МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Курсовая работа

Студент: Малахов А. В.
Вариант: 8
Группа: М8О-103М-20
Преподаватель: Иванов И.Э

Москва
2020

Содержание

1 Задание

Составить уравнение теплового баланса и определить стационарное распределение температуры внутри плоской стенки и тепловой поток через стенку толщиной h, разделяющую гермообъем возвращаемого космического аппарата с температурой внутри T_1 от внешнего атмосферного газового потока с температурой T_2

2 Физическая модель

Предполагается, что стенка толщиной h имеет сложную слоистую структуру, характеризуемую переменным коэффициентом теплопроводности $k(x) = A + Bx$ распределенными источниками тепла $f(x) = Cexp(-D(x - \frac{h}{E})^2)$.
X – координата поперек стенки. Коэффициент теплоотдачи в процессе теплообмена между высокотемпературным газовым потоком и стенкой α_2, между газовой средой внутри гермообъема и стенкой α_1.

$T_1 = 20$ C
$T_2 = 800$ C
$\alpha_1 = 25.0 \frac{\text{вт}}{\text{м}^2\text{град}}$
$\alpha_2 = 46.5 \frac{\text{вт}}{\text{м}^2\text{град}}$

3 Задание численного метода

A) Численные методы: конечно-разностный метод решения краевых задач для ОДУ.

4 Описание моделей

Уравнение теплопроводности внутри стенки

$$\frac{\partial}{\partial x}[k(x) * \frac{\partial T(x)}{\partial x}] = f(x)$$

Краевые условия на левом конце x=0 (условия теплообмена между стенкой и газом гермообъема)

$$\alpha_1(T_1 - T(0)) = -k(0) * \frac{\partial T(x)}{\partial x}|_{x=0}$$

Краевые условия на правом конце x=h (условия теплообмена между стенкой и газом атмосферы).

$$\alpha_2(T_2 - T(h)) = -k(h) * \frac{\partial T(x)}{\partial x}|_{x=h}$$

5 Задание параметров

$$h = 50\text{мм} = 0.05\text{м}$$
$$A = 55\frac{\text{вт}}{\text{мград}}$$
$$B = -150\frac{\text{вт}}{\text{м}^2\text{град}}$$
$$C = +10\frac{\text{вт}}{\text{м}^3}$$
$$D = 100\frac{1}{\text{м}^2}$$
$$E = 2$$

6 Переход к СЛАУ

Раскроем производную в уравнении теплопроводности внутри стенки:

$$\frac{\partial}{\partial x}[k(x) * \frac{\partial T(x)}{\partial x}] = \frac{\partial k(x)}{\partial x} * \frac{\partial T(x)}{\partial x} + k(x) * \frac{\partial^2 T(x_i)}{\partial x^2}$$

Для решения данного диф. уравнения необходимо найти значения сеточной функции во всех ее узлах. Для этого будем использовать следующие конечно-разностные аппроксимации:

$$\frac{\partial T(x_i)}{\partial x} = \frac{T(x_{i+1}) - T(x_{i-1})}{2\Delta x}$$

$$\frac{\partial^2 T(x_i)}{\partial x^2} = \frac{\frac{T(x_{i+1})-T(x_i)}{\Delta x} - \frac{T(x_i)-T(x_{i-1})}{\Delta x}}{\Delta x} = \frac{T(x_{i+1}) - 2T(x_i) + T(x_{i-1})}{\Delta x^2}$$

Составим систему уравнений:

$$\begin{cases} \frac{k(x_{i+1})-k(x_{i-1})}{2\Delta x} * \frac{T(x_{i+1})-T(x_{i-1})}{2\Delta x} + k(x_i) * \frac{T(x_{i+1})-2T(x_i)+T(x_{i-1})}{\Delta x^2} = F(x_i), i = \overline{1, h-1} \\ \alpha_1(T_1 - T(0)) = -k(0) * \frac{T(x_1)-T(x_0)}{\Delta x}|_{x_0} = 0 \\ \alpha_2(T_2 - T(x_n)) = -k(x_n) * \frac{T(x_n)-T(x_{n-1})}{\Delta x}|_{x_n} = h \end{cases}$$

