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

Студент Лучина Е.Д

Группа ИУ7-61Б

Преподаватель Градов В.М.

Москва.
2020 г.

Цель работы

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

Физическое содержание задачи

Постановки задач в данной лабораторной работе и работе №3 во многом совпадают. Отличия заключаются в следующем:

1. Сформулированная в данной работе математическая модель описывает нестационарное температурное поле $T(x,\ t)$, зависящее от координаты x и меняющееся во времени.
2. Свойства материала стержня привязаны к температуре, т.е. Теплоемкость и коэффициент теплопроводности $c(T),\ k(T)$ зависят от Т , тогда как в работе №3 $k(x)$ зависит от координаты, а $c = 0$.
3. При $x = 0$ цилиндр нагружается тепловым потоком $F(t)$, в общем случае зависящим от времени, а в работе №3 поток был постоянный.

Если в настоящей работе задать поток постоянным, т.е. $F(t) = const$, то будет происходить формирование температурного поля от начальной температуры T_0 до некоторого установившегося (стационарного) распределения $T(x,\ t)$. Это поле в дальнейшем с течением времени меняться не будет и должно совпасть с температурным распределением $T(x)$, получаемым в лаб. работе №3, если все параметры задач совпадают, в частности, вместо $k(T)$ надо использовать $k(x)$ из лаб. работы №3. Это полезный факт для тестирования программы.

Если после разогрева стержня положить поток $F(t) = 0$, то будет происходить остывание, пока температура не выровняется по всей длине и не станет равной T_0 . При произвольной зависимости потока $F(t)$ от времени температурное поле будет как-то сложным образом отслеживать поток.

Замечание. Варьируя параметры задачи, следует обращать внимание на то, что решения, в которых температура превышает примерно 2000К, физического смысла не имеют и практического интереса не представляют.

Исходные данные

Функция $T(x,\ t)$ описывается квазилинейным уравнением параболического типа:

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

В обозначениях последующих формул

$$F(t) =- k(T)\frac{T}{x}\ ;\ p(x) = \frac{2}{R}\alpha(x)\ ,\ f(x) = \frac{2T_0}{R}\alpha(x) \qquad (2)$$

Краевые условия:

$$\begin{aligned} & t=0,\ T(x,\ 0)=T_0 \\ & x=0,\ -k(T(0))\frac{T}{x}=F_0 \\ & x=l,\ -k(T(l))\frac{T}{x}=\alpha_N(T(l)-T_0) \end{aligned} \quad (3)$$

Значение параметров для отладки

Все размерности соблюдены

$k(T)=\alpha_1(b_1+c_1T^{m_1}),$ Вт/см К

$c(T)=a_2+b_2T^{m_2}-\frac{c_2}{T^2}$, Дж/см^{3}К

$a_1=0.0134$ $a_2=2.049$	$b_1=1$ $b_2=0.563\times10^{-3}$	$c_1=4.35\times10^{-4}$ $c_2=0.528\times10^{5}$	$m_1=1$ $m_2=1$	$a_0=0.05,$ Вт/см^{2}К $\alpha_N=0.01,$ Вт/см^{2}К

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

Отсюда найдем константы

$c=-d\alpha_0;\ d=\frac{\alpha_N l}{\alpha_N-\alpha_0}$

$l=10$см

$R=0.5$см

$T_0=300K$

$F(t)=50$ Вт/см^2 - постоянное значение для отладки

Разностная схема и аналоги краевых условий

Разностная схема и алгоритм получения разностных аналогов краевых условий описаны в лекции 14.

Разностная схема

$$\widehat{A}_n\widehat{y}_{n-1}-\widehat{B}_n\widehat{y}_n+\widehat{C}_n\widehat{y}_{n+1}=-\widehat{D}_n\quad,\ 1\le n\le N-1 \qquad (4)$$

где

$$\widehat{A}_n=\widehat{\chi}_{n-1/2}\frac{\tau}{h}$$

$$\widehat{C}_n=\widehat{\chi}_{n+1/2}\frac{\tau}{h}$$

$$\widehat{B}_n=\widehat{A}_n+\widehat{C}_n+\widehat{c}_nh+p_nh\tau$$

$$\widehat{D}_n=f_nh\tau+\widehat{c}y_nh$$

Разностный аналог краевого условия при $x=0$

$$(\tfrac{h}{8}\widehat{c}_{1/2}+\tfrac{h}{4}\widehat{c}_0+\widehat{\chi}_{1/2}\tfrac{\tau}{h}+\tfrac{\tau h}{8}p_{1/2}+\tfrac{\tau h}{4}p_0)\widehat{y}_0+(\tfrac{h}{8}\widehat{c}_{1/2}-\widehat{\chi}_{1/2}\tfrac{\tau}{h}+\tfrac{\tau h}{8}p_{1/2})\widehat{y}_1=$$

