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

Студент Соколов Ефим
Группа ИУ7-63Б
Дисциплина Моделирование

Преподаватель: ________ Градов В.М.
подпись, дата Фамилия, И.О.

Оценка ____________________

Москва — 2021 г.

Цель работы

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

Постановка задачи

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:

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

В обозначениях уравнения лекции:

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

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

$$\hat{A}_n\hat{y}_{n-1} - \hat{B}_n\hat{y}_n + \hat{D}\hat{y}_{n+1} = -\hat{F}_n \tag{2}$$

$$\begin{gathered}(\tfrac{h}{8}\hat{c}_{\frac{1}{2}} + \tfrac{h}{4}\hat{c}_0 + \hat{\chi}_{\frac{1}{2}}\tfrac{\tau}{h} + \tfrac{\tau h}{8}p_{\frac{1}{2}} + \tfrac{\tau h}{4}p_0)\hat{y}_0 + (\tfrac{h}{8}\hat{c}_{\frac{1}{2}} - \hat{\chi}_{\frac{1}{2}}\tfrac{\tau}{h} + \tfrac{\tau h}{8}p_{\frac{1}{2}})\hat{y}_1 = \\ = \tfrac{h}{8}\hat{c}_{\frac{1}{2}}(y_0 + y_1) + \tfrac{h}{4}\hat{c}_0 y_0 + \hat{F}\tau + \tfrac{\tau h}{4}(\hat{f}_{\frac{1}{2}} + \hat{f}_0)\end{gathered} \tag{3}$$

Разностная схема с разностным краевым условием при $x = l$:

$$\int_{x_{N-\frac{1}{2}}}^{x_N} dx \int_{t_m}^{t_{m+1}} c(T)\frac{\partial T}{\partial t}dt =$$
$$\int_{x_{N-\frac{1}{2}}}^{x_N} dx \int_{t_m}^{t_{m+1}} f(x)dt - \int_{t_m}^{t_{m+1}} dt \int_{x_{N-\frac{1}{2}}}^{x_N} \frac{\partial F}{\partial x}dx - \int_{x_{N-\frac{1}{2}}}^{x_N} dx \int_{t_m}^{t_{m+1}} p(x)Tdt \quad (4)$$

Применим метод правых прямоугольников для интегралов из правой части:

$$\int_{x_{N-\frac{1}{2}}}^{x_N} \hat{c}(\hat{T}-T)dx = \int_{x_{N-\frac{1}{2}}}^{x_N} \hat{f}\tau dx - \int_{t_m}^{t_{m+1}}(F_N - F_{N-\frac{1}{2}})dt - \int_{x_{N-\frac{1}{2}}}^{x_N} p\hat{T}\tau dx \quad (5)$$

Интеграл $\int_{t_m}^{t_{m+1}}(F_N - F_{N-\frac{1}{2}})dt$ решим с помощью метода правых прямоугольников, а оставшиеся методом трапеций:

$$\frac{h}{4}[\hat{c}_N(\hat{y}_N - y_N) + \hat{c}_{N-\frac{1}{2}}(\hat{y}_{N-\frac{1}{2}} - y_{N-\frac{1}{2}})] =$$
$$= \frac{h}{4}\tau(\hat{f}_N - \hat{f}_{N-\frac{1}{2}}) - \tau(\hat{F}_N - \hat{F}_{N-\frac{1}{2}}) - \frac{h}{4}\tau(p_N\hat{y}_N + p_{N-\frac{1}{2}}\hat{y}_{N-\frac{1}{2}}) \quad (6)$$

Подставим в выражения для потока:

$$\frac{h}{4}[\hat{c}_N(\hat{y}_N - y_N) + \hat{c}_{N-\frac{1}{2}}(\frac{\hat{y}_N+\hat{y}_{N-1}}{2} - \frac{y_N+y_{N-1}}{2})] =$$
$$= \frac{h}{4}\tau(\hat{f}_N - \hat{f}_{N-\frac{1}{2}}) - \tau(\alpha_N(\hat{y}_N - T_0) - \hat{\chi}_{N-\frac{1}{2}}\frac{\hat{y}_{N-1}-\hat{y}_N}{h}) - \quad (7)$$
$$-\frac{h}{4}\tau(p_N\hat{y}_N + p_{N-\frac{1}{2}}\frac{\hat{y}_N+\hat{y}_{N-1}}{2})$$

