|  |  |
| --- | --- |
| 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 №** | 3 |

**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 ОДУ второго порядка с краевыми условиями II и III рода.

**Цель работы:** Получение навыков разработки алгоритмов решения краевой задачи при реализации моделей, построенных на ОДУ второго порядка.

**Исходные данные:**

1. Дано:

*np* = 1.4 – коэффициент преломления,

*l* = 0.2 см – толщина слоя,

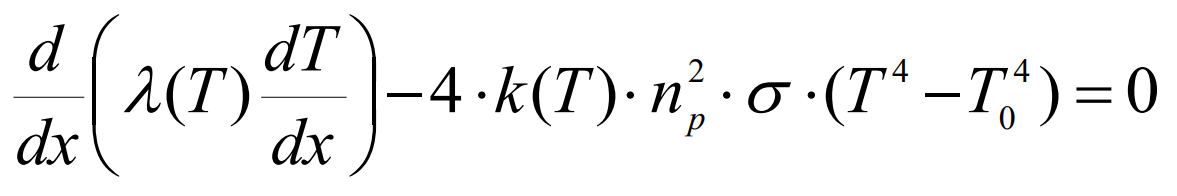
*T0* = 300К – температура окружающей среды,

*σ* = 5.668 10-12 Вт/(см2K4)- постоянная Стефана- Больцмана,

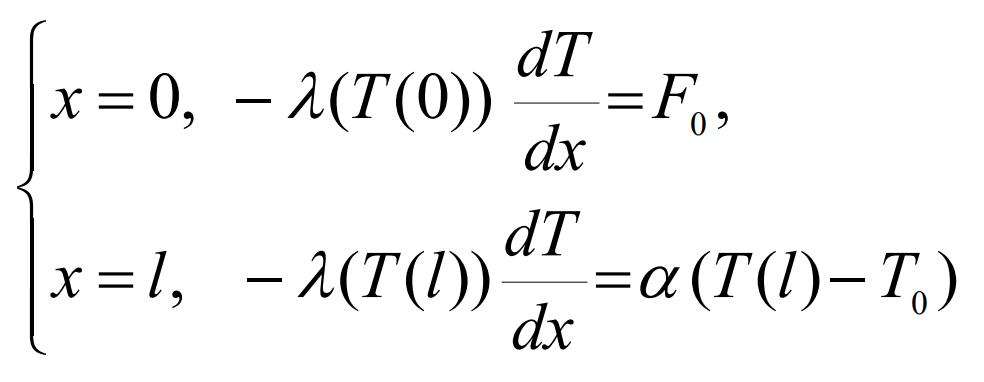
*F0* = 100 Вт/см2 - поток тепла,

*α* = 0.05 Вт/(см2 К) – коэффициент теплоотдачи.

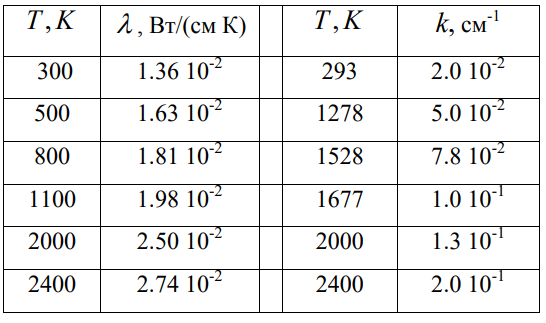
1. Уравнение:



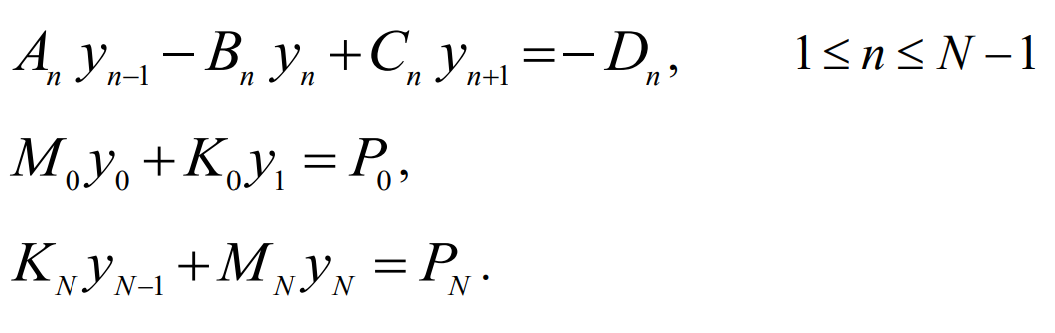
1. Краевые условия



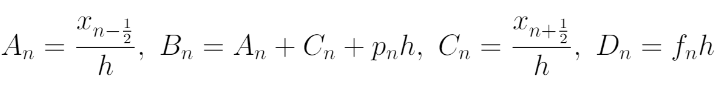
Функции *lambda(T)* и *k(T)* заданы таблицей:



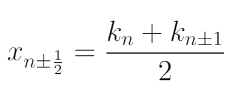
1. Разностная схема



Где,



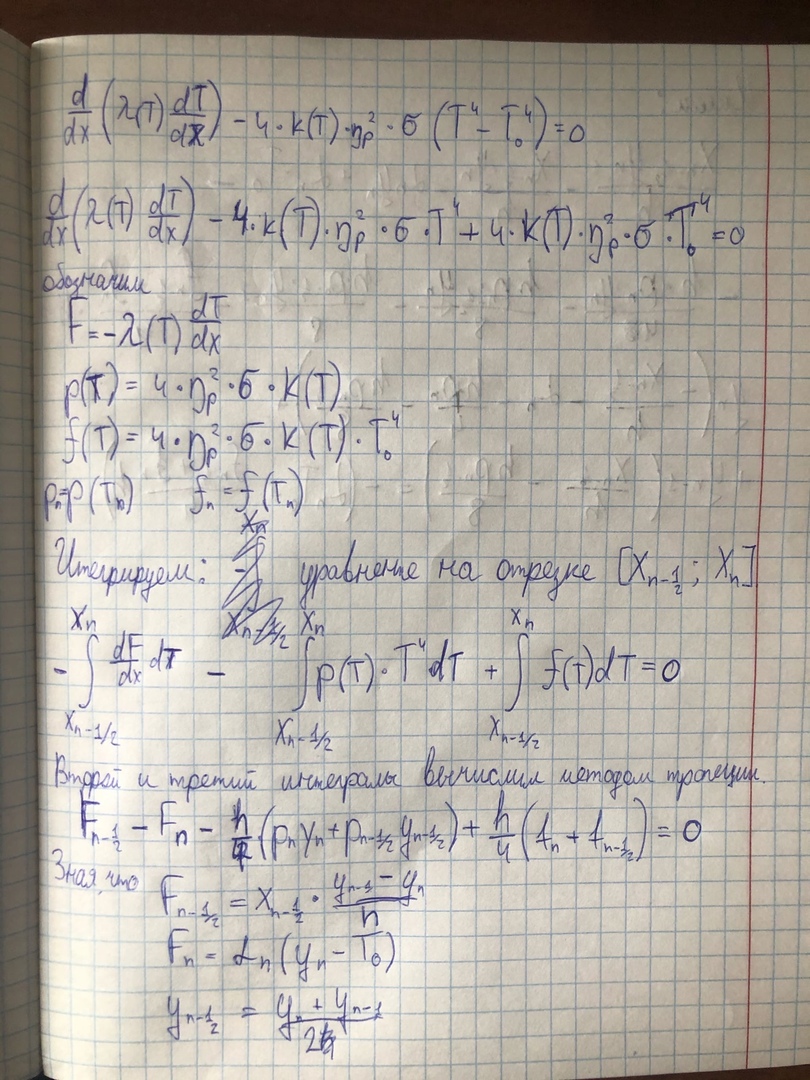
Способ вычисления – метод средних:



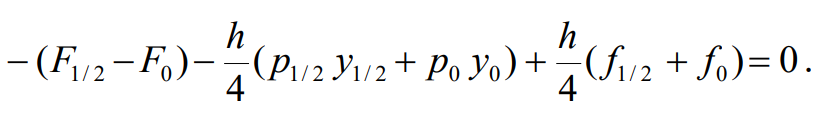
Система совместно с краевыми условиями решается методом прогонки.

1. Краевые условия

Обозначим:

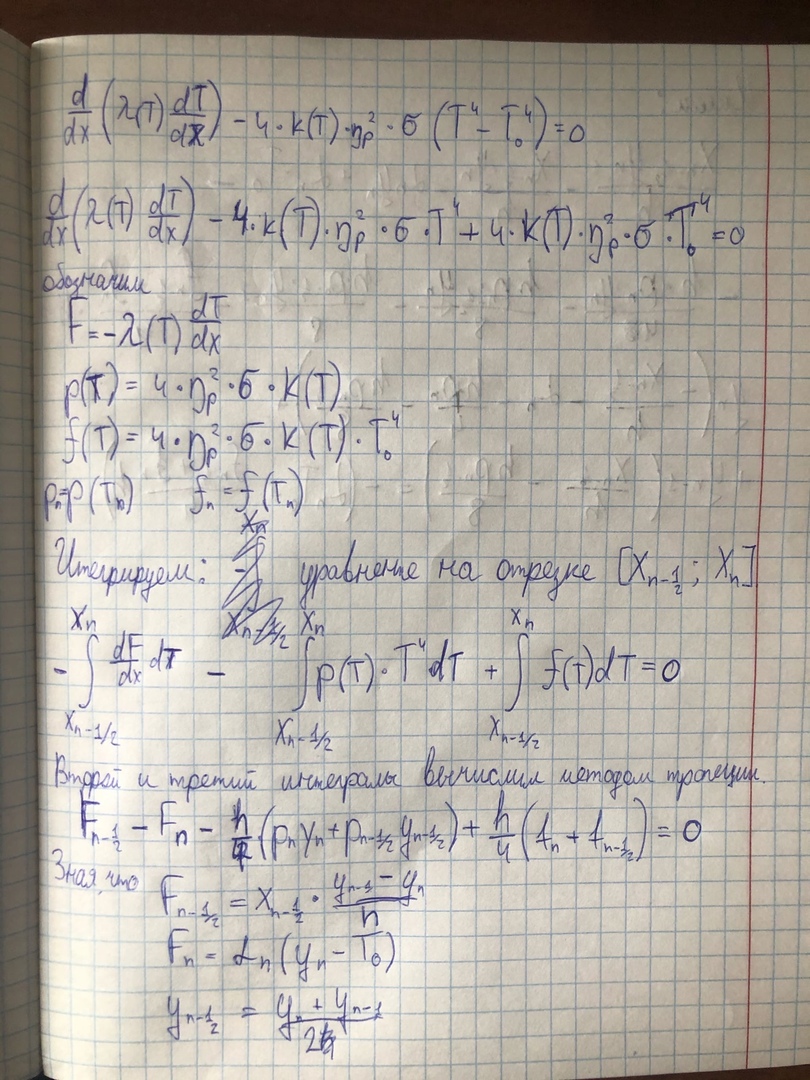


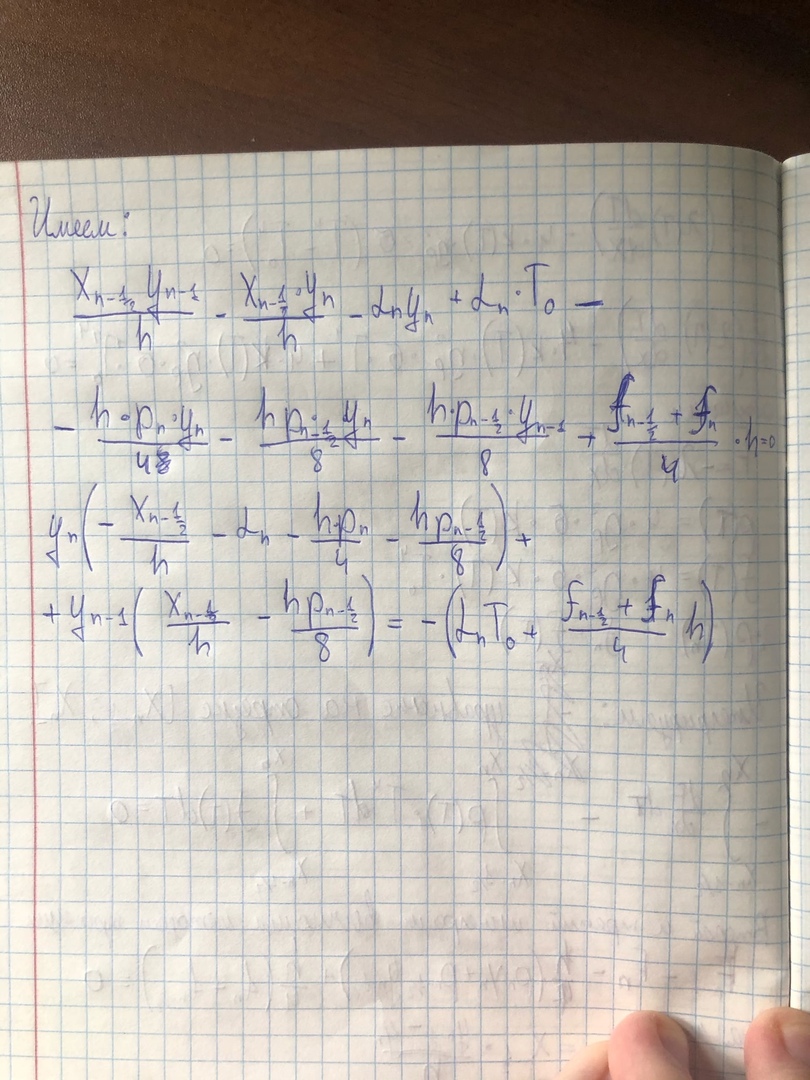
Разностные аналоги краевых условий при *x = 0*:



, взяты из лекции 7.

Разностные аналоги краевых условий при x = l:



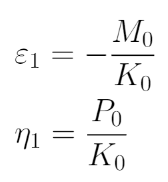


Опираясь на уравнения краевых условий, находим коэффициенты K0, M0, P0, KN, MN, PN.