Преобразуем систему к следующему виду:

$$\begin{cases} [\frac{k(x_{i-1})+4k(x_i)-k(x_{i+1})}{4\Delta x^2}] * T(x_{i-1}) + [\frac{-2k(x_i)}{\Delta x^2}] * T(x_i) + [\frac{k(x_{i-1})+4k(x_i)-k(x_{i+1})}{4\Delta x^2}] * T(x_{i+1}) = \\ = F(x_i), i = \overline{1, h-1} \\ T(x_0) + [\frac{-k(x_0)}{k(x_0)+\Delta x a_1}]T(x_1) = \frac{\Delta x a_1 T_1}{k(x_0)+\Delta x a_1}|_{x_0=0} \\ [\frac{-k(x_0)}{k(x_0)+\Delta x a_1}]T(x_{n-1}) + T(x_n) = \frac{\Delta x a_2 T_2}{k(x_n)+\Delta x a_2}|_{x_n=h} \end{cases}$$

Данную систему можно представить как трехдиагональную матрицу. А данную матрицу можно разрешить с помощью метода прогонки.

7 Метод прогонки

Метод прогонки применим к СЛАУ представленным в виде трехдиагональной матрицы, имеющим вид:

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = f_i$$

Полученную СЛАУ можно представить в виде:

$$a_i T(x_{i-1}) + b_i T(x_i) + c_i T(x_{i+1}) = F(x_i)$$

Данный метод находит решение СЛАУ за 2 подхода.
В 1 этапе происходит поиск коэффициентов P_i и Q_i

$$P_1 = \frac{-c_1}{b_1}$$

$$Q_1 = \frac{-f_1}{b_1}$$

$$P_i = \frac{-c_1}{b_1 + a_i P_{i-1}}, i = \overline{2, n-1}$$

$$Q_i = \frac{f_i - a_i Q_{i-1}}{b_1 + a_i P_{i-1}}, i = \overline{2, n}$$

Во 2 этапе расчитываются значения неизвестных

$$T(x_n) = Q_n$$

$$T(x_i) = P_i * T(x_{i+1}) + Q_i, \text{ где } i = \overline{1, n-1}$$

Достаточное условие для применимости метода прогонки выражается в следующих соотношениях:

$$|c_1| \leq 1,\ |a_n| \leq 1$$

$$|c_1| + |a_n| \leq 2,\ |b_i| \geq |c_i| + |a_i|, \text{ где } i = \overline{2, n-1}$$

8 Нахождение теплового потока

Тепловой поток можно найти по следующей формуле:

$$q = -k(x) grad(T(x))$$

В данной задаче расчет идет по оси OX, тогда тепловой поток будем искать по формуле:

$$q = -k(x) * \frac{\partial T(x)}{\partial x}$$

Заменим производную нв онечно разностную апроксимацию:

$$\frac{\partial T(x_0)}{\partial x} = \frac{T(x_0+\Delta x) - T(x_0)}{\Delta x}, \text{ где } x_0 = 0$$

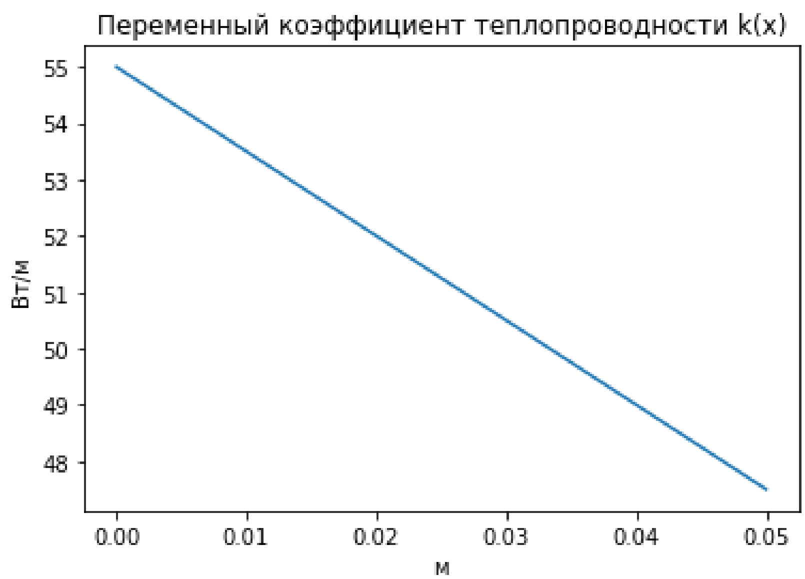
$$\frac{\partial T(x_n)}{\partial x} = \frac{T(x_n) - T(x_n+\Delta x)}{\Delta x}, \text{ где } x_n = h$$

$$\frac{\partial T(x_i)}{\partial x} = \frac{T(x_i+\Delta x) - T(x_i-\Delta x)}{\Delta 2x}, \text{ где } i = \overline{1, n-1}$$

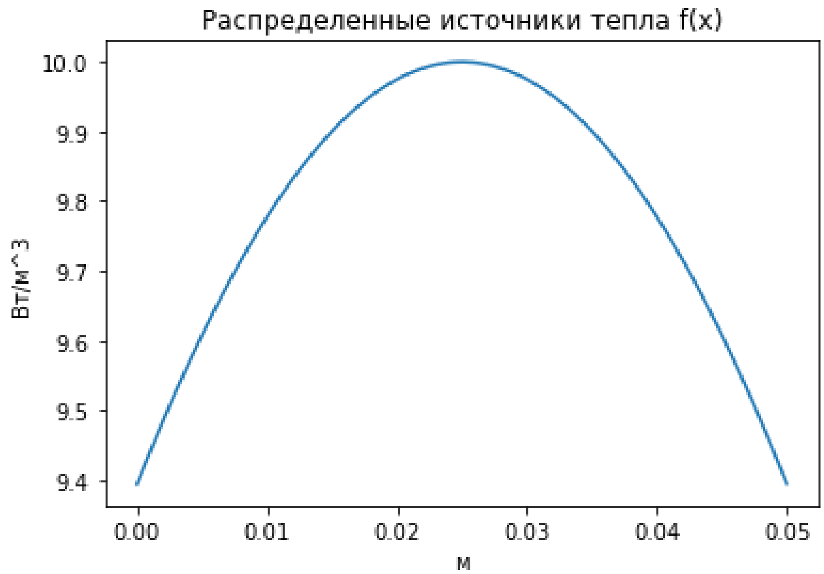
9 Результаты работы программы

Результатом работы программы являются следующие графики:

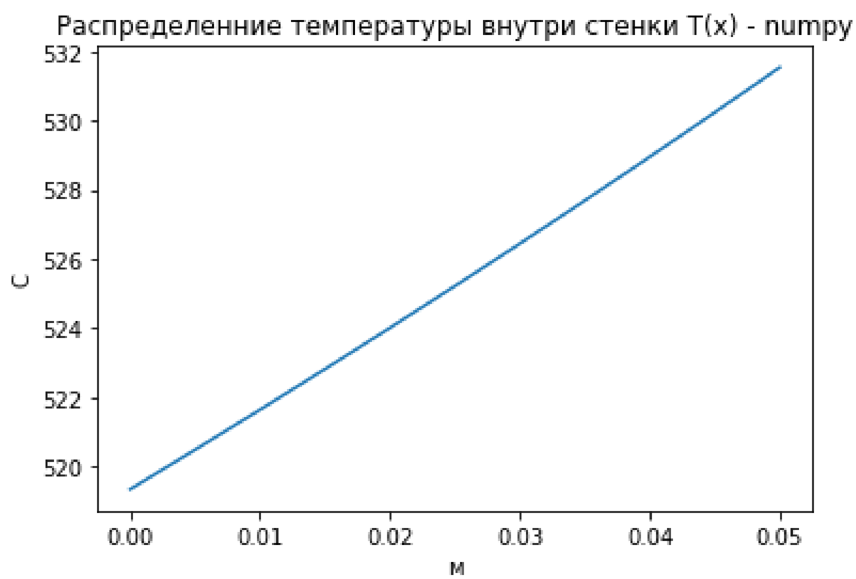
1. Переменный коэффициент теплопроводности k(x)



