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

Тема Модели на основе ДУ в частных производных с краевыми условиями II и III рода

Студент Якуба Д. В.

Группа ИУ7-63Б

Оценка (баллы) ______________

Преподаватель Градов В. М.

Москва
2021 г.

Лабораторная работа по теме «Модели на основе ДУ в частных производных с краевыми условиями II и III рода»

Тема:

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

Цель работы:

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

Задание:

1. Задана математическая модель.

Уравнение для функции $T(x,t)$

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

Краевые условия:

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

В обозначениях уравнения лекции

$$p(x) = \frac{2}{R}\alpha(x), f(u) \equiv f(x) = \frac{2T_0}{R}\alpha(x)$$

2. 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 при $x = 0$. Для этого надо проинтегрировать на отрезке $\left[x_{N-1/2}, x_N\right]$ выписанное выше уравнение и учесть, что поток $\widehat{F_N} = \alpha_N(\widehat{y_N} - T_0)$, а $\widehat{F_{N-1/2}} = \widehat{\chi_{N-1/2}}\frac{\widehat{y_{N-1}} - \widehat{y_N}}{h}$

3. Значение параметров для отладки:

$k(T) = a_1(b_1 + c_1 T^{m_1})$, Вт/см К,

$c(T) = a_2 + b_2\ T^{m_2} - \dfrac{c_2}{T^2}$, Дж/см^{3}К.

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.

$\alpha(x) = \dfrac{c}{x-d}$,

$\alpha_0 = 0.05$ Вт/см^2 К,

$\alpha_N = 0.01$ Вт/см^2 К,

$l = 10$ см,

$T_0 = 300$К,

$R = 0.5$ см,

$F(t) = 50$ Вт/см^2 (для отладки принять постоянным).

Результат

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

Пусть $u \equiv T$, $F = -k(u)\frac{\partial u}{\partial x}$.

В таком случае уравнение

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

При факте того, что

$$p(x) = \frac{2}{R}\alpha(x), f(u) \equiv f(x) = \frac{2T_0}{R}\alpha(x)$$

Примет вид:

$$c(u)\frac{\partial u}{\partial t} = -\frac{\partial F}{\partial x} - p(x)u + f(u)$$

Проинтегрируем данное выражение:

$$\int\limits_{x_{N-1/2}}^{x_N} \mathrm{dx} \int\limits_{t_m}^{t_{m+1}} c(u)\frac{\partial u}{\partial t}dt = -\int\limits_{t_m}^{t_{m+1}} dt \int\limits_{x_{N-1/2}}^{x_N} \frac{\partial F}{\partial x}dx - \int\limits_{x_{N-1/2}}^{x_N} dx \int\limits_{t_m}^{t_{m+1}} p(x)udt + \int\limits_{x_{N-1/2}}^{x_N} dx \int\limits_{t_m}^{t_{m+1}} f(u)dt$$

То есть

$$\int\limits_{x_{N-1/2}}^{x_N} \hat{c}(\hat{u}-u)dx = \int\limits_{t_m}^{t_{m+1}} \left(F_{N-1/2} - F_N\right)dt - \int\limits_{x_{N-1/2}}^{x_N} p\hat{u}\tau dx + \int\limits_{x_{N-1/2}}^{x_N} \hat{f}\tau dx$$

Для первого интеграла в правой части воспользуемся методом правых прямоугольников, а для остальных – методом трапеций.

$$\frac{h}{4}\left[\widehat{c_N}(\widehat{y_N} - y_N) + \widehat{c_{N-1/2}}\left(\widehat{y_{N-1/2}} - y_{N-1/2}\right)\right] = -\left(\widehat{F_N} - \widehat{F_{N-1/2}}\right)\tau - \left(p_N\widehat{y_N} + p_{N-1/2}\widehat{y_{N-1/2}}\right)\tau\frac{h}{4} + \left(\widehat{f_N} + \widehat{f_{N-1/2}}\right)\tau\frac{h}{4}$$

Учитывая $\widehat{F_N} = \alpha_N(\widehat{y_N} - T_0)$, $\widehat{F_{N-1/2}} = \widehat{\chi_{N-1/2}}\frac{\widehat{y_{N-1}}-\widehat{y_N}}{h}$, $\widehat{y_{N-1/2}} = \frac{\widehat{y_N}+\widehat{y_{N-1}}}{2}$, $y_{N-1/2} = \frac{y_N+y_{N-1}}{2}$ получим:

$$\frac{h}{4}\left[\widehat{c_N}(\widehat{y_N} - y_N) + \widehat{c_{N-1/2}}\left(\frac{\widehat{y_N} + \widehat{y_{N-1}}}{2} - \frac{y_N + y_{N-1}}{2}\right)\right] = -\left(\alpha_N(\widehat{y_N} - T_0)\widehat{\chi_{N-1/2}}\frac{\widehat{y_{N-1}} - \widehat{y_N}}{h}\right)\tau - \left(p_N\widehat{y_N} + p_{N-1/2}\frac{\widehat{y_N} + \widehat{y_{N-1}}}{2}\right)\tau\frac{h}{4} + \left(\widehat{f_N} + \widehat{f_{N-1/2}}\right)\tau\frac{h}{4}$$

Приведём данное уравнение к разностному аналогу краевого условия при $x = l$:

$$\left(\widehat{c_N}\frac{h}{4} + \widehat{c_{N-\frac{1}{2}}}\frac{h}{8} + \alpha_N\tau + \widehat{\chi_{N-\frac{1}{2}}}\frac{\tau}{h} + p_N\frac{\tau h}{4} + p_{N-\frac{1}{2}}\frac{\tau h}{8}\right)\widehat{y_N} + \left(\widehat{c_{N-\frac{1}{2}}}\frac{h}{8} - \widehat{\chi_{N-\frac{1}{2}}}\frac{\tau}{h} + p_{N-\frac{1}{2}}\frac{\tau h}{8}\right)\widehat{y_{N-1}} = \frac{h}{4}\left(\widehat{c_N}y_N + \widehat{c_{N-\frac{1}{2}}}\frac{y_N + y_{N-1}}{2}\right) + \alpha_N T_0\tau + \left(\widehat{f_N} + \widehat{f_{N-\frac{1}{2}}}\right)\tau\frac{h}{4}$$

Требуется найти левые и правые коэффициенты:

$$\begin{cases} \widehat{K_0}\widehat{y_0} + \widehat{M_0}\widehat{y_1} = \widehat{p_0} \\ \widehat{A_n}\widehat{y_{n-1}} - \widehat{B_n}\widehat{y_n} + \widehat{D_n}\widehat{y_{n+1}} = -\widehat{F_n} \\ \widehat{K_n}\widehat{y_N} + \widehat{M_{N-1}}\widehat{y_{N-1}} = P_N \end{cases}$$

Данная система решается методом итераций:

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

2. График зависимости температуры $T(x, t_m)$ от координаты x при нескольких фиксированных значениях времени t_m (аналогично рисунку в лекции) при заданных выше параметрах.

На рисунке представлены графики зависимости температуры от координаты при фиксированных t.

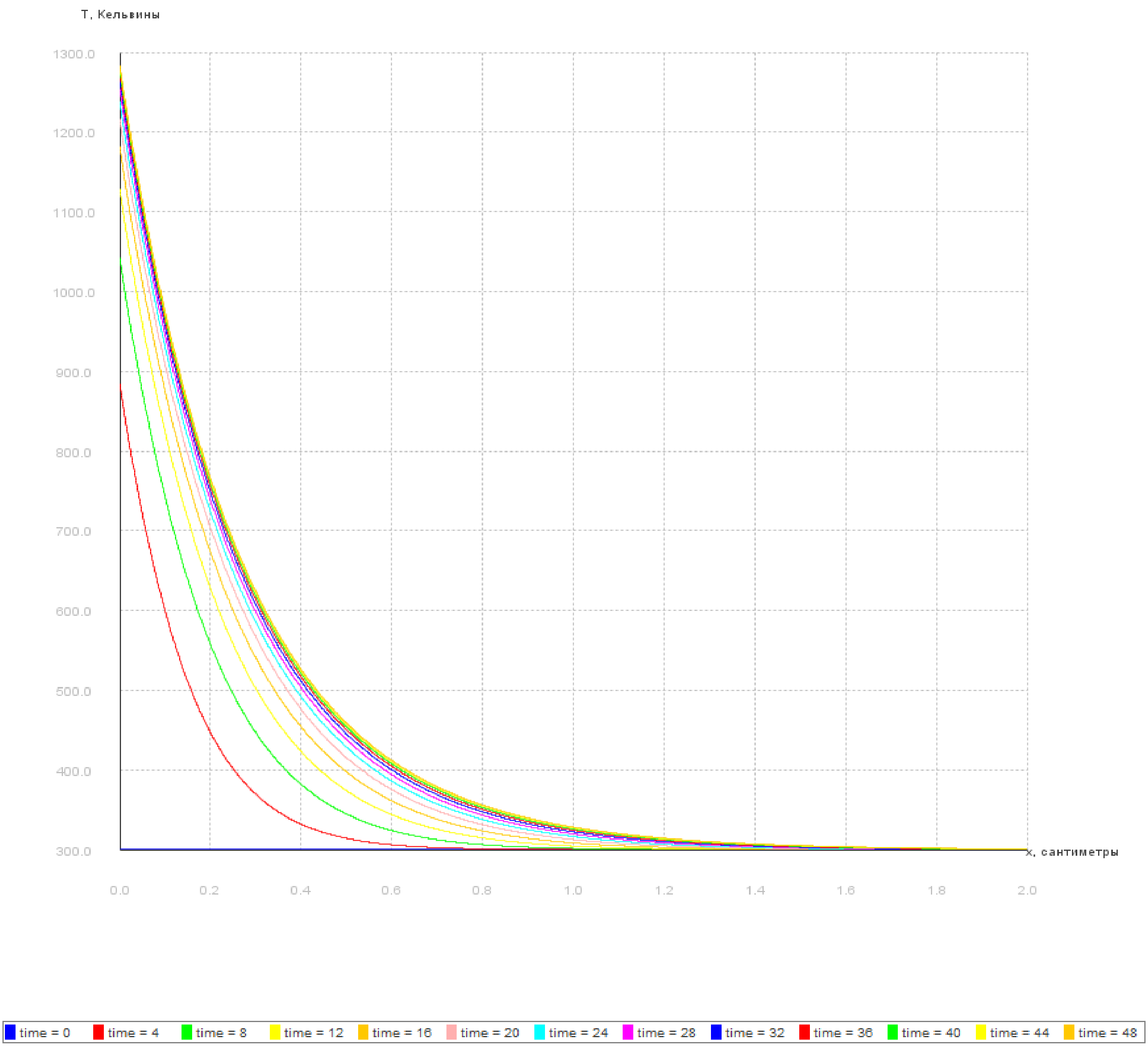
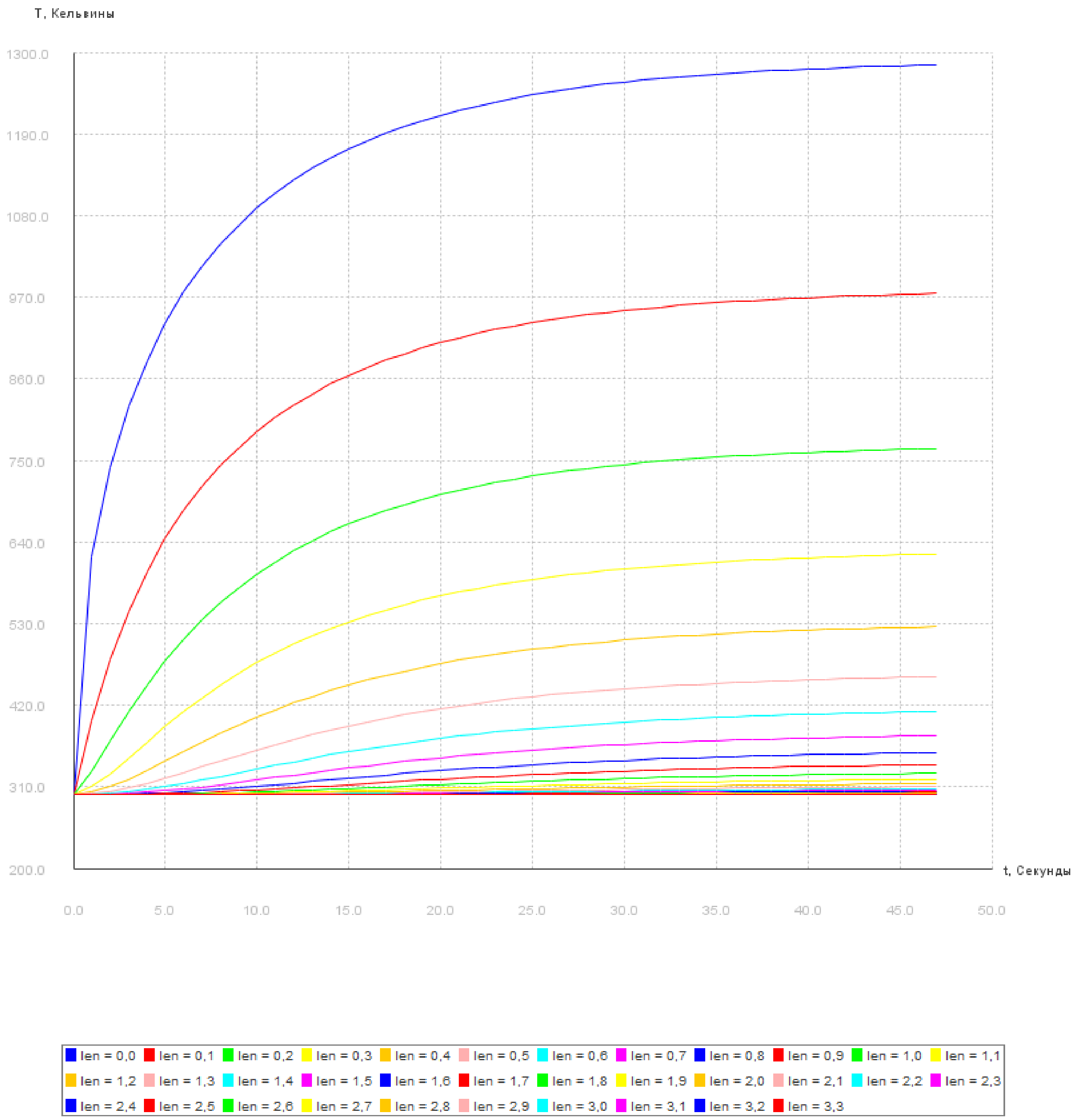


График зависимости $T(x_n, t)$ при нескольких фиксированных значениях координаты x_n. Обязательно представить случай $n = 0$, то есть $x = x_0 = 0$.

На рисунке представлены графики зависимости температуры от времени при фиксированных x.



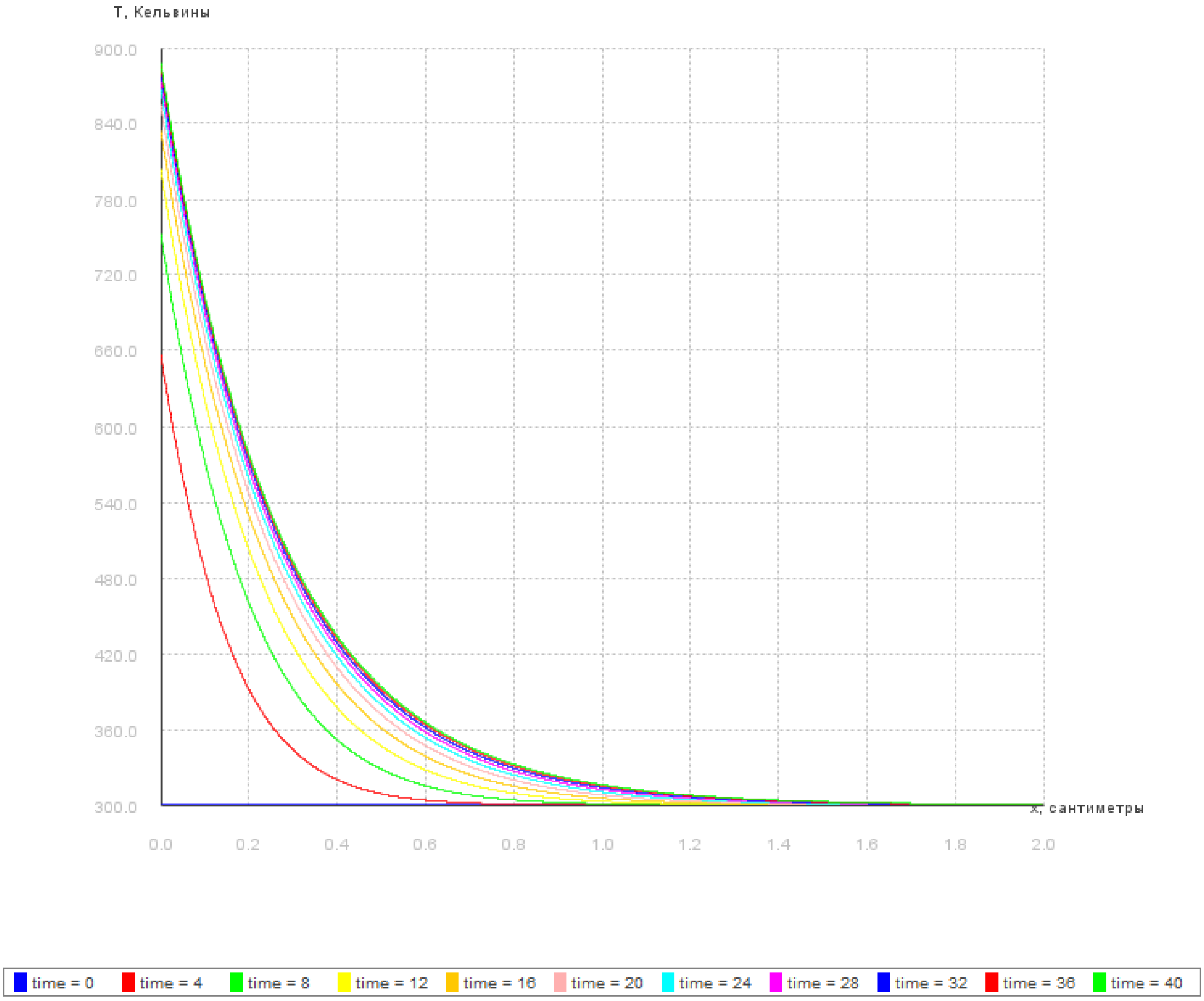
Контрольные вопросы

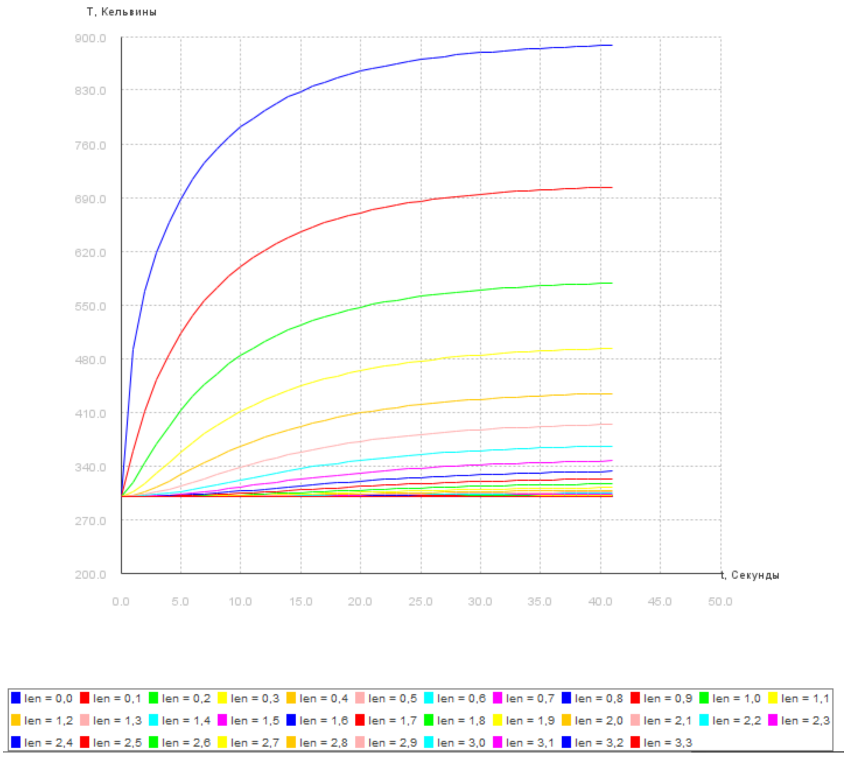
1. Приведите результаты тестирования программы (графики, общие соображения, качественный анализ)

Ответ:

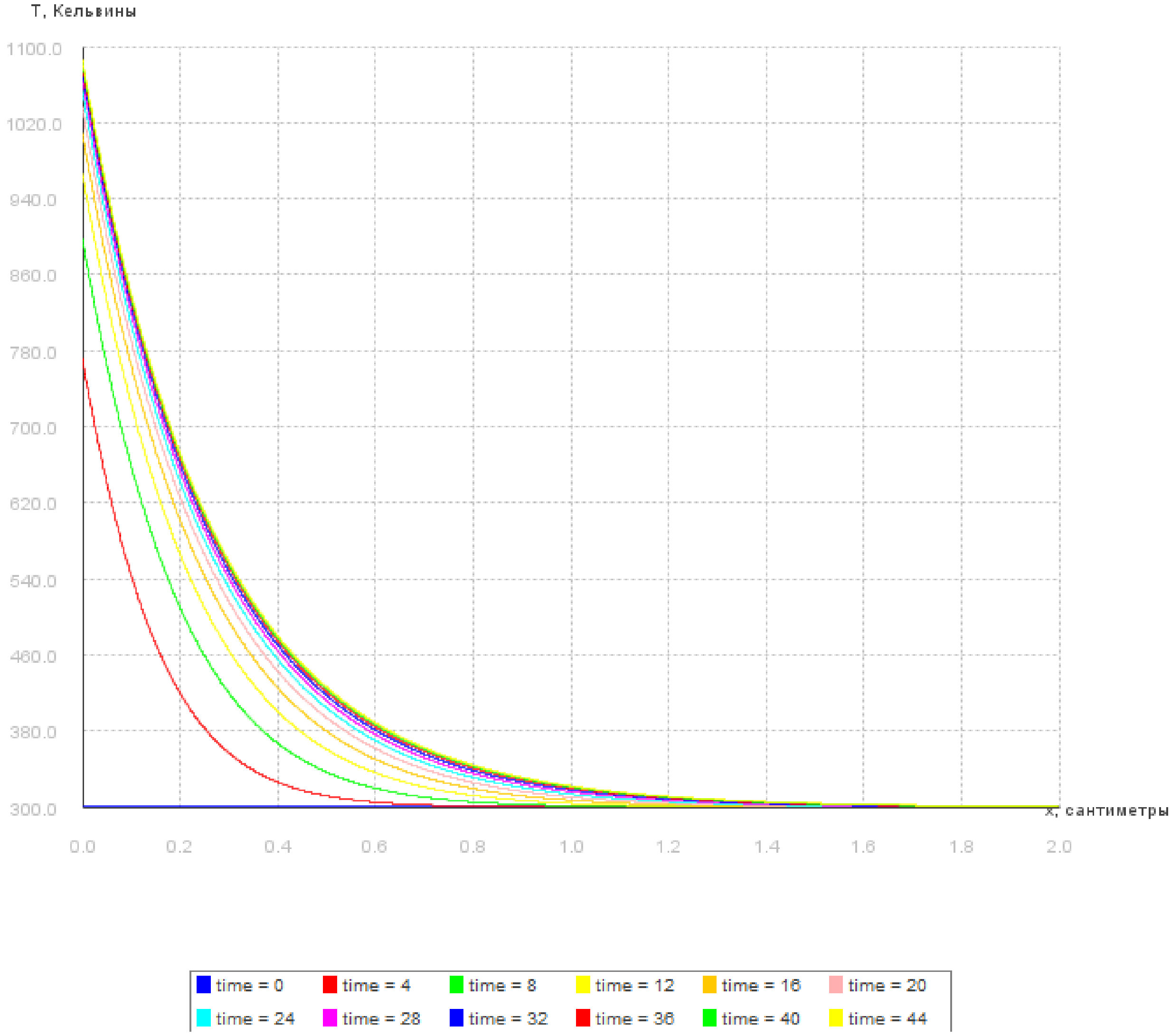
При постепенном увеличении теплового потока можем видеть возрастание значений температур:

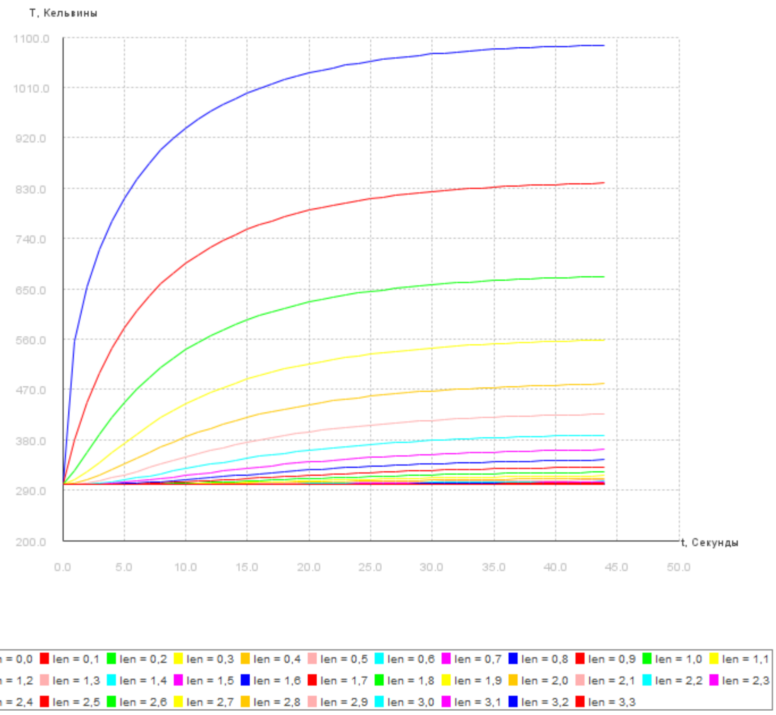
$$F_0 = 30$$



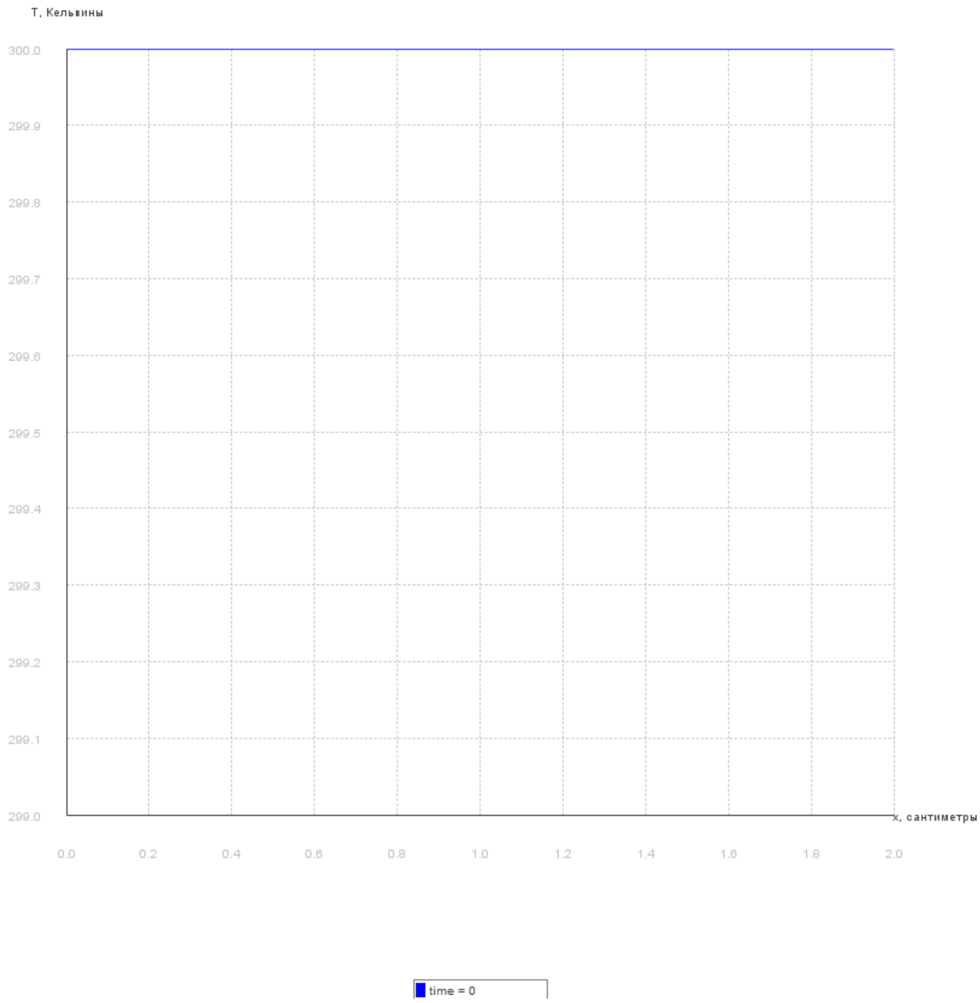

T, Кельвины
900.0
830.0
760.0
690.0
620.0
550.0
480.0
410.0
340.0
270.0
200.0
t, Секунды
0.0
5.0
10.0
15.0
20.0
25.0
30.0
35.0
40.0
45.0
50.0
len = 0,0 len = 0,1 len = 0,2 len = 0,3 len = 0,4 len = 0,5 len = 0,6 len = 0,7 len = 0,8 len = 0,9 len = 1,0 len = 1,1
len = 1,2 len = 1,3 len = 1,4 len = 1,5 len = 1,6 len = 1,7 len = 1,8 len = 1,9 len = 2,0 len = 2,1 len = 2,2 len = 2,3
len = 2,4 len = 2,5 len = 2,6 len = 2,7 len = 2,8 len = 2,9 len = 3,0 len = 3,1 len = 3,2 len = 3,3

$$F_0 = 40$$


T, Кельвины
1100.0
1020.0
940.0
860.0
780.0
700.0
620.0
540.0
460.0
380.0
300.0
x, сантиметры
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
2.0
time = 0 time = 4 time = 8 time = 12 time = 16 time = 20
time = 24 time = 28 time = 32 time = 36 time = 40 time = 44