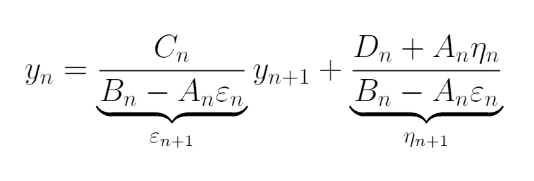
1. Решение.

Для решения системы, используем метод прогонки итеративно.

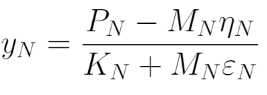
С начала каждой итерации вычисляются начальные значения прогоночных коэффициентов:



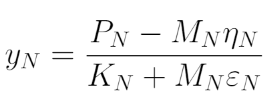
Далее в каждой итерации вычисляются массивы прогоночных коэффициентов:



Последнее значение функции в последней точке:



Затем в каждой итерации обратной прогонкой находятся значения:



**Листинг программы**

Листинг 1. Дано.

np = 1.4  
l = 0.2  
T0 = 300  
T\_CONST = 400  
siqma = 5.668 \* 10\*\*-12  
F0 = 100  
alpha = 0.05  
h = 10\*\*-4  
E = 10\*\*-6

Листинг 2. Таблицы.

table1 = {  
 **'T'**: [300, 500, 800, 1100, 2000, 2400],  
 **'lambda'**: [1.36 \* 10\*\*-2, 1.63 \* 10\*\*-2, 1.81 \* 10\*\*-2, 1.92 \* 10\*\*-2, 2.5 \* 10\*\*-2, 2.74 \* 10\*\*-2],  
}  
  
table2 = {  
 **'T'**: [293, 1278, 1528, 1677, 2000, 2400],  
 **'k'**: [2 \* 10\*\*-2, 5 \* 10\*\*-2, 7.8 \* 10\*\*-2, 1 \* 10\*\*-1, 1.3 \* 10\*\*-1, 2 \* 10\*\*-1],  
}

Листинг 3. Линейная интерполяция.

**def** linear(x, y):  
 **def** p(value):  
 i = 1  
 **while** value > x[i] **and** i < len(x) - 1:  
 i += 1  
 **return** y[i - 1] + (y[i] - y[i - 1]) / (x[i] - x[i - 1]) \* (value - x[i - 1])  
 **return** p  
l\_t = linear(table1[**'T'**], table1[**'lambda'**])  
k\_t = linear(table2[**'T'**], table2[**'k'**])

Листинг 4. Вспомогательные функции.

**def** T(x):**return** T\_CONST - (T\_CONST - T0) \* x / l  
  
  
**def** lambda\_x(n):  
 **return** l\_t(t[n])  
  
  
*# x + 1/2***def** x2(n):  
 **return** (lambda\_x(n) + lambda\_x(n + 1)) / 2  
  
  
*# x - 1/2***def** \_x2(n):  
 **return** (lambda\_x(n) + lambda\_x(n - 1)) / 2  
  
  
**def** p2(n):  
 **return** (p(n) + p(n + 1)) / 2  
  
  
**def** \_p2(n):  
 **return** (p(n) + p(n - 1)) / 2  
  
  
**def** p(n):  
 **return** 4 \* np \* np \* siqma \* k\_t(t[n]) \* t[n]\*\*3  
  
  
**def** f2(n):  
 **return** (f(n) + f(n + 1)) / 2  
  
  
**def** \_f2(n):  
 **return** (f(n) + f(n - 1)) / 2  
  
  
**def** f(n):  
 **return** 4 \* np \* np \* siqma \* k\_t(t[n]) \* T0\*\*4  
  
  
**def** a(n):  
 **return** (l\_t(t[n]) + l\_t(t[n - 1])) / 2 / h  
  
  
**def** b(n):  
 **return** a(n) + c(n) + 4 \* np \* np \* siqma \* k\_t(t[n]) \* t[n]\*\*3 \* h  
  
  
**def** c(n):  
 **return** (l\_t(t[n]) + l\_t(t[n + 1])) / 2 / h  
  
  
**def** d(n):  
 **return** 4 \* np \* np \* siqma \* k\_t(t[n]) \* T0\*\*4 \* h

Листинг 5. Краевые условия.

K0 = x2(0) + h \*\* 2 / 8 \* p2(0) + h \*\* 2 / 4 \* p(0)  
M0 = h \*\* 2 / 8 \* p2(0) - x2(0)  
P0 = h \* F0 + h \*\* 2 / 4 \* (f2(0) + f(0))  
  
KN = - \_x2(steps) / h - alpha - h \* p(steps) / 4 - h \* \_p2(steps) / 8  
MN = \_x2(steps) / h - h \* \_p2(steps) / 8  
PN = - (alpha \* T0 + (\_f2(steps) + f(steps)) / 4 \* h)

Листинг 6. Условие выхода из итерации по температуре и по балансу энергии.