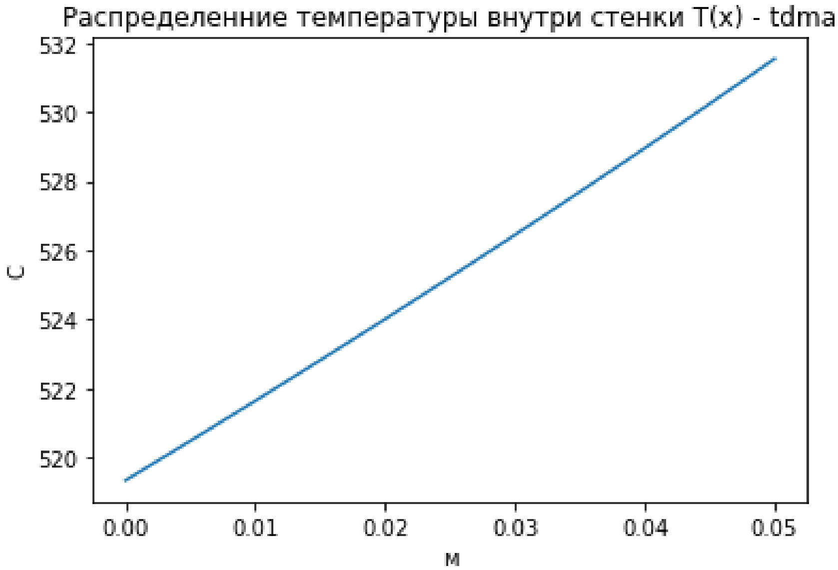
2. Распределенные источники тепла f(x)



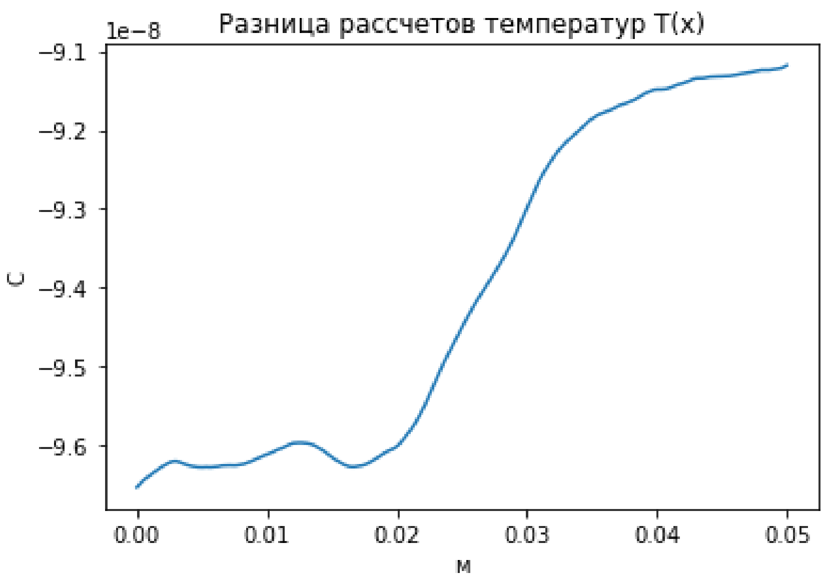
3. Распределенние температуры внутри стенки T(x) - numpy



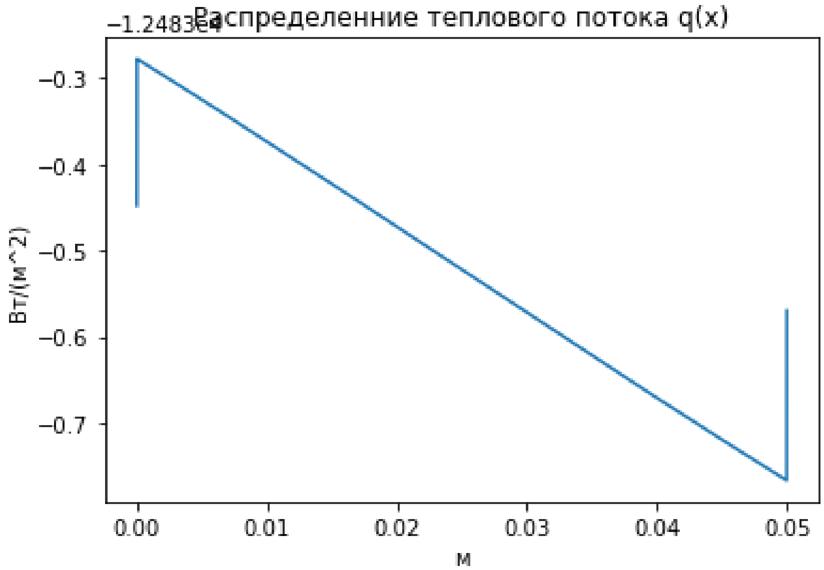
4. Распределенние температуры внутри стенки T(x) - tdma



