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 Сушина А.Д.

Группа ИУ7-61б

Оценка (баллы) _____

Преподаватель Градов В.М.

Москва.
2020 г

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

Исходные данные.

1. Задана математическая модель.

Уравнение для функции $T(x,t)$

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

Краевые условия

$$\left\{\begin{array}{c} t=0,\ T(x,0)=T_0 \\ x=0,\ -k(T(0))\dfrac{\partial T}{\partial x}=F_0 \\ x=l,\ -k(T(l))\dfrac{\partial T}{\partial x}=\alpha_N(T(l)-T_0) \end{array}\right\} \tag{2}$$

В обозначениях уравнения (14.1) лекции №14

$$p(x)=\frac{2}{R}\alpha(x),\quad f(u)\equiv f(x)=\frac{2T_0}{R}\alpha(x) \tag{3}$$

2. Разностная схема с разностным краевым условием при $x=0$ получена в Лекции №14 (14.6),(14.7) и может быть использована в данной работе.

Разностная схема:

$$\widehat{A}_n\widehat{y_{n-1}}-\widehat{B}_n\widehat{y}_n+\widehat{D}_n\widehat{y_{n+1}}=-\widehat{F}_n,\quad 1\le n\le N-1 \tag{4}$$

$$\widehat{A}_n=\widehat{X}_{n-\frac{1}{2}}\frac{\tau}{h} \tag{5}$$

$$\widehat{B}_n=\widehat{A}_n+\widehat{D}_n+\widehat{c}_n h+p_n h\tau \tag{6}$$

$$\widehat{D}_n=\widehat{X}_{n+\frac{1}{2}}\frac{\tau}{h} \tag{7}$$

$$\widehat{F}_n=f_n\tau h+\widehat{c}_n y_n h \tag{8}$$

Разностные аналоги краевых условий при x=0:

$$\left(\frac{h}{8}\widehat{c}_{\frac{1}{2}}+\frac{h}{4}\widehat{c}_0+\widehat{X}_{\frac{1}{2}}\frac{\tau}{h}+\frac{\tau h}{8}p_{\frac{1}{2}}+\frac{\tau h}{4}p_0\right)\widehat{y}_0+\left(\frac{h}{8}\widehat{c}_{\frac{1}{2}}-\widehat{X}_{\frac{1}{2}}\frac{\tau}{h}+\frac{\tau h}{8}p_{\frac{1}{2}}\right)\widehat{y}_1=$$

$$=\frac{h}{8}\widehat{c}_{\frac{1}{2}}(y_0+y_1)+\frac{h}{4}\widehat{c}_0 y_0+\widehat{F}\tau+\frac{\tau h}{4}(\widehat{f}_{\frac{1}{2}}+\widehat{c}_0) \tag{9}$$

Самостоятельно надо получить интегро-интерполяционным методом разностный аналог краевого условия при $x = l$. Для этого надо проинтегрировать на отрезке [xN-1/2, xN] выписанное выше уравнение (1) и учесть, что поток

$$F_N = \alpha_N (\widehat{y_N} - T_0) \tag{10}$$

$$F_{N-\frac{1}{2}} = \widehat{X_{N-\frac{1}{2}}} \frac{\widehat{y_{N-1}} - \widehat{y_N}}{h} \tag{11}$$

3. Значения параметров для отладки (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/см3К.

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 =0.528 105, m_2 =1.

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

α_0 =0.05 Вт/см2 К,

α_N =0.01 Вт/см2 К,

l = 10 см,

T_0 =300К,

R =0.5 см,

$F(t)$ =50 Вт/см2 (для отладки принять постоянным).

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 T, 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 2000K, физического смысла не имеют и практического интереса не представляют.

Код программы

Код программы представлен на листингах 1-2.