Приведем к общему виду:

$$(\frac{h}{4}\hat{c}_N + \frac{h}{8}\hat{c}_{N-\frac{1}{2}} + \alpha_N\tau + \frac{\tau}{h}\chi_{N-\frac{1}{2}} + \frac{h}{4}\tau p_N + \frac{h}{8}\tau p_{N-\frac{1}{2}})y_N +$$
$$+(\frac{h}{8}\hat{c}_{N-\frac{1}{2}} - \frac{\tau}{h}\chi_{N-\frac{1}{2}} + \frac{h}{8}\tau p_{N-\frac{1}{2}})y_{N-1} = \quad (8)$$
$$= \frac{h}{4}\hat{c}_N y_N + \frac{h}{8}\hat{c}_{N-\frac{1}{2}}(y_N + y_{N-1}) + \tau\alpha_N T_0 + \frac{h}{4}\tau(\hat{f}_N + \hat{f}_{N-\frac{1}{2}})$$

Применим простую аппроксимацию:

$$p_{N-\frac{1}{2}} = \frac{p_N + p_{N-1}}{2}, \hat{f}_{N-\frac{1}{2}} = \frac{\hat{f}_N + \hat{f}_{N-1}}{2}, \hat{c}_{N-\frac{1}{2}} = \frac{\hat{c}_N + \hat{c}_{N-1}}{2}$$

Если $c(u) = 0$ и сократить τ формула (8) перейдёт формулу для разностного краевого условия при $x = l$ из предыдущей лабораторной работы.

Значения параметров для отладки (все размерности согласованы):

$$k(T) = a_1(b_1 + c_1 T^{m_1}),$$

$$c(T) = a_2 + b_2 T^{m_2} - \frac{c_2}{T^2},$$

$$a_1 = 0.0134, b_1 = 1, c_1 = 4.35 * 10^{-4}, m_1 = 1,$$

$$a_2 = 2.049, b_2 = 0.563 * 10^{-3}, c_2 = 0.528 * 10^5, m_2 = 1,$$

$$\alpha(x) = \frac{c}{x-d}, \alpha_0 = 0.05, \alpha_N = 0.01,$$

$$l = 10, T_0 = 300, R = 5,$$

$$F(t) = 50.$$

Листинг кода

На листинге 1 приведен код программы на языке Python 3.

```
import matplotlib.pyplot as plt
import numpy as np
from math import fabs

a1 = 0.0134
b1 = 1
c1 = 4.35e-4
m1 = 1
a2 = 2.049
b2 = 0.563e-3
c2 = 0.528e5
m2 = 1
alpha0 = 0.05
alphaN = 0.01
l = 10
T0 = 300
R = 0.5
F0 = 50
h = 1e-3
t = 1

```

```
def k(T):
    return a1 * (b1 + c1 * T ** m1)


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


def alpha(x):
    d = (alphaN * l) / (alphaN - alpha0)
    c = - alpha0 * d
    return c / (x - d)


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


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


def func_plus_half(x, step, func):
    return (func(x) + func(x + step)) / 2


def func_minus_half(x, step, func):
    return (func(x) + func(x - step)) / 2


def A(T):
    return t / h * func_minus_half(T, t, k)


def D(T):
    return t / h * func_plus_half(T, t, k)


def B(x, T):
    return A(T) + D(T) + c(T) * h + p(x) * h * t

