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 № 5

Тема: Исследование математической модели на основе технологии вычислительного эксперимента.

Студент Оберган Т.М.

Группа ИУ7-65Б

Оценка (баллы) ________________

Преподаватель Градов В.М.

Москва.
2020 г.

Оглавление

Исходные данные 3
Листинг 4
Результаты работы 9

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

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,

$T_0 = 300$К,

$R = 0.5$ см,

2. Поток тепла $F(t)$ при $x = 0$

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

Листинг

Листинг 1 – главный модуль

```
import numpy as np
from math import fabs

from model import *
from visualization import draw_graphs

# Прогонка
def get_T_new(T_prev, t):
    A_list, B_list, C_list, F_list = get_coeffs(T_prev)
    K0, M0, P0 = left_boundary_condition(T_prev, t)
    KN, MN, PN = right_boundary_condition(T_prev)

    ksi = [-M0 / K0]
    eta = [P0 / K0]

    n = len(A_list)
    for i in range(n):
        denominator = B_list[i] - A_list[i] * ksi[i]
        ksi.append(C_list[i] / denominator)
        eta.append((F_list[i] + A_list[i] * eta[i]) / denominator)

    T_new = [0] * (n + 2)
    T_new[-1] = (PN - MN * eta[i]) / (KN + MN * ksi[i])

    for i in range(len(ksi) - 1, -1, -1):
        T_new[i] = ksi[i] * T_new[i+1] + eta[i]

    return T_new


def get_result():
    res = []
    curr_T = [T0 for i in np.arange(0, l, x_step)]

    res.append(curr_T)

    t = t_step
    t_total = t
    while True:
        if t >= period:
            t_values.append(t)
            t = 0
        T_prev = curr_T

        max_diff = 1
        while max_diff > eps:
            T_new = get_T_new(T_prev, t)

            max_diff = 0
            for i in range(len(T_prev)):
                diff = fabs((T_new[i] - T_prev[i]) / T_new[i])
                if max_diff < diff:
                    max_diff = diff

            T_prev = T_new

        curr_T = T_new
        res.append(T_new)
```

```
        update_t_values(t_step)
        t += t_step
        t_total += t_step

        if t_total > t_last:
            break

    return res

if __name__ == '__main__':
    # Вычисление результата
    res = get_result()

    # Отображение графиков
    draw_graphs(res, l, t_last, x_step, t_step)
```