Листинг 1. Functions.py

```
import numpy as np
from data import data
from math import fabs
a1 = data['a1']
b1 = data['b1']
c1 = data['c1']
m1 = data['m1']
a2 = data['a2']
b2 = data['b2']
c2 = data['c2']
m2 = data['m2']
alpha0 = data['Alpha0']
alphaN = data['AlphaN']
l = data['l']
T0 = data['t0']
R = data['R']
F0 = data['F0']
h = data['h']
t = data['t']

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

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

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

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

def F(x, T):
```

```
    return f(x) * h * t + c(T) * T * h

def left_condition(y):
    c0 = c(y[0])
    c12 = func_plus_half(y[0], t, c)
    k12 = func_plus_half(y[0], t, k)
    p0 = p(0)
    p12 = p(h / 2)
    K0 = h / 8 * c12 + h / 4 * c0 + \
        k12 * t / h + \
        t * h / 8 * p12 + t * h / 4 * p0
    M0 = h / 8 * c12 - \
        k12 * t / h + \
        t * h * p12 / 8
    P0 = h / 8 * c12 * (y[0] + y[1]) + \
        h / 4 * c0 * y[0] + \
        F0 * t + t * h / 8 * (3 * f(0) + f(h))
    return K0, M0, P0

def right_condition(y):
    cn12 = func_minus_half(y[-1], t, c)
    cn = c(y[-1])
    kn12 = func_minus_half(y[-1], t, k)
    pn12 = p(l - h / 2)
    pn = p(l)
    fn = f(l)
    fn12 =  f(l - h / 2)
    KN = h / 8 * cn12 + h / 4 * cn + \
        kn12 * t / h + t * alphaN + \
        t * h / 8 * pn12 + t * h / 4 * pn
    MN = h / 8 * cn12 - \
        kn12 * t / h + \
        t * h * pn12 / 8
    PN = h / 8 * cn12 * (y[-1] + y[-2]) + \
        h / 4 * cn * y[-1] + t * alphaN * T0 + \
        t * h / 4 * (fn + fn12)
    return KN, MN, PN

def calculate(prev):
    K0,M0,P0 = left_condition(prev)
    KN,MN,PN = right_condition(prev)
    eps = [0, -M0 / K0]
    eta = [0,  P0 / K0]

    x = h
    n = 1
    while (x + h < l):
        new_eps = D(prev[n]) / (B(x, prev[n]) - A(prev[n]) * eps[n])
        new_eta = (F(x, prev[n]) + A(prev[n]) * eta[n]) / (B(x, prev[n]) - A(prev[n]) * eps[n])
        eps.append(new_eps)
        eta.append(new_eta)
        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 get_result():
    step1 = int(l / h)
    T = [T0] * (step1 + 1)
    T_new = [0] * (step1 + 1)
    ti = 0
    res = []
    res.append(T)
    lent = len(T)
    while True:
        prev = T
        while True:
            T_new = calculate(prev)
            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)]
    return res, x, te
```

Листинг 2. main.py

```
import matplotlib.pyplot as plt
import data as const
from functions import get_result
from tkinter import *
root = Tk()
varList = {
      'a1' : StringVar(),
    'b1' : StringVar(),
    'c1' : StringVar(),
    'm1': StringVar(),
    'a2': StringVar(),
    'b2': StringVar(),
    'c2': StringVar(),
    'm2': StringVar(),
    "Alpha0": StringVar(),
    "AlphaN": StringVar(),
    "l": StringVar(),
    "t0": StringVar(),
    "R": StringVar(),
    "F0": StringVar(),
    "h": StringVar(),
    't': StringVar(),
```

```
}

def create_grid(root):
	i = 0
	for var in varList.keys():
		label = Label(root, text=var)
		label.grid(row=i, column=0, sticky="e")
		entry = Entry(root,width=10,textvariable=varList[var])
		entry.grid(row=i, column=1)
		entry.insert(0, str(const.resetData[var]))
		i+=1

def check_is_num():
	for var in varList.values():
		try:
			float(var.get())
		except ValueError:
			return False
	return True

def start_work(Event):
	if not check_is_num():
		print("WARNING NOT DIGIT")
		return
	for var in varList.keys():
		const.data[var] = float(varList[var].get())
	res, x, te = get_result()
	plt.subplot(1, 2, 1)
	step1 = 0
	for i in res:
		if (step1 % 2 == 0):
			plt.plot(x, i[:-1])
		step1 +=1
	plt.title('T(x)')
	plt.plot(x , res[-1][:-1])
	plt.xlabel("x, sm")
	plt.ylabel("T, K")
	plt.grid()
	plt.subplot(1, 2, 2)
	h = const.data['h']
	step2 = 0
	while (step2 < 1/3):
		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__':

	btn = Button(root, text="START")
	create_grid(root)
	btn.bind("<Button-1>", start_work)
	btn.grid(column=1, padx=10, pady=10)
	root.mainloop()
```

Результаты работы.

