**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 дифференциальных уравнений в частных производных с краевыми условиями II и III рода.

**Студент** Мирзоян С. А.

**Группа** ИУ7-65Б

**Оценка (баллы)** _______________

**Преподаватель** Градов В.М.

Москва
2020 г.

**Цель работы**. Получение навыков проведения исследований компьютерной математической модели, построенной на квазилинейном уравнении параболического типа.

Исследование проводится с помощью программы, созданной в лабораторной **работе №4**.

**Исходные данные.**

1. Значения параметров (все размерности согласованы)

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

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

Порядки величин (как в лаб. работе №4):

$a_1$=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.

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

Порядки величин (как в лаб. работе №4):

$\alpha_0 = 0.05$ Вт/см$^2$ К,

$\alpha_N = 0.01$ Вт/см$^2$ К,

$l =$ 10 см,

$T_0 =$300К,

$R = 0.5$ см,

2. Поток тепла $F(t)$ при $x = 0$

$F(t) = \frac{F_{max}}{t_{max}} t \exp(-(t/t_{max} - 1))$, где $F_{max}, t_{max}$ - амплитуда импульса потока и время её достижения (Вт/см$^2$ и с).

## Листинг

```
from numpy import arange
import matplotlib.pyplot as plt

class Data:
    x0 = 0
    l = 10            # Длина стержня (cm)
    R = 0.5           # Радиус стержня (cm)
    Tenv = 300        # Температура окружающей среды (K)
    F0 = 100         # Плотность теплового потока (W / (cm^2 * K))
    k0 = 0.1          # Коэффициент теплопроводности в начале стержня (W / (cm * K))
    kN = 0.2          # Коэффициент теплопроводности в конце стержня (W / (cm * K))
    alpha0 = 1e-2     # Коэффициент теплоотдачи в начале стержня (W / (cm^2 * K))
    alphaN = 9e-2  # Коэффициент теплоотдачи в конце стержня (W / (cm^2 * K))
    h = 1e-2
    bk = (kN * l) / (kN - k0)
    ak = - k0 * bk
    b_alpha = (alphaN * l) / (alphaN - alpha0)
    a_alpha = - alpha0 * b_alpha


    @staticmethod
    def k(x):
        return Data.ak / (x - Data.bk)

    @staticmethod
    def alpha(x):
        return Data.a_alpha / (x - Data.b_alpha)

    @staticmethod
    def Xn_plus_half(x):
        return (2 * Data.k(x) * Data.k(x + Data.h)) / \
               (Data.k(x) + Data.k(x + Data.h))

    @staticmethod
    def Xn_minus_half(x):
        return (2 * Data.k(x) * Data.k(x - Data.h)) / \
```

```
                    (Data.k(x) + Data.k(x - Data.h))

        @staticmethod
        def p(x):
            return 2 * Data.alpha(x) / Data.R

        @staticmethod
        def f(x):
            return 2 * Data.alpha(x) / Data.R * Data.Tenv


    def thomas_algorithm(A, B, C, D, K0, M0, P0, KN, MN, PN):  # Tridiagonal matrix algorithm
        # Initial values
        xi = [None, - M0 / K0]
        eta = [None, P0 / K0]

        # Straight running
        for i in range(1, len(A)):
            x = C[i] / (B[i] - A[i] * xi[i])
            e = (D[i] + A[i] * eta[i]) / (B[i] - A[i] * xi[i])

            xi.append(x)
            eta.append(e)

        # print(xi)
        # print(eta)

        # Reverse running
        y = [(PN - MN * eta[-1]) / (KN + MN * xi[-1])]

        for i in range(len(A) - 2, -1, -1):
            y_i = xi[i + 1] *  y[0] + eta[i + 1]

            y.insert(0, y_i)

        return y



