iМИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

Физико-механический институт

Высшая школа прикладной математики и вычислительной физики

Отчет по курсовому проекту  
по курсу:

ЧИСЛЕННОЕ РЕШЕНИЕ КРАЕВЫХ ЗАДАЧ

на тему:

Исследование нестационарного поля температур в плоской неограниченной пластине с использованием явного метода конечных разностей

Работу выполнил:

студент группы 5030301/20001

Медведев Кирилл Максимович

Преподаватель:

к.т.н., доц. Плетнев А. А.

Санкт-Петербург

2024

**Оглавление**

[1 Постановка задачи 3](#_Toc164886861)

[1.1 Физическая постановка задачи 3](#_Toc164886862)

[1.2 Математическая постановка задачи 3](#_Toc164886863)

[3 Метод решения №2. Явный метод конечных разностей 5](#_Toc164886864)

[4 Тестовый расчёт 9](#_Toc164886865)

[5 Результаты решения задачи 9](#_Toc164886866)

[5.1 Зависимости температуры от координаты 9](#_Toc164886867)

[5.2 Зависимости температуры от времени 11](#_Toc164886868)

[5.3 Решения на последовательности сгущающихся сеток 12](#_Toc164886869)

[5.4 Решения при нарушении условия устойчивости 12](#_Toc164886870)

[6 Выводы 13](#_Toc164886871)

[Приложения 14](#_Toc164886872)

# 1 Постановка задачи

## 1.1 Физическая постановка задачи

Дана стальная неограниченная пластина со следующими характеристиками [1].

Находящаяся в следующих условиях:

Для следующих коэффициентов теплоотдачи требуется провести исследование нестационарного температурного поля, используя метод Фурье

## 1.2 Математическая постановка задачи

|  |  |
| --- | --- |
| Обозначения | Рисунок |
| α – коэффициент конвективной теплоотдачи,  δ – толщина пластины,  τ – время,  𝑇 – температура пластины,  𝑇𝑒 – температура окружающей среды,  𝑇𝑤 – температура на границе пластины,  𝑞 – плотность теплового потока  𝑥 – координата, м |  |

Уравнение теплопроводности в размерных переменных имеет вид:

Начальные и граничные условия в размерных переменных

Приведём его к безразмерному виду, которое и будем в дальнейшем решать, с помощью следующей замены переменных:

Безразмерное уравнение теплопроводности имеет вид:

Начальные и граничные условия в безразмерных переменных:

# 3 Метод решения №2. Явный метод конечных разностей

Основные идеи метода заключаются в следующем:

**1. Дискретизация.**

Область непрерывного изменения аргумента (координата X) разбивается на конечное число интервалов (или ячеек), в пределах каждого интервала размещается точка (узел), в которой задается значение искомой функции (температуры) для этого интервала. Совокупность узлов с упорядоченной нумерацией называется конечно-разностной сеткой.

Для безразмерных координат:

Аналогичная дискретизация выполняется для оси времени. В результате искомая непрерывная функция становится сеточной.

Где i – номер узла по координате; n – номер узла по времени

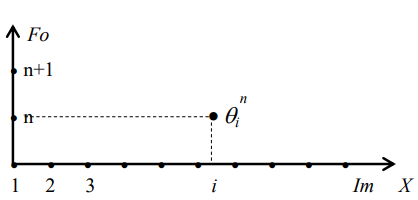


Рис 1. Расположение и нумерация узлов конечно разностной сетки.

Будем использовать равномерную сетку. Её параметры следующие:

**2. Аппроксимация**

Производные в безразмерном уравнении теплопроводности заменяются следующими конечно-разностными аналогами.

Уравнение теплопроводности в безразмерных переменных (4) будет иметь теперь вид.

Из него получаем выражение для :

Заметим, что выражение не должно быть отрицательным, иначе увеличение приведет к уменьшению, что нефизично. Из этого следует следующее неравенство, являющееся условием устойчивости явной схемы.

Выражения для левой и правой границы получается из соответствующих ГУ (3) и (4)

**3. Решение алгебраических уравнений**

В результате аппроксимации исходная задача сведена к системе алгебраических уравнений, записанных относительно значений температуры в узлах сетки. Совокупность значений сеточной функции, соответствующих одному значению индекса “n”, называется временным слоем.

Слой “n” называется текущим – он известен.

Слой “n+1” называется будущим – его необходимо посчитать.

В построенном алгоритме значения температуры в узлах на «будущем» временном слое вычисляются по формулам (5), (6), (7) независимо и **явно** через температуры в узлах на «текущем» временном слое. Поэтому данная разностная схема называется явной. Ее достоинством является простота реализации, а недостатком – ограничение на шаг по времени.

Для вычисления Fomax используется кусочно-линейная аппроксимация.

* Bi <1,25 Fomax=
* 1,25 ≤ Bi ≤20 Fomax=2,76
* Bi>20 Fomax=1,1

Изображение выглядит как текст, диаграмма, снимок экрана, Шрифт

Автоматически созданное описание

Рис 2. Блок схема вычислительного алгоритма

# 4 Тестовый расчёт

Тестовый расчёт проведём для третьего варианта; в точке x=d/4, tau=344,727; полученный результат сравним с результатом метода Фурье.   
Параметры сетки: ,

|  |  |
| --- | --- |
| полученная методом Фурье | полученная методом конечных разностей |
| 238,515 | 238,546 |

Метод конечных разностей написан корректно. Отличия имеются лишь во втором знаке.

# 5 Результаты решения задачи

Для различных вариантов коэффициента теплоотдачи c помощью написанной программы (приложение 1) получим результаты расчёта, которые представим в виде графиков, используя пакет SciDavis

## 5.1 Зависимости температуры от координаты

Для разных коэффициентов теплоотдачи (чисел Био) построим три зависимости температуры от координаты в три различных момента безразмерного времени: 0.1\*FoMax, 0.5\*FoMax, 0.9\*FoMax.

Параметры сетки:

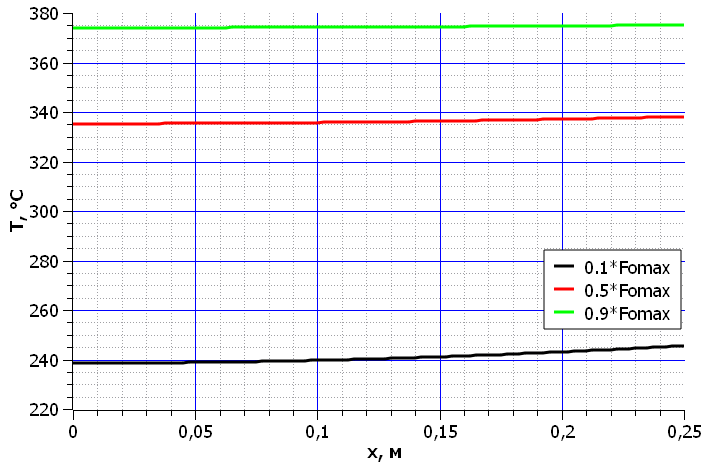


Рис 3. Зависимость температуры от координаты при Bi = 0,091; FoMax = 25,66

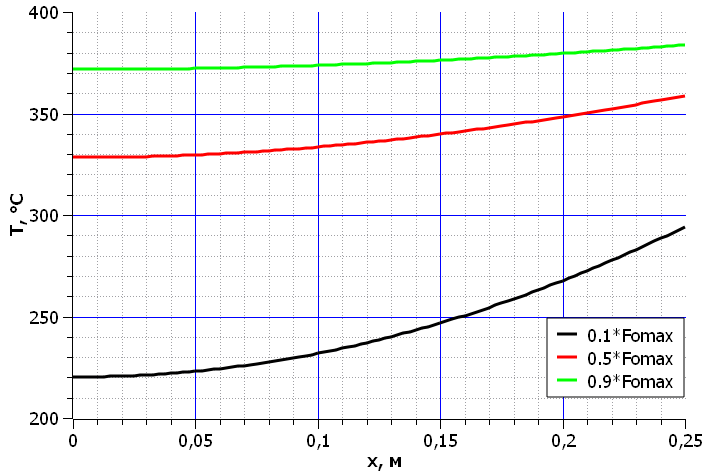


Рис 4. Зависимость температуры от координаты при Bi = 1,36; FoMax = 2,507

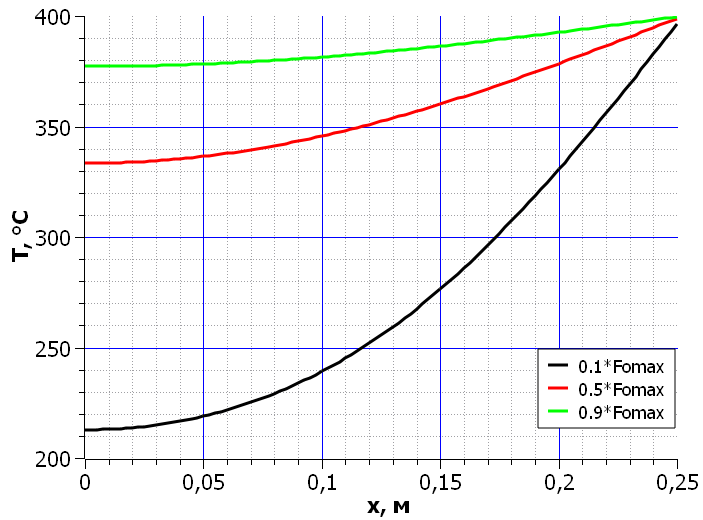


Рис 5. Зависимость температуры от координаты при Bi = 90,9; FoMax = 1,10

## 5.2 Зависимости температуры от времени

Для двух крайних значений числа Био построим зависимости температуры от времени в трёх сечениях пластины: x= 0, x= d/4, x = d/2.

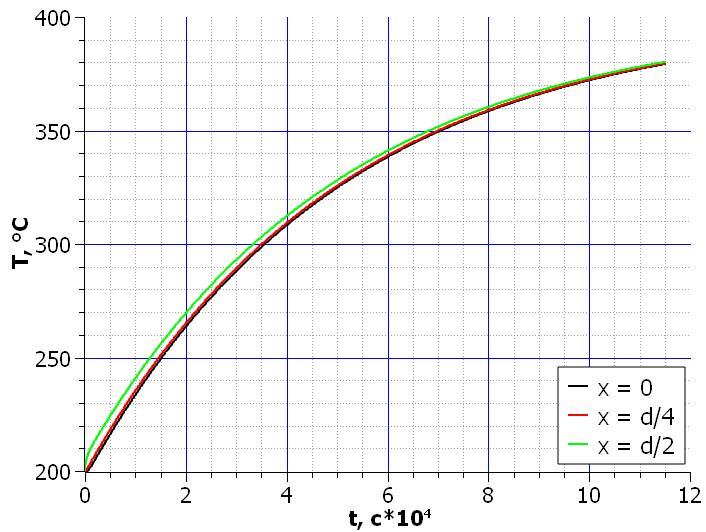


Рис 6. Зависимость температуры от времени в трех сечениях пластины при Bi = 0,091

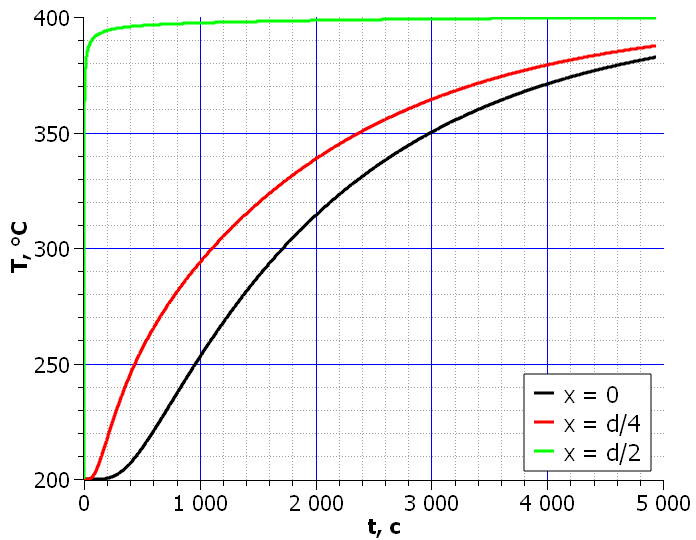


Рис 7. Зависимость температуры от времени в трех сечениях пластины при Bi = 90,9

## 5.3 Решения на последовательности сгущающихся сеток

Рассмотрим, как меняется решение для первого варианта(Bi = 0,091) в точке x = d/4, Fo = 1.0 на последовательности сгущающихся сеток.