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

Для получения разностного аналога краевого условия при $x = l$, надо проинтегрировать на отрезке [xN-1/2, xN] выписанное выше уравнение (1) и учесть (10) и (11)

Обозначим $F = -k(T)\frac{\partial T}{\partial x}$ (12)

Тогда (1) можно записать в виде:

$$c(T)\frac{\partial T}{\partial t} = -\frac{\partial F}{\partial x} - p(x)T + f(x) \quad (13)$$

где

$$p(x) = \frac{2}{R}\alpha(x)$$

$$f(x) = \frac{2T_0}{R} * \alpha(x)$$

Проитегрируем (13):

$$\int_{X_{N-\frac{1}{2}}}^{X_N} dx \int_{t_m}^{t_{m+1}} c(T)\frac{\partial T}{\partial t} dt = -\int_{t_m}^{t_{m+1}} dt \int_{X_{N-\frac{1}{2}}}^{X_N} \frac{\partial F}{\partial x} dx - \int_{X_{N-\frac{1}{2}}}^{X_N} dx \int_{t_m}^{t_{m+1}} p(x)T dt + \int_{X_{N-\frac{1}{2}}}^{X_N} dx \int_{t_m}^{t_{m+1}} f(x)\,dt \quad (14)$$

$$\int_{X_{N-\frac{1}{2}}}^{X_N} \widehat{c_n}(\hat{T} - T)dx = -\int_{t_m}^{t_{m+1}} (F_n - F_{N-\frac{1}{2}})dt - \int_{X_{N-\frac{1}{2}}}^{X_N} p\hat{T}\,\tau dx + \int_{X_{N-\frac{1}{2}}}^{X_N} \hat{f}\,\tau dx$$

Вычисляем интегралы

$$\frac{h}{4}\left(\widehat{c_N}(\widehat{y_N} - y_N) - \widehat{c_{N-\frac{1}{2}}}\left(\frac{\widehat{y_N} + \widehat{y_{N-1}}}{2} - \frac{y_N + y_{N+1}}{2}\right)\right) =$$

$$= -\tau\left(\alpha_N(\widehat{y_N} - T_0) - \widehat{X_N}\frac{\widehat{y_N} + \widehat{y_{N-1}}}{h}\right) - \left(p_N\widehat{y_N} - p_{n-\frac{1}{2}}\frac{\widehat{y_N} + \widehat{y_{N-1}}}{2}\right)\frac{\tau h}{4} + \left(\widehat{f_N} - \widehat{f_{N-\frac{1}{2}}}\right)\frac{\tau h}{4} \quad (15)$$

Приведем к виду $\widehat{K_N}\widehat{y_N} + \widehat{M_N}\widehat{y_{N-1}} = \widehat{P_N}$. Получим:

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

$$= \alpha_N\tau T_0 + \frac{h}{4}\widehat{C_N}y_N + \frac{h}{8}\widehat{c_{N-\frac{1}{2}}}(y_N + y_{N-1}) + \frac{h}{4}\tau(\widehat{f_N} + \widehat{f_{N-\frac{1}{2}}}) \quad (16)$$

Для функций c, X, p будет принята простая аппроксимация.

$$p_{N-\frac{1}{2}} = \frac{p_{N-1} + p_N}{2} \tag{17}$$

Из (9) и (16) получим $K_0, M_0, P_0, K_N, M_N, P_N$.

Получим систему:

$$\begin{cases} \widehat{K_0}\,\widehat{y_0} + \widehat{M_0}\,\widehat{y_1} = \widehat{P_0} \\ \widehat{A_n}\,\widehat{y_{n-1}} - \widehat{B_n}\,\widehat{y_n} + \widehat{D_n}\,\widehat{y_{n+1}} = -\widehat{F_n}, 1 \le n \le N-1 \\ \widehat{K_N}\,\widehat{y_N} + \widehat{M_{N-1}}\,\widehat{y_{N-1}} = \widehat{P_N} \end{cases} \tag{18}$$

Эту систему можно решить методом итераций. Пусть i — номер итерации

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

2. График зависимости температуры $T(x, t_m)$ от координаты x при нескольких фиксированных значениях времени t_m (аналогично рисунку в лекции №14) при заданных выше параметрах. Обязательно представить распределение $T(x,t)$ в момент времени, соответствующий установившемуся режиму, когда поле перестает меняться с некоторой точностью (например, $\left| \frac{T(t+\tau) - T(t)}{T(t+\tau)} \right| < 10^{-4}$), т.е. имеет место выход на стационарный режим. На этой стадии левая часть дифференциального уравнения близка к нулю, и на самом деле решается уравнение из лабораторной работы №3 (отличие только в том, что там было линейное уравнение).