```

```python

    def left_boundary_conditions():
        X_half = Data.Xn_plus_half(Data.x0)
        p1 = Data.p(Data.x0 + Data.h)
        f1 = Data.f(Data.x0 + Data.h)

        p0 = Data.p(Data.x0)
        f0 = Data.f(Data.x0)

        p_half = (p0 + p1) / 2

        K0 = X_half + Data.h * Data.h * p_half / 8 + Data.h * Data.h * p0 / 4
        M0 = Data.h * Data.h * p_half / 8 - X_half
        P0 = Data.h * Data.F0 + Data.h * Data.h * (3 * f0 + f1) / 4

        return K0, M0, P0


    def right__boundary_conditions():
        X_half = Data.Xn_minus_half(Data.l)

        pN = Data.p(Data.l)
        pN1 = Data.p(Data.l - Data.h)
        fN = Data.f(Data.l)
        fN1 = (2 * Data.alpha(Data.l - Data.h)) / Data.R * Data.Tenv

        KN = - (X_half + Data.alphaN * Data.h) / Data.h - Data.h * (5 * pN + pN1) / 16
        MN = X_half / Data.h - Data.h * (pN + pN1) / 16
        PN = - Data.alphaN * Data.Tenv - Data.h * (3 * fN + fN1) / 8

        return KN, MN, PN


    def calc_coefficients():
        A = []
        B = []
        C = []
        D = []

        for i in arange(Data.x0, Data.l, Data.h):
```

```
        An = Data.Xn_minus_half(i) / Data.h
        Cn = Data.Xn_plus_half(i) / Data.h
        Bn = An + Cn + Data.p(i) * Data.h
        Dn = Data.f(i) * Data.h

        A.append(An)
        B.append(Bn)
        C.append(Cn)
        D.append(Dn)

    return A, B, C, D


if __name__ == "__main__":
    a, b, c, d = calc_coefficients()
    # print(a)
    # print(b)
    # print(c)
    # print(d)

    k0, m0, p0 = left_boundary_conditions()
    # print(k0)
    # print(m0)
    # print(p0)

    kN, mN, pN = right__boundary_conditions()
    # print(kN)
    # print(mN)
    # print(pN)

    T = thomas_algorithm(a, b, c, d, k0, m0, p0, kN, mN, pN)
    print(T)
    x = arange(Data.x0, Data.l, Data.h)

    plt.title('Heating the rod')
    plt.grid(True)
    plt.plot(x, T, 'r', linewidth=0.5)
    plt.xlabel("Length (cm)")
    plt.ylabel("Temperature (K)")

    plt.savefig("plot.png")
```

```

        plt.show()
```

**Ответы на вопросы:**

1. Провести исследование по выбору оптимальных шагов по времени $\tau$ и пространству $h$ . Шаги должны быть максимально большими при сохранении устойчивости разностной схемы и заданной точности расчета.

Рассмотреть влияние на получаемые результаты амплитуды импульса $F_{\max}$ и времени $t_{\max}$ (определяют крутизну фронтов и длительность импульса).

Точность расчета можно оценить разными способами.

1) Уменьшая шаги и наблюдая сходимость решений, как это делалось в лаб. работе №1.

2) Проверяя, соблюдается ли при выбранных $h,\tau$ баланс мощности после выхода на стационарное распределение температуры (в установившемся режиме), реализующееся при $F(t)$ =const, т.е. в этом режиме должно выполняться условие: подводимая мощность равна отводимой. Имеем

$$\pi R^2 (F_0 - F_N) = 2\pi R \int_0^l \int \alpha [T(x, t_M) - T_0] dx,$$

окончательно

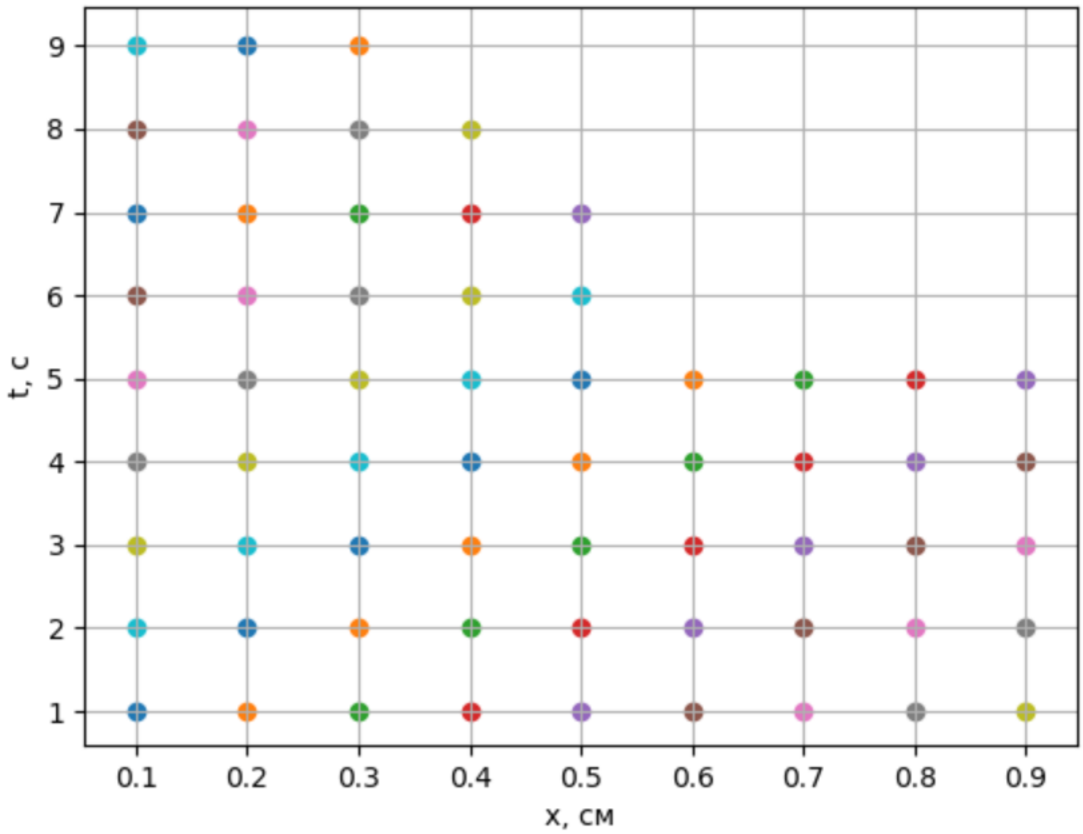
$$\left| \frac{F_0 - F_N}{\frac{2}{R} \int_0^l \int \alpha [T(x, t_M) - T_0] dx} - 1 \right| \le \varepsilon . \qquad (1)$$

Задать точность $\varepsilon$ примерно $10^{-2}$. Здесь $t_M$ - время выхода на стационарный режим, т.е. когда температура перестает меняться с заданной точностью (см. лаб. работу №4).

*Замечание.* Варьируя параметры задачи, следует иметь ввиду, что решения, в которых температура превышает значения примерно 2000К, физического смысла не имеют и практического интереса не представляют.

**Результат исследования:**

Так как параметры неравенства (1) не зависят от $\tau$, это значит, что, баланс мощности после выхода на стационарное распределение температуры не зависит от шага по времени. Это можно наблюдать на следующем графике




Можно заметить, что баланс при шаге h > 0.3 см не соблюдается.

Далее принимаем h = const = 0.3 и изменяем шаг по времени

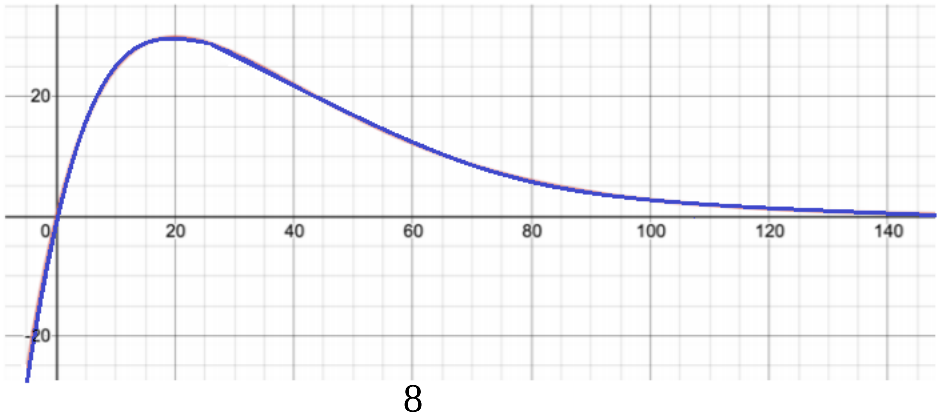


Рис.1 График функции $F(t) = \frac{F_{max}}{t_{max}}\ e^{-\frac{t}{t_{max-1}}}$, при $F_{max} = 30, t_{max} = 20$

| x,см | T(t_i=18c), tau = 9c | T(t_i=18c), tau = 6c | T(t_i=18c), tau = 3c | T(t_i=18c), tau = 1c | T(t_i=18c), tau = 0.5c | T(t_i=18c), tau = 0.25c |
|---|---|---|---|---|---|---|
| 0.00 | 698.6470 | 704.7200 | 711.0016 | 715.4807 | 716.6590 | 717.2604 |
| 0.30 | 418.3414 | 420.1297 | 421.8280 | 423.0352 | 423.3699 | 423.5453 |
| 0.60 | 336.0523 | 336.0652 | 335.7837 | 335.4113 | 335.2971 | 335.2377 |
| 0.90 | 311.1787 | 310.8767 | 310.3318 | 309.7716 | 309.5989 | 309.5070 |
| 1.20 | 303.5058 | 303.2774 | 302.9144 | 302.5578 | 302.4482 | 302.3897 |
| 1.50 | 301.1088 | 300.9848 | 300.8021 | 300.6329 | 300.5826 | 300.5560 |
| 1.80 | 300.3532 | 300.2950 | 300.2157 | 300.1482 | 300.1294 | 300.1197 |
| 2.10 | 300.1132 | 300.0881 | 300.0568 | 300.0330 | 300.0269 | 300.0239 |
| 2.40 | 300.0365 | 300.0263 | 300.0147 | 300.0070 | 300.0053 | 300.0044 |
| 2.70 | 300.0118 | 300.0078 | 300.0037 | 300.0014 | 300.0010 | 300.0008 |

Рис. 2.1: Таблица $T(x, 18)$.

| x,см | T(t_i=72c), tau = 9c | T(t_i=72c), tau = 6c | T(t_i=72c), tau = 3c | T(t_i=72c), tau = 1c | T(t_i=72c), tau = 0.5c | T(t_i=72c), tau = 0.25c |
|---|---|---|---|---|---|---|
| 0.00 | 475.2750 | 475.2583 | 475.0150 | 474.7235 | 474.6338 | 474.5865 |
| 0.30 | 377.2207 | 377.5177 | 377.6289 | 377.5903 | 377.5658 | 377.5512 |
| 0.60 | 334.5280 | 334.9302 | 335.2299 | 335.3617 | 335.3849 | 335.3951 |
| 0.90 | 315.6171 | 315.9582 | 316.2606 | 316.4326 | 316.4709 | 316.4893 |
| 1.20 | 307.1179 | 307.3534 | 307.5833 | 307.7299 | 307.7652 | 307.7826 |
| 1.50 | 303.2566 | 303.4000 | 303.5499 | 303.6530 | 303.6790 | 303.6920 |
| 1.80 | 301.4907 | 301.5704 | 301.6581 | 301.7218 | 301.7385 | 301.7469 |
| 2.10 | 300.6810 | 300.7220 | 300.7688 | 300.8043 | 300.8138 | 300.8187 |
| 2.40 | 300.3099 | 300.3296 | 300.3525 | 300.3703 | 300.3752 | 300.3777 |
| 2.70 | 300.1403 | 300.1491 | 300.1593 | 300.1673 | 300.1696 | 300.1707 |
| 3.00 | 300.0632 | 300.0668 | 300.0708 | 300.0740 | 300.0748 | 300.0753 |
| 3.30 | 300.0283 | 300.0296 | 300.0309 | 300.0319 | 300.0322 | 300.0323 |
| 3.60 | 300.0126 | 300.0130 | 300.0133 | 300.0134 | 300.0134 | 300.0134 |
| 3.90 | 300.0056 | 300.0056 | 300.0056 | 300.0055 | 300.0054 | 300.0054 |
| 4.20 | 300.0024 | 300.0024 | 300.0023 | 300.0022 | 300.0021 | 300.0021 |

Рис. 2.2: Таблица $T(x, 72)$.

| x,см | T(t_i=144c), tau = 9c | T(t_i=144c), tau = 6c | T(t_i=144c), tau = 3c | T(t_i=144c), tau = 1c | T(t_i=144c), tau = 0.5c | T(t_i=144c), tau = 0.25c |
|---|---|---|---|---|---|---|
| 0.00 | 310.8666 | 310.6081 | 310.3689 | 310.2208 | 310.1853 | 310.1678 |
| 0.30 | 305.2961 | 305.0905 | 304.8990 | 304.7802 | 304.7516 | 304.7375 |
| 0.60 | 302.7241 | 302.5850 | 302.4533 | 302.3707 | 302.3508 | 302.3410 |
| 0.90 | 301.4626 | 301.3757 | 301.2913 | 301.2375 | 301.2244 | 301.2179 |
| 1.20 | 300.8112 | 300.7600 | 300.7085 | 300.6748 | 300.6665 | 300.6624 |
| 1.50 | 300.4604 | 300.4319 | 300.4018 | 300.3814 | 300.3763 | 300.3738 |
| 1.80 | 300.2653 | 300.2503 | 300.2336 | 300.2218 | 300.2188 | 300.2173 |
| 2.10 | 300.1542 | 300.1469 | 300.1382 | 300.1317 | 300.1300 | 300.1291 |
| 2.40 | 300.0899 | 300.0867 | 300.0826 | 300.0793 | 300.0784 | 300.0779 |
| 2.70 | 300.0524 | 300.0513 | 300.0496 | 300.0480 | 300.0476 | 300.0474 |
| 3.00 | 300.0304 | 300.0302 | 300.0297 | 300.0292 | 300.0290 | 300.0289 |
| 3.30 | 300.0176 | 300.0177 | 300.0177 | 300.0176 | 300.0176 | 300.0176 |
| 3.60 | 300.0101 | 300.0103 | 300.0105 | 300.0106 | 300.0106 | 300.0106 |
| 3.90 | 300.0057 | 300.0059 | 300.0062 | 300.0063 | 300.0063 | 300.0063 |

Рис. 2.3: Таблица $T(x, 144)$.

Зафиксировав $xj$ и сравнивая значения $T(xj\ , ti)$ при разных шагах $\tau$, можно заметить, что наиболее оптимальным шагом по времени является $\tau$ = 1 с., так как при $\tau$ > 1 погрешность $T$ увеличивается, при $\tau$ < 1 погрешность уже незначительная и наблюдается сходимость решений.

2. График зависимости температуры $T(0,t)$ при 3-4 значениях параметров $a_2$ и/или $b_2$ теплоемкости.

*Справка*. С ростом теплоемкости темп нарастания температуры снижается.

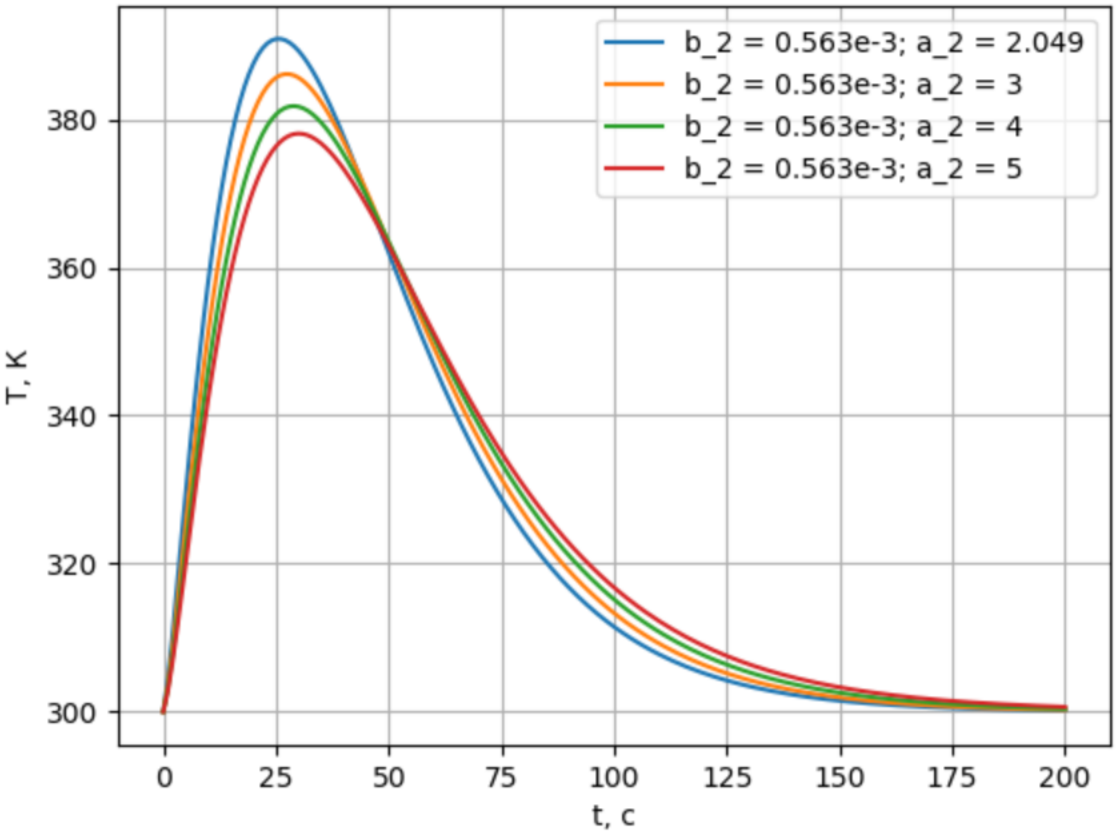



Рис. 3: График функции $T(0,\,t)$ при значениях параметров $a2$ и/или $b2$ теплоемкости.

Из графика видно, что с увеличением значения теплоемкости рост температуры снижается

3. График зависимости температуры $T(0,t)$ (т.е. при $x=0$) в частотном режиме теплового нагружения. Импульсы следуют один за другим с заданной частотой $\nu$ (частота определяется количеством импульсов в 1 секунду).

Показать, что при большом количестве импульсов температурное поле начинает в точности воспроизводиться от импульса к импульсу.

Продемонстрировать, как по мере роста частоты импульсов размах колебаний температуры уменьшается (вплоть до нуля), т.е. реализуется квазистационарный режим, при котором в торец поступает постоянный поток $F_c = \nu \int_0^{t_u} F(t)\,dt$. Здесь $t_u$ -

длительность импульса, определяемая как момент времени, когда $\frac{F(t_u)}{F_{max}} \approx 0.05$.
Если взять прямоугольные импульсы длительностью $t_u$, т.е. $F(t)$=const=$F_0$, то $F_c = \nu\, F_0\, t_u$.

*Справка.* Полученное температурное поле должно совпасть с результатом расчета $T(x)$ по программе лаб. работы №3 при $F_0 = F_c$, разумеется при всех одинаковых параметрах модели, в частности, вместо $k(T)$ надо использовать $k(x)$ из лаб. работы №3.

Графики:

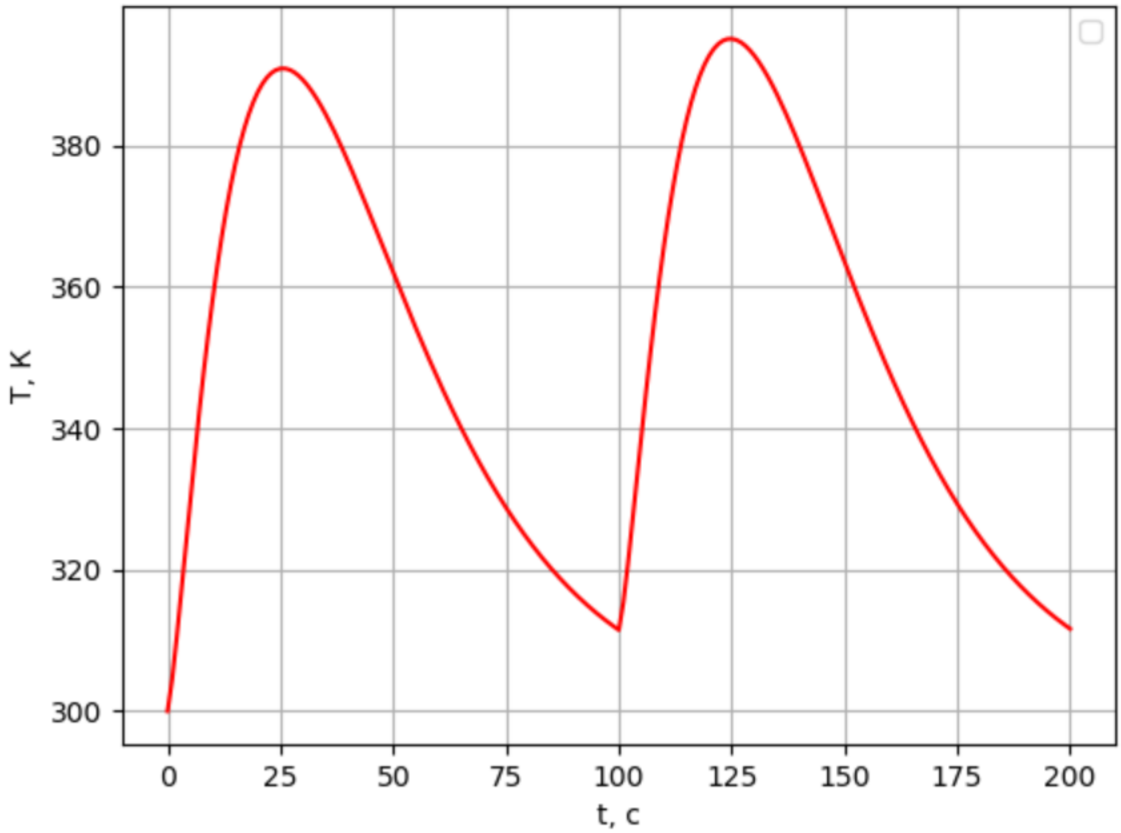



Рис. 4. $\nu = 0.01\frac{\text{Вт}}{\text{см}^2}$, h = 0.3 см, t = 1 с (значения, взяты из №1)

На графике можно наблюдать следующее явление. К моменту достижения амплитуды второго импульса первый импульс еще не успел до конца затухнуть, поэтому температура во время второго импульса больше, чем во время первого.

Теперь постепенно будем увеличивать частоту, пока не увидим совпадение с графиком из лабораторной работы №3.

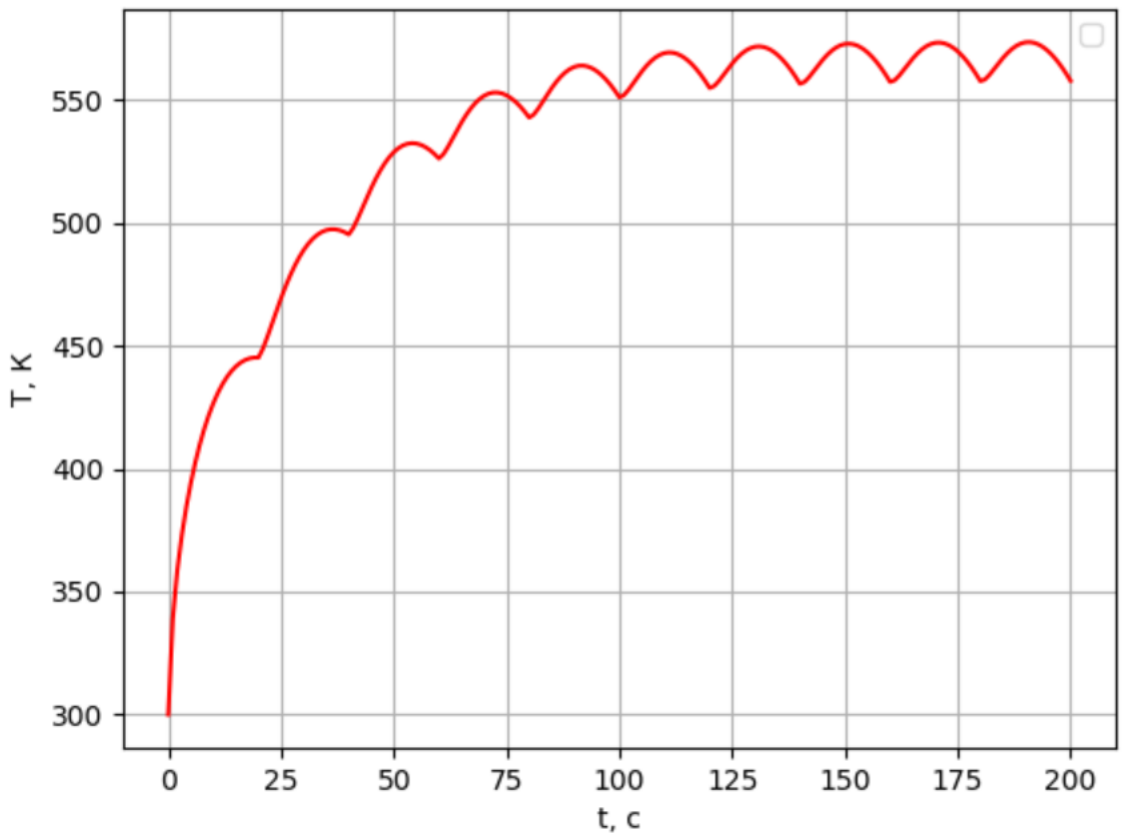


Рис. 5.1. $\nu = 0.05 \frac{\text{Вт}}{\text{см}^2}$,

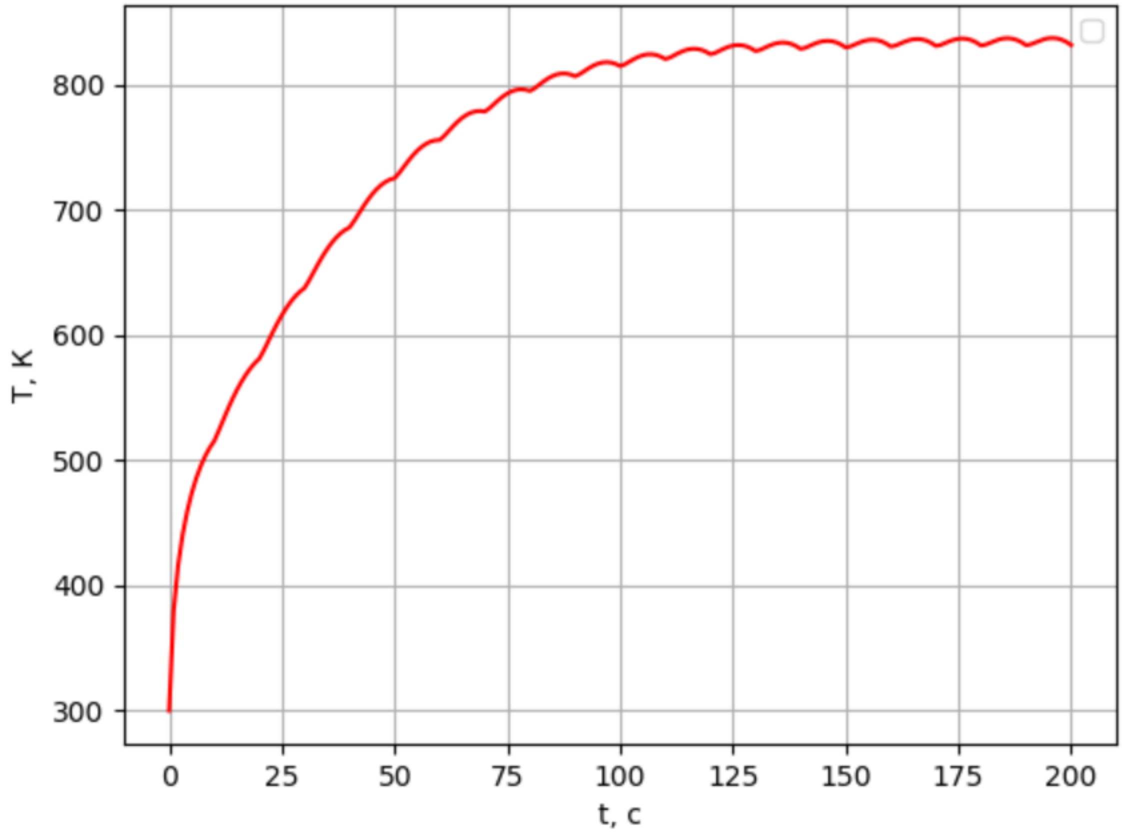



Рис. 5.2. $\nu = 0.1\frac{\text{Вт}}{\text{см}^2}$,

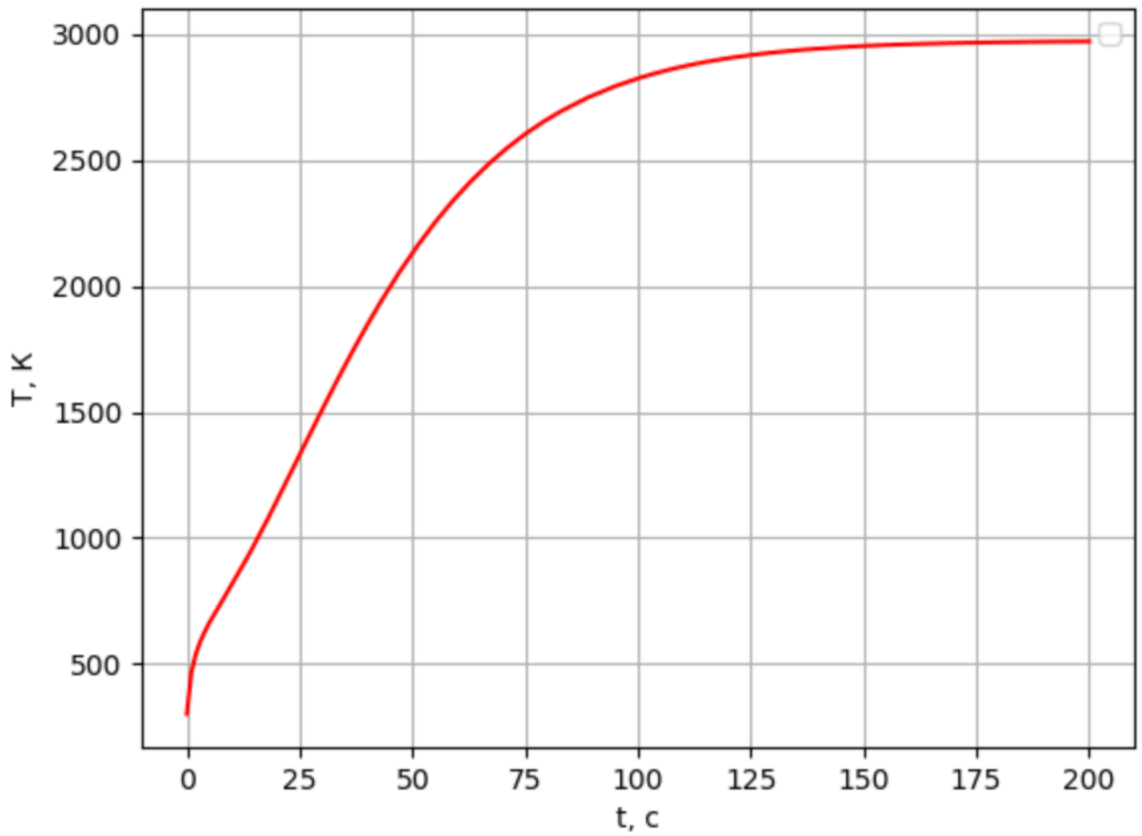



Рис. 5.2. $\nu = 0.5\frac{\text{Вт}}{\text{см}^2}$,

Можно заметить, что при $\nu = 0.5\frac{\text{Вт}}{\text{см}^2}$ температурное поле в точности воспроизводиться от импульса к импульсу.

Также можно заметить, что при тех же параметрах, что и в лабораторной работе №3, графики функций T(x) совпадают