```

```python

def F(x, T):
    return f(x) * h * t + c(T) * T * h


def left_boundary_condition(T_prev):
    K0 = h / 8 * func_plus_half(T_prev[0], t, c) + h / 4 * c(T_prev[0]) + \
         func_plus_half(T_prev[0], t, k) * t / h + \
         t * h / 8 * p(h / 2) + t * h / 4 * p(0)
    M0 = h / 8 * func_plus_half(T_prev[0], t, c) - \
         func_plus_half(T_prev[0], t, k) * t / h + \
         t * h * p(h / 2) / 8
    P0 = h / 8 * func_plus_half(T_prev[0], t, c) * (T_prev[0] + T_prev[1]) + \
         h / 4 * c(T_prev[0]) * T_prev[0] + \
         F0 * t + t * h / 8 * (3 * f(0) + f(h))
    return K0, M0, P0


def right_boundary_condition(T_prev):
    KN = h / 8 * func_minus_half(T_prev[-1], t, c) + h / 4 * c(T_prev[-1]) + \
         func_minus_half(T_prev[-1], t, k) * t / h + t * alphaN + \
         t * h / 8 * p(l - h / 2) + t * h / 4 * p(l)
    MN = h / 8 * func_minus_half(T_prev[-1], t, c) - \
         func_minus_half(T_prev[-1], t, k) * t / h + \
         t * h * p(l - h / 2) / 8
    PN = h / 8 * func_minus_half(T_prev[-1], t, c) * (T_prev[-1] + T_prev[-2]) + \
         h / 4 * c(T_prev[-1]) * T_prev[-1] + t * alphaN * T0 + \
         t * h / 4 * (f(l) + f(l - h / 2))
    return KN, MN, PN


def run(T_prev, K0, M0, P0, KN, MN, PN):
    eps = [0, -M0 / K0]
    eta = [0, P0 / K0]
    x = h
    n = 1
```

```

    while (x + h < l):
        eps.append(D(T_prev[n]) / (B(x, T_prev[n]) - A(T_prev[n]) * eps[n]))
        eta.append((F(x, T_prev[n]) + A(T_prev[n]) * eta[n]) / (B(x, T_prev[n]) - A(T_prev[n]) * eps[n]))
        n += 1
        x += h

    y = [0] * (n + 1)
    y[n] = (PN - MN * eta[n]) / (KN + MN * eps[n])
    for i in range(n - 1, -1, -1):
        y[i] = eps[i + 1] * y[i + 1] + eta[i + 1]

    return y


def main():
    step1 = int(l / h)
    T = [T0] * (step1 + 1)
    T_new = [0] * (step1 + 1)
    ti = 0
    res = []
    res.append(T)

    while True:
        prev = T
        while True:
            K0, M0, P0 = left_boundary_condition(prev)
            KN, MN, PN = right_boundary_condition(prev)
            T_new = run(prev, K0, M0, P0, KN, MN, PN)
            max = fabs((T[0] - T_new[0]) / T_new[0])

            for step2, j in zip(T, T_new):
                d = fabs(step2 - j) / j
                if d > max:
                    max = d
            if max < 1:
                break

            prev = T_new
```

```
            res.append(T_new)
            ti += t

            check_eps = 0
            for i, j in zip(T, T_new):
                if fabs((i - j) / j) > 1e-2:
                    check_eps = 1
            if check_eps == 0:
                break
            T = T_new

    x = [i for i in np.arange(0, l, h)]
    te = [i for i in range(0, ti, t)]
    step1 = 0
    for i in res:
        if (step1 % 2 == 0):
            plt.plot(x, i[:-1])
        step1 += 1

    plt.plot(x, res[-1][:-1])
    plt.xlabel("x, cm")
    plt.ylabel("T, K")
    plt.grid()
    plt.show()

    step2 = 0
    while (step2 < l / 3):
        point = [j[int(step2 / h)] for j in res]
        plt.plot(te, point[:-1])
        step2 += 0.1

    plt.xlabel("t, sec")
    plt.ylabel("T, K")
    plt.grid()
    plt.show()


if __name__ == "__main__":
    main()
```

Листинг 1: Код программы

Выполнение заданий лабораторной работы

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

Вывод разностный аналог краевого условия при $x = l$ приведен выше (уравнения (4)-(8)).