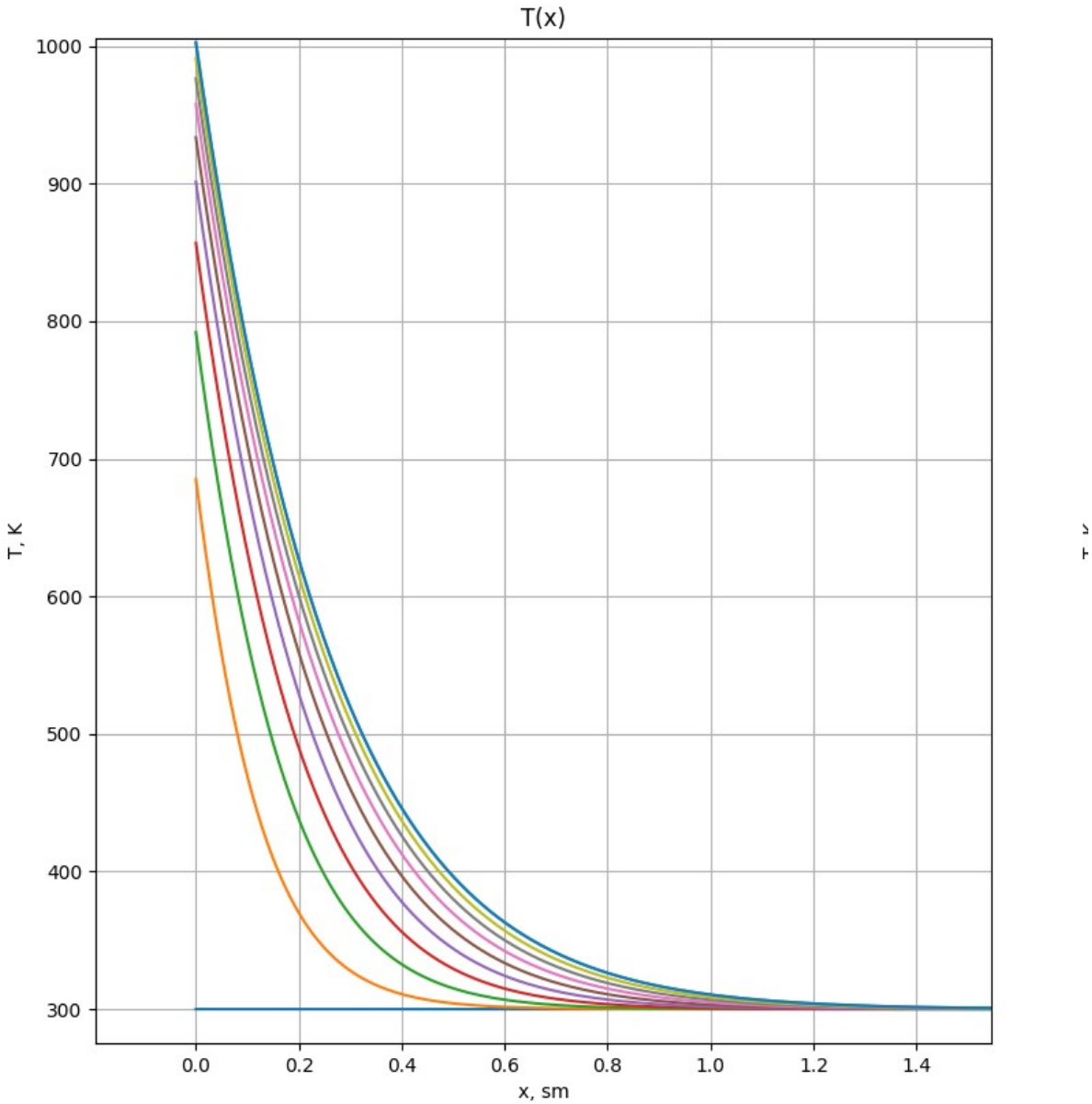

Рис 1. График зависимости температуры $T(x,t_m)$ от координаты x при нескольких фиксированных значениях времени t_m

3. График зависимости $T(x_n,t)$ при нескольких фиксированных значениях координаты x_n.

