|  |  |
| --- | --- |
| 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 «Программное обеспечение ЭВМ и информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Лабораторная работа № 3**

**По курсу «Моделирование»**

|  |  |
| --- | --- |
| **Тема: «Программно-алгоритмическая реализация моделей на основе ОДУ второго порядка с краевыми условиями II и III рода»**  **Студент Горячев В. Г.**  **Группа ИУ7-65Б**  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель Градов В. М.** |  |

Москва

2021 г

**Цель работы**

Получение навыков разработки алгоритмов решения краевой задачи при

реализации моделей, построенных на ОДУ второго порядка.

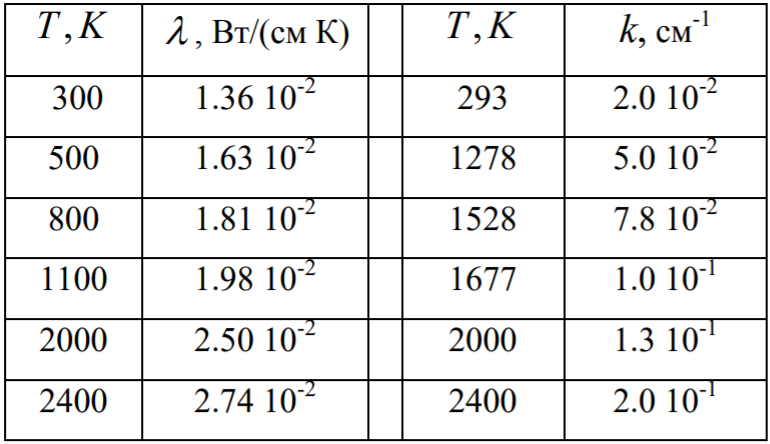
**Исходные данные**

1. Задана математическая модель. Квазилинейное уравнение для функции :

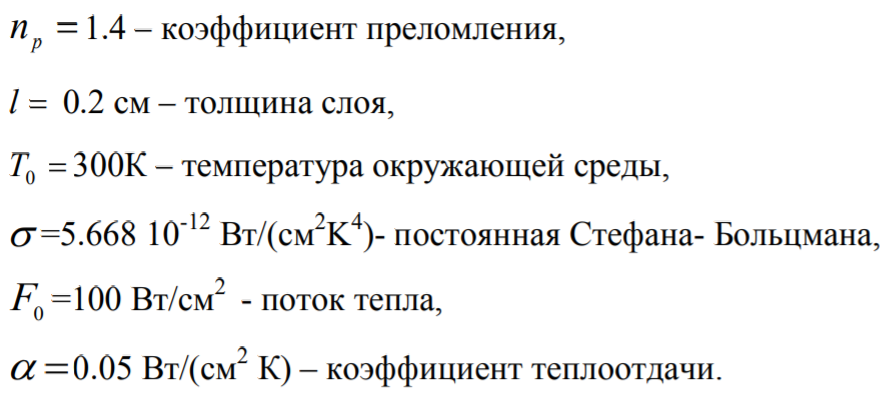
Краевые условия:

1. Функции , заданы таблицей

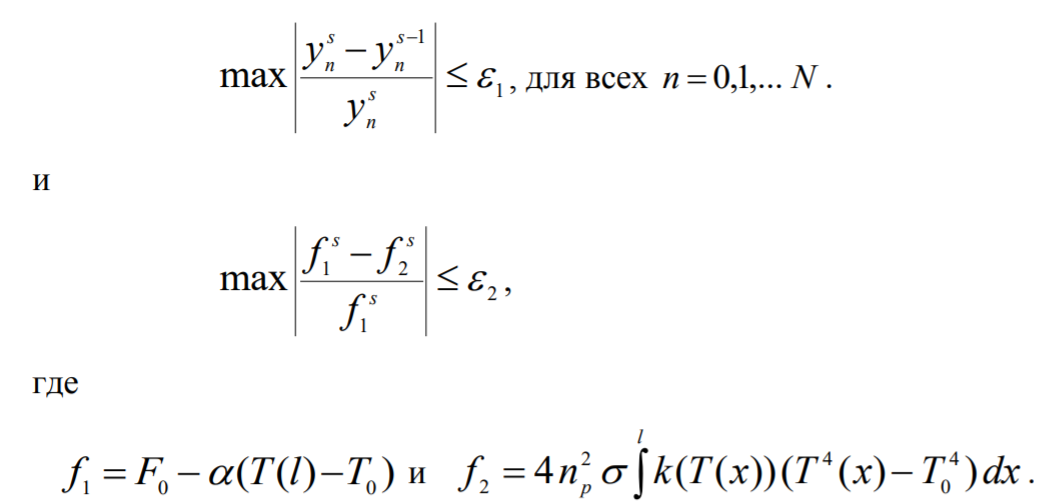
Таблица 1



1. Разностная схема с разностным краевым условием при получена в лекции №7 , и может быть использована в данной работе. Самостоятельно надо получить интегро-интерполяционным методом разностный аналог краевого условия при .
2. Значения параметров для отладки:



1. Выход из итераций организовать по температуре и балансу энергии, т.е.