$= \frac{h}{8}\widehat{c}_{1/2}(y_0 + y_1) + \frac{h}{4}\widehat{c}_0 y_0 + \widehat{F}\tau + \frac{\tau h}{4}(\widehat{f}_{1/2} + \widehat{f}_0)$
(5)

$$K_0\widehat{y}_0 + M_0\widehat{y}_1 = P_0 \qquad (6)$$
$$K_0 = \frac{h}{8}\widehat{c}_{1/2} + \frac{h}{4}\widehat{c}_0 + \widehat{\chi}_{1/2}\frac{\tau}{h} + \frac{\tau h}{8}p_{1/2} + \frac{\tau h}{4}p_0$$
$$M_0 = \frac{h}{8}\widehat{c}_{1/2} - \widehat{\chi}_{1/2}\frac{\tau}{h} + \frac{\tau h}{8}p_{1/2}$$
$$P_0 = \frac{h}{8}\widehat{c}_{1/2}(y_0 + y_1) + \frac{h}{4}\widehat{c}_0 y_0 + \widehat{F}\tau + \frac{\tau h}{4}(\widehat{f}_{1/2} + \widehat{f}_0)$$

Разностный аналог краевого условия при $x = l$

Проинтегрируем уравнение (1) на отрезке $[x_{N-1/2}, x_N]$ и на интервале времени $[t_m,\ t_{m+1}]$

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

Методом правых треугольников приближенно вычислим внутренние интегралы по t. Тем самым следует ожидать порядок точности $O(\tau)$ по переменной t .
Получим

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

Первый интеграл правой части (по времени) вычислим также методом правых треугольников, а остальные (интегралы по координате) - методом средних.

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

Подставим сюда выражение для потока
$\widehat{F}_N = \alpha_N(\widehat{y}_N - T_0)$, а $\widehat{F}_{N-1/2} = \widehat{\chi}_{N-1/2}\frac{\widehat{y}_{N-1} - \widehat{y}_N}{h}$
И заменим
$\widehat{y}_{N-1/2} = \frac{\widehat{y}_N + \widehat{y}_{N-1}}{2}$; $y_{N-1/2} = \frac{y_N + y_{N-1}}{2}$
Получим

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

Преобразуем

$$K_N\widehat{y}_N + M_N\widehat{y}_{N-1} = P_N \qquad (8)$$

где

$$K_N = \frac{h}{4}\widehat{c}_N + \frac{h}{8}\widehat{c}_{N-1/2} + \widehat{\chi}_{N-1/2}\frac{\tau}{h} + \frac{\tau h}{4}p_N + \frac{\tau h}{8}p_{N-1/2} + \alpha_N\tau$$

$$M_N = \frac{h}{8}\widehat{c}_{N-1/2} - \widehat{\chi}_{N-1/2}\frac{\tau}{h} + \frac{\tau h}{8}p_{N-1/2}$$
$$P_N = \frac{h}{4}\widehat{c}_N y_N + \frac{h}{8}\widehat{c}_{N-1/2}(y_N + y_{N+1}) + \alpha_N T_0\tau + (\widehat{f}_N + \widehat{f}_{N-1/2})\tau\frac{h}{4}$$

В итоге система квазилинейных разностных уравнений примет канонический вид:

$$\begin{aligned} &K_0\widehat{y}_0 + M_0\widehat{y}_1 = P_0 \\ &\widehat{A}_n\widehat{y}_{n-1} - \widehat{B}_n\widehat{y}_n + \widehat{C}_n\widehat{y}_{n+1} = -\widehat{D}_n \ , \qquad 1 \le n \le N-1 \\ &K_N\widehat{y}_N + M_N\widehat{y}_{N-1} = P_N \end{aligned} \qquad (9)$$

Решение системы

Данную систему (9) можно решить методом прогонки.

Прогоночные коэффициенты можно найти по следующим рекуррентным формулам:

$$\xi_{n+1} = \frac{C_n}{B_n - A_n\xi_n}\ ;\ \eta_{n+1} = \frac{D_n + A_n\eta_n}{B_n - A_n\xi_n} \qquad (10)$$

начальные коэффициенты прогонки

$$\xi_1 = \frac{M_0}{K_0};\ \eta_1 = \frac{P_0}{K_0} \qquad (11)$$

Алгоритм решения

Для каждого фиксированного t:

- Найти $K_0,\ M_0,\ P_0,\ K_N,\ M_N,\ P_N$
- Найти $\widehat{A}_n,\ \widehat{B}_n,\ \widehat{C}_n,\ \widehat{D}_n$
- Найти прогоночные коэффициенты (прямой ход)
- Найти значения T (обратный ход) по формулам