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. Обязательно представить распределение $T(x, t_m)$ в момент времени, соответствующий установившемуся режиму, когда поле перестает меняться с некоторой точностью т.е. имеет место выход на стационарный режим. На этой стадии левая часть дифференциального уравнения близка к нулю. График зависимости $T(x_n, t)$ при нескольких фиксированных значениях координаты x_n. Обязательно представить случай $n = 0$, т.е. $x = x_0 = 0$.

На рисунке 1 приведены графики зависимости температуры $T(x, t_m)$ от координаты x при нескольких фиксированных значениях времени t_m.

На рисунке 2 приведены графики зависимости $T(x_n, t)$ при нескольких фиксированных значениях координаты x_n.

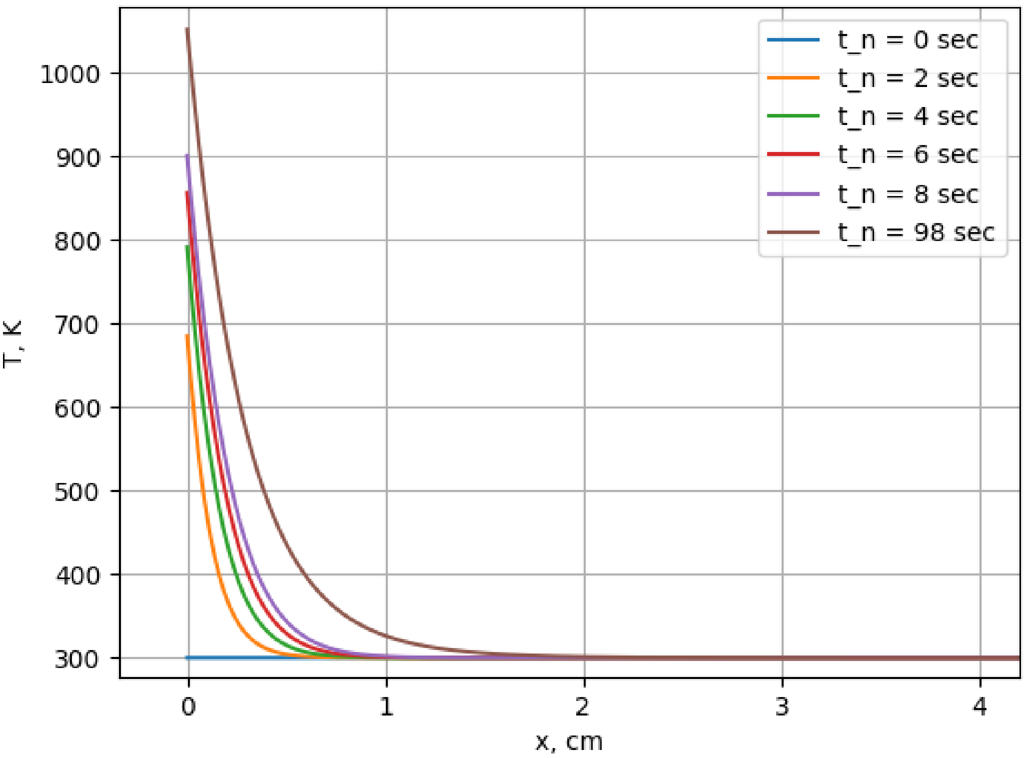

Рисунок 1: Графики зависимости $T(x, t_m)$

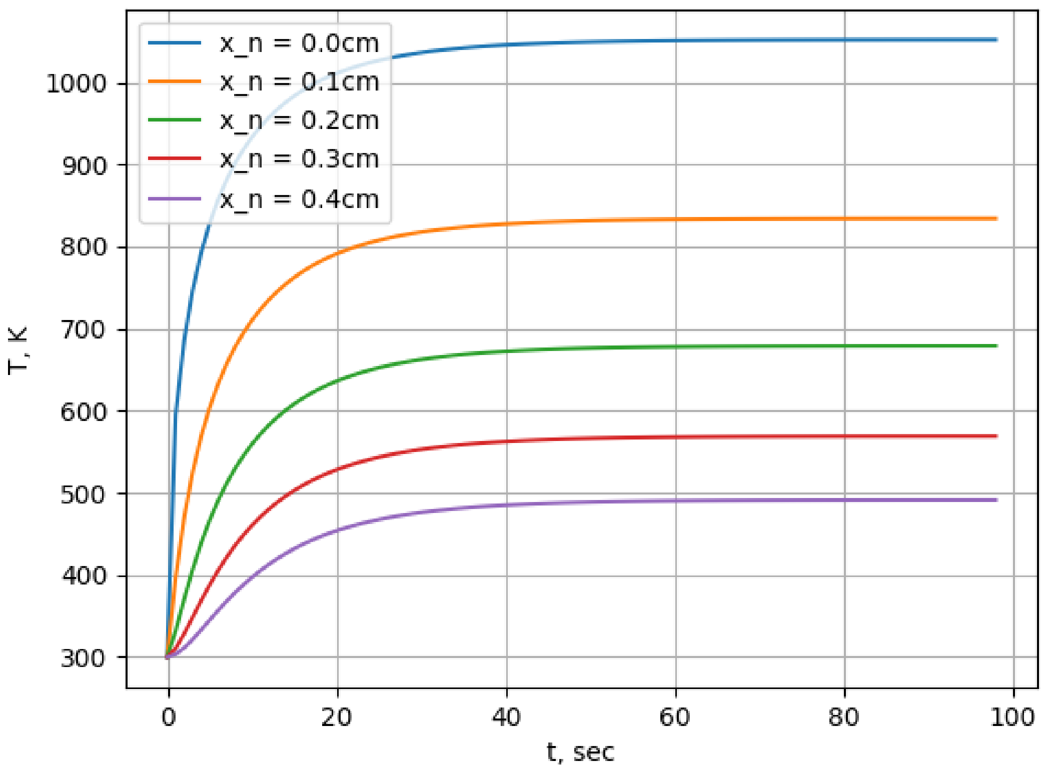

Рисунок 2: Графики зависимости $T(x_n, t)$

Ответы на контрольные вопросы

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

1. При отрицательном тепловом потоке (например, при $F = -5$) слева идет съем тепла. На рисунках 3 и 4 приведены графики зависимости температуры от координаты и от времени, соответственно.

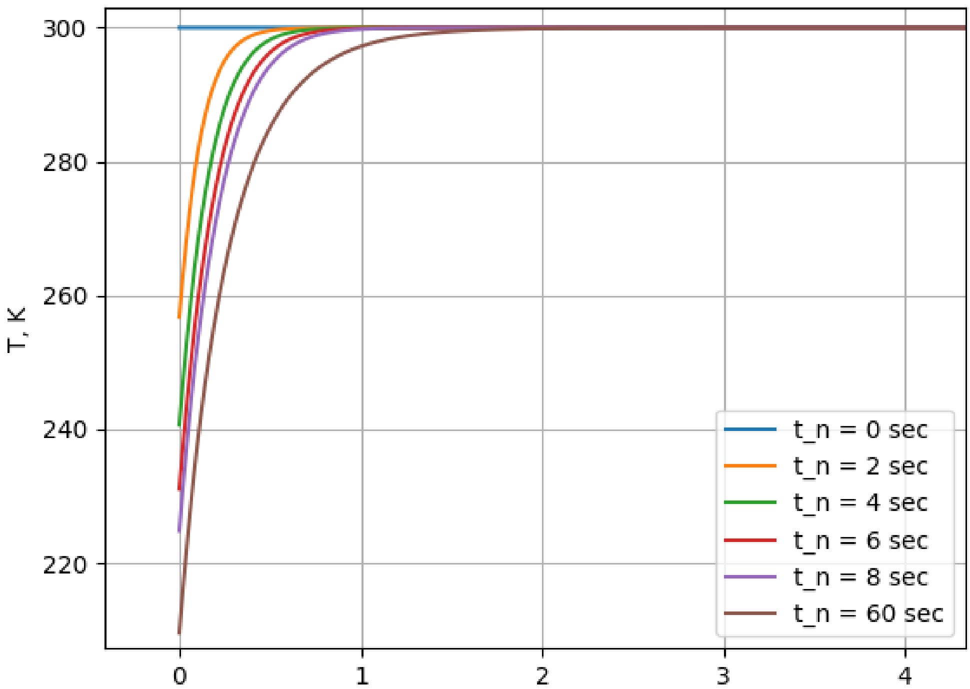

Рисунок 3: Графики зависимости $T(x, t_n)$ при $F = -5$

2. Если тепловой поток равен нулю ($F = 0$), то температура стрежня будет равняться температуре окружающей среды, $T = 300$ (см. рис. 5).

