8 *
                                           Functions.p(Constants.l -
                                           Constants.h / 2) +
                                        Constants.t * Constants.h / 4 *
                                           Functions.p(Constants.l);
```

```
                double MN = Constants.h / 8 * c_minus - Constants.t /
                    Constants.h * k_minus +
                                Constants.t * Constants.h / 8 *
                                    Functions.p(Constants.l -
                                    Constants.h / 2);

                double PN = Constants.h / 8 * c_minus * (T[T.Count -
                    1] + T[T.Count - 2]) + Constants.h / 4 *
                    Functions.c(T[T.Count - 1]) * T[T.Count - 1] +
                                Constants.t * Constants.alphaN *
                                    Constants.T0 + Constants.t *
                                    Constants.h / 4 *
                                    (Functions.f(Constants.l) +
                                    Functions.f(Constants.l -
                                    Constants.h / 2));

                double[] result = {KN, MN, PN};
                return result;
            }

            static public List<double> GetNewT(List<double> T)
            {
                double[] cond = GetLeftConditions(T);
                double K0 = cond[0], M0 = cond[1], P0 = cond[2];

                cond = GetRightConditions(T);
                double KN = cond[0], MN = cond[1], PN = cond[2];

                List<double> xi = new List<double>();
                xi.Add(0); xi.Add(-M0 / K0);

                List<double> eta = new List<double>();
                eta.Add(0); eta.Add(P0 / K0);

                double x = Constants.h, Tn, denominator, next_xi,
                    next_eta;
                int n = 1;

                while (x + Constants.h < Constants.l)
                {
                        Tn = T[n];
                        denominator = (Functions.B(x, Tn) -
                            Functions.A(Tn) * xi[n]);

                        next_xi = Functions.D(Tn) / denominator;
                        next_eta = (Functions.F(x, Tn) +
                            Functions.A(Tn) * eta[n]) / denominator;

                        xi.Add(next_xi);
                        eta.Add(next_eta);

                        n++;
                        x += Constants.h;
                }

                List<double> T_new = new List<double>();
                for (int i = 0; i < n + 1; i++)
                        T_new.Add(0);

                T_new[n] = (PN - MN * eta[n]) / (KN + MN * xi[n]);

                for (int i = n - 1; i > -1; i--)
```

```
                                T_new[i] = xi[i + 1] * T_new[i + 1] + eta[i +
                                    1];

                        return T_new;
                }
        }
}
```

```
        namespace src
        {
                public static class Constants
                {
                        public const double a1 = 0.0134;
                        public const double b1 = 1;
                        public const double c1 = 4.35e-4;
                        public const double m1 = 1;
                        public const double a2 = 2.049;
                        public const double b2 = 0.563e-3;
                        public const double c2 = 0.528e5;
                        public const double m2 = 1;
                        public const double alpha0 = 0.05;
                        public const double alphaN = 0.01;
                        public const double l = 10;
                        public const double T0 = 300;
                        public const double R = 0.5;
                        public const double F0 = 50;
                        public const double h = 1e-3;
                        public const double t = 1;
                        public const double eps = 1e-2;
                }
        }
```

```
using System;

namespace src
{
        class Functions
        {
                static public double k(double T)
                {
                        return Constants.a1 * (Constants.b1 + Constants.c1 *
                            Math.Pow(T, Constants.m1));
                }

                static public double c(double T)
                {
                        return Constants.a2 + Constants.b2 * Math.Pow(T,
                            Constants.m2) - (Constants.c2 / Math.Pow(T, 2));
                }

                static public double alpha(double x)
                {
                        double d = (Constants.alphaN * Constants.l) /
                            (Constants.alphaN - Constants.alpha0);
                        double c = -Constants.alpha0 * d;
                        return c / (x - d);
                }

                static public double p(double x)
                {
                        return alpha(x) * 2 / Constants.R;
                }
```

```
                static public double f(double x)
                {
                        return alpha(x) * 2 * Constants.T0 / Constants.R;
                }

                static public double ApproximationPlus(Func<double, double> f,
                   double n, double step)
                {
                        return (f(n) + f(n + step)) / 2;
                }

                static public double ApproximationMinus(Func<double, double>
                   f, double n, double step)
                {
                        return (f(n) + f(n - step)) / 2;
                }

                static public double A(double T)
                {
                        return Constants.t / Constants.h *
                           ApproximationMinus(k, T, Constants.t);
                }

                static public double D(double T)
                {
                        return Constants.t / Constants.h *
                           ApproximationPlus(k, T, Constants.t);
                }

                static public double B(double x, double T)
                {
                        return A(T) + D(T) + Constants.h * c(T) + Constants.h
                           * Constants.t * p(x);
                }

                static public double F(double x, double T)
                {
                        return Constants.h * Constants.t * f(x) + T *
                           Constants.h* c(T);
                }


        }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.IO;

namespace src
{
        class Program
        {
                static void SaveData(List<double> res, string fileName)
                {
                        FileStream file = new FileStream(fileName,
                           FileMode.Create, FileAccess.Write);
                        StreamWriter writer = new StreamWriter(file);

                        foreach (var elem in res)
                        {
                                writer.WriteLine(elem);
```

```
        }

        writer.Close();
        file.Close();
    }

    static void Task()
    {
        List<double> T = Enumerable.Range(1, (int)(Constants.l
            / Constants.h) + 1).Select(x =>
            (double)Constants.T0).ToList();
        List<double> TNew = Enumerable.Range(1,
            (int)(Constants.l / Constants.h) + 1).Select(x =>
            0d).ToList();
        List<double> TPrev = new List<double>();

        double currentMax = 1d, ti = 0;

        bool epsilonCondition = true;
        int file_i = 0;

        SaveData(T, $"results/data/task_1_{file_i++}.txt");

        while (epsilonCondition)
        {
            TPrev = T;
            currentMax = 1d;

            while (currentMax >= 1)
            {
                TNew = Conditions.GetNewT(T);
                currentMax = Math.Abs((T[0] - TNew[0])
                    / TNew[0]);

                var arr_d = T.Zip(TNew, (T_i, Tnew_i)
                    => Math.Abs(T_i - Tnew_i) /
                    Tnew_i);
                currentMax = arr_d.Max();

                TPrev = TNew;
            }

            SaveData(TNew,
                $"results/data/task_1_{file_i++}.txt");

            ti += Constants.t;

            epsilonCondition = false;
            int flag = T.Zip(TNew, (T_i, Tnew_i) =>
                Math.Abs(T_i - Tnew_i) /
                Tnew_i).Count(elem => elem >
                Constants.eps);
            if (flag != 0)
                epsilonCondition = true;

            T = TNew;

            // List<double> ys = new List<double>();
            // for (double s = 0; s < Constants.l / 3d> ;
                s += 0.1)
            // {
            //      Console.WriteLine(s);
            //      ys.Add(TNew[(int)(s / Constants.h)]);
```

```
                                // }
                                // SaveData(ys,
                                    $"results/data/task_2_{file_i++}.txt");
                        }
                }

                static void Main(string[] args)
                {
                        // Output with point.
                        System.Threading.Thread.CurrentThread.CurrentCulture =
                           new System.Globalization.CultureInfo("en-US");
                        Task();
                }
        }
}
```

0.3 Экспериментальная часть

Результаты работы программы

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

2. График зависимости температуры $T(x, t_m)$ от координаты x при нескольких фкисированных значених времени t_m при заданных выше параметрах.