Листинг 2 – модуль данных модели и вычислительных функций

```
from math import exp

# Параметры
l = 10
T0 = 300
Fmax = 50
tmax = 20
t_last = 300
period = 160
t = 1
x_step, t_step = 0.1, 1
eps = 1e-2


a1 = 0.0134
b1 = 1
c1 = 4.35e-4
m1 = 1
a2 = 2.049
b2 = 0.563e-3
c2 = 0.528e5
m2 = 1

alpha_0 = 0.05
alpha_N = 0.01

R = 0.5

k_0 = 0.4
k_N = 0.1
b_arg = (k_N * l) / (k_N - 0.4)
a_arg = k_0 * (-b_arg)
d_arg = (alpha_N * l) / (alpha_N - alpha_0)
c_arg = alpha_0 * (-d_arg)

t_values = []


def update_t_values(h_t):
    for i in range(len(t_values)):
        t_values[i] += h_t
```

```python
# Исходные данные модели
def k(T):
    return a1 * (b1 + c1 * T ** m1)


def k_x(x):
    return a_arg / (x - b_arg)


def get_c(T):
    return a2 + b2 * T ** m2 - (c2 / T ** 2)


def alpha(x):
    return c_arg / (x - d_arg)


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


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


# Поток тепла
def F0(t):
    return Fmax / tmax * t * exp(1 - (t / tmax))


def get_total_F(t):
    total_F = F0(t)
    for cur_t in t_values:
        total_F += F0(cur_t)
    return total_F


# Простая аппроксимация
def approc_plus_half(func, n, step):
    return (func(n) + func(n + step)) / 2


def approc_minus_half(func, n, step):
    return (func(n) + func(n - step)) / 2


# Краевые условия
# При x = 0
def left_boundary_condition(T_prev, t):
    c_0 = get_c(T_prev[0])
    c_half = (c_0 + get_c(T_prev[1])) / 2

    p_0 = p(0)
    p_half = approc_plus_half(p, 0, x_step)
    X_half = approc_plus_half(k_x, 0, x_step)

    Ft = get_total_F(t)

    K0 = x_step * (c_half / 8 + c_0 / 4 + t_step / 8 * p_half +
                   (t_step / 4 * p_0)) + X_half * t_step / x_step

    M0 = x_step / 8 * c_half - t_step / x_step * X_half + \
         x_step / 8 * t_step * p_half
```

```
    P0 = x_step * (c_half * (T_prev[0] + T_prev[1]) / 8 +
                   c_0 * T_prev[0] / 4 + t_step *
                   (approc_plus_half(f, 0, x_step) + f(0)) / 4) + \
         Ft * t_step

    return K0, M0, P0


# При x = N
def right_boundary_condition(T_prev):
    T_prev_N = T_prev[-1]
    c_minus = approc_minus_half(get_c, T_prev_N, t)
    k_minus = approc_minus_half(k, T_prev_N, t)
    cN = get_c(T_prev_N)

    KN = x_step / 8 * c_minus + x_step / 4 * cN + t / x_step * k_minus + t *
alpha_N + \
         t * x_step / 8 * p(l - x_step / 2) + t * x_step / 4 * p(l)

    MN = x_step / 8 * c_minus - t / x_step * k_minus + t * x_step / 8 * p(l -
x_step / 2)

    PN = x_step / 8 * c_minus * (T_prev_N + T_prev[-2]) + \
         x_step / 4 * cN * T_prev_N + t * alpha_N * T0 + t * x_step / 4 * (f(l)
+ f(l - x_step / 2))

    return KN, MN, PN


def get_coeffs(T_prev):
    A, B, C, F = [], [], [], []

    for i in range(1, len(T_prev) - 1):
        cur_x = i * x_step

        a = approc_plus_half(k_x, cur_x, x_step) * t_step / x_step
        c = approc_minus_half(k_x, cur_x, x_step) * t_step / x_step

        A.append(a)
        C.append(c)
        B.append(a + c + get_c(T_prev[i]) * x_step + p(cur_x) * x_step * t_step)
        F.append(f(cur_x) * x_step * t_step + get_c(T_prev[i]) * T_prev[i] *
x_step)

    return A, B, C, F
```

Листинг 3 – модуль построения графиков

```
import matplotlib.pyplot as plt
import numpy as np


def draw_graphs(res, x_max, t_max, x_step, t_step):
    # Графики
    lenres = len(res)
    t_last = len(res[0])
    res_cutted = [i[0:t_last:] for i in res]

    # -- Трехмерный
    x, y = np.mgrid[0:lenres:1, 0:t_last:1]
    z = np.array([np.array(i) for i in res_cutted])

    fig_3d = plt.figure()
    xyz = fig_3d.add_subplot(111, projection='3d')
    xyz.plot_surface(x, y, z, cmap='inferno')
    fig_3d.show()

    # -- Проекции
    fig, (first_graph, second_graph) = plt.subplots(
        nrows=1, ncols=2,
        figsize=(8, 4))

    # ------- Первая cm - K
    x = list(np.arange(0, 10, x_step))
    x_cutted = x[:t_last:]
    step1 = 10
    for i in res_cutted[::step1]:
        first_graph.plot(x_cutted, i)
    first_graph.plot(x_cutted, res_cutted[-1])
    first_graph.set_xlabel("x, cm")
    first_graph.set_ylabel("T, K")
    first_graph.grid()

    # ------- Вторая sec - K
    step2 = 5
    te = list(range(0, t_max, t_step))
    for i in np.arange(0, x_max / 3, 0.2 * step2):
        line = [j[int(i / x_step)] for j in res]
        second_graph.plot(te, line[:-1])
    second_graph.set_xlabel("t, sec")
    second_graph.set_ylabel("T, K")
    second_graph.grid()
    fig.show()
```

Результаты работы

1. Провести исследование по выбору оптимальных шагов по времени τ 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, τ 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

$$\pi R^2 (F_0 - F_N) = 2\pi R \iint_0^l \alpha [T(x, t_M) - T_0] dx$$

окончательно

$$\left| \frac{F_0 - F_N}{\frac{2}{R} \iint_0^l \alpha [T(x, t_M) - T_0] dx} - 1 \right| \le \varepsilon$$

Задать точность $\varepsilon \approx 10^{-2}$. 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.