5. Разница рассчетов температур T(x)

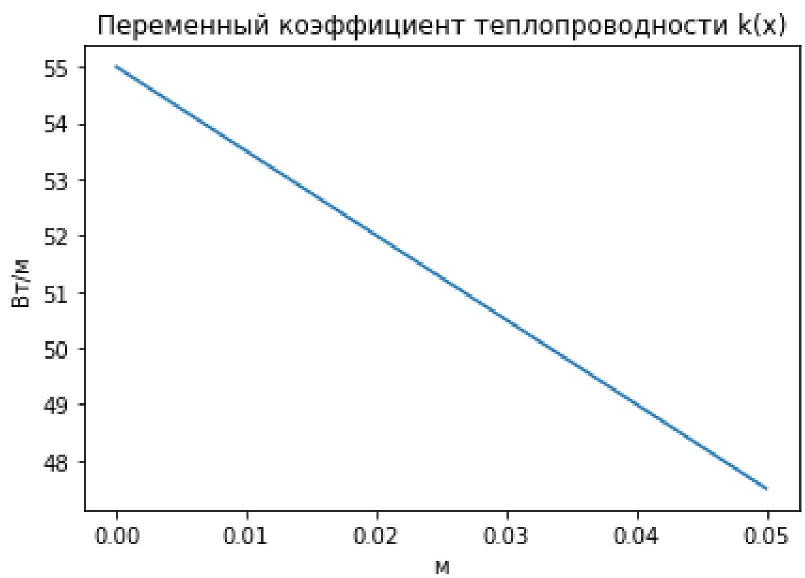


6. Распределенние теплового потока q(x)

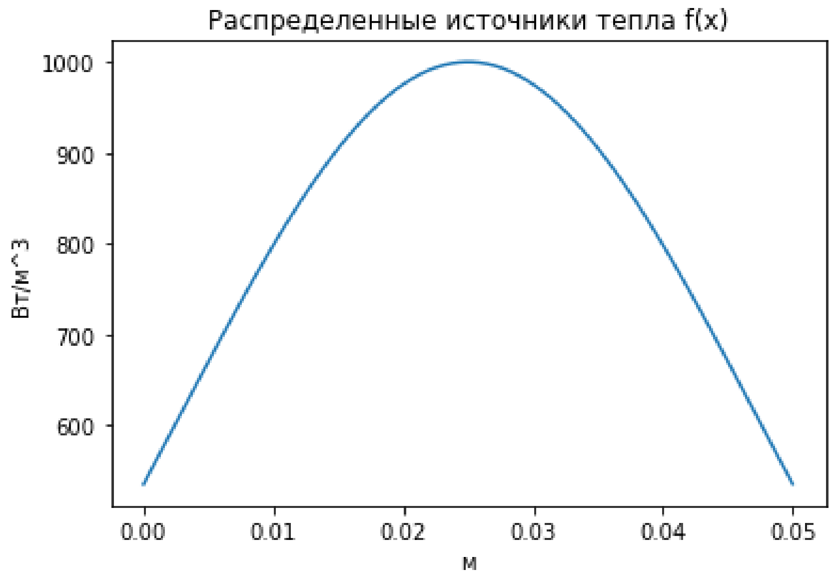


Также были произведены расчеты для C=1000 и D=1000

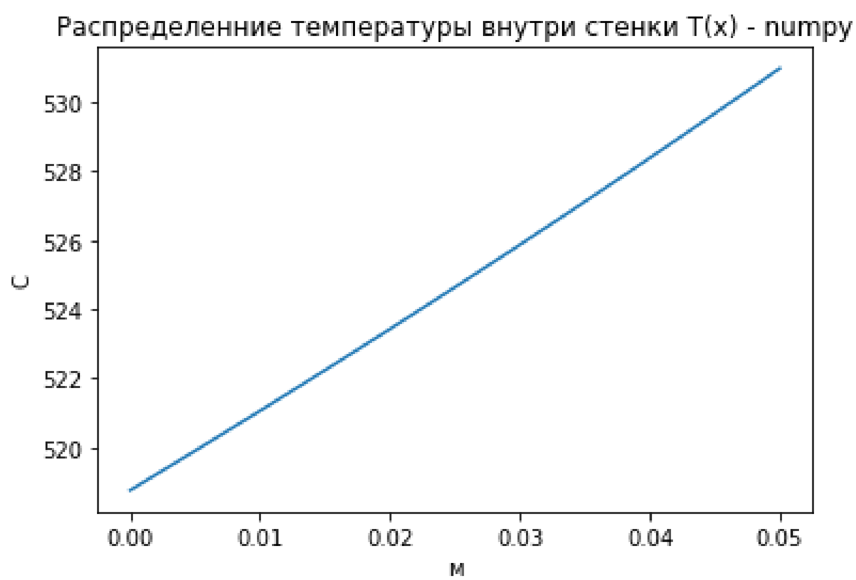
7. Переменный коэффициент теплопроводности k(x) для C=1000 и D=1000



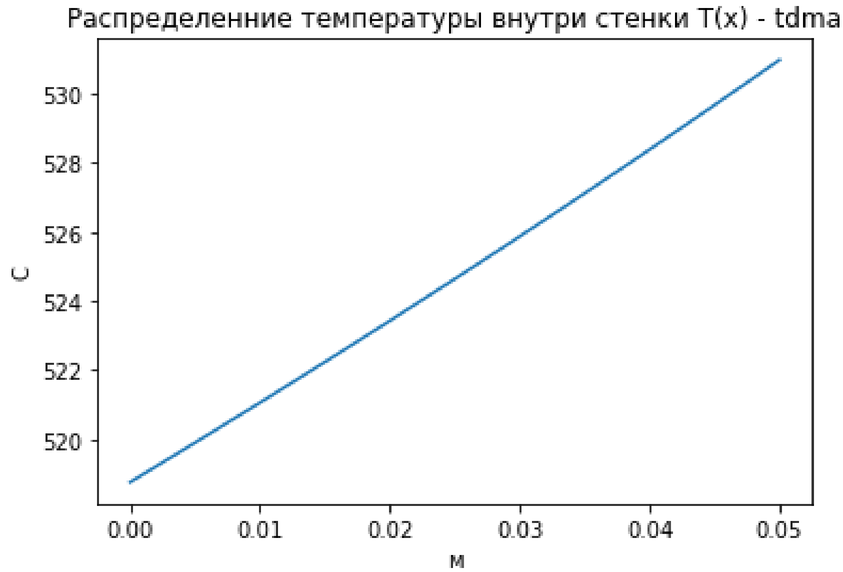
8. Распределенные источники тепла f(x) для C=1000 и D=1000



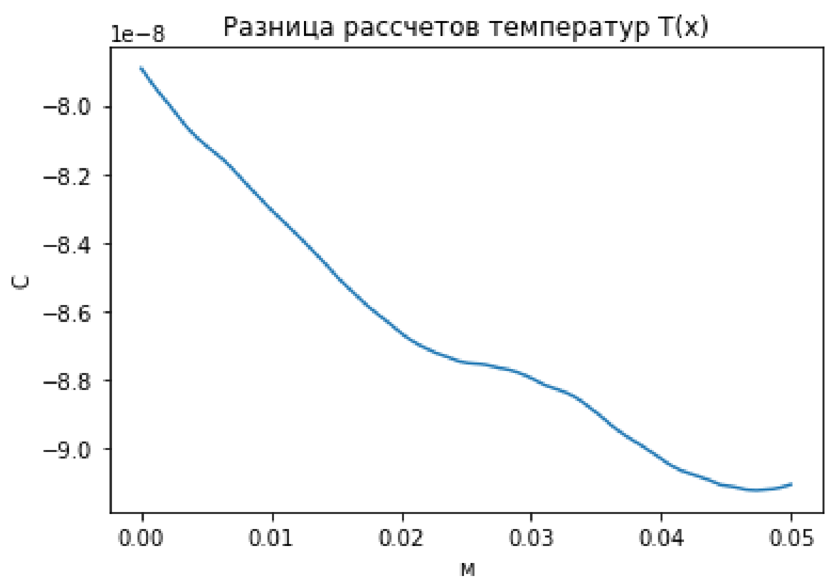
9. Распределенние температуры внутри стенки T(x) - numpy для C=1000 и D=1000



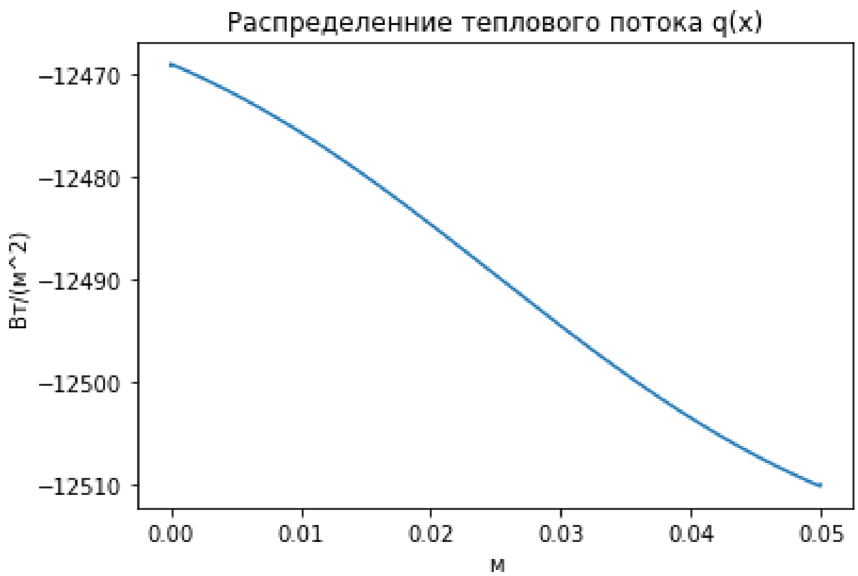
10. Распределенние температуры внутри стенки T(x) - tdma для C=1000 и D=1000



11. Разница рассчетов температур T(x) для C=1000 и D=1000



12. Распределенние теплового потока q(x) для C=1000 и D=1000



Из графиков видно, что разница в результатах между методом прогонки и методом из numpy составляет 10^{-8}. В данной задаче такая разница является не значительной, так как результаты исчисляются сотнями.
Если говорить о самих результатах, то они следующие: "Температура возрастает линейно от холодной стенки к горячей с небольшой вогнутостью".

10 Заключение

В ходе данной работы было составлено уравнение теплового баланса и определено стационарное распределение температуры внутри плоской стенки, а также тепловой поток через стенку, разделяющую гермообъем возвращаемого космического аппарата от внешнего атмосферного газового потока.
Для решения СЛАУ была написана программа на языке Python, а после произведен анализ ее работы и разбор результатов.