**if** max(ys) <= eps1 **and** max(fs) <= eps2:  
 **break**

Листинг 7. Прямой ход.

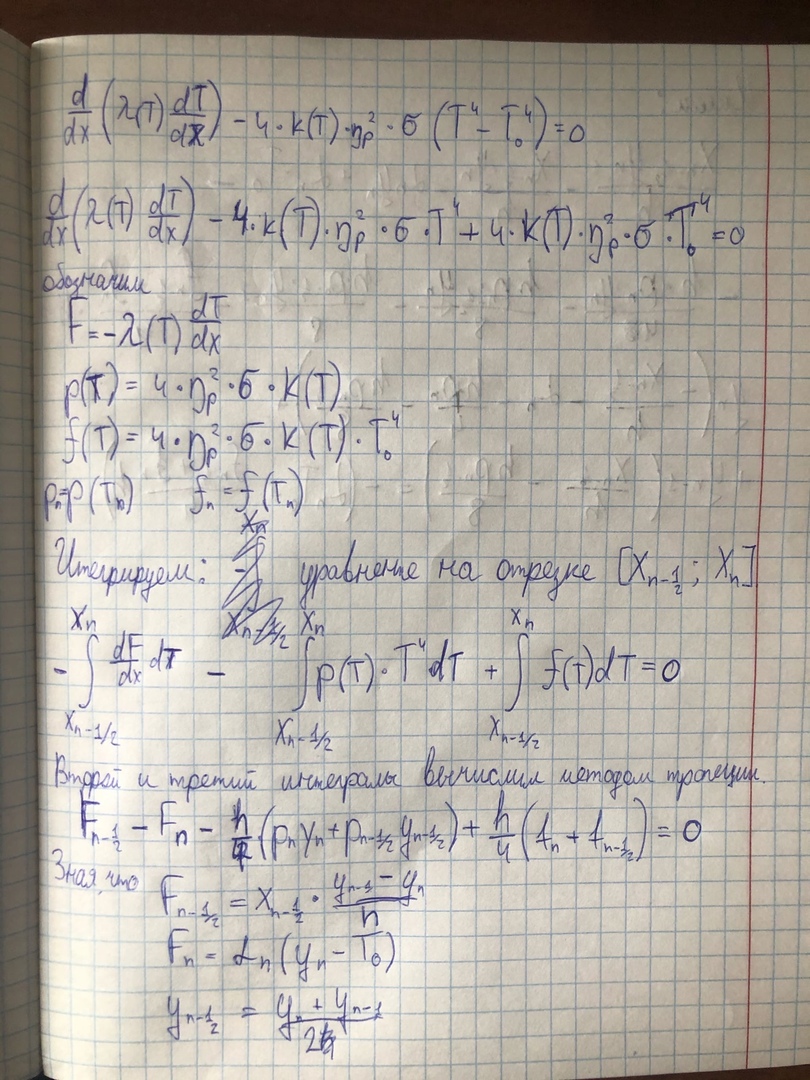
**while** x + h < l:  
 eps.append(c(n) / (b(n) - a(n) \* eps[n]))  
 eta.append((d(n) + a(n) \* eta[n]) / (b(n) - a(n) \* eps[n]))  
 n += 1  
 x += h