Шаг по времени следующий:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Im |  | (Метод Фурье) | (МКР) | Ошибка | Относительная ошибка |
| 11 | 1.0 | 216.246 | 217.994 | 1.748 | 0.81% |
| 21 | 217.096 | 0.85 | 0.39% |
| 41 | 216.665 | 0.419 | 0.19% |
| 81 | 216.454 | 0.208 | 0.09% |

## 5.4 Решения при нарушении условия устойчивости

Зададим шаг по времени на 8% больше допустимого, нарушив условие устойчивости. Следующее решение получается для третьего варианта (Bi = 90,9) при x = d/2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

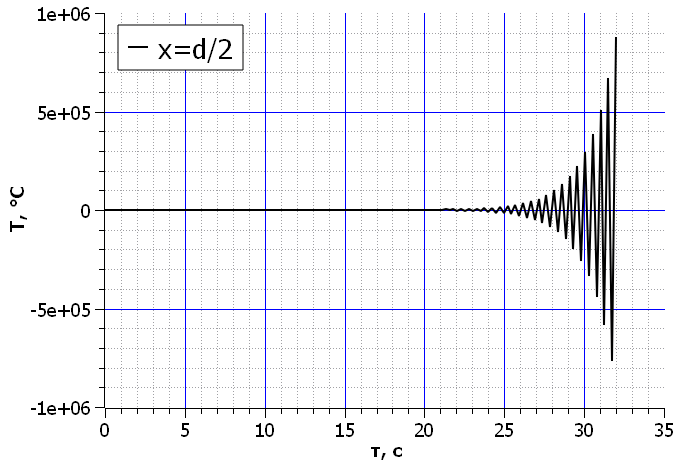


Рис 8 Расходимость решения при нарушении условия устойчивости

# 6 Выводы

1. При измельчении сетки в два раза ошибка решения уменьшается примерно в два раза, что говорит о первом асимптотическом порядке точности метода:
2. Точность в 3 значащие цифры достигается для 81 узла по координате. Относительная погрешность составляет менее 1%.
3. При нарушении условий сходимости решения получаются неправильными и неограниченно, немонотонно растут по модулю.
4. Программная реализация метода конечных разностей проще метода Фурье. Количество строк в процедуре решения составляет 11, против 35 у метода Фурье. Стоит также отметить, что метод Фурье может оказаться принципиально не реализуемым, в отличии от метода конечных разностей.

Ссылки на литературу:

1. <http://thermalinfo.ru/Sets/cache/supercache/thermalinfo.ru/svojstva-materialov/metally-i-splavy/teploemkost-stali/index.html>

# Приложения

Приложение 1

Код программы

1 Program Main

program Main

use methods

implicit none

integer :: i,j, N, m(2), k, Xof(3)

real(8) :: l, d, T0, Te, eps, p, c

real(8) :: a(3), Bi(3), FoMax(3), dFo, X, Fof(3,3), Fo, dX

real(8) :: time1, time2, treads

type(point), dimension(:), allocatable :: p\_c, p\_f, data1, data2, data3

call CPU\_TIME(time1)

!Создаём объект символизирующий решение в какой-либо точке

l = 55d0

d = 0.5d0

a(1) = 20d0

a(2) = 300d0

a(3) = 20000d0

T0 = 200d0 + 273.15d0

Te = 400d0 + 273.15d0

N = 10 !Число разбиений Im-1

dX = 1d0/real(N, 8)

dFo = (dX\*\*2)\*0.1d0

p = 7900d0

c = 500d0

!Вычисление чисел Bio

Bi(1) = atoBi(a(1), d, l, 1)

Bi(2) = atoBi(a(2), d, l, 1)

Bi(3) = atoBi(a(3), d, l, 1)

FoMax(1) = getFoMax(Bi(1))

FoMax(2) = getFoMax(Bi(2))

FoMax(3) = getFoMax(Bi(3))

Fof(:,1) = 0.1d0\*FoMax(:)

Fof(:,2) = 0.5d0\*FoMax(:)

Fof(:,3) = 0.9d0\*FoMax(:)

Xof(1) = dint(xtoX(0d0, d, 1)/dX) + 1

Xof(2) = dint(xtoX(d/4d0, d, 1)/dX) + 1

Xof(3) = dint(xtoX(d/2d0, d, 1)/dX) + 1

10 format(f18.9, 3(a,f18.9))

20 format(f18.9,3(a,f18.9))

open(1, file = "T1(x).csv")

open(2, file = "T2(x).csv")

open(3, file = "T3(x).csv")

open(4, file="Ta1(tau).csv")

open(5, file="Ta3(tau).csv")

write(1,\*) "X", ";", "T(a1, Fo11)", ";", "T(a1, Fo12)", ";", "T(a1, Fo13)"

write(2,\*) "X", ";", "T(a2, Fo21)", ";", "T(a2, Fo22)", ";", "T(a2, Fo23)"

write(3,\*) "X", ";", "T(a3, Fo31)", ";", "T(a3, Fo32)", ";", "T(a3, Fo33)"

write(4,\*) "tau", ";", "Ta1(X1)", ";", "Ta1(X2)", ";", "Ta1(X3)"

write(5,\*) "tau", ";", "Ta3(X1)", ";", "Ta3(X2)", ";", "Ta3(X3)"

allocate(p\_c(N+1), p\_f(N+1), data1(N+1), data2(N+1), data3(N+1))

call set\_ics(dX, 0d0, 1d0, p\_c, N)

do while(p\_c(1) % Fo <= FoMax(1) + dFo)

call FDM\_emp(dFo, dX, p\_c, p\_f, Bi(1), N) !Расчитали будущий на основе текущего

call add\_data(data1, data2, data3, p\_c, N, Fof(1,:), dFo, T0, Te, d)

write(4, 20) tautoFo(p\_c(1) % Fo, l, d, c, p, -1), ';', TtoTheta(p\_c(Xof(1)) % Theta, T0, Te, -1), ';',TtoTheta(p\_c(Xof(2)) % Theta, T0, Te, -1), ';',TtoTheta(p\_c(Xof(3)) % Theta, T0, Te, -1)

p\_c = p\_f

end do

do i = 1, N+1

write(1, 10) data1(i) % X,';', data1(i) % Theta,';', data2(i) % Theta,';', data3(i) % Theta

end do

call set\_ics(dX, 0d0, 1d0, p\_c, N)

do while(p\_c(1) % Fo <= FoMax(2) + dFo)

CALL FDM\_emp(dFo, dX, p\_c, p\_f, Bi(2), N)

call add\_data(data1, data2, data3, p\_c, N, Fof(2,:), dFo, T0, Te, d)

p\_c = p\_f

end do

do i = 1, N+1

write(2, 10) data1(i) % X,';', data1(i) % Theta,';', data2(i) % Theta,';', data3(i) % Theta

end do

call set\_ics(dX, 0d0, 1d0, p\_c, N)

do while(p\_c(1) % Fo <= FoMax(3) + dFo)

CALL FDM\_emp(dFo, dX, p\_c, p\_f, Bi(3), N)

call add\_data(data1, data2, data3, p\_c, N, Fof(3,:), dFo, T0, Te, d)

write(5, 20) tautoFo(p\_c(1) % Fo, l, d, c, p, -1), ';', TtoTheta(p\_c(Xof(1)) % Theta, T0, Te, -1), ';',TtoTheta(p\_c(Xof(2)) % Theta, T0, Te, -1), ';',TtoTheta(p\_c(Xof(3)) % Theta, T0, Te, -1)

p\_c = p\_f

end do

do i = 1, N+1

write(3, 10) data1(i) % X,';', data1(i) % Theta,';', data2(i) % Theta,';', data3(i) % Theta

end do

close(1)

close(2)

close(3)

deallocate(p\_c, p\_f, data1, data2, data3)

call CPU\_TIME(time2)

write(\*,\*) dabs(time1 - time2)

pause

end program Main2 Module Methods

module methods

implicit none

type :: point

real(8) :: X, Fo, Theta

end type

contains

subroutine set\_ics(dX, Fo, Theta, p\_c, N)

type(point), dimension(:) :: p\_c

real(8) :: dX, Fo, Theta

integer :: i, N

do i = 1, N+1

p\_c(i) % X = dX\*(i-1d0)

p\_c(i) % Fo = 0d0

p\_c(i) % Theta = 1d0

end do

end subroutine

subroutine add\_data(d1, d2, d3, p\_c, N, Fof, dFo, T0, Te, d)

type(point), dimension(:) :: p\_c, d1, d2, d3

real(8), dimension(:) :: Fof

real(8) dFo, T0, Te, d

integer :: i, N

if (dabs(p\_c(1) % Fo - Fof(1)) <= dFo) then

d1 % Fo = Fof(3)

do i = 1, N+1

d1(i) % Theta = TtoTheta(p\_c(i) % Theta, T0, Te, -1)

d1(i) % X = xtoX(p\_c(i) % X, d, -1)

end do

else if (dabs(p\_c(1) % Fo - Fof(2)) <= dFo) then

d2 % Fo = Fof(2)

do i = 1, N+1

d2(i) % Theta = TtoTheta(p\_c(i) % Theta, T0, Te, -1)

d2(i) % X = xtoX(p\_c(i) % X, d, -1)

end do

else if (dabs(p\_c(1) % Fo - Fof(3)) <= dFo) then

d3 % Fo = Fof(3)

do i = 1, N+1

d3(i) % Theta = TtoTheta(p\_c(i) % Theta, T0, Te, -1)

d3(i) % X = xtoX(p\_c(i) % X, d, -1)

end do

end if

end subroutine

subroutine FDM\_emp(dFo, dX, p\_c, p\_f, Bi, N)

type(point), dimension(:) :: p\_c, p\_f

real(8) :: dFo, dX, Bi

integer :: i, N

p\_f % Fo = p\_c % Fo + dFo

p\_f % X = p\_c % X

do i = 2, N

p\_f(i) % Theta = (dFo/(dX\*\*2))\*(p\_c(i+1) % Theta + p\_c(i-1) % Theta) + (p\_c(i) % Theta)\*(1d0 - 2d0\*dFo/(dX\*\*2))

end do

p\_f(1) % Theta = p\_f(2) % Theta

p\_f(N+1) % Theta = (p\_f(N) % Theta)/(1d0 + dX\*Bi)

end subroutine

function getFoMax(Bi) result(FoMax)

real(8), intent(in) :: Bi

real(8) :: FoMax

if(Bi < 1.25d0) then

FoMax = 3.11d0\*((Bi)\*\*(-0.88d0))

else if (Bi > 20d0) then

FoMax = 1.10d0

else

FoMax = 2.76d0\*((Bi)\*\*(-0.31d0))

end if

end function

function xtoX(x, d, m) result(X1)

real(8), intent(in) :: x, d

integer, intent(in) :: m

real(8) :: X1

if(m == 1) then

X1 = 2d0\*x/d

else

X1 = d\*x/2d0

end if

end function

function TtoTheta(T, T0, Te, m) result(Theta)

real(8), intent(in) :: T, T0, Te

integer, intent(in) :: m

real(8) :: Theta

if(m == 1) then

Theta = (T - Te)/(T0 - Te)

else

Theta = T\*(T0 - Te) + Te - 273.15d0

end if

end function

function tautoFo(tau, l, d, c, p, m) result(Fo1)

real(8), intent(in) :: tau, l, d, c, p

integer, intent(in) :: m

real(8) :: Fo1

if(m == 1) then

Fo1 = tau\*l\*4d0/(c\*p\*(d\*\*2))

else

Fo1 = c\*p\*(d\*\*2)\*tau/(4d0\*l)

end if

end function

function atoBi(a,d,l,m) result(Bi)

real(8), intent(in) :: a, d, l

integer, intent(in) :: m

real(8) :: Bi

if(m == 1) then

Bi = a\*d/(2d0\*l)

else

Bi = 2d0\*l\*a/d

end if

end function

end module