3. Если после разогрева стержня сделать тепловой поток равным 0, то стержень будет остывать пока температура не выровняется по всей длине и не станет равной температуре окружающей среды (см. рис. 6).

4. При произвольной зависимости потока $F(t)$ от времени температурное поле будет как-то сложным образом отслеживать поток (см.

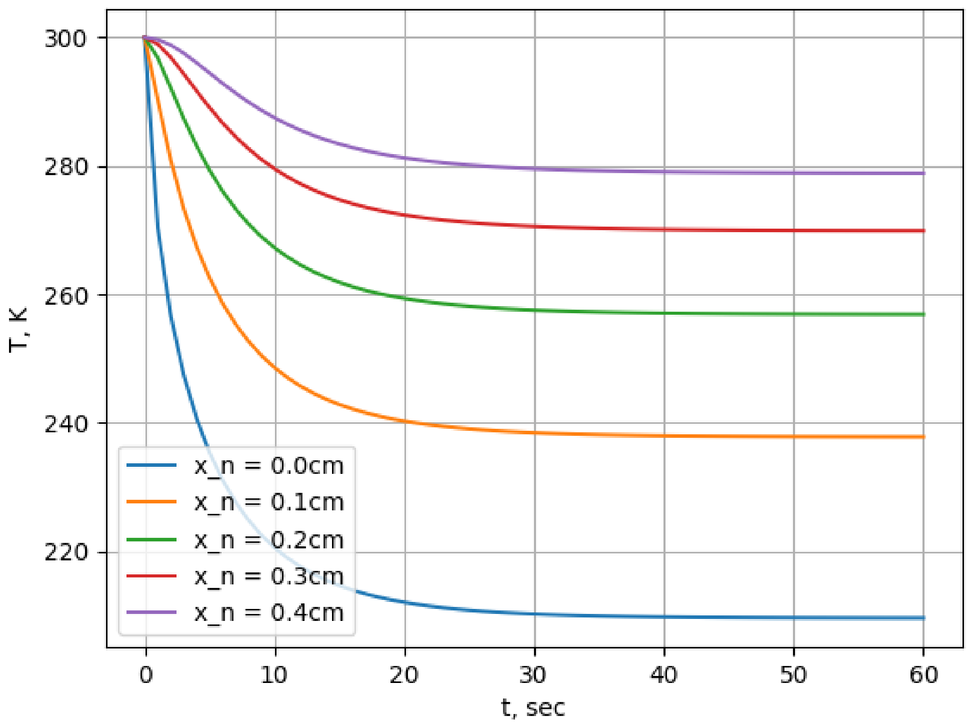

Рисунок 4: Графики зависимости $T(x_n, t)$ при $F = -5$

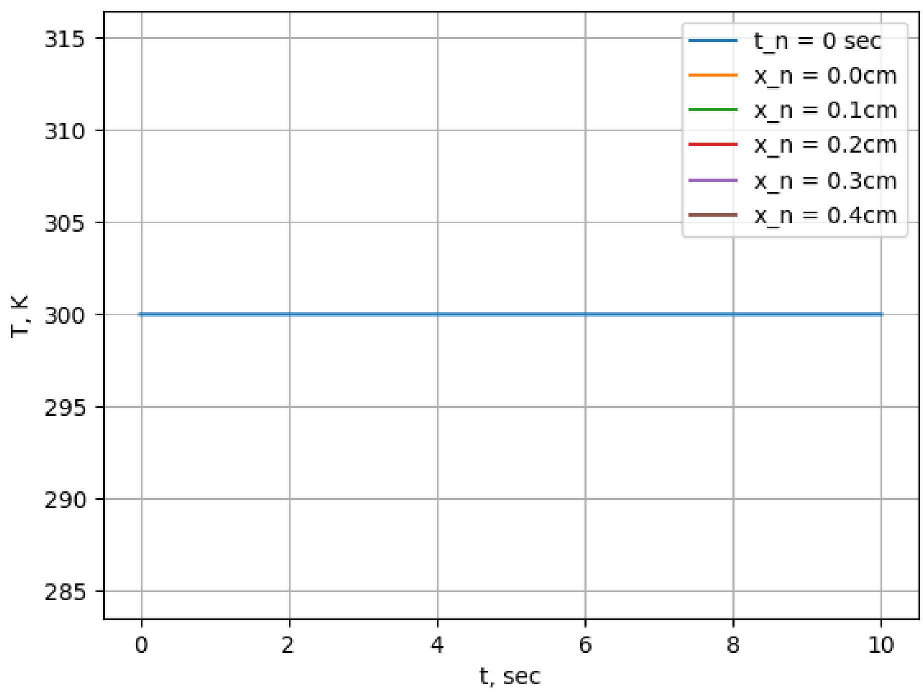

Рисунок 5: Графики зависимости $T(x_n, t)$ при $F = 0$ (все графики совпадают)

рис. 7). На рисунке 8 приведен график зависимости теплового потока от времени (поток линейно изменялся со значения $F = 50$ до $F = -5$ и затем фиксировался на этом значении).