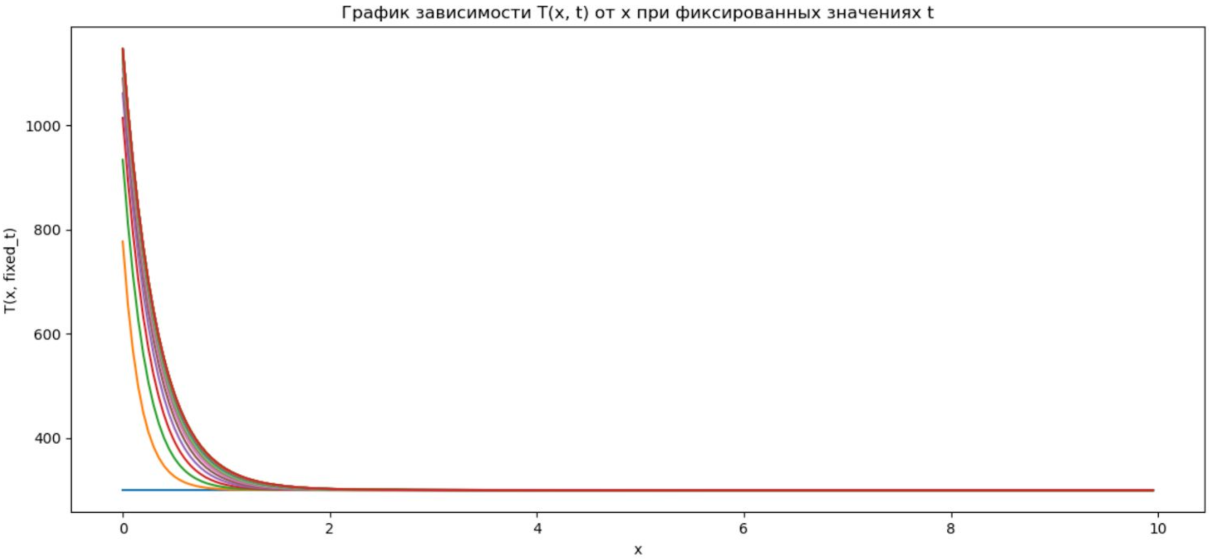
$$T_N = \frac{P_N - M_N\eta_N}{K_N + M_N\xi_N}$$
$$T_n = \xi_{n+1}T_{n+1} + \eta_{n+1}$$

В результате получим матрицу описывающую температурное поле $T(x,\ t)$, где строке соответствует фиксированный момент времени, а столбцу - фиксированная координаты вдоль длины стержня.

Результат работы

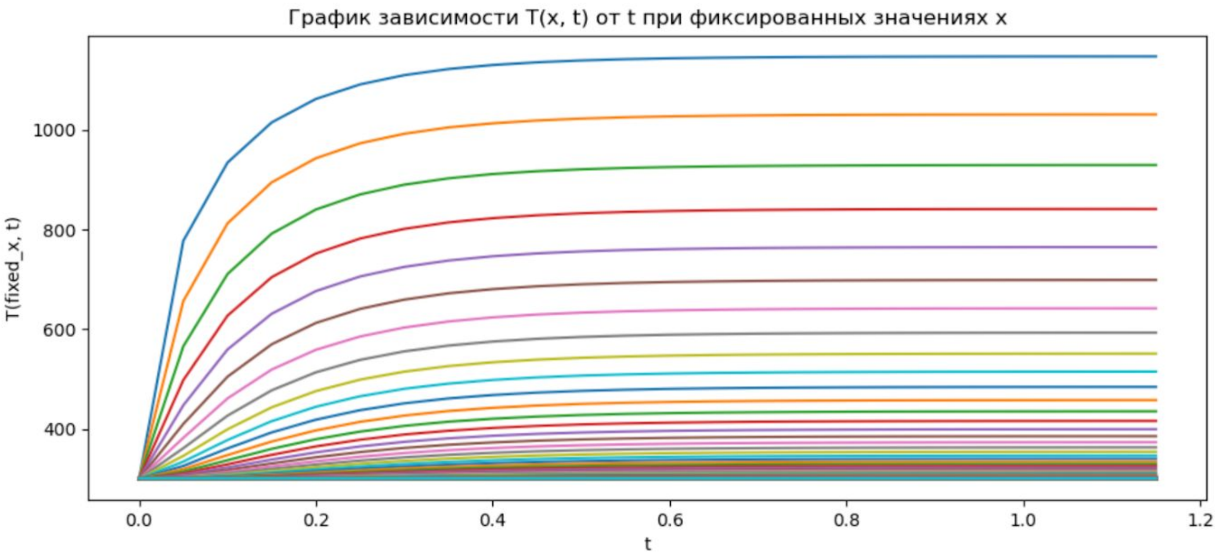
Разностный аналог краевого условия при $x = l$ и его краткий вывод интегро-интерполяционным методом представлен в пункте Разностный аналог краевого условия при $x = l$.