Обязательно представить случай *n*=0, т.е. $x=x_0=0$.

Синий график — x=0

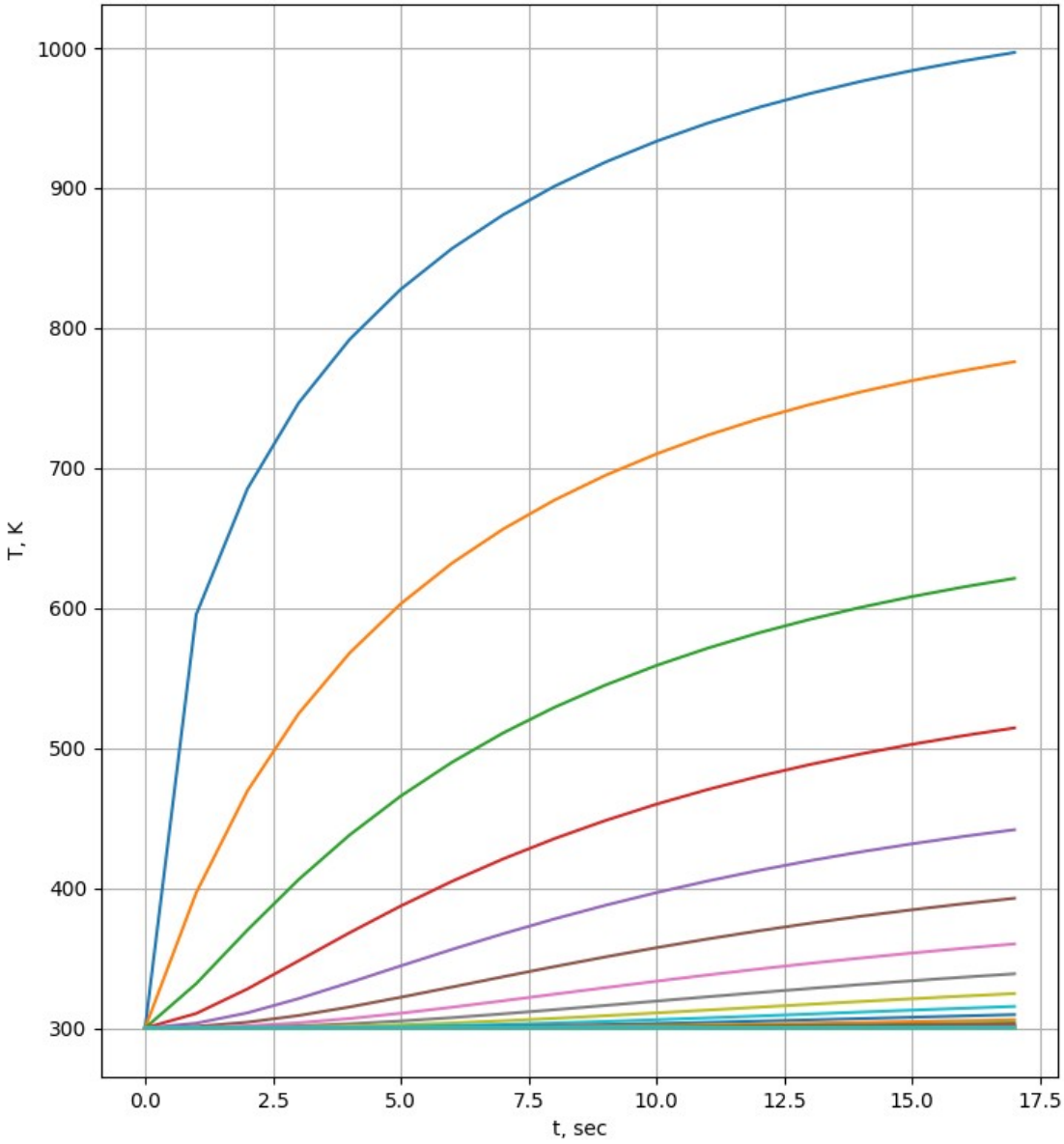

Рис 2. График зависимости $T(x_n, t)$ при нескольких фиксированных значениях координаты x_n

Вопросы при защите лабораторной работы.