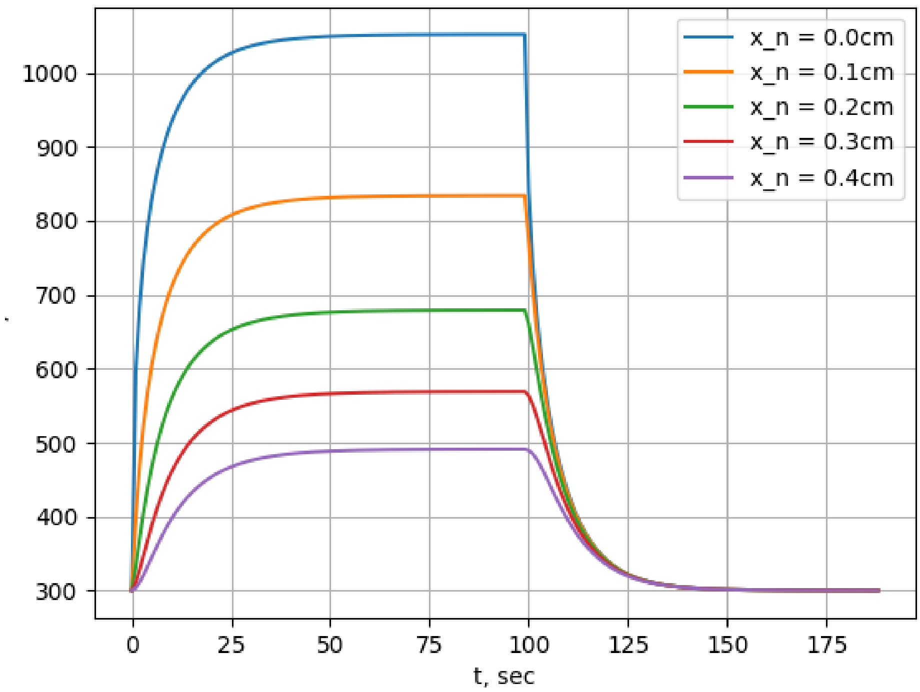

Рисунок 6: Графики зависимости $T(x_n, t)$

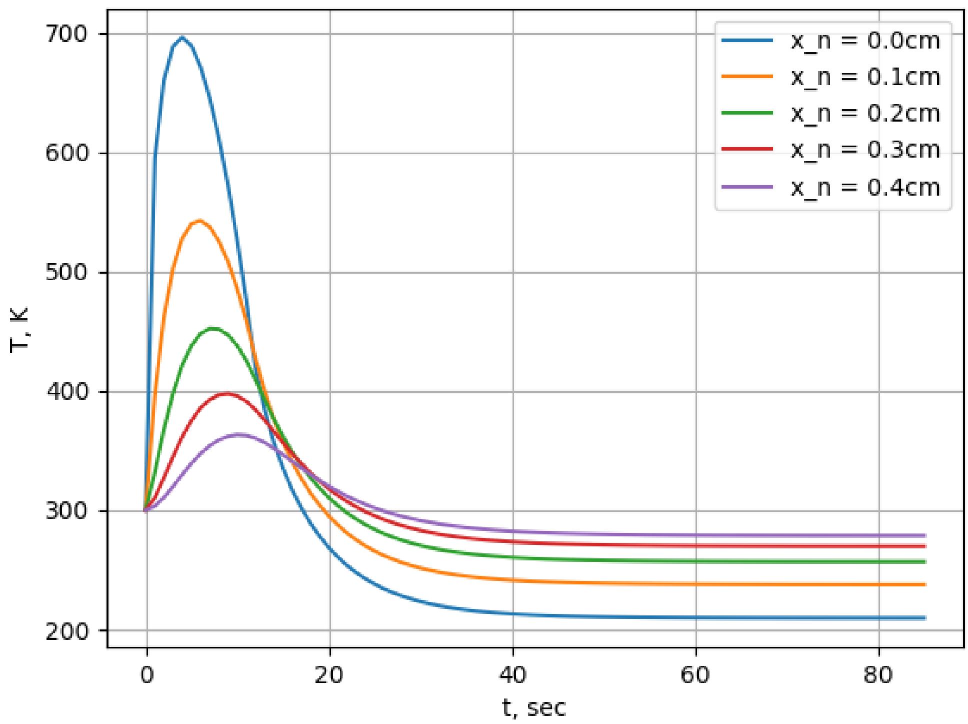