Fmax = 50

tmax = 20

t_last = 300

period = 170

Исследование шага по пространству:

Примем t_step = 1

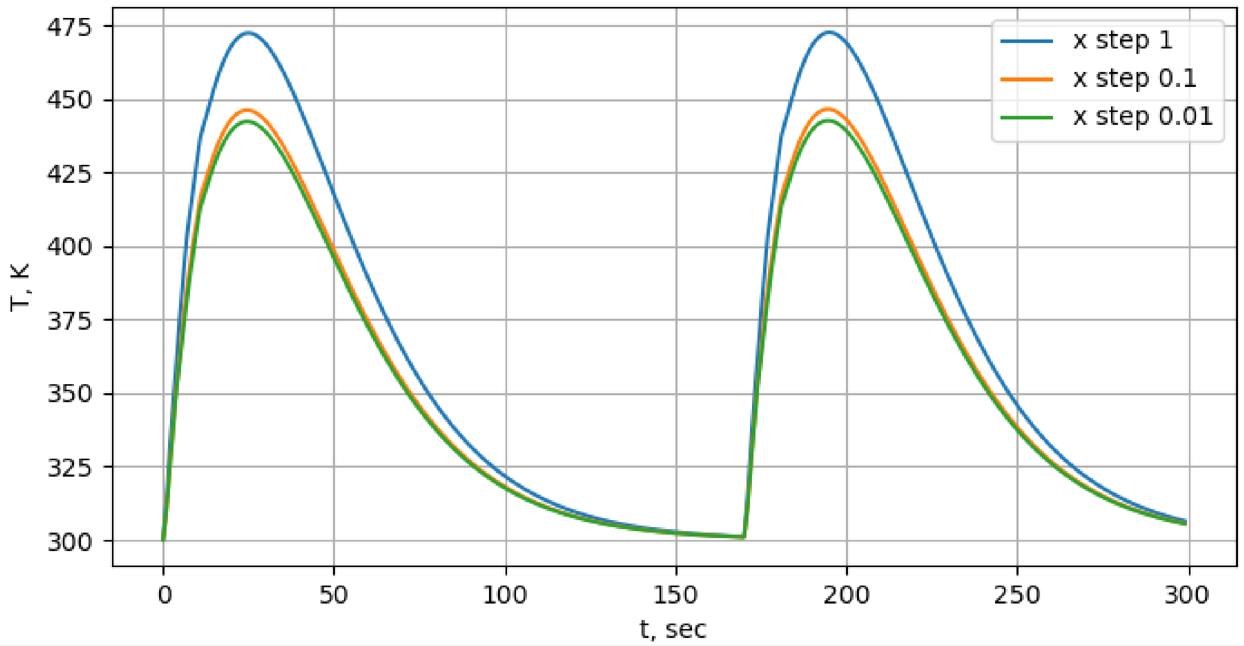

Рис. 1 – варьирование значения x_step

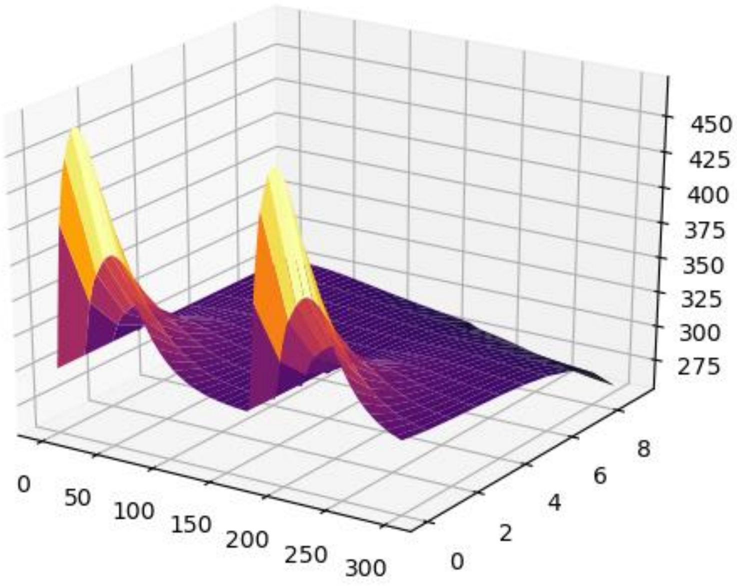

Рис. 2 – трехмерный график время – пространство – температура,
при x_step = 1

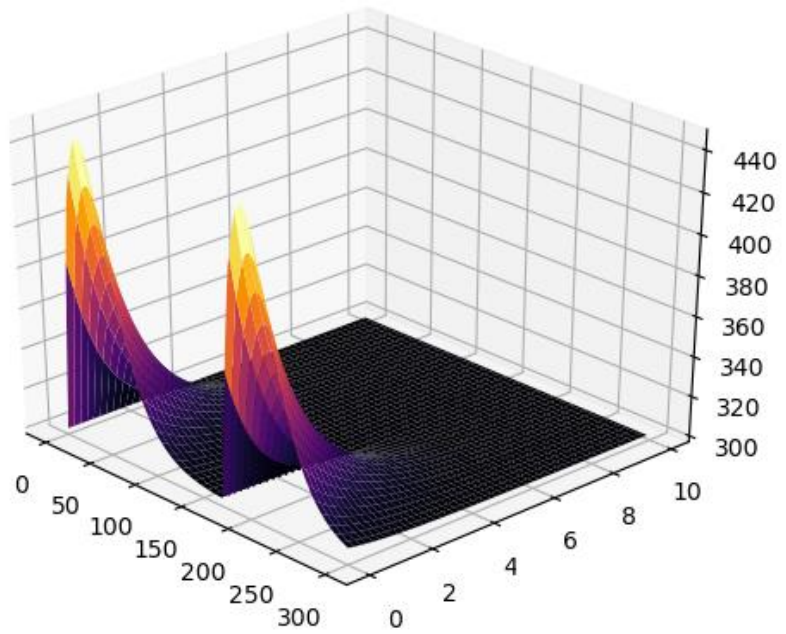

Рис. 3 – x_step = 0.1

На рис.1 и рис.2 видно, что шаг 1 дает недостаточную точность расчетов. Особенно на рис. 2 заметно, как график «сползает вниз», чего быть не должно.

В качестве шага по x будет принято значение 0.1. **x_step = 0.1**

Исследование шага по времени

Примем x_step = 0.1

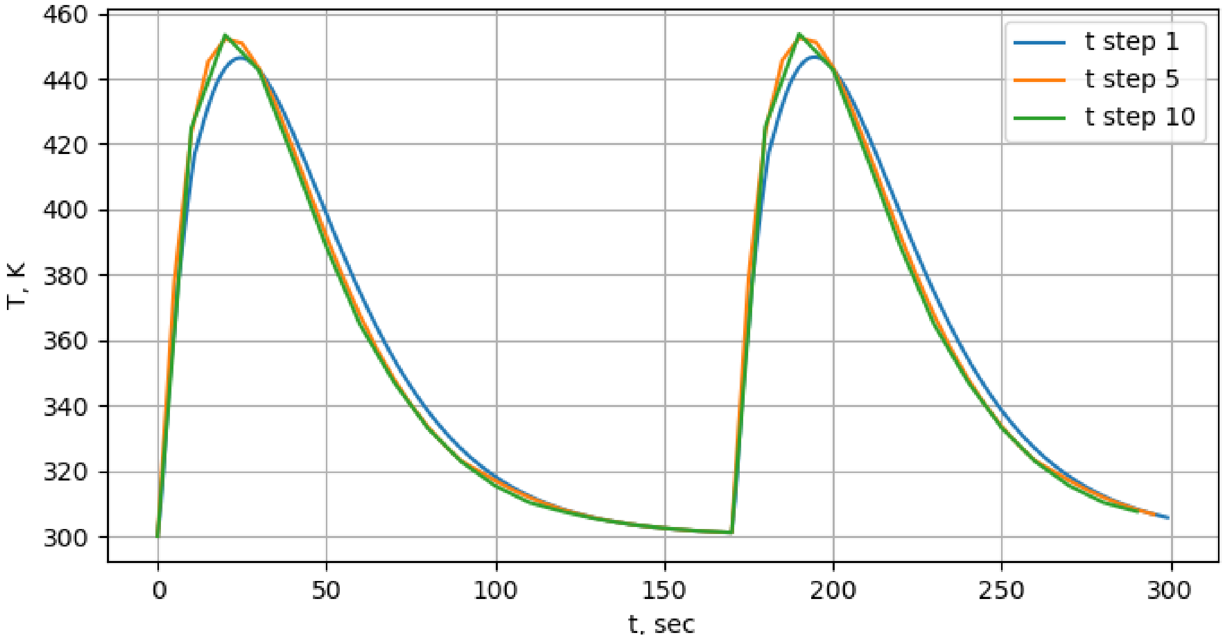

Рис. 4 – варьирование значения t_step

В качестве значения t_step будет принято 1. **t_step = 1**

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.