**Код программы (значимые участки)**

Константы и подготовка данных:

|  |
| --- |
| eps\_ys = 1e-5; eps\_energy = 1e-3  coef\_relax = 0.1  max\_iters = 100  F0 = 100.0  T0 = 300.0  length = 0.2  h = 1e-3  N = int(length / h)  coef\_n = 1.4  alpha = 0.05  sigma = 5.668e-12  table\_lambda = np.array([  [300, 500, 800, 1100, 2000, 2400],  [1.36e-2, 1.63e-2,1.81e-2, 1.98e-2, 2.50e-2, 2.74e-2]  ], dtype=np.double)  table\_k = np.array([  [293, 1278, 1528, 1677, 2000, 2400],  [2.0e-2, 5.0e-2, 7.8e-2, 1.0e-1, 1.3e-1, 2.0e-1],  ], dtype=np.double)  table\_lambda[0] = np.log(table\_lambda[0])  table\_lambda[1] = np.log(table\_lambda[1])  table\_k[0] = np.log(table\_k[0])  table\_k[1] = np.log(table\_k[1])  main\_const = 4.0 \* coef\_n \* coef\_n \* sigma  T0\_pow4 = pow(T0, 4)  def **get\_lambda**(T):  return np.exp(np.interp(T, table\_lambda[0], table\_lambda[1]))  def **get\_k**(T):  return np.exp(np.interp(T, table\_k[0], table\_k[1])) |

Составляющие уравнение, разностную схему функции и элементы алгоритма прогонки

|  |
| --- |
| # эта функция не выделяется, однако я оставил её  # объявления явным (но не использую)  def **p**(T):  return 0  def **f**(T):  return -main\_const \* get\_k(T) \* (pow(T, 4) - T0\_pow4)  def **cappa**(a, b):  return (2 \* a \* b) / (a + b)  def **left**(Ts):  K0 = cappa(get\_lambda(Ts[0]), get\_lambda(Ts[1]))  M0 = -K0  P0 = h \* F0 + h \* (f(Ts[0]) + f(Ts[1])) / 4 \* h  return K0, M0, P0  def **right**(Ts):  KN = -cappa(get\_lambda(Ts[-1]), get\_lambda(Ts[-2]))  MN = alpha \* h  PN = T0 \* MN  MN -= KN  return KN, MN, PN  def **progonka**(A, B, C, D, K0, M0, P0, KN, MN, PN):  \*\*\* |

Итерация и отслеживание условий выхода

|  |
| --- |
| def **calc\_iteration**(Ts):  K0, M0, P0 = left(Ts)  KN, MN, PN = right(Ts)    A = np.empty(N - 1, dtype=np.double)  B = np.empty(N - 1, dtype=np.double)  C = np.empty(N - 1, dtype=np.double)  D = np.empty(N - 1, dtype=np.double)  X = np.arange(h, length, h)  for n, x in enumerate(X):  lambda\_prev = get\_lambda(Ts[n])  lambda\_cur = get\_lambda(Ts[n + 1])  lambda\_next = get\_lambda(Ts[n + 2])  A[n] = cappa(lambda\_prev, lambda\_cur) / h  C[n] = cappa(lambda\_cur, lambda\_next) / h  B[n] = A[n] + C[n]  D[n] = f(Ts[n]) \* h    return progonka(A, B, C, D, K0, M0, P0, KN, MN, PN)  def **f1**(T):  return F0 - alpha \* (T[-1] - T0)  def **f2**(T):  x = np.arange(0, length + h, h)  yi = [get\_k(Ti) \* (pow(Ti, 4) - T0\_pow4) for Ti in T]  return main\_const \* integrate.simpson(yi, x)  def **not\_energy\_cond**(ys, eps):  return np.max(abs((f1(ys) - f2(ys)) / f1(ys))) > eps  def **not\_ys\_cond**(ys, ys0, eps):  return np.max(abs((ys - ys0) / ys)) > eps |

**Вывод краевого условия**

Представить разностный аналог краевого условия при и его краткий вывод интегро-интерполяционным методом.