Рисунок 7: Графики зависимости $T(x_n, t)$ при изменяющемся значении $F(t)$

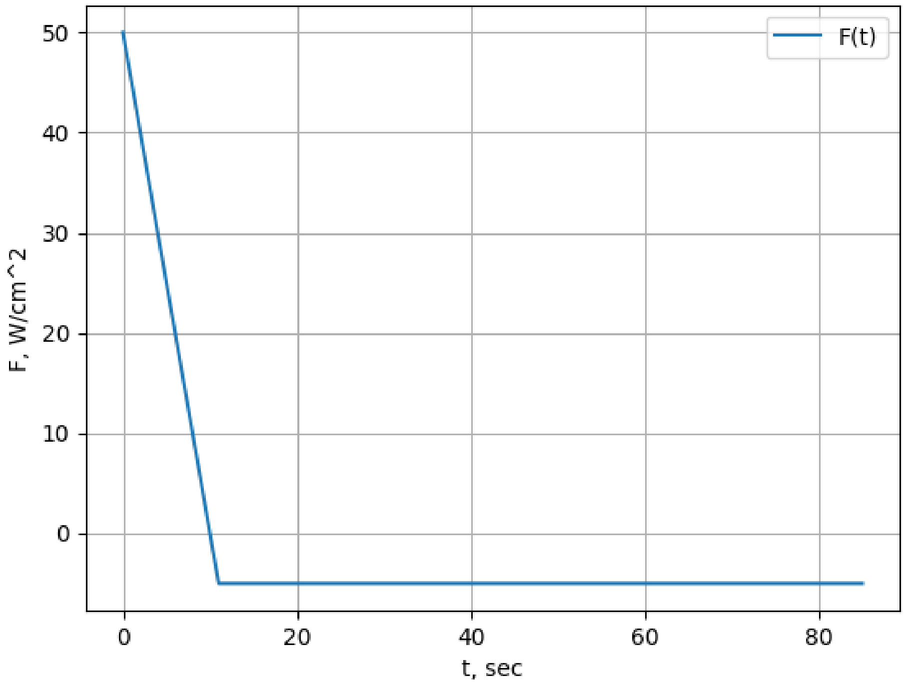

Рисунок 8: Графики зависимости $F(t)$