Fmax = 50; tmax = 20; t_last = 300; period = 300

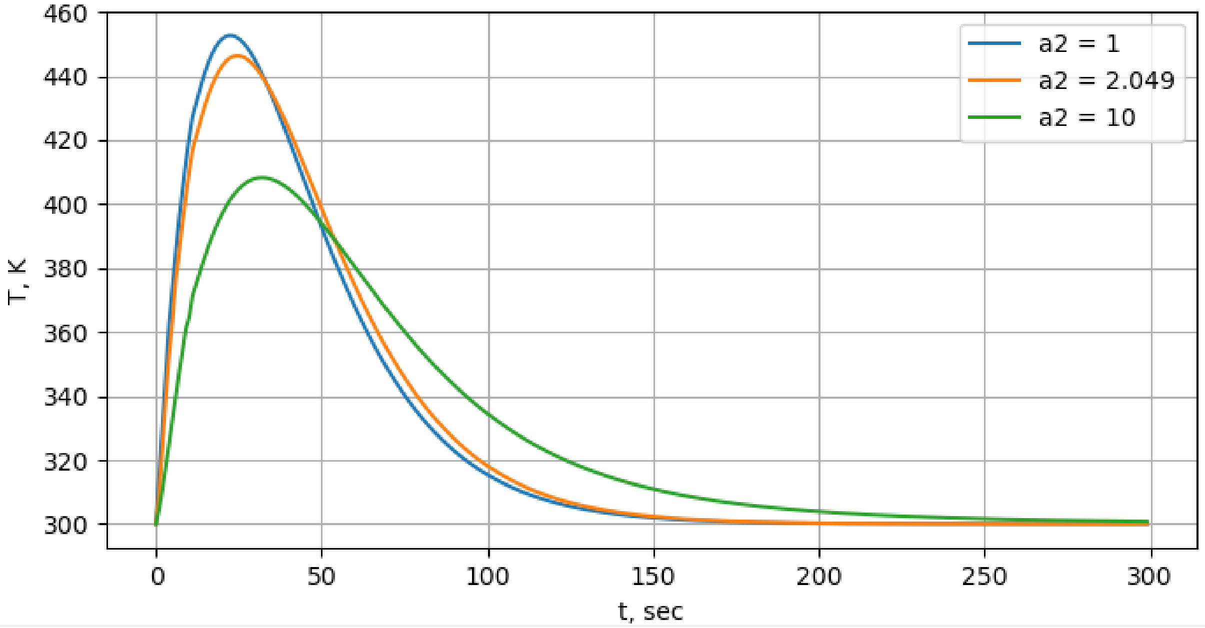

Рис. 4 – варьирование значения a2

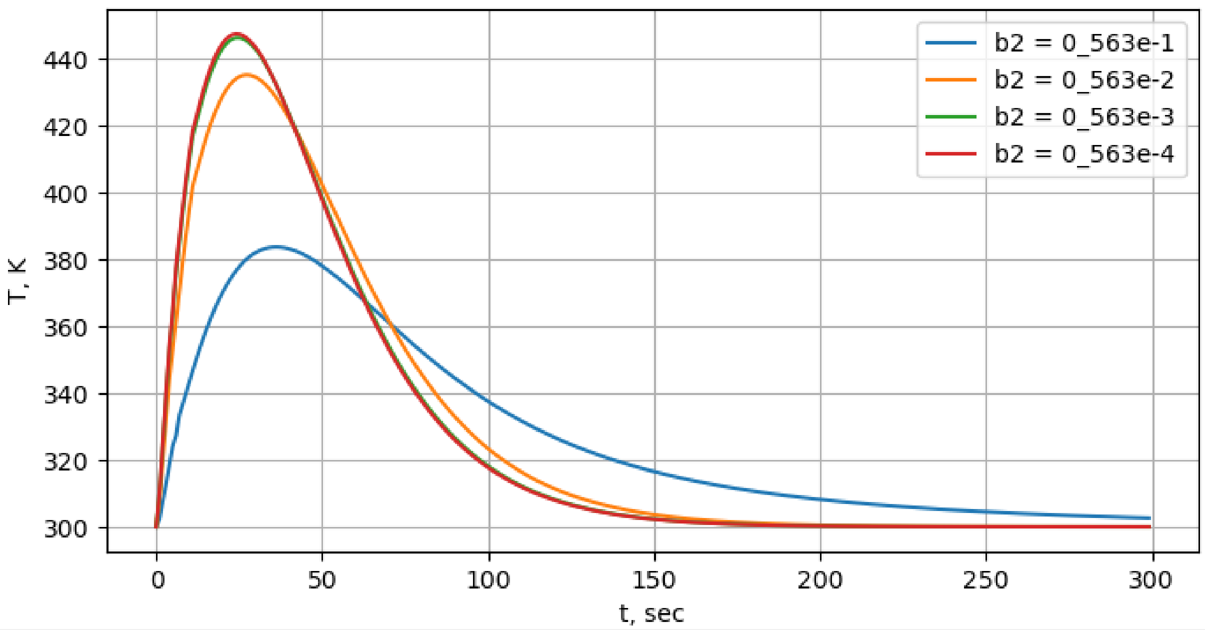

Рис. 5 – варьирование значения b2

3. График зависимости температуры $T(0,t)$ в частотном режиме теплового нагружения. Импульсы следуют один за другим с заданной частотой v (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 $F_c = v\int_0^{t_u} F(t)dt$. 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 $F_c = vF_0 t_u$.

Исследование этой части будет проводиться при значениях: Fmax = 50, tmax = 10 С разными значениями period.