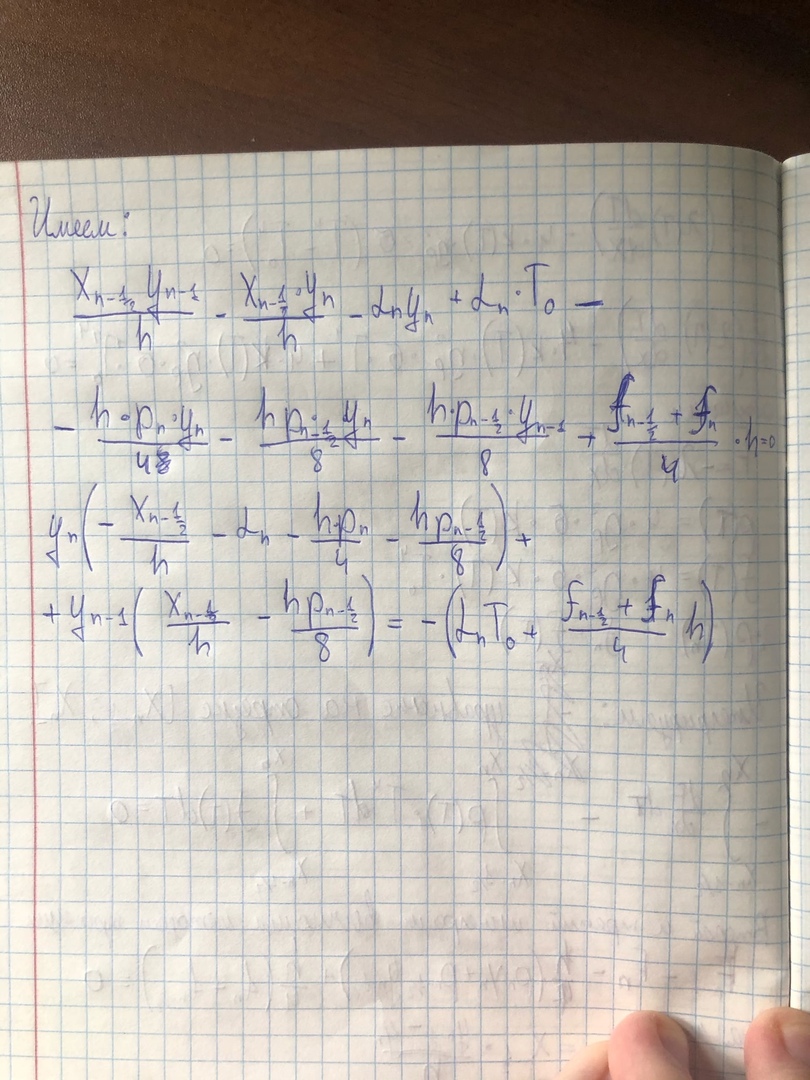
Листинг 8. Обратный ход.

t[n] = (PN - MN \* eta[n]) / (KN + MN \* eps[n])  
**for** k **in** range(n - 1, -1, -1):  
 t[k] = eps[k + 1] \* t[k + 1] + eta[k + 1]

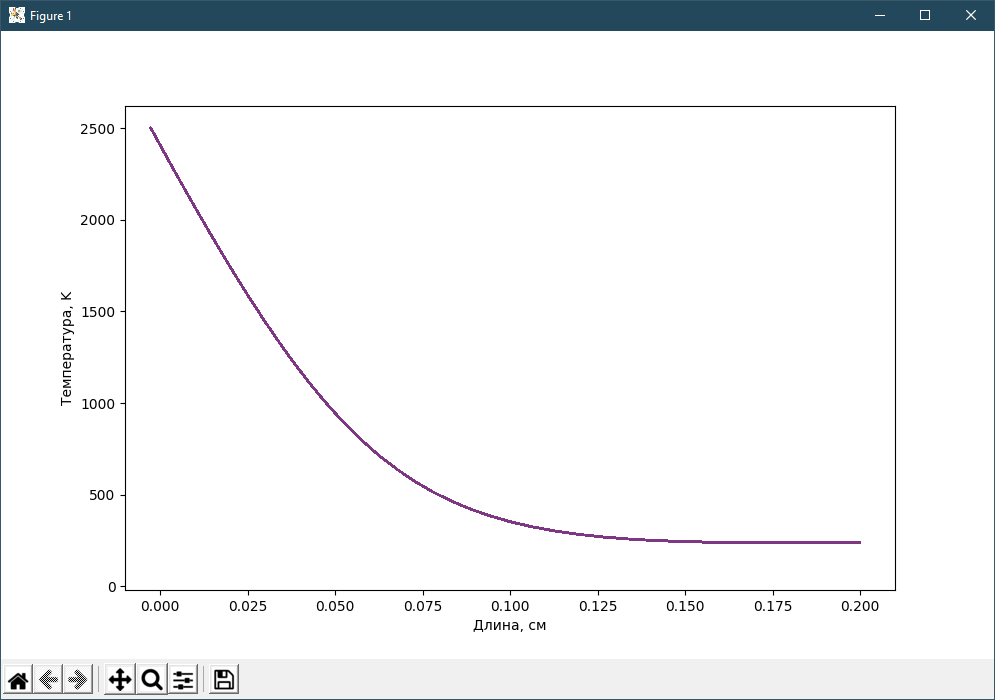
**Результат программы**

1. Разностный аналог краевого условия при x=l:

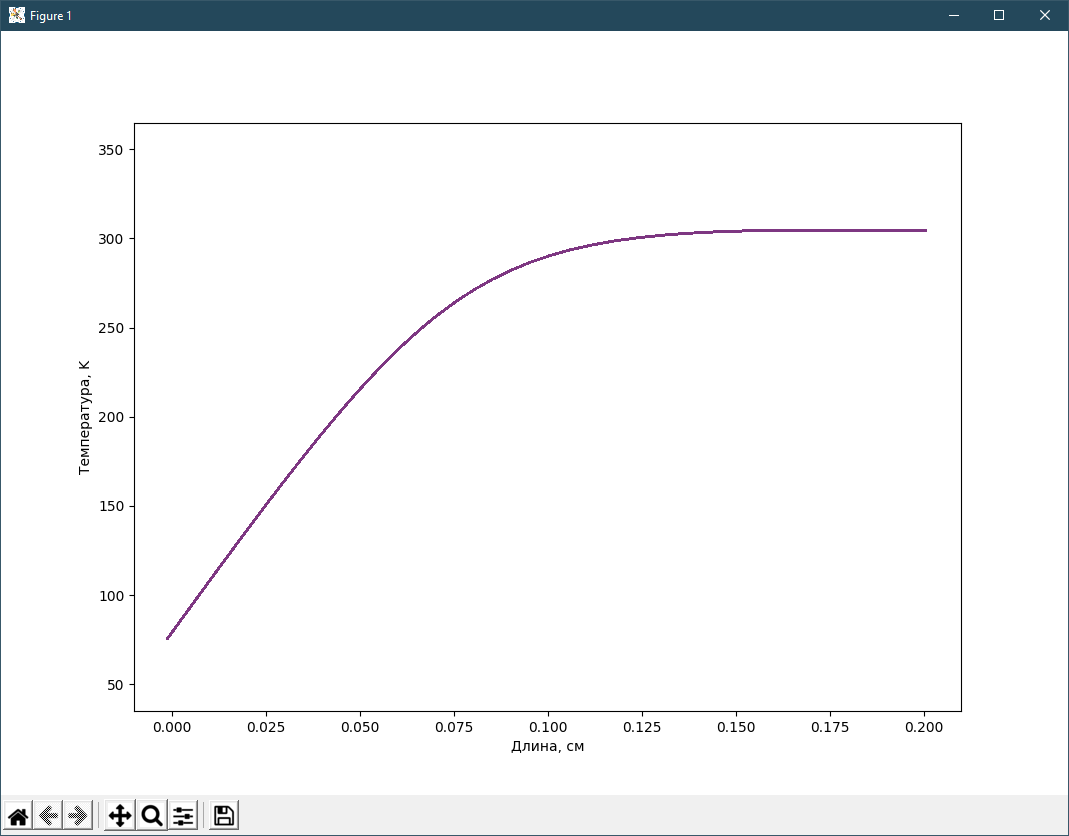




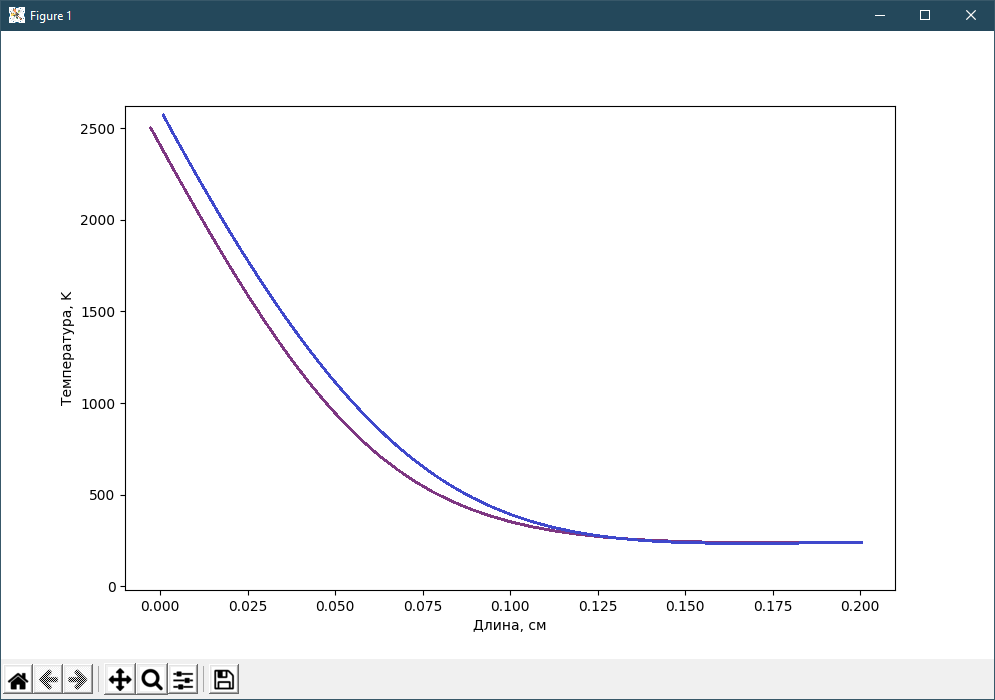
1. График зависимости температуры T(x) от координаты x при заданных условиях.