Ответы на вопросы дать письменно в Отчете о лабораторной работе.

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

Если принять F0=-10. При отрицательном тепловом потоке слева идет съем тепла.

Для ЛР4:

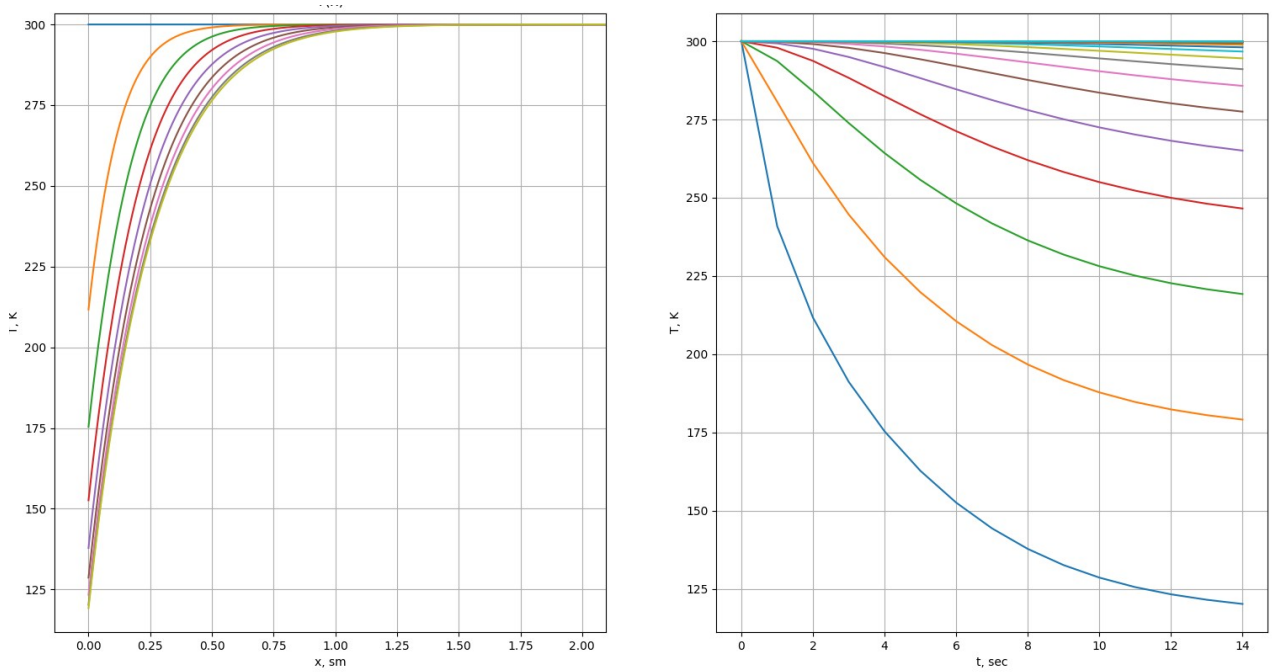

Рис 3. Результат работы программы при f0=-10

Для ЛР3:

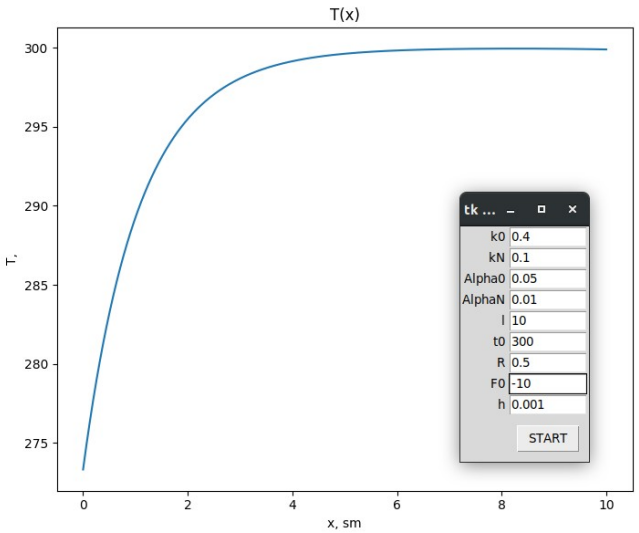

Рис 4. Результат работы ЛР3 при f0=-10

Если принять F0 =0.Тепловое нагружение отсутствует, причин для нагрева нет, температура стержня должна быть равна температуре окружающей среды T0.