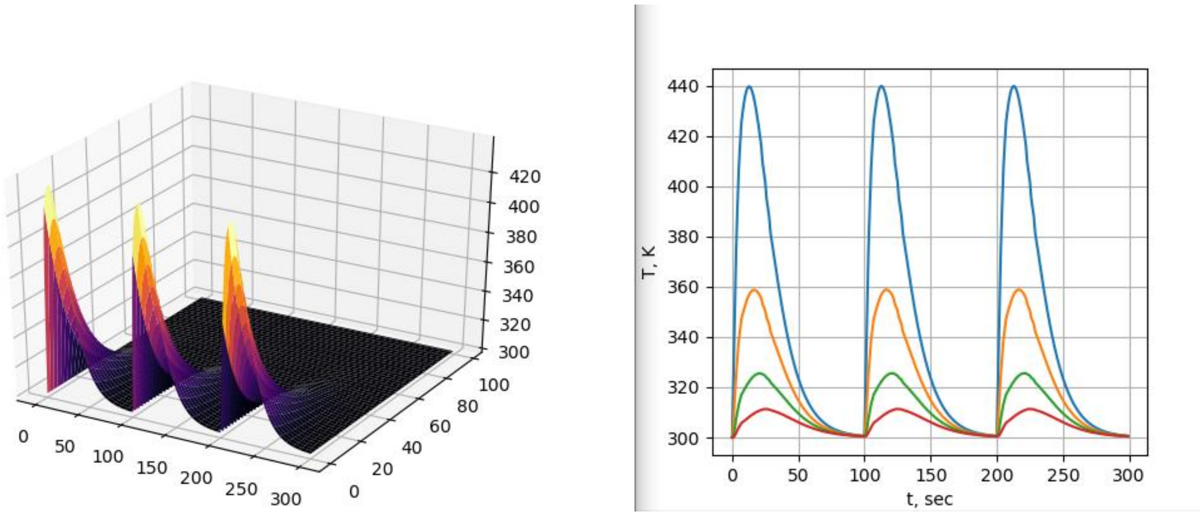

Рис. 6 – Fmax = 50, tmax = 10, period = 100

На этом графике видно, что температурное поле в точности воспроизводится от импульса к импульсу.

При уменьшении периодичности подачи импульсов можно увидеть, как происходит наложение:

Замечание по частотному режиму.
Вариант перекрытия импульсов сложнее для расчета и малоинтересен для практики. При этом заранее понятно, что размах колебаний температуры будет уменьшаться с ростом частоты. Когда же перекрытия импульсов нет, эффект уменьшения размаха обусловливается тепловой инерцией материала, здесь физика и числа другие.