Исходная система:

Приведём уравнение к виду

Для этого вводятся обозначения:

Интегрируем

Учитывая, что

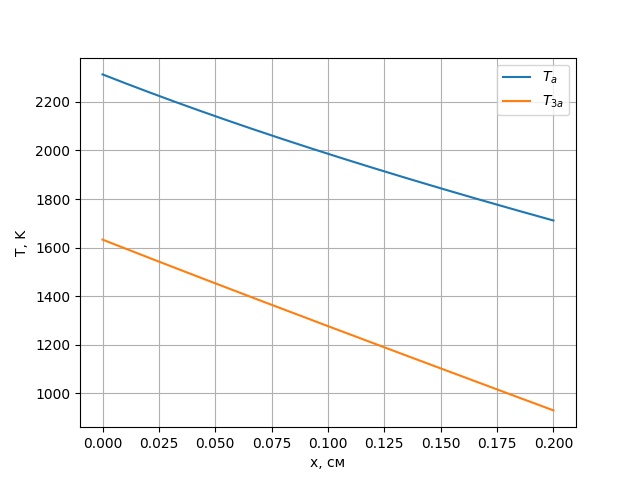
Получаем

Полученное краевое условие записывается в виде

где

**Результаты работы программы**

1. Графики зависимости при заданных параметрах и график в тех же условиях, но с увеличенным в 3 раза параметром . Совмещённо с пунктом 4 из оригинального задания.



Первый график:

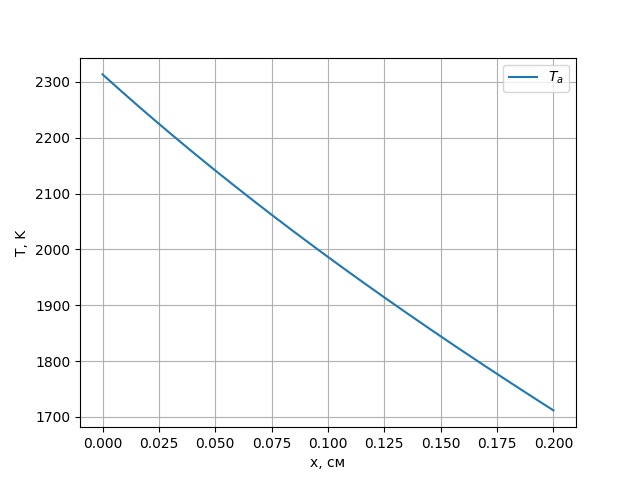
* Итерации: 22
* Максимальная разность у: 0.0002061
* Баланс энергий: 0.0002326

Второй график:

* Итерации: 64
* Максимальная разность у: 9.4e-06
* Баланс энергий: 0.0116244

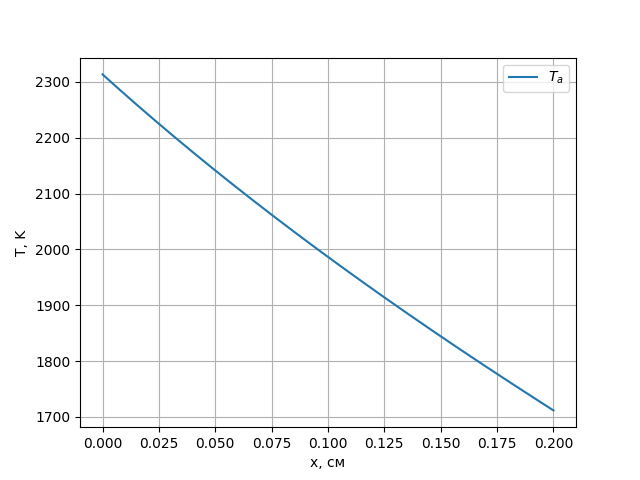
Увеличение параметра заметно снизило температуру стержня в целом и усилило скорость охлаждения. Побочным эффектом стало троекратное количество итераций, необходимое для схождения решения по обоим условиям.