ЛР4:

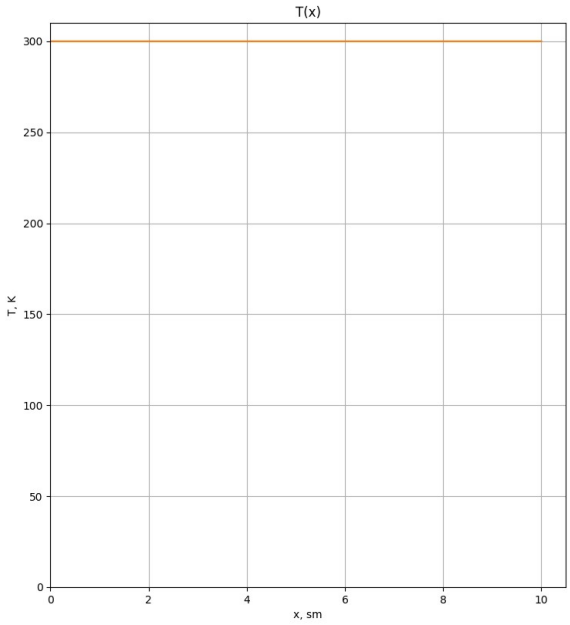

Рис 5. Результат работы программы

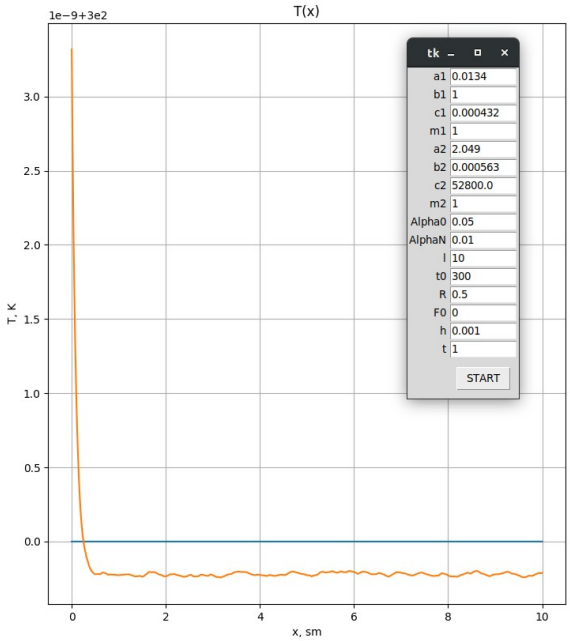

Рис 6. Погрешность

2. Выполните линеаризацию уравнения (14.8) по Ньютону, полагая для простоты, что все коэффициенты зависят только от одной переменной $\coprod y_n$. Приведите линеаризованный вариант уравнения и опишите алгоритм его решения. Воспользуйтесь процедурой вывода, описанной в лекции №8.

Система уравнений:

$$\begin{cases} \widehat{K}_0 \widehat{y}_0 + \widehat{M}_0 \widehat{y}_1 = \widehat{P}_0 \\ \widehat{A}_n \widehat{y_{n-1}} - \widehat{B}_n \widehat{y}_n + \widehat{D}_n \widehat{y_{n+1}} = -\widehat{F}_n, 1 \le n \le N-1 \\ \widehat{K}_N \widehat{y}_N + \widehat{M_{N-1}} \widehat{y_{N-1}} = \widehat{P}_N \end{cases} \quad (19)$$