11 Код программы

```
#!/usr/bin/env python
# Coding: utf-8


import math
import matplotlib.pyplot as plt
import numpy as np


T1 = 20
T2 = 800
ALFA1 = 25.0
ALFA2 = 46.5

h = 0.05
A = 55
B = -150
C = 10
D = 100
E = 2

n = 5000

dx = h / n
fx = []
kx = []


def f(x):
    return C * math.exp((-1) * D * (x * dx - h / E) ** 2)


def k(x):
    return A + B * x * dx


def suffiCienCy_CheCk(matrix, n):
    if (
        (abs(matrix[0][1]) > 1)
        or (abs(matrix[n - 1][n - 1]) > 1)
        or (abs(matrix[0][1]) + abs(matrix[n - 1][n - 1]) >= 2)
    ):
        raise ExCeption(
```

```
            "This matrix does not satisfy the suffiCient Condition of the sweep 
        )

    for i in range(1, n - 1):
        if round(abs(matrix[i][i - 1]) + abs(matrix[i][i + 1]) - abs(matrix[i][i
            raise ExCeption(
                "This matrix does not satisfy the suffiCient Condition of the sw
            )


def TDMA(matrix, f, n):
    suffiCienCy_CheCk(matrix, n)
    P = [0] * (n)
    Q = [0] * (n)

    P[0] = (-1) * matrix[0][1] / matrix[0][0]
    Q[0] = f[0] / matrix[0][0]

    for i in range(1, n - 1):
        tmp = matrix[i][i] + matrix[i][i - 1] * P[i - 1]
        P[i] = (-1) * matrix[i][i + 1] / tmp
        Q[i] = (f[i] - matrix[i][i - 1] * Q[i - 1]) / tmp

    Q[n - 1] = (f[n - 1] - matrix[n - 1][n - 2] * Q[n - 2]) / (
        matrix[n - 1][n - 1] + matrix[n - 1][n - 2] * P[n - 2]
    )

    result = [0] * (n)
    for i in reversed(range(n)):
        if i == n - 1:
            result[i] = Q[i]
        else:
            result[i] = P[i] * result[i + 1] + Q[i]
    return result


for i in range(n):
    kx.append(k(i))
    fx.append(f(i))

matrix = [0] * (n)


matrix[0] = [0] * (n)
matrix[0][0] = 1
matrix[0][1] = (-1) * kx[0] / (kx[0] + dx * ALFA1)
```

```
for i in range(1, n - 1):
    matrix[i] = [0] * (n)
    matrix[i][i - 1] = (kx[i - 1] + 4 * kx[i] - kx[i + 1]) / (4 * (dx ** 2))
    matrix[i][i] = (-2) * kx[i] / (dx ** 2)
    matrix[i][i + 1] = ((-1) * kx[i - 1] + 4 * kx[i] + kx[i + 1]) / (4 * (dx **


matrix[n - 1] = [0] * (n)
matrix[n - 1][n - 1] = 1
matrix[n - 1][n - 2] = (-1) * kx[n - 1] / (kx[n - 1] + dx * ALFA2)

f = [0] * (n)
f[0] = ALFA1 * T1 / (kx[0] / dx + ALFA1)
f[n - 1] = ALFA2 * T2 / (kx[n - 1] / dx + ALFA2)
for i in range(1, n - 1):
    f[i] = fx[i]


temp_numpy = np.linalg.solve(matrix, f)
temp_tdma = TDMA(matrix, f, n)


heat_flow = [0] * (n)

heat_flow[0] = (-1) * kx[0] * (temp_tdma[1] - temp_tdma[0]) / dx

for i in range(1, n - 1):
    heat_flow[i] = (-1) * kx[i] * (temp_tdma[i + 1] - temp_tdma[i - 1]) / (2 * d

heat_flow[n - 1] = (-1) * kx[n - 1] * (temp_tdma[n - 1] - temp_tdma[n - 2]) / dx


x = []
for i in range(n):
    x.append(i * dx)
plt.plot(x, kx)
plt.title("k(x)")
plt.xlabel("m")
plt.ylabel("Bt/m")
plt.show()

plt.plot(x, fx)
plt.title("f(x)")
plt.xlabel("m")
```

```
plt.ylabel("Bt/m^3")
plt.show()

plt.plot(x, temp_numpy)
plt.title("T(x) - numpy")
plt.xlabel("m")
plt.ylabel("C")
plt.show()

plt.plot(x, temp_tdma)
plt.title("T(x) - tdma")
plt.xlabel("m")
plt.ylabel("C")
plt.show()

plt.plot(x, list(map(lambda x, y: x - y, temp_numpy, temp_tdma)))
plt.title("T(x)")
plt.xlabel("m")
plt.ylabel("C")
plt.show()

plt.plot(x, heat_flow)
plt.title("q(x)")
plt.xlabel("m")
plt.ylabel("Bt/(m^2)")
plt.show()
```