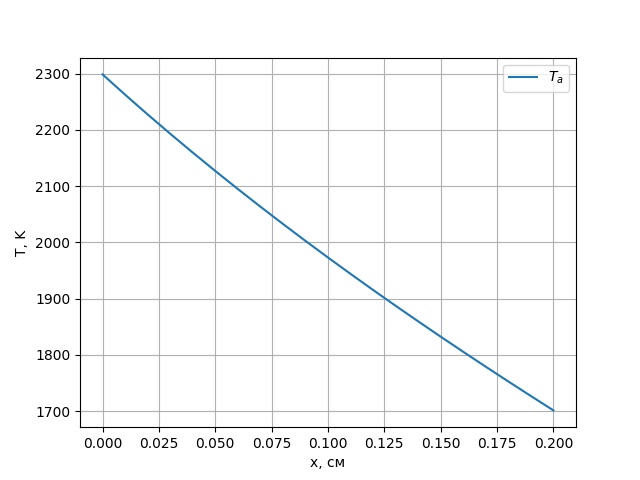
Теперь изменим начальную предполагаемую температуру стержня с 500 до 1200, а затем – вернём исходное значение и уменьшим/увеличим шаг.



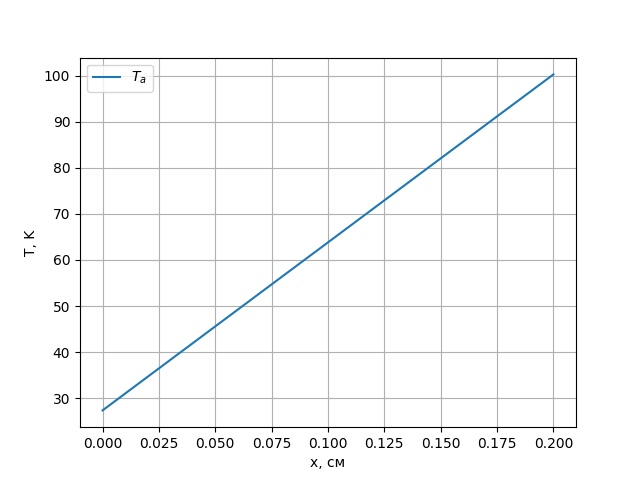
Видимых изменений нет, наблюдается небольшое изменение температуры на концах отрезка, но их не привожу, потому что эти изменения не больше погрешности.

Увеличение шага в 10 раз приводит к небольшому сдвигу графика вниз (слева) и повышению количества итераций до 34 при изначальных 22-х. Уменьшение в 5 раз – почти не приносит изменений (справа) и повышает кол-во итераций до 26.



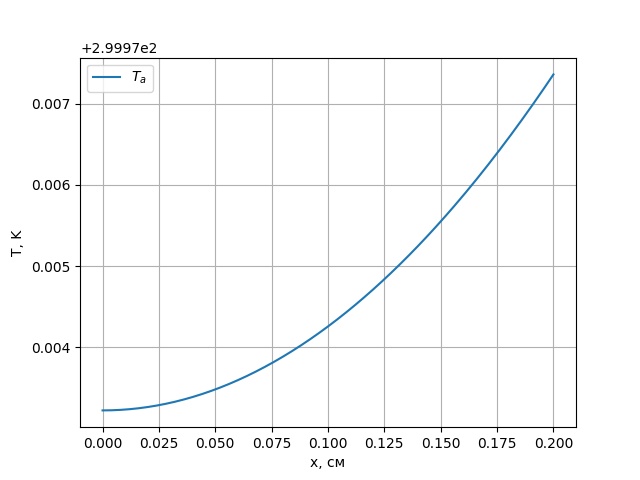


1. График зависимости при .



Итерации: 67. Теплосъём слева приводит к значительному охлаждению пластины.

1. График зависимости при .



Несмотря на причудливый вид графика, все значения лежат около температуры окружающей среды – 300К. Решение получено за 42 итерации. и является ответом на главный вопрос жизни, вселенной и всего такого.

1. Для указанного в задании исходного набора параметров привести данные по балансу энергии. Каковы использованные в работе значения точности выхода из итераций по температуре и по балансу энергии?

Использовались точности и соответственно.

Значения: f1 = 29.4198549 и f2 = 29.413012.

**Вопросы при защите лабораторной работы**

1. Какие способы тестирования программы можете предложить?

Как и всегда при работе с моделями, проверять работу программы с разными входными условиями, которые можно проверить на практике опытами и измерениями, проверить результат работы при вводе заведомо недостоверных данных, чтобы узнать, соответствует ли модель реальному объекту и с такой стороны. Например, задать начальные температуры, невозможные для фигурирующего в описании задачи кварца или сапфира.

1. Получите простейший разностный аналог нелинейного краевого условия при . Производную аппроксимируйте односторонней разностью.

Дано уравнение вида

Заменим производную разностью, после чего умножим на h и сгруппируем слагаемые: