|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 **ПРОГРАМНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.03 ПРОГРАММНАЯ ИНЖЕНЕРИЯ**

**Отчет**

|  |  |
| --- | --- |
| **По лабораторной работе №** | 4 |

**Дисциплина:** Моделирование

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-62Б |  |  | Н.А. Гарасев |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | В.М. Градов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2021

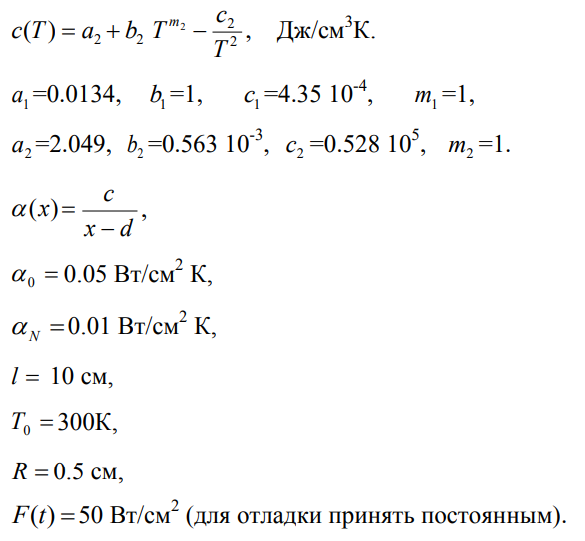
**Тема:** Программно-алгоритмическая реализация моделей на основе дифференциальных уравнений в частных производных с краевыми условиями II и III рода.

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

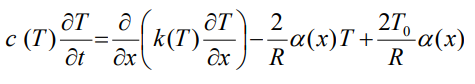
**Исходные данные:**

1. Дано:

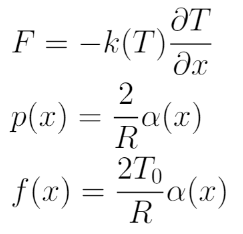
*K(T) = a1(b1 +c1Tm1),* Вт/см К,



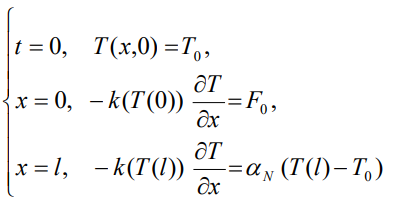
1. Уравнение:



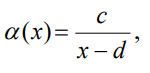
1. В обозначениях уравнения лекции



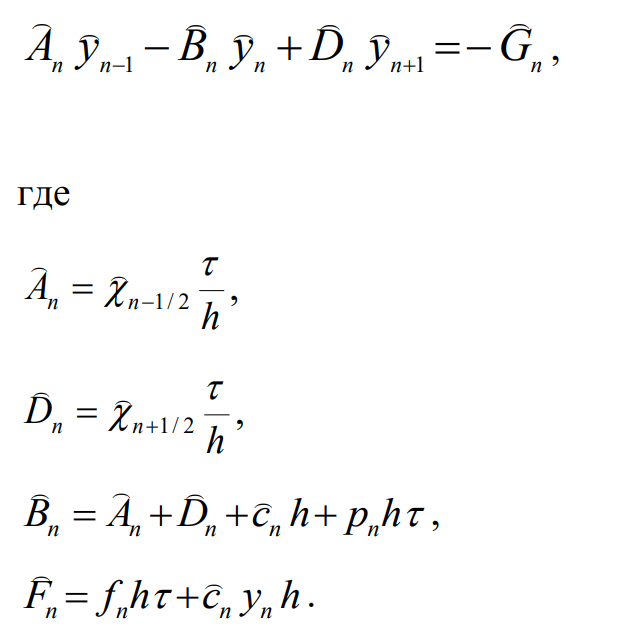
1. Краевые условия



Функция a(x) по условию:

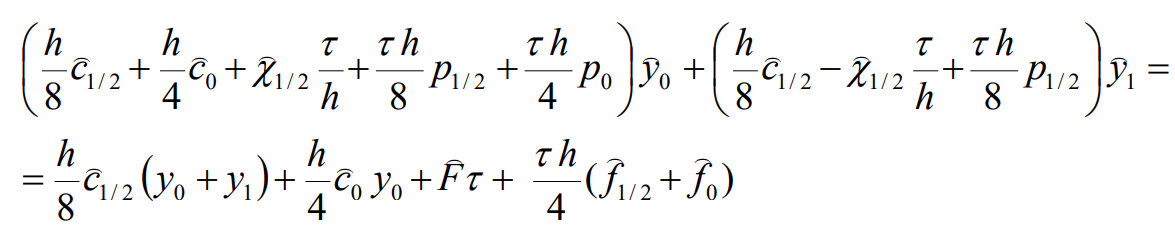


1. Разностная схема



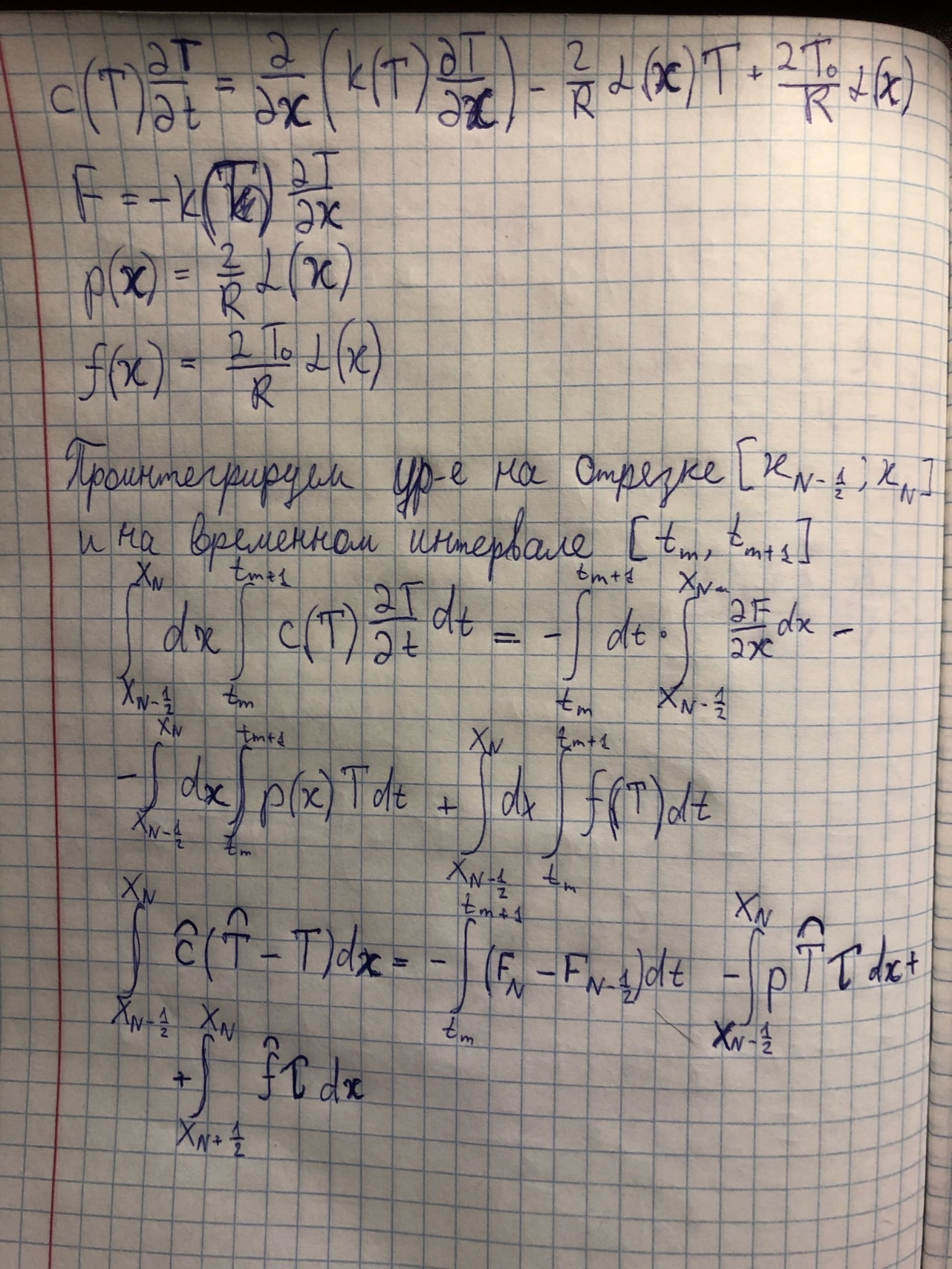
1. Краевые условия

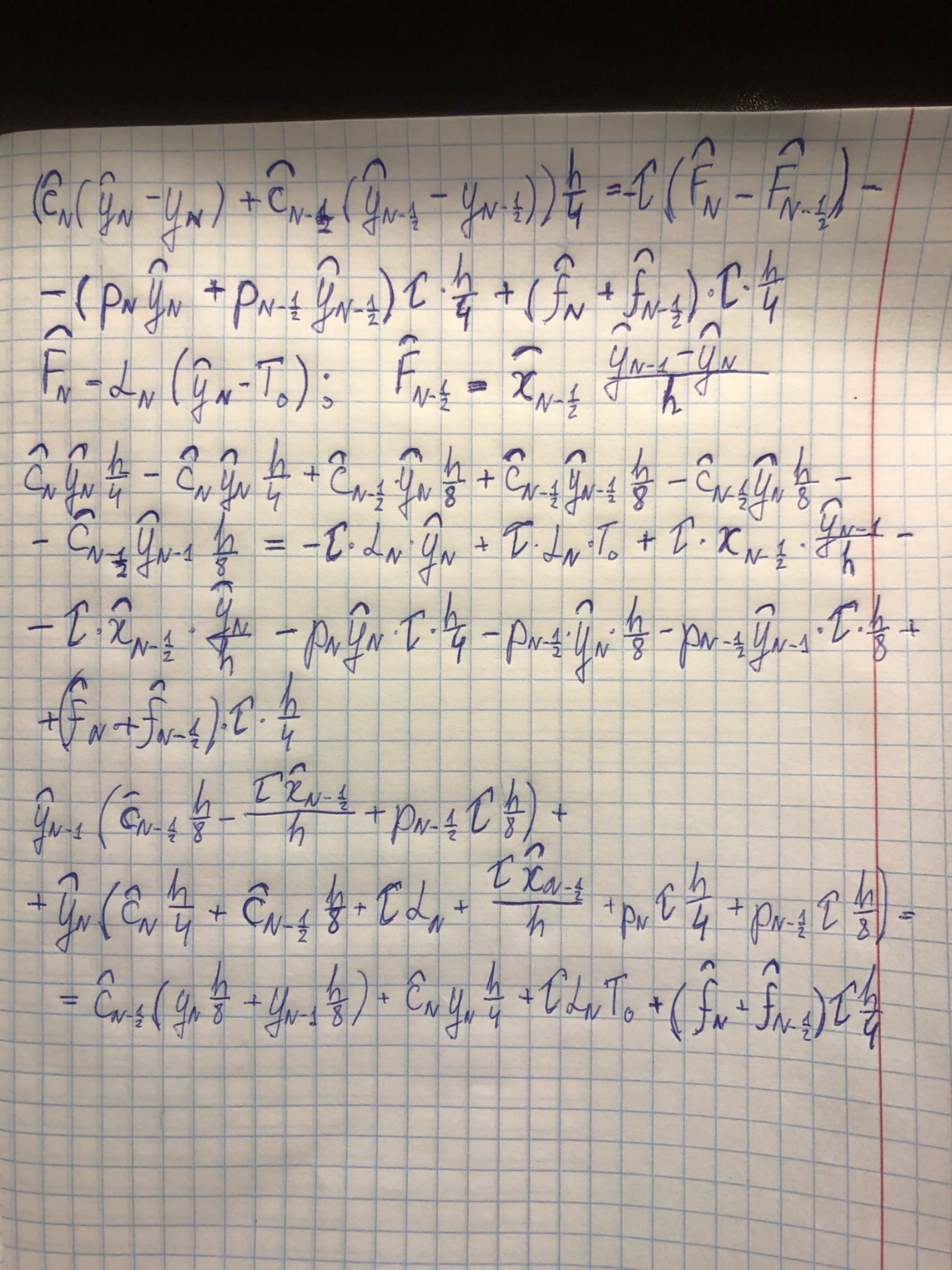
Разностные аналоги краевых условий при *x = 0*:



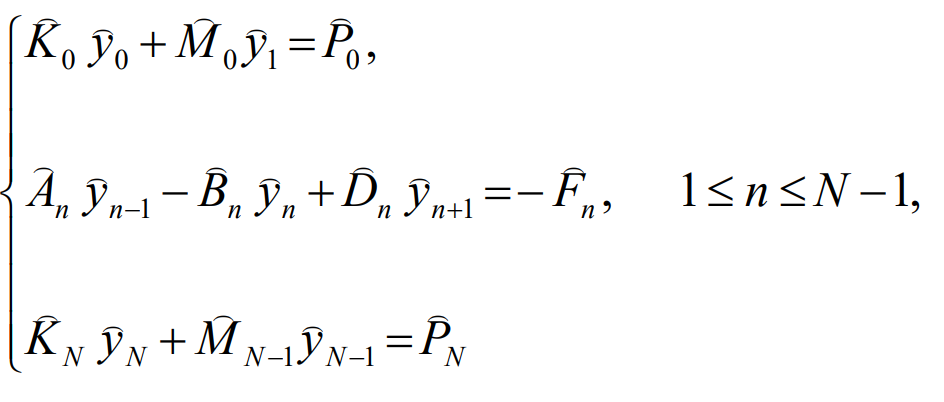
, взяты из лекций 10-11.

Разностные аналоги краевых условий при x=l:

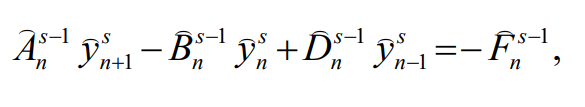




Используя полученные краевые условия, можно найти коэффициенты *K0, M0, P0, KN, MN, PN*.

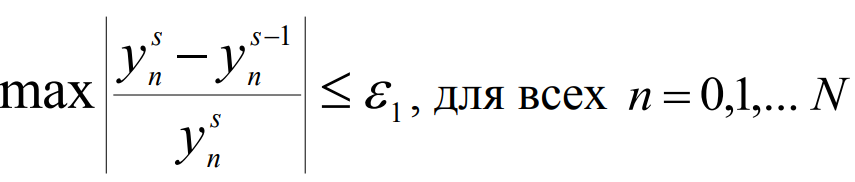
****

1. Метод простых итераций.



, где s – номер итерации.

Итерации прекращаются при условии:



**Листинг программы**

Листинг 1. Данные.

**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 = 0.01  
t = 1

Листинг 2. Вспомогательные функции

**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\_1\_2(x, step, func):  
 **return** (func(x) + func(x + step)) / 2  
  
**def** func\_minus\_1\_2(x, step, func):  
 **return** (func(x) + func(x - step)) / 2  
  
**def** A(T):  
 **return** func\_minus\_1\_2(T, t, k) \* t / h  
  
**def** D(T):  
 **return** func\_plus\_1\_2(T, t, k) \* t / h  
  
**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

Листинг 3. Расчет коэффициентов.

K0 = h / 8 \* func\_plus\_1\_2(prevT[0], t, c) + h / 4 \* c(prevT[0]) + \  
 func\_plus\_1\_2(prevT[0], t, k) \* t / h + \  
 t \* h / 8 \* p(h / 2) + t \* h / 4 \* p(0)  
  
M0 = h / 8 \* func\_plus\_1\_2(prevT[0], t, c) - \  
 func\_plus\_1\_2(prevT[0], t, k) \* t / h + \  
 t \* h \* p(h / 2) / 8  
  
P0 = h / 8 \* func\_plus\_1\_2(prevT[0], t, c) \* (prevT[0] + prevT[1]) + \  
 h / 4 \* c(prevT[0]) \* prevT[0] + \  
 F0 \* t + t \* h / 8 \* (3 \* f(0) + f(h))  
  
KN = h / 8 \* func\_minus\_1\_2(prevT[-1], t, c) + h / 4 \* c(prevT[-1]) + \  
 func\_minus\_1\_2(prevT[-1], t, k) \* t / h + t \* alphaN + \  
 t \* h / 8 \* p(l - h / 2) + t \* h / 4 \* p(l)  
  
MN = h / 8 \* func\_minus\_1\_2(prevT[-1], t, c) - \  
 func\_minus\_1\_2(prevT[-1], t, k) \* t / h + \  
 t \* h \* p(l - h / 2) / 8  
  
PN = h / 8 \* func\_minus\_1\_2(prevT[-1], t, c) \* (prevT[-1] + prevT[-2]) + \  
 h / 4 \* c(prevT[-1]) \* prevT[-1] + t \* alphaN \* T0 + \  
 t \* h / 4 \* (f(l) + f(l - h / 2))

Листинг 4. Прямой ход.

eps = [0, -M0 / K0]  
eta = [0, P0 / K0]  
  
x = h  
n = 1  
**while** (x + h < l):  
 eps.append(D(prevT[n]) / (B(x, prevT[n]) - A(prevT[n]) \* eps[n]))  
 eta.append((F(x, prevT[n]) + A(prevT[n]) \* eta[n]) / (B(x, prevT[n]) \  
 - A(prevT[n]) \* eps[n]))  
 n += 1  
 x += h

Листинг 5. Обратный ход.

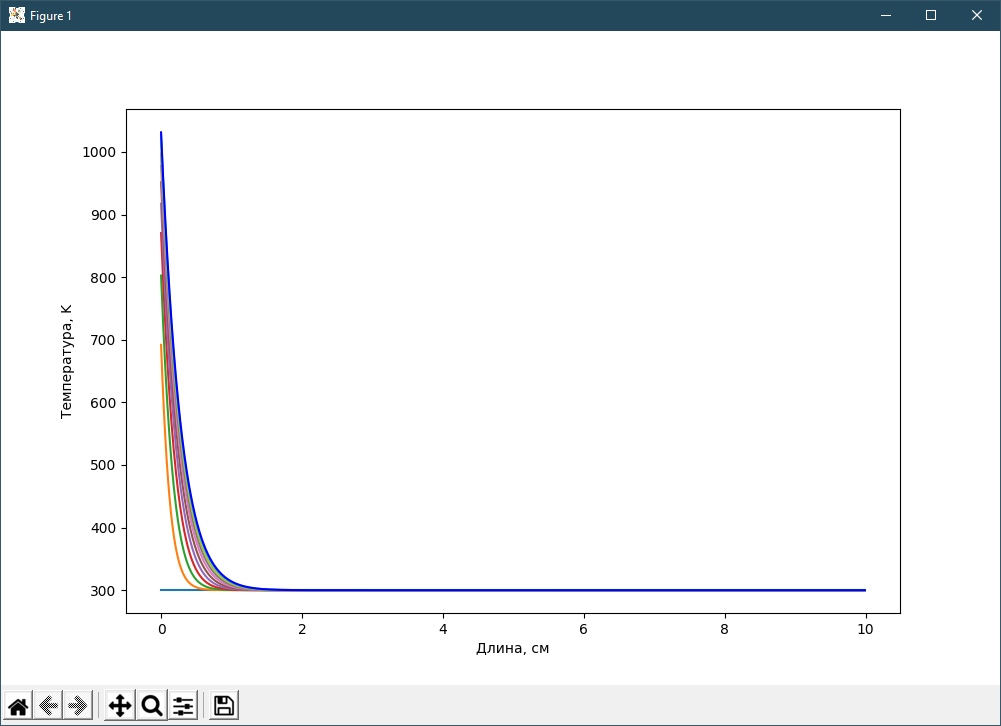
y = [0] \* (n + 1)  
y[n] = (PN - MN \* eta[n]) / (KN + MN \* eps[n])  
  
**for** i **in** range(n - 1, -1, -1):  
 y[i] = eps[i + 1] \* y[i + 1] + eta[i + 1]  
  
**return** y

Листинг 6. Проверка выхода из цикла.

**def** check\_eps(T, newT):  
 **for** i, j **in** zip(T, newT):  
 **if** fabs((i - j) / j) > 1e-2:  
 **return True  
 return False  
  
def** check\_iter(T, newT):  
 max = fabs((T[0] - newT[0]) / newT[0])  
 **for** i, j **in** zip(T, newT):  
 d = fabs(i - j) / j  
 **if** d > max:  
 max = d  
 **return** max < 1

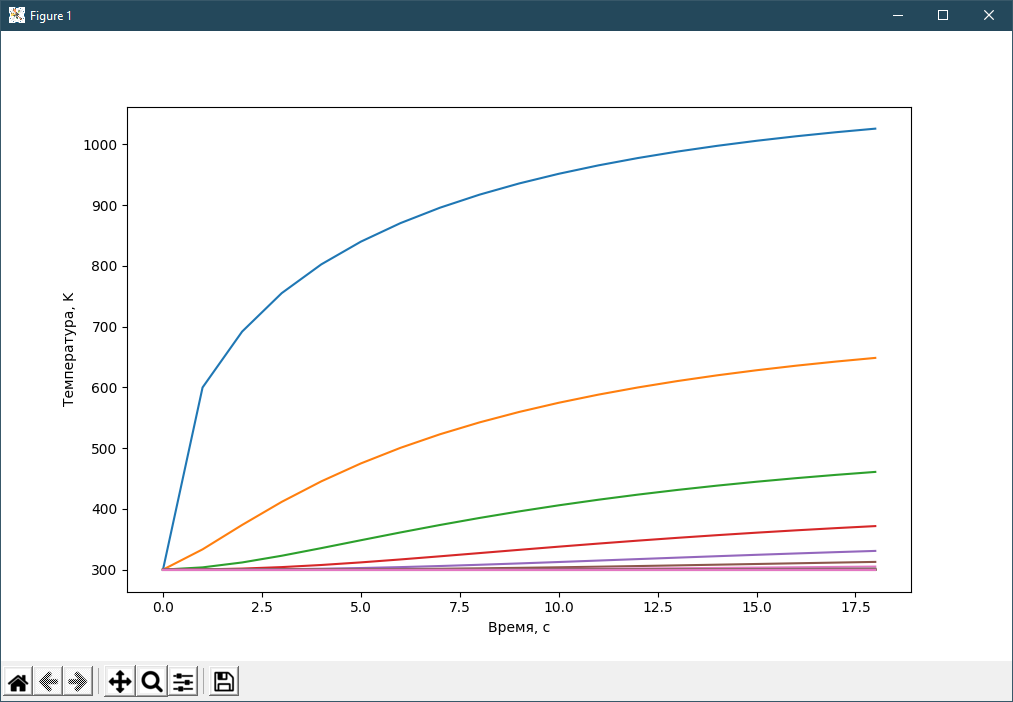
**Результаты работы.**

График зависимости температуры от координаты при нескольких фиксированных значениях времени.