График зависимости температуры $T(x,\ t)$ от координаты x при нескольких фиксированных значениях времени t_m (с шагом $\tau = 5$) при заданных выше параметрах.



Последний красный график (где температура изменяется в бОльшем диапазоне) представляет распределение $T(x,\ t)$ в момент времени, соответствующий установившемуся режиму, когда поле перестает меняться с некоторой точностью $\frac{T(t+\tau)-T(t)}{T(t+\tau)} < 10^{-4}$, т.е. имеет место выход на стационарный режим. Нижний синий график, где температура постоянна и равна T0, - температура в начальный момент времени.

График зависимости $T(x_n, t)$ при нескольких фиксированных значениях координаты x_n (с шагом $h = 0.05$).



Синий верхний график соответствует x = l, нижний - x = 0

Листинг кода программы

Программа написана на языке python. Для отображения графиков используется matplotlib.

Функции описывающие исходные уравнения явления и параметры для отладки

```
l = 10
R = 0.5
T0 = 300
F0 =  50

a1 = 0.0134
a2 = 2.049
b1 = 1
b2 = 0.564e-3
c1 = 4.35e-4
c2 = 0.528e5
m1 = 1
m2 = 1
def k(T): return a1 * (b1 + c1 * T ** m1)
def cap(T): return a2 + b2 * T ** m2 - (c2 / T / T)

alpha0 = 0.05
alphaN = 0.01
d = alphaN * l / (alphaN - alpha0)
c = -d * alpha0
def alpha(x): return c / (x - d)
def p(x): return 2 * alpha(x) / R
def f(x): return 2 * T0 * alpha(x) / R
def Hi(T1, T2):
    k1 = k(T1)
    k2 = k(T2)
    return 2 * k1 * k2 / (k1 + k2)

h = 0.05
tau = 5
N = (int)(l / h)
```

Поиск коэффициентов разностной схемы и разностных аналогов краевых условий K_0, M_0, P_0; K_N, M_N, P_N; A_n, B_n, C_n, D_n

```
def left_border(prev_iter, prev_T):
    Hi12 = Hi(prev_iter[0], prev_iter[1])
    p0 = p(0)
    p1 = p(h)
    p12 = (p0 + p1) / 2
    f0 = f(0)
    f1 = f(h)
    f12 = (f0 + f1) / 2
    cap0 = cap(prev_iter[0])
    cap1 = cap(prev_iter[1])
    cap12 = (cap0 + cap1) / 2

    tauh4 = tau * h / 4
    h8cap12 = h/8 * cap12
    Hi12tauh = Hi12 * tau / h

    K0 = h8cap12 + h/4*cap0 + Hi12tauh + tauh4/2*p12 + tauh4*p0
    M0 = h8cap12 - Hi12tauh + tauh4/2*p12
    P0 = h8cap12*(prev_T[0]+prev_T[1]) + h/4*cap0*prev_T[0] +F0*tau + tauh4*(f12+f0)
    return (K0, M0, P0)

def right_border(prev_iter, prev_T):
```

```
    last = len(prev_iter) - 1
    HiN12 = Hi(prev_iter[last], prev_iter[last - 1])
    pN = p(l)
    pN1 = p(l - h)
    pN12 = (pN + pN1) / 2
    fN = f(l)
    fN1 = f(l - h)
    fN12 = (fN + fN1) / 2
    capN = cap(prev_iter[last])
    capN1 = cap (prev_iter[last - 1])
    capN12 = (capN + capN1) / 2

    tauh4 = tau * h / 4
    h8capN12 = h/8 * capN12
    HiN12tauh = HiN12 * tau / h

    KN = h/4*capN + h8capN12 + HiN12tauh + tauh4 * pN + tauh4/2*pN12 + alphaN*tau
    MN = h8capN12 - HiN12tauh + tauh4/2*pN12
    PN = h/4*capN*prev_T[last] + h/8*capN12*(prev_T[last] + prev_T[last - 1]) +
alphaN*T0*tau + tauh4 * (fN + fN12)
    return (KN, MN, PN)

def coefs(prev_iter, prev_T):
    A_arr = [0 for x in range(N)]
    B_arr = [0 for x in range(N)]
    C_arr = [0 for x in range(N)]
    D_arr = [0 for x in range(N)]
    for i in range (1, N - 1):
        x = i * h
        A_arr[i] = Hi(prev_iter[i], prev_iter[i - 1]) * tau / h
        C_arr[i] = Hi(prev_iter[i], prev_iter[i + 1]) * tau / h
        B_arr[i] = A_arr[i] + C_arr[i] + cap(prev_iter[i])* h + p(x) * h * tau
        D_arr[i] = f(x) * h * tau + cap(prev_iter[i]) * prev_T[i] * h

    return (A_arr, B_arr, C_arr, D_arr)
```