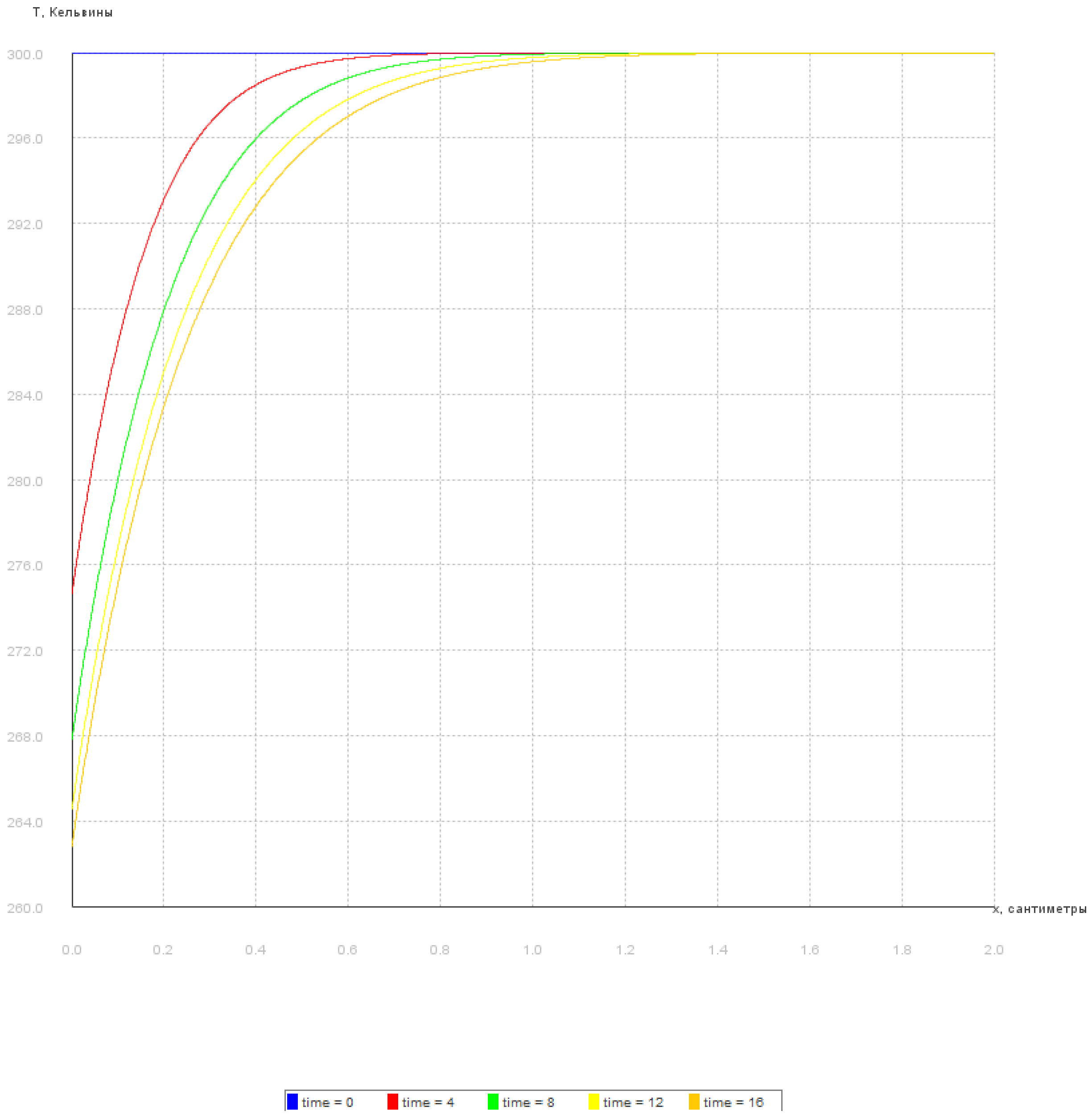


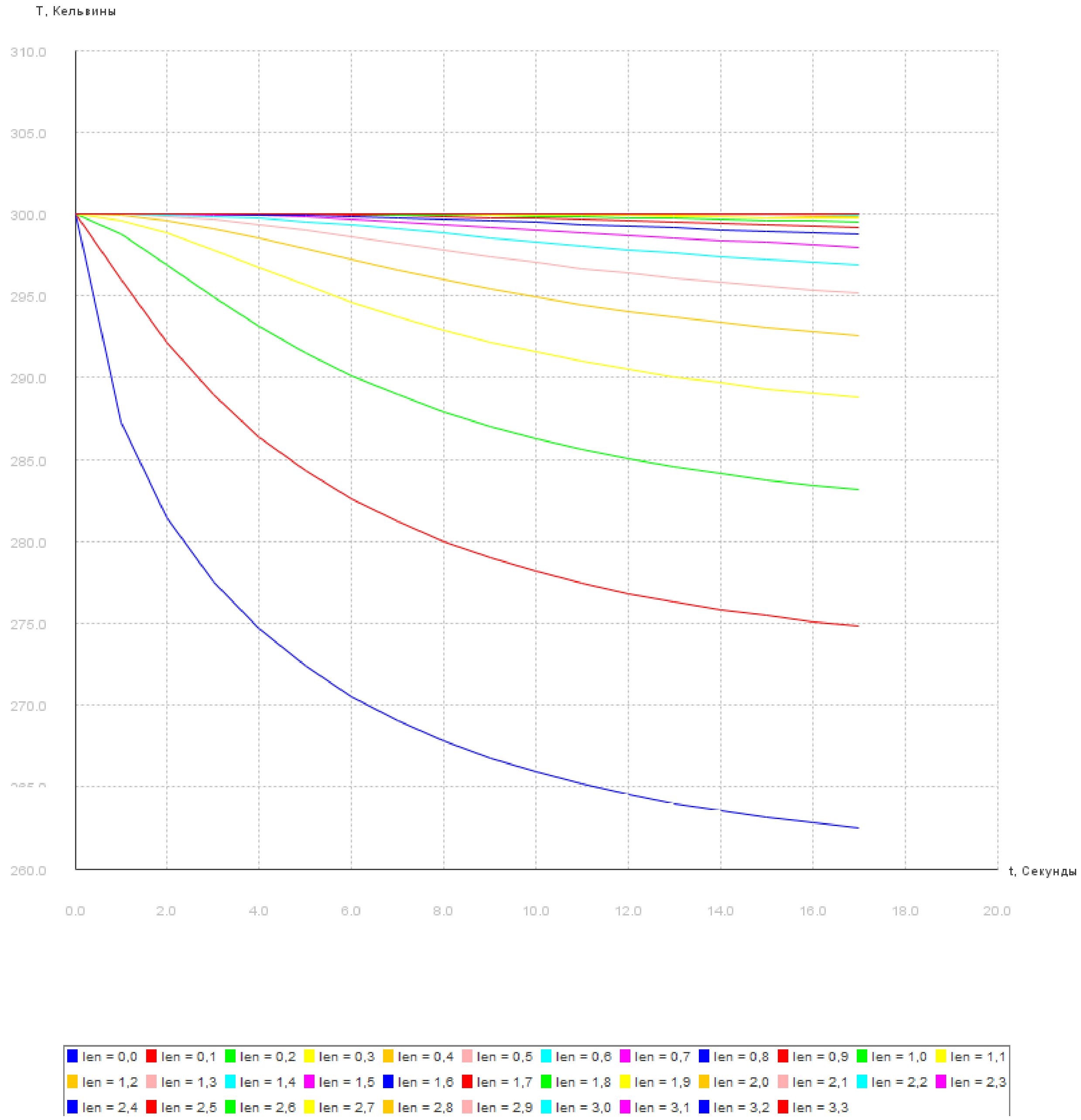
При тепловом потоке, равном нулю ($F_0 = 0$), температура не должна изменяться:



При указании отрицательного значения теплового потока ($F_0 = -2$, например) должно происходить охлаждение слева:

```
val fZero = -2.0
```