При фиксированных значениях времени t = 0, 2, 4, 8, …. Синяя кривая (самая нижняя) соответствует установившемуся режиму, когда поле меняется с точность eps = 1e-3.

График зависимости температуры от времени при нескольких фиксированных значениях координаты.

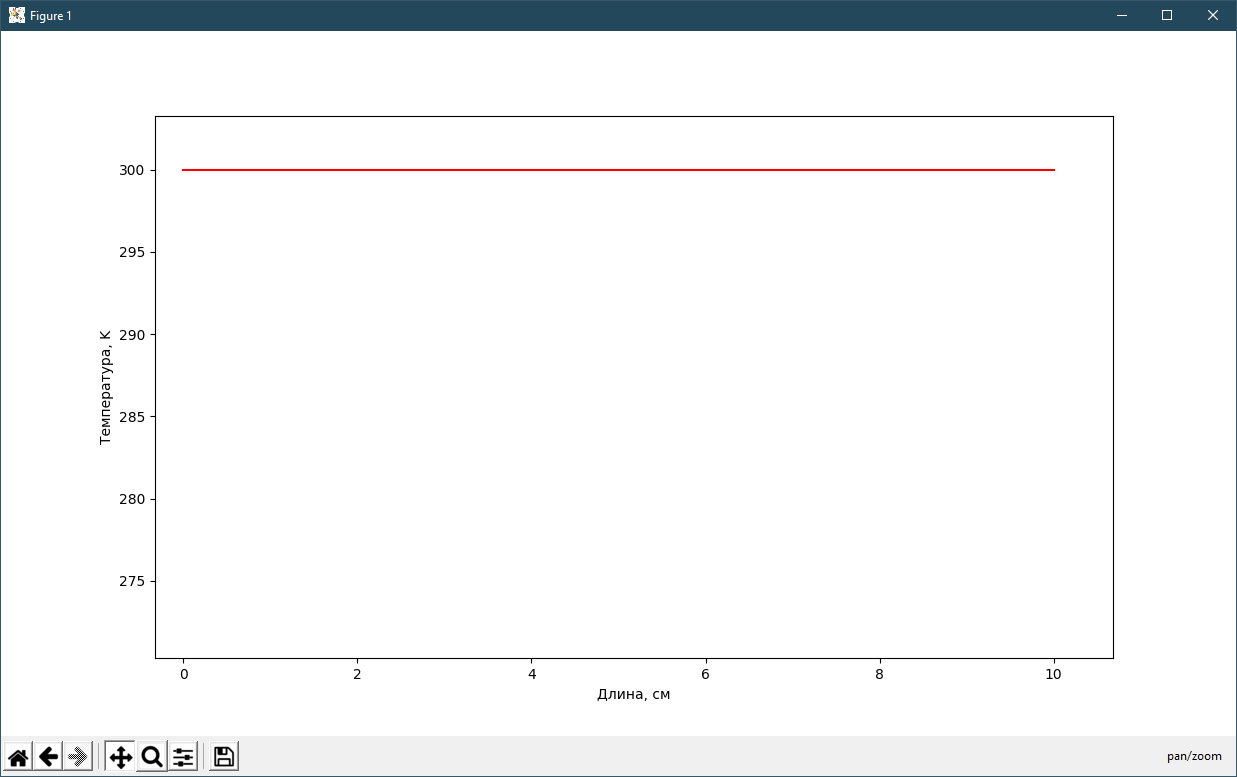


Длина *x* изменяется от 0,2 до 3,2 с шагом 0,2.

**Ответы на вопросы.**

Протестируем программу на примерах из третьей лабораторной работы.

При F0 = 0, температура стержня должна оставаться неизменной и должна быть равна температуре внешней среды T0.



При отрицательном тепловом потоке (F0) происходит съем тепла. Чему соответствуют графики, представленные ниже.