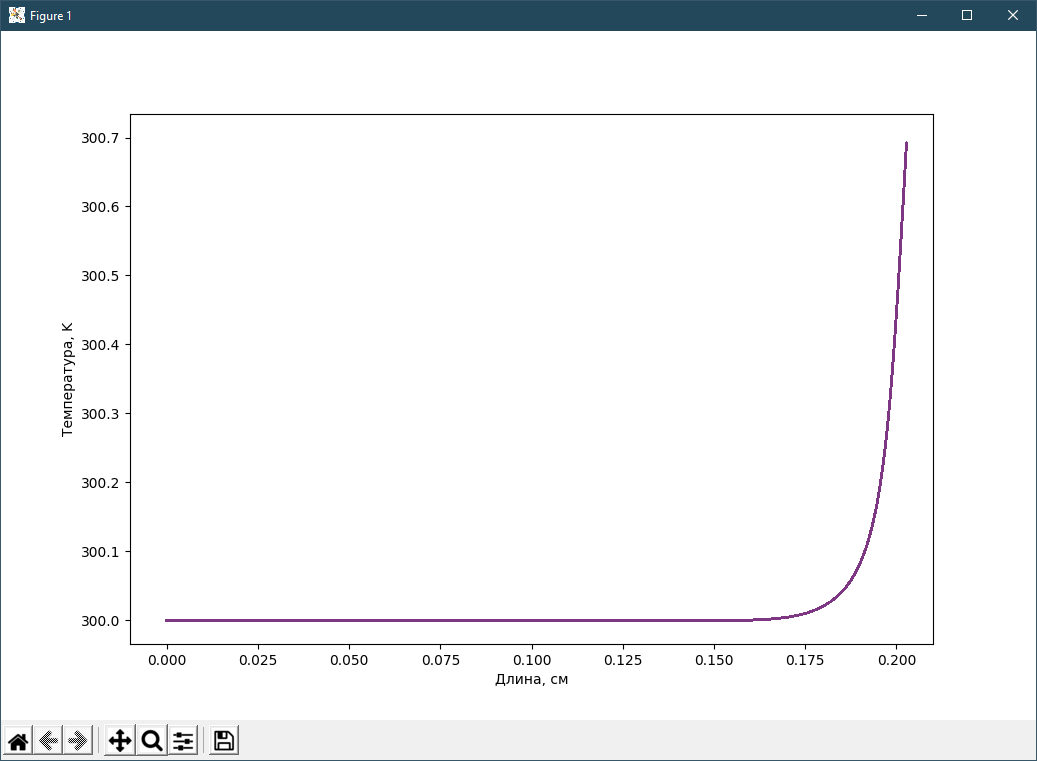
1. График зависимости T(x) при F0 = -10Вт/См2



1. График зависимости T(x) при увеличенных значениях alpha (alpha = 3 \* alpha). Сравнить с п.2.



1. График зависимости T(x) при F0 = 0.



1. Для указанного в задании исходного набора параметров привести данные по балансу энергии, т.е. значения величин:

Точность выхода eps\_1 (по температуре) = 0,069

Точность выхода eps\_2 (по балансу) = 1,12

**Вопросы при защите лабораторной работы.**

1. **Какие способы тестирования программы можно предложить?**

Опираясь на физические законы, то при F0 > 0 происходит охлаждение пластины, при F0 < 0 – нагревание пластины. Также при увеличении показателя теплосъема, уровень температур должен снижаться, а градиент увеличиваться.

1. **Получите простейший разностный аналог нелинейного краевого условия** 𝒙=𝒍, −𝒌(𝒍)𝒅𝑻𝒅𝒙=𝜶𝑵(𝑻(𝒍)−𝑻𝟎) + 𝝋(𝑻), где 𝝋(𝑻) **– заданная функция. Производную аппроксимируйте односторонней разностью.**

Аппроксимируем производную:

Подставим в исходное уравнение:

Учитывая, что :

Приведем подобные и получим уравнение относительно

1. **Опишите алгоритм применения метода прогонки, если при x = 0 краевое условие квазилинейное (как в настоящей работе), а при x = l, как в п.2.**

Для прямого хода нужно найти начальные прогоночные коэффициенты по формулам:

где коэффициенты были получены в лекции 7. Затем по формулам находим последующие прогоночные коэффициенты:

Получим , решив полученное уравнение в п.2, например, методом дихотомии. Далее по прогоночной формуле можно найти все значения неизвестных :

**4. Опишите алгоритм определения единственного значения сеточной функции yp в одной заданной точке p. Использовать встречную прогонку, т.е. комбинацию правой и левой прогонок (лекция №8). Оба краевых условия линейные.**

1. Для начала нужно вычислить начальные прогоночные коэффициенты.

Для правой прогонки они вычисляются по:

Для левой прогонки по:

2. Найдем прогоночные коэффициенты:

Для левой прогонки:

Для правой прогонки:

3. Левая и правая прогонки:

4. Выразим :