Метод прогонки

```
def ksi_etta(K0, M0, P0, A_arr, B_arr, C_arr, D_arr):
    ksi_arr = [None] * N
    etta_arr = [None] * N
    ksi_arr[1] = -M0 / K0
    etta_arr[1] = P0 / K0
    for i in range(1, N - 1):
        denominator = B_arr[i] - A_arr[i] * ksi_arr[i]
        ksi_arr[i + 1] = C_arr[i] / denominator
        etta_arr[i + 1] = (D_arr[i] + A_arr[i] * etta_arr[i]) / denominator
    return (ksi_arr, etta_arr)

def get_T(KN, MN, PN, ksi_arr, etta_arr):
    T_arr = [None] * N
    T_arr[N - 1] = (PN - MN * etta_arr[N - 1]) / (KN + MN * ksi_arr[N - 1])
    for i in range(N - 2, -1, -1):
        T_arr[i] = ksi_arr[i + 1] * T_arr[i + 1] + etta_arr[i + 1]
    return T_arr
```

Основная верхнеуровневая функция solve, возвращает матрицу, которая описывает температурное поле T(x, t). И вспомогательная функция для определения факта достижения нужной точности.

```
def find_max_diff(T_arr_cur, T_arr_prev):
    length = len(T_arr_cur)
    maxv = abs((T_arr_cur[0] - T_arr_prev[0]) / T_arr_cur[0])
    for i in range(1, length):
        v = abs((T_arr_cur[i] - T_arr_prev[i]) / T_arr_cur[i])
        if (maxv < v):
            maxv = v
    return maxv

def solve():
    eps = 0.0001

    result = []
    cur_T = [T0 for i in range(N)]
    result.append(cur_T)

    tmp = float('Inf')
    t = 0
    while (tmp > eps):
        prev_T = cur_T
        cur_iter = cur_T
        tmp = float('Inf')
        while (tmp > eps):
            prev_iter = cur_iter
            K0, M0, P0 = left_border(prev_iter, prev_T)
            KN, MN, PN = right_border(prev_iter, prev_T)
            A_arr, B_arr, C_arr, D_arr = coefs(prev_iter, prev_T)
            ksi_arr, etta_arr = ksi_etta(K0, M0, P0, A_arr, B_arr, C_arr, D_arr)
            cur_iter = get_T(KN, MN, PN, ksi_arr, etta_arr)
            tmp = find_max_diff(cur_iter, prev_iter)
        cur_T = cur_iter
        result.append(cur_T)
        t += tau
        tmp = find_max_diff(cur_T, prev_T)
    return (result)
```

Построение графиков

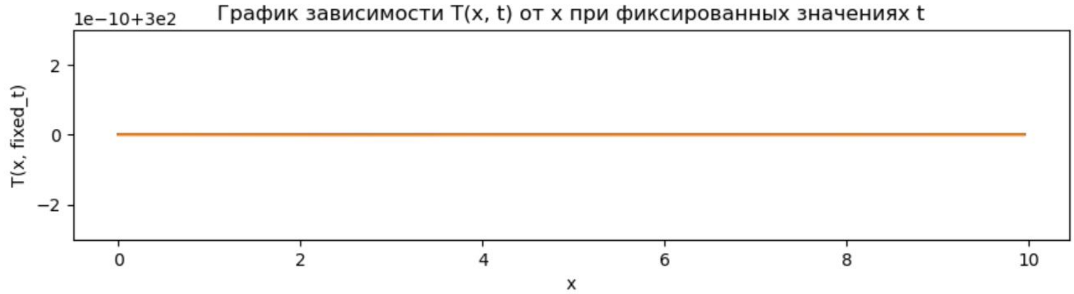
```
from matplotlib import pyplot as plt
def graph_fix_t_T_of_x(result):
    x = [i * h for i in range (N)]
    for i in result:
        plt.plot(x, i)
    plt.ylabel('T(x, fixed_t)')
    plt.xlabel('x')
    plt.title('График зависимости T(x, t) от x при
фиксированных значениях t')
    plt.show()

def graph_fix_x_T_of_t(result):
    length = len(result)
    t = [i * tau for i in range (length)]
```