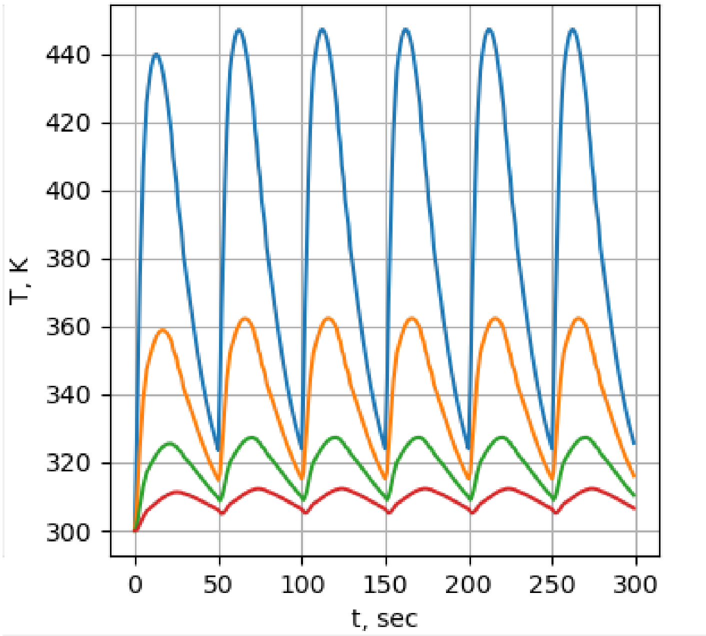

Рис. 7 – Fmax = 50, tmax = 10, period = 50

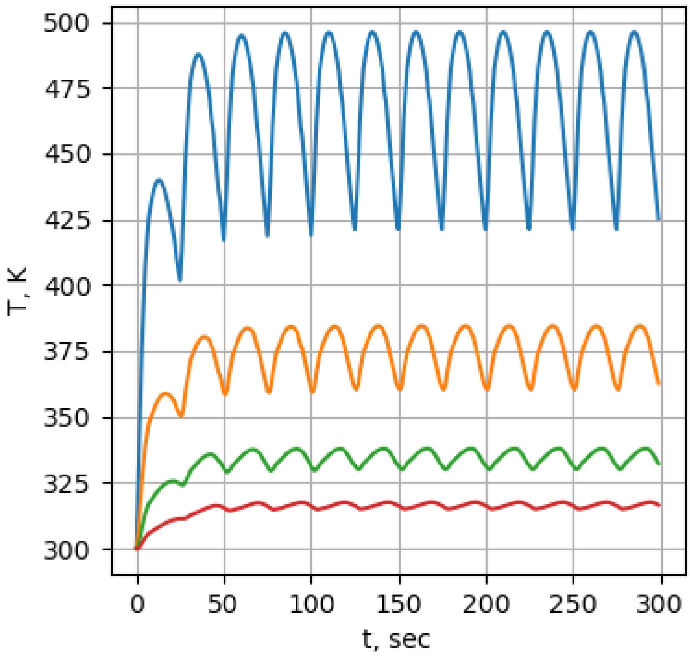

Рис. 8 – Fmax = 50, tmax = 10, period = 25

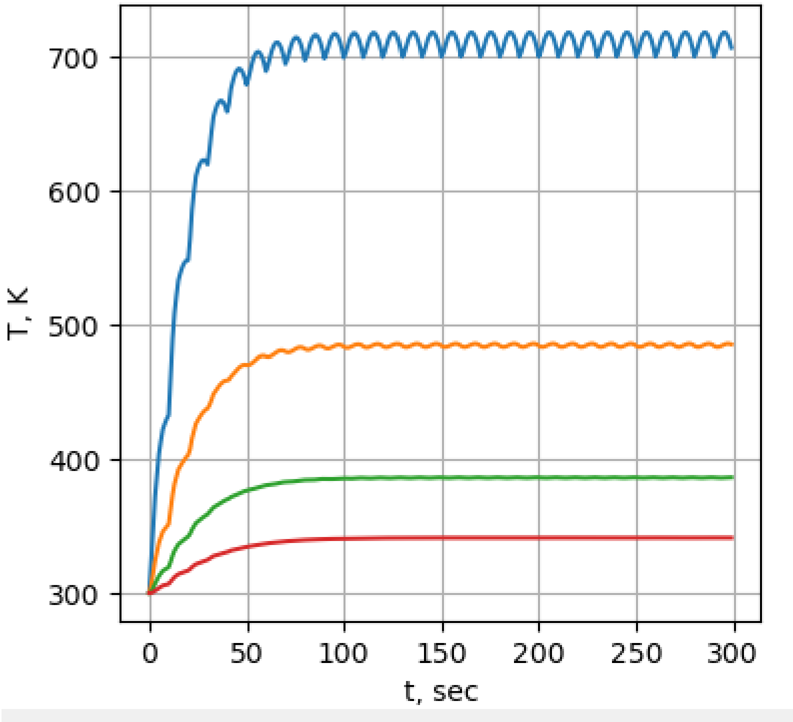

Рис. 9 – Fmax = 50, tmax = 10, period = 10

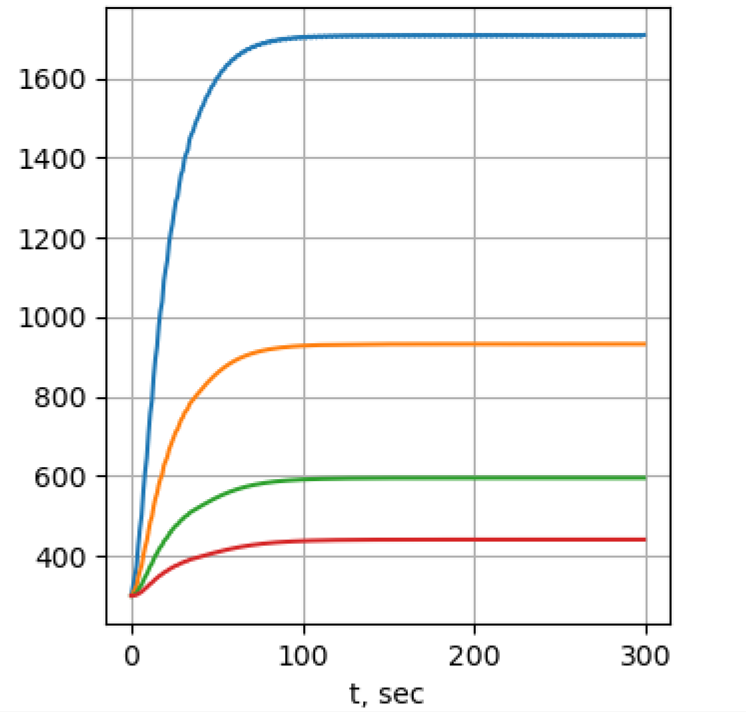

Рис. 10 – Fmax = 50, tmax = 10, period = 3