Код программы

```
import kotlin.math.abs
import kotlin.math.pow
import javax.swing.*
import org.math.plot.Plot2DPanel

class Parameters()
{
    val a1 = 0.0134
    val b1 = 1.0
    val c1 = 4.35e-4
    val m1 = 1.0
    val a2 = 2.049
    val b2 = 0.563e-3
    val c2 = 0.528e5
    val m2 = 1.0
    val alphaZero = 0.05
```

```
    val alphaN = 1e-2
    val l = 10.0
    val tZero = 300.0
    val r = 5e-1
    val fZero = 50.0
    val h = 1e-3
    val t = 1.0
    val epsilon = 1e-3
}

val parameters = Parameters()

fun plusApprox(function: (Double) -> Double, n: Double, step: Double): Double
{
    return (function(n) + function(n + step)) / 2
}

fun minusApprox(function: (Double) -> Double, n: Double, step: Double): Double
{
    return (function(n) + function(n - step)) / 2
}

val kFun = { x: Double -> parameters.a1 * (parameters.b1 + parameters.c1 *
parameters.m1.pow(x)) }

val cFun = { x: Double -> parameters.a2 + parameters.b2 * x.pow(parameters.m2) -
(parameters.c2 / x.pow(2)) }

fun alphaFun(x: Double): Double
{
    val s1 = (parameters.alphaN * parameters.l) / (parameters.alphaN -
parameters.alphaZero)
    val s2 = -parameters.alphaZero * s1
    return s2 / (x - s1)
}

val pFun = { x: Double -> alphaFun(x) * 2 / parameters.r }

val fFun = { x: Double -> alphaFun(x) * 2 * parameters.tZero / parameters.r }

val aAFun = { x: Double -> parameters.t / parameters.h * minusApprox(kFun, x,
parameters.t) }

val dDFun = { x: Double -> parameters.t / parameters.h * plusApprox(kFun, x,
parameters.t) }

val bBFun =
    { x: Double, t: Double -> aAFun(t) + dDFun(t) + parameters.h * cFun(t) +
parameters.h * parameters.t * pFun(x) }

val fFFun = { x: Double, t: Double -> parameters.h * parameters.t * fFun(x) + t *
parameters.h * cFun(t) }

fun leftexitConditions(tList: MutableList<Double>): Triple<Double, Double, Double>
{
    val c = plusApprox(cFun, tList[0], parameters.t)
    val k = plusApprox(kFun, tList[0], parameters.t)

    val kZero =
        parameters.h / 8 * c + parameters.h / 4 * cFun(tList[0]) +
                parameters.t / parameters.h * k + parameters.t * parameters.h / 8 *
pFun(
```

```
            parameters.h / 2
        ) + parameters.t * parameters.h / 4 * pFun(0.0)

    val mZero =
        parameters.h / 8 * c - parameters.t / parameters.h * k +
                parameters.t * parameters.h / 8 * pFun(parameters.h / 2)

    val pZero =
        parameters.h / 8 * c * (tList.first() + tList[1]) +
                parameters.h / 4 * cFun(tList.first()) * tList.first() +
                parameters.fZero * parameters.t + parameters.t * parameters.h / 8 *
(3 * fFun(
            0.0
        ) + fFun(parameters.h))

    return Triple(kZero, mZero, pZero)
}

fun rightexitConditions(tList: MutableList<Double>): Triple<Double, Double, Double>
{
    val c = minusApprox(cFun, tList.last(), parameters.t)
    val k = minusApprox(kFun, tList.last(), parameters.t)

    val kN =
        parameters.h / 8 * c + parameters.h / 4 * cFun(tList.last()) +
                parameters.t / parameters.h * k +
                parameters.t * parameters.alphaN +
                parameters.t * parameters.h / 8 * pFun(
            parameters.l - parameters.h / 2
        ) + parameters.t * parameters.h / 4 * pFun(parameters.l)

    val mN =
        parameters.h / 8 * c - parameters.t / parameters.h * k +
                parameters.t * parameters.h / 8 * pFun(parameters.l - parameters.h
/ 2)

    val pN =
        parameters.h / 8 * c * (tList.last() + tList[tList.size - 2]) +
                parameters.h / 4 * cFun(tList.last()) * tList.last() +
                parameters.t * parameters.alphaN * parameters.tZero +
                parameters.t * parameters.h / 4 * (fFun(
            parameters.l
        ) + fFun(parameters.l - parameters.h / 2))

    return Triple(kN, mN, pN)
}

fun formNewTList(list: MutableList<Double>): MutableList<Double>
{
    val zeroTriple = leftexitConditions(list)
    val nTriple = rightexitConditions(list)

    val xiList: MutableList<Double> = mutableListOf(0.0, -zeroTriple.second /
zeroTriple.first)
    val etaList: MutableList<Double> = mutableListOf(0.0, zeroTriple.third /
zeroTriple.first)

    var curX = parameters.h
    var curN = 1

    while (curX + parameters.h < parameters.l)
    {
```

```
        val curT = list[curN]
        val dm = bBFun(curX, curT) - aAFun(curT) * xiList[curN]

        xiList.add(dDFun(curT) / dm)
        etaList.add((fFFun(curX, curT) + aAFun(curT) * etaList[curN]) / dm)

        curX += parameters.h
        curN++
    }

    val outT = mutableListOf<Double>()
    for (i in 0..curN)
        outT.add(0.0)

    outT[curN] =
        (nTriple.third - nTriple.second * etaList[curN]) / (nTriple.first +
nTriple.second * xiList[curN])

    for (i in curN - 1 downTo 0)
        outT[i] = xiList[i + 1] * outT[i + 1] + etaList[i + 1]

    return outT
}

fun simpleIteration(): Pair<MutableList<MutableList<Double>>, Double>
{
    var tList = mutableListOf<Double>()
    var newTList = mutableListOf<Double>()

    for (i in 0..(parameters.l / parameters.h).toInt())
    {
        tList.add(parameters.tZero)
        newTList.add(0.0)
    }

    val outList = mutableListOf(tList)

    var curT = 0.0
    var exitCondition = true
    while (exitCondition)
    {
        var tempList = tList
        var max = 1.0

        while (max >= 1)
        {
            newTList = formNewTList(tempList)
            max = abs((tList.first() - newTList.first()) / newTList.first())

            for (ind in tList.indices)
            {
                if (abs((tList[ind] - newTList[ind]) / newTList[ind]) > max)
                    max = abs((tList[ind] - newTList[ind]) / newTList[ind])
            }

            tempList = newTList
        }

        outList.add(newTList)
        curT += parameters.t

        exitCondition = false
```

```
        for (ind in tList.indices)
        {
            if (abs(tList[ind] - newTList[ind]) / newTList[ind] >
parameters.epsilon)
            {
                exitCondition = true
                break
            }
        }

        tList = newTList
    }

    return Pair(outList, curT)
}

fun main()
{
    val out = simpleIteration()

    val xList = mutableListOf<Double>()
    var i = 0.0
    while (i < 2)
    {
        xList.add(i)
        i += parameters.h
    }

    val plot = Plot2DPanel()
    for (curY in out.first.indices)
    {
        if (curY % 4 == 0)
            plot.addLinePlot("time = $curY", xList.toDoubleArray(),
out.first[curY].toDoubleArray())
    }

    val frame = JFrame("T(x)")
    plot.addLegend("SOUTH")

    plot.setAxisLabel(0, "x, сантиметры")
    plot.setAxisLabel(1, "T, Кельвины")
    frame.setSize(1000, 1000)
    frame.contentPane = plot
    frame.isVisible = true

    val secList = mutableListOf<Double>()
    i = 0.0
    while (i < out.second && secList.size != out.first.size)
    {
        secList.add(i)
        i += parameters.t
    }

    val sPlot = Plot2DPanel()

    var k = 0.0
    while (k < parameters.l / 3)
    {
        val curList = mutableListOf<Double>()
        for (curF in out.first)
            curList.add((curF[(k / parameters.h).toInt()]))
        sPlot.addLinePlot("len = %.1f".format(k), secList.toDoubleArray(),
```

```
curList.toDoubleArray())

        k += 0.1
    }
    sPlot.addLegend("SOUTH")
    val newFrame = JFrame("T(t)")
    newFrame.setSize(1000, 1000)
    sPlot.setAxisLabel(0, "t, Секунды")
    sPlot.setAxisLabel(1, "T, Кельвины")
    newFrame.contentPane = sPlot
    newFrame.isVisible = true
}
```