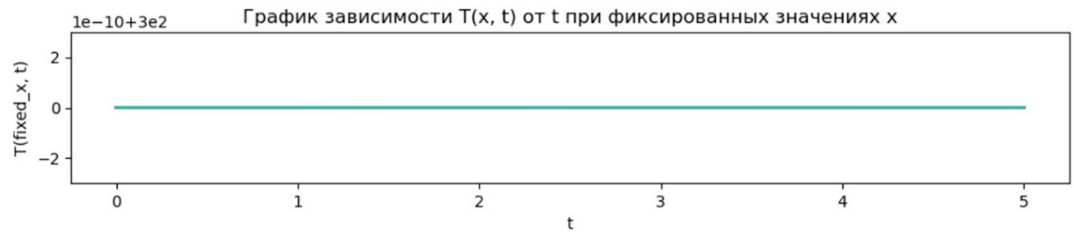
```
    for i in range(N):
     fixed_x = [result[j][i] for j in range(length)]
        print(fixed_x)
        plt.plot(t, fixed_x)
    plt.ylabel('T(fixed_x, t)')
    plt.xlabel('t')
    plt.title('График зависимости T(x, t) от t при
фиксированных значениях x')
    plt.show()
```

Вопросы при защите лабораторной работы

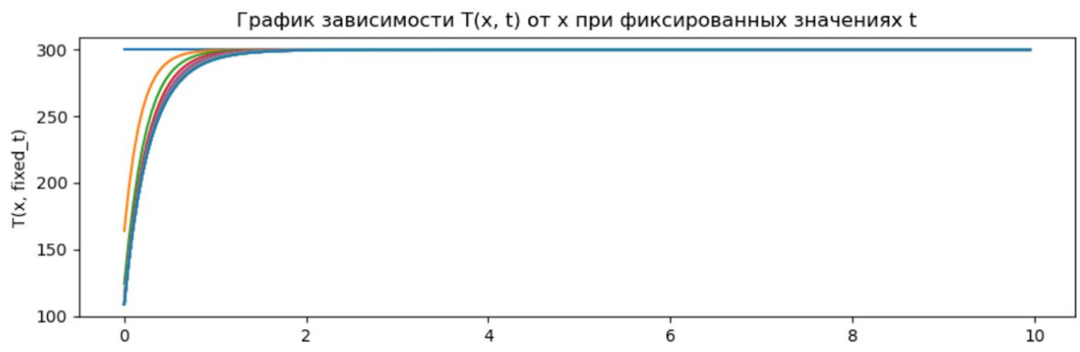
Приведите результаты тестирования программы (графики, общие соображения, качественный анализ). Учесть опыт выполнения лабораторной работы №3.

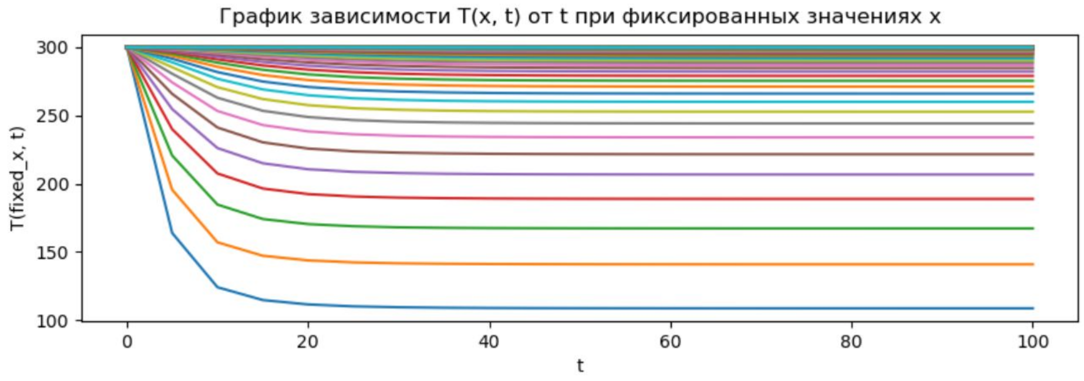
Можно рассмотреть разный поток F. При нулевом F нагревание (или охлаждение) отсутствует, температурное поле будет неизменно равно T0.



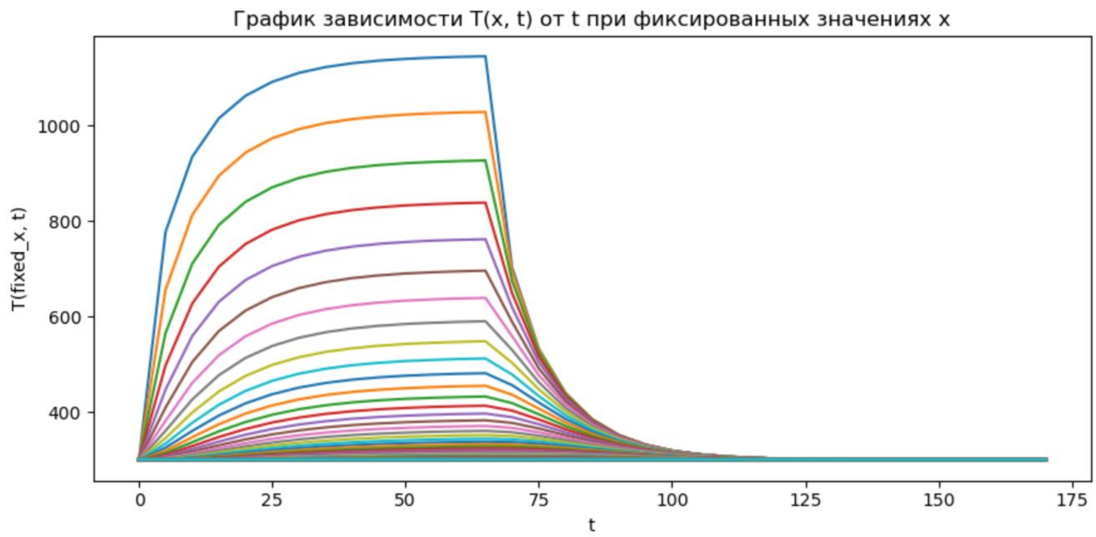


При положительном потоке мы видели как температура на левом конце стержня со временем увеличивается. При отрицательном потоке должны наблюдать его охлаждение.

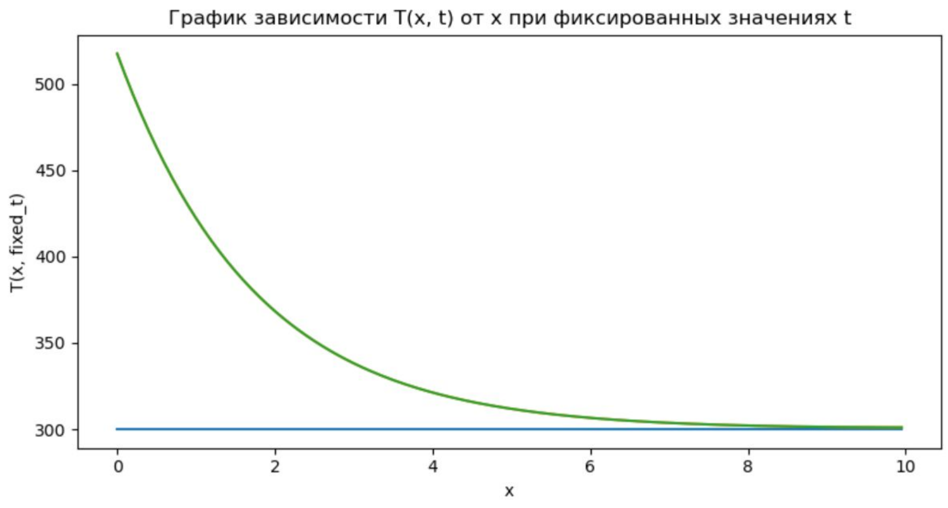




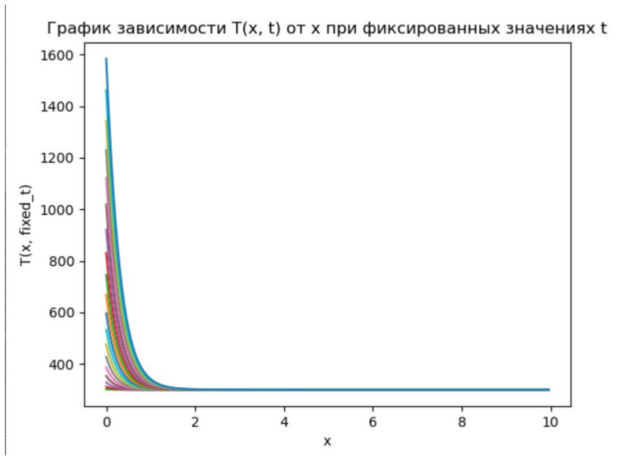
Если после разогрева стержня положить поток $F(t) = 0$, то будет происходить остывание, пока температура не выровняется по всей длине и не станет равной T_0. Здесь поток прекращается при $t = 60$

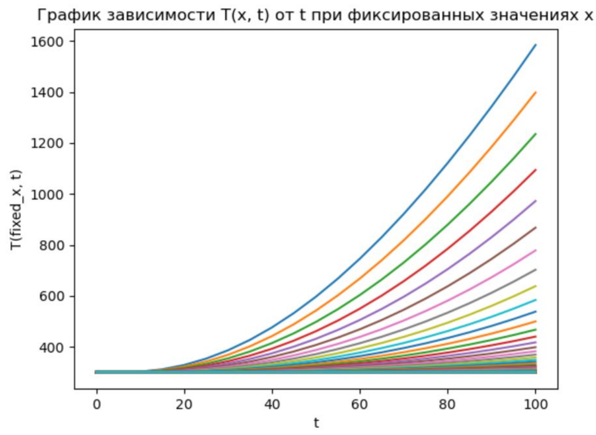


Если убрать зависимость коэффициента теплопроводности k от T, заменив его коэффициентом, зависящим только от координаты, как в предыдущей лабораторной работе, а теплоемкость c обнулить, то при выходе на стационарный режим полученный график идентичен графику полученному в лабораторной №3.



При произвольной зависимости потока $F(t)$ от времени температурное поле будет как-то сложным образом отслеживать поток. Например $F(t) = T^2/100$.





Выполните линеаризацию уравнения (9) по Ньютону, полагая для простоты, что все коэффициенты зависят только от одной переменной $\widehat{y}_n$. Приведите линеаризованный вариант уравнения и опишите алгоритм его решения. Воспользуйтесь процедурой вывода, описанной в лекции №8.

$$
\begin{aligned}
&K_0\widehat{y}_0 + M_0\widehat{y}_1 = P_0 \\
&\widehat{A}_n\widehat{y}_{n-1} - \widehat{B}_n\widehat{y}_n + \widehat{C}_n\widehat{y}_{n+1} = -\widehat{D}_n \quad , \qquad\qquad 1 \le n \le N-1 \\
&K_N\widehat{y}_N + M_N\widehat{y}_{N-1} = P_N
\end{aligned}
$$

Выполним линеаризацию методом Ньютона по переменным $\widehat{y}_{n-1}, \widehat{y}_n, \widehat{y}_{n+1}$. Обозначим текущую итерацию s, а предыдущую итерацию (s - 1). Предполагается, что на предыдущей итерации значения известны. Производные коэффициентов по всем переменным, кроме $\widehat{y}_n$, будут равны нулю, производная $\widehat{B}\widehat{y}_n$ будет посчитана как производная произведения функций.

$$
\begin{aligned}
&(\widehat{A}_n\widehat{y}_{n-1} - \widehat{B}_n\widehat{y}_n + \widehat{C}_n\widehat{y}_{n+1} + \widehat{D}_n)\Big|_{s-1} + \widehat{A}_n^{(s-1)}\Delta\widehat{y}^{(s)}_{n-1} + \\
&+ \Big(\frac{\partial\widehat{A}_n}{\partial\widehat{y}_n}\widehat{y}_{n-1} - \frac{\partial\widehat{B}_n}{\partial\widehat{y}_n}\widehat{y}_n - \widehat{B}_n + \frac{\partial\widehat{C}_n}{\partial\widehat{y}_n}\widehat{y}_{n+1} + \frac{\partial\widehat{D}_n}{\partial\widehat{y}_n}\Big)\Big|_{s-1}\Delta\widehat{y}_n^{(s)} + \widehat{C}^{(s-1)}_n\Delta\widehat{y}^{(s)}_{n+1} = 0
\end{aligned}
$$

Приведем к каноническому виду

$$
\begin{aligned}
&A_n\widehat{y}_{n-1} - B_n\widehat{y}_n + C_n\widehat{y}_{n+1} = -D_n \quad \text{, где} \\
&A_n = \widehat{A}_n^{(s-1)} \\
&B_n = \Big(\frac{\partial\widehat{A}_n}{\partial\widehat{y}_n}\widehat{y}_{n-1} - \frac{\partial\widehat{B}_n}{\partial\widehat{y}_n}\widehat{y}_n - \widehat{B}_n + \frac{\partial\widehat{C}_n}{\partial\widehat{y}_n}\widehat{y}_{n+1} + \frac{\partial\widehat{D}_n}{\partial\widehat{y}_n}\Big)\Big|_{s-1} \\
&C_n = \widehat{C}^{(s-1)}_n \\
&D_n = (\widehat{A}_n\widehat{y}_{n-1} - \widehat{B}_n\widehat{y}_n + \widehat{C}_n\widehat{y}_{n+1} + \widehat{D}_n)\Big|_{s-1}
\end{aligned}
$$

Эту систему уравнений с трехдиагональной матрицей также можно решить методом прогонки с краевыми условиями:

$$\Delta\widehat{y}^{(s)}{}_0 = 0; \qquad \Delta\widehat{y}^{(s)}{}_N = 0$$

В результате находятся все $\Delta\widehat{y}^{(s)}{}_n$, после чего находятся все значения функции для текущей итерации по следующей формуле:

$$\widehat{y}^{(s)}{}_n = \widehat{y}^{(s-1)}{}_n + \Delta\widehat{y}^{(s)}{}_n$$

В качестве начального приближения $\widehat{y}^{0}{}_n$ можно задать y_n с предыдущего шага $t = t_m$.

Итерационный процесс сходится при условии $max[\frac{\Delta\widehat{y}^{(s)}{}_n}{\widehat{y}^{(s)}{}_n}] \le \varepsilon,\ 1 \ \le n \le N$.