Коэффициенты зависят от $\widehat{y}_n$. Исходя из этого получим:

$$\left(\widehat{A}_n \widehat{y_{n-1}} - \widehat{B}_n \widehat{y}_n + \widehat{D}_n \widehat{y_{n+1}} + \widehat{F}_n\right)\Big|_{i-1} + \widehat{A}_n^{(i-1)} \Delta \widehat{y_{n-1}^{(i)}} +$$

$$+ \left(\frac{\partial \widehat{A}_n}{\partial \widehat{y}_n} \widehat{y_{n-1}} - \frac{\partial \widehat{B}_n}{\partial \widehat{y}_n} \widehat{y}_n - \widehat{B}_n + \frac{\partial \widehat{D}_n}{\partial \widehat{y}_n} \widehat{y_{n+1}} + \frac{\partial \widehat{F}_n}{\partial \widehat{y}_n}\right)\Big|_{(i-1)} * \Delta \widehat{y}_n^{(i)} + \widehat{D}_n^{(i-1)} \Delta \widehat{y_{n+1}}^{(i)} = 0 \quad (20)$$

Канонический вид:

$$A_n \Delta \widehat{y_{n-1}^{(i)}} - B_n \Delta \widehat{y}_n^{(i)} + D_n \Delta \widehat{y_{n+1}}^{(i)} = -F_n, \quad (21)$$

где:

$$A_n = \widehat{A}_n^{(i-1)} \quad (22)$$

$$B_n=\left(\frac{-\partial\widehat{A_n}}{\partial\widehat{y_n}}\widehat{y_{n-1}}+\frac{\partial\widehat{B_n}}{\partial\widehat{y_n}}\widehat{y_n}+\widehat{B_n}-\frac{\partial\widehat{D_n}}{\partial\widehat{y_n}}\widehat{y_{n+1}}-\frac{\partial\widehat{F_n}}{\partial\widehat{y_n}}\right)\Bigg|_{i-1)} \quad (23)$$

$$D_n=\widehat{D_n}^{(i-1)} \quad (24)$$

$$F_n=\left(\widehat{A_n}\widehat{y_{n-1}}-\widehat{B_n}\widehat{y_n}+\widehat{D_n}\widehat{y_{n+1}}+\widehat{F_n}\right)\Big|_{i-1} \quad (25)$$

Подставим краевые условия и получим их в каноническом виде:

$$K_0\Delta\widehat{y_0}^{(i)}+M_0\Delta\widehat{y_1}^{(i)}=P_0$$

$$K_N\Delta\widehat{y_N}^{(i)}+M_{N-1}\Delta\widehat{y_{N-1}}^{(i)}=P_N$$

где

$$K_0=\widehat{K_0}^{(i-1)} \quad (26)$$

$$M_0=\widehat{M_0}^{(i-1)} \quad (27)$$

$$P_0=\left(\widehat{K_0}\widehat{y_0}+\widehat{M_0}\widehat{y_1}-\widehat{P_0}\right)\Big|_{i-1)} \quad (28)$$

$$K_N=\widehat{K_N}^{(i-1)} \quad (29)$$

$$M_N=\widehat{M_{N-1}}^{(i-1)} \quad (30)$$

$$P_N=\left(\widehat{K_N}\widehat{y_N}+\widehat{M_{N-1}}\widehat{y_{N-1}}-\widehat{P_N}\right)\Big|_{i-1)} \quad (31)$$

Получаем систему:

$$\begin{cases} K_N\Delta\widehat{y_N}^{(i)}+M_{N-1}\Delta\widehat{y_{N-1}}^{(i)}=P_N \\ K_0\Delta\widehat{y_0}^{(i)}+M_0\Delta\widehat{y_1}^{(i)}=P_0 \\ A_n\Delta\widehat{y_{n-1}}^{(i)}-B_n\Delta\widehat{y_n}^{(i)}+D_n\Delta\widehat{y_{n+1}}^{(i)}=-F_N \end{cases} \quad (32)$$

Для решения системы необходимо найти все $\Delta\widehat{y_n}^{(i)}$. Зная приближение (i-1), можно найти приближение (i). Найдём значение искомой функции в узлах:

$$\widehat{y_n}^{(i)}=\widehat{y_n}^{(i-1)}+\widehat{y_n}^{(i)} \quad (33)$$

Условие завершения:

$$max\left|\frac{\Delta\widehat{y_n}^{(i)}}{\widehat{y_n}^{(i)}}\right|\le\varepsilon,n=1,'N \quad (34)$$