Министерство образования и науки Российской Федерации

Новосибирский государственный технический университет

Кафедра прикладной математики

Уравнения математической физики

Курсовой проект

Факультет ПМИ

Группа ПМ-01

Студент Жигалов П.С.

Преподаватель Персова М.Г.

Вариант 82, 3

Новосибирск

2013

Содержание

[Содержание 2](#_Toc358611803)

[1. Постановка задачи 3](#_Toc358611804)

[1.1. Формулировка задания 3](#_Toc358611805)

[1.2. Постановка задачи 3](#_Toc358611806)

[2. Теоретическая часть 3](#_Toc358611807)

[2.1. Дискретизация по времени 3](#_Toc358611808)

[2.2. Вариационная постановка 3](#_Toc358611809)

[2.3. Конечноэлементная дискретизация 4](#_Toc358611810)

[2.4. Переход к локальным матрицам 4](#_Toc358611811)

[2.5. Учет краевых условий 5](#_Toc358611812)

[2.6. Метод решения СЛАУ 6](#_Toc358611813)

[3. Описание разработанной программы 6](#_Toc358611814)

[3.1. Структуры данных, используемые для задания расчетной области и конечноэлементной сетки 6](#_Toc358611815)

[3.2. Структура основных модулей программы 6](#_Toc358611816)

[3.2.1 fem\_module 7](#_Toc358611817)

[3.2.2 cgm\_module 7](#_Toc358611818)

[3.2.3 param\_module 7](#_Toc358611819)

[4. Описание тестирования программы 8](#_Toc358611820)

[4.1. Тест 1 8](#_Toc358611821)

[4.2. Тест 2 8](#_Toc358611822)

[4.3. Тест 3 8](#_Toc358611823)

[4.4. Тест 4 9](#_Toc358611824)

[5. Исследования 10](#_Toc358611825)

[5.1. Определение порядка аппроксимации по времени 10](#_Toc358611826)

[6. Выводы 10](#_Toc358611827)

[7. Тексты основных модулей программы 10](#_Toc358611828)

[7.1. fem\_module 10](#_Toc358611829)

[7.2. param\_module 19](#_Toc358611830)

1. Постановка задачи

1.1. Формулировка задания

МКЭ для трёхмерной начально-краевой задачи для уравнения параболического типа в декартовой системе координат. Базисные функции трилинейные на параллелепипедах. Схема Кранка-Николсона для аппроксимации по времени. Краевые условия всех типов. Коэффициент диффузии  – кусочно-постоянная функция. Матрицу СЛАУ генерировать в разреженном строчном формате. Для решения СЛАУ использовать МСГ или ЛОС с неполной факторизацией.

1.2. Постановка задачи

Параболическая начально-краевая задача для функции  определяется дифференциальным уравнением

(1) ,

заданным в некоторой области  с границей , начальным условием

(2) 

и краевыми условиями

(3) , (4) , (5) ,

в которых  - значение искомой функции  на границе , а  - значение на  производной функции  по направлению внешней нормали к поверхности , .

Коэффициент  - кусочно-постоянная функция, коэффициент  - некоторая известная функция.

Вид конечных элементов – параллелепипеды.

Вид базисных функций – трилинейные.

2. Теоретическая часть

2.1. Дискретизация по времени

Положим, что ось времени  разбита на так называемые временные слои со значениями , , а значения искомой функции  и параметров ,  и  уравнения (1) будем обозначать соответственно через ,  и , которые уже не зависят от времени , но остаются функциями пространственных координат.

Схема Кранка-Николсона для уравнения (1), при условии, что  и  не зависят от , а  - зависит, будет выглядеть следующим образом:

(6)  , , 

2.2. Вариационная постановка

Пусть  – некоторая пробная функция из пространства , тогда:

(7) .

Применим к (7) формулу Грина, преобразуем и перегруппируем:

(8) 

2.3. Конечноэлементная дискретизация

Заменим пространство  на конечномерное пространство , которое определим как линейное пространство, натянутое на базисные функции . Заменим в (8) функцию  аппроксимирующей ее функцией , а функцию  - функцией .

Поскольку любая функция  может быть представлена в виде линейной комбинации , получим:

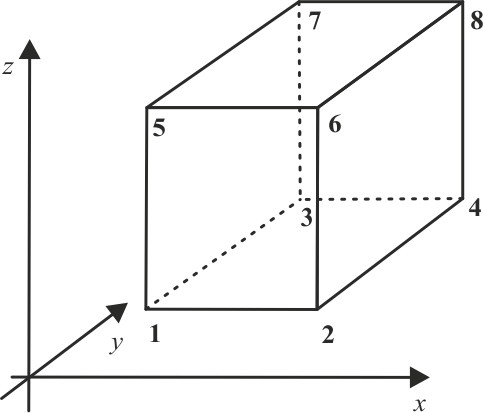
(9) 

Решение  может быть представлено в виде: (10) , причем  компонент вектора весов  могут быть фиксированы и определены из условия . Подставляя (10) в (9) получим СЛАУ для вектора весов :

(11) 

2.4. Переход к локальным матрицам

Введем локальную нумерацию узлов в соответствии с рисунком и определим базисные функции на конечном элементе. На отрезках , ,  зададим по две одномерные линейные функции:

(12.1) ,

(12.2) ,

(12.3) .

Тогда трилинейные базисные функции представляются в виде произведения функций (12.1)-(12.3). Так как функции (12.1)-(12.3) заданы на конечном элементе, во всех прочих конечных элементах они будут равны нулю. Таким образом, трилинейные базисные функции на каждом конечном элементе могут быть представлены в виде:

 ,  , 

 ,  , 

 ,  (13)

Выражения для вычисления компонент локальных матриц в этом случае принимают вид:

(14)  – матрица жесткости

(15)  – матрица массы

Компоненты локальных матриц будут иметь размерность 8x8 и примут вид:

После вычислений итоговые матрицы примут вид:

(16)  (17) 





 также раскладываем по базису: , тогда вектор правой части имеет вид (18) .

Глобальная матрица собирается путем добавления к ней локальных матриц в соответствующие места элементов. Про-филь матрицы формируется следующим образом: , если  и  узлы находятся в одном локальном элементе.

2.5. Учет краевых условий

Краевые условия первого рода будем учитывать следующим образом: в глобальной матрице и глобальном векторе находим соответствующую глобальному номеру краевого узла строку, и ставим в диагональный элемент глобальной матрицы единицу, а в вектор правой части значение краевого условия, заданное в исходном узле. Строку матрицы необходимо обнулить, при этом учесть симметричную структуру матрицы. Краевые условия первого рода должны быть учтены в последнюю очередь.

Рассмотрим краевые условия третьего рода (5). Необходимо вычислить интегралы  и .

Положим параметр  константой на каждом конечном элементе, а функцию  разложим по базису . В данном случае, базис получится вырожденным в билинейный, так как краевые условия заданы на гранях, т.е. одна из переменных фиксирована.

Вычислив интеграл , получим локальную матрицу: , где:

Умножив  на  получим добавку в вектор правой части, а сама матрица  будет добавкой в глобальную матрицу (или правую часть, необходимо учесть (11)).





Рассмотрим краевые условия второго рода (4). Необходимо вычислить интеграл . Производится это аналогично третьим, с той разницей, что добавка есть лишь в правую часть.

2.6. Метод решения СЛАУ

Для решения СЛАУ использован метод сопряженных градиентов с предобуславливанием неполным разложением Холесского (-разложение). Данный метод является достаточно эффективным для данной задачи, так как в СЛАУ используется симметричная матрица.

3. Описание разработанной программы

3.1. Структуры данных, используемые для задания расчетной области и конечноэлементной сетки

Для задания расчетной области и конечноэлементной сетки в программе используются следующие структуры (упрощенно):

1) для описания конечных элементов используется тип данных *local\_area*, содержащий в себе номера узлов и значения параметра  в данной области

type :: local\_area ! конечные элементы

integer :: mas\_(8) ! номера узлов

double precision :: lambda\_ ! lambda в области

end type

2) для хранения координат узлов используется тип данных *cross*, содержащий в себе координаты ,  и.

type :: cross ! координаты узлов

double precision :: mas\_(3) ! x,y,z соотв.

end type

3) тип данных *fem* используется для хранения массивов типа *local\_area* и *cross.*

type :: fem

type(local\_area),private,allocatable :: matr(:) ! конечные элементы

type(cross),private,allocatable :: set(:) ! координаты узлов

end type

Для внешнего хранения или ввода расчетной области и конечноэлементной сетки используются файлы следующего назначения:

|  |  |  |
| --- | --- | --- |
| ***Имя файла*** | ***Что хранится*** | ***Способ задания данных*** |
| cross.txt | Узлы сетки | Первая строка – число узлов, последующие – их координаты. |
| local.txt | Конечные элементы | Первая строка – число конечных элементов, далее номера узлов, определяющих конечные элементы и значения . |
| bound1.txt | Первые краевые | Номер узла, на котором задано условие |
| bound2.txt | Вторые краевые | Четыре номера узла, задающих грань |
| bound3.txt | Третьи краевые | Четыре номера узла, задающих грань, значение |

3.2. Структура основных модулей программы

Основные модули программы:

* *fem\_module* – основной модуль программы;
* *cgm\_module* – модуль решения СЛАУ методом МСГ;
* *list\_module* – модуль, реализующий класс «список»;
* *param\_module* – модуль, в котором задаются все необходимые параметры;

3.2.1 fem\_module

Основной модуль программы, содержит описание и реализацию основного класса программы – *fem*, основные и вспомогательные типы данных для представления расчетной области и конечноэлементной сетки, все необходимые данные, такие как локальные матрицы.

Структура всех типов данных и классов:

type :: local\_area конечные элементы

integer :: mas\_(8) номера узлов

double precision :: lambda\_ lambda в области

end type

type :: cross координаты узлов

double precision :: mas\_(3) x, y, z соотв.

end type

type :: limits границы, для вторых-третьих краевых

double precision :: min\_(3), max\_(3)

end type

type :: local\_matrix локльные матрицы

double precision :: m\_1(8,8) матрица массы

double precision :: g\_x(8,8) матрица жесткости (x)

double precision :: g\_y(8,8) матрица жесткости (y)

double precision :: g\_z(8,8) матрица жесткости (z)

double precision :: c\_1(4,4) матрица c\_1

end type

type :: fem

type(slae) :: gl\_matr СЛАУ

type(local\_area), private, allocatable :: matr(:) конечные элементы

type(cross), private, allocatable :: set(:) координаты узлов

type(local\_matrix), private :: lc\_matr локальные матрицы

double precision :: t\_curr = 0d0, t\_old = 0d0 значение времени

double precision, allocatable :: q\_curr(:), q\_old(:) вектора весов

type(limits), private :: lim границы, для вторых-третьих краевых

contains

procedure :: calc\_ функция запуска вычислений

procedure :: read\_ функция считывания исходных данных (кроме краевых)

procedure, private :: do\_make\_matrix формирование матрицы

procedure, private :: do\_make\_bound учет краевых условий

procedure, private :: add добавлениие эл -та x к (i, j) эл -ту матрицы

procedure :: dealloc\_ освобождение памяти

procedure :: grid\_gen генератор сеток

end type

3.2.2 cgm\_module

Модуль программы, обеспечивающий решение СЛАУ в разряженном строчно-столбцовом формате методом сопряженных градиентов с предобуславливанием неполным разложением Холесского (-разложение).

Структура всех типов данных и классов:

type :: slae СЛАУ

integer :: n

integer,allocatable :: ig(:),jg(:)

double precision,allocatable :: di(:),gg(:),f(:),x(:)

contains

procedure :: calc\_cgm решение СЛАУ методом МСГ

procedure :: alloc\_all выделение памяти под СЛАУ

procedure :: dealloc\_all освобождение памяти

procedure,private :: do\_holessky\_decompose выполнение разложения Холесского

procedure,private :: do\_solve\_slae решение СЛАУ для матрицы с разложением Холесского

procedure,private :: mult\_mtrx\_vctr умножение матрицы на вектор

procedure,private :: mult\_scalar вычисление скалярного произведения

procedure,private :: residual вычисление невязки

procedure,private :: norm\_2 вычисление квадрата нормы вектора

end type

3.2.3 param\_module

Модуль программы, обеспечивающий хранение и использования всех параметров для работы программы.

Структура всех типов данных и классов:

double precision, parameter :: t\_start = 0d0, t\_delta = 0.0125d0, t\_end = 1d0 <

double precision, parameter :: x\_start = 0d0, x\_delta = 1d0, x\_end = 10d0 < параметры

double precision, parameter :: y\_start = 0d0, y\_delta = 1d0, y\_end = 10d0 < генератора

double precision, parameter :: z\_start = 0d0, z\_delta = 1d0, z\_end = 10d0 < сеток

double precision, parameter :: lambda\_grid = 1d0 <

function u\_(x, y, z, t) функция 

function f\_(x, y, z, t) функция правой части уравнения

function sigma\_(x, y, z, t) функция параметра 

function grad\_u\_(x, y, z, t) функция градиента 

4. Описание тестирования программы

4.1. Тест 1

*Цель теста:* проверка вклада 

*Параметры теста:* , , , 

*Параметры области:* , , , , ,  (27 узлов)

*Параметры времени:* ,  *Краевые условия:* везде первые

*Результаты:*

***Временной слой Относительная погрешность***

Time = 0.1000000E+00 Diff = 0.145E-16

Time = 0.2000000E+00 Diff = 0.143E-16

Time = 0.3000000E+00 Diff = 0.282E-16

Time = 0.4000000E+00 Diff = 0.278E-16

Time = 0.5000000E+00 Diff = 0.137E-16

Time = 0.6000000E+00 Diff = 0.135E-16

Time = 0.7000000E+00 Diff = 0.266E-16

Time = 0.8000000E+00 Diff = 0.394E-16

Time = 0.9000000E+00 Diff = 0.259E-16

Time = 0.1000000E+01 Diff = 0.000E+00

4.2. Тест 2

*Цель теста:* проверка вклада 

*Параметры теста:* , , , 

*Параметры области:* , , , , ,  (27 узлов)

*Параметры времени:* ,  *Краевые условия:* везде первые

*Результаты:*

***Временной слой Относительная погрешность***

Time = 0.1000000E+00 Diff = 0.582E-16

Time = 0.2000000E+00 Diff = 0.162E-15

Time = 0.3000000E+00 Diff = 0.977E-16

Time = 0.4000000E+00 Diff = 0.174E-15

Time = 0.5000000E+00 Diff = 0.514E-16

Time = 0.6000000E+00 Diff = 0.158E-15

Time = 0.7000000E+00 Diff = 0.217E-15

Time = 0.8000000E+00 Diff = 0.695E-16

Time = 0.9000000E+00 Diff = 0.322E-15

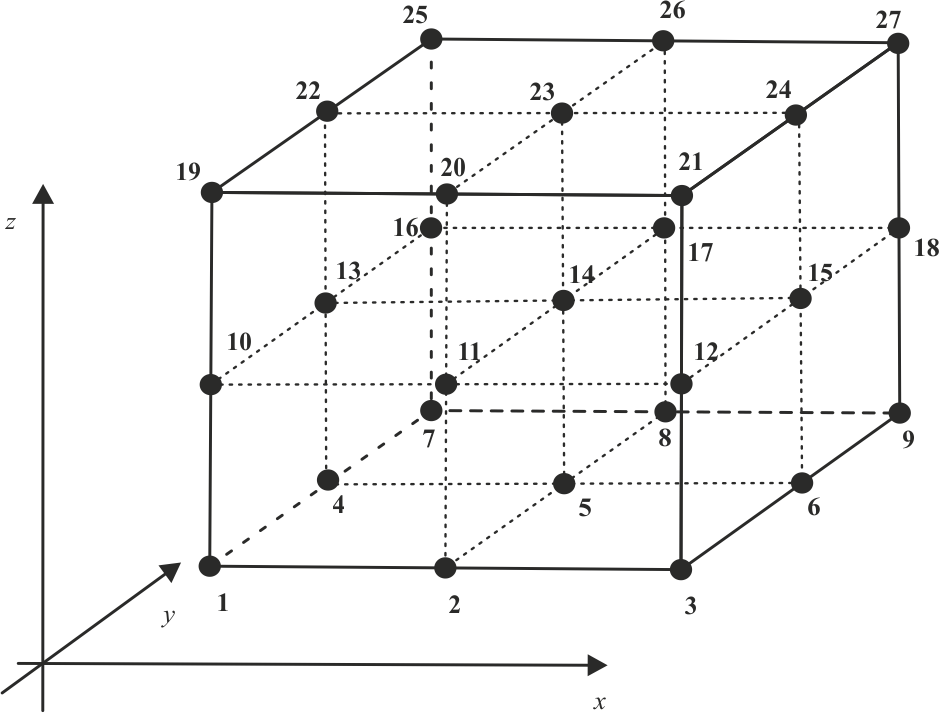
Time = 0.1000000E+01 Diff = 0.243E-15

4.3. Тест 3

*Цель теста:* проверка вторых и третьих краевых условий

*Параметры теста:* , , , 

*Параметры области:* , , , , ,  (27 узлов)

*Параметры времени:* , 

*Краевые условия:*

* первые: узлы 1 - 9, 13, 15, 19 - 27
* вторые: грань 1 - 3 - 19 - 21
* третьи: грань 7 - 9 - 25 - 27

*Результаты:*

***Временной слой Относительная погрешность***

Time = 0.1000000E+00 Diff = 0.155E-05

Time = 0.2000000E+00 Diff = 0.308E-05

Time = 0.3000000E+00 Diff = 0.459E-05

Time = 0.4000000E+00 Diff = 0.606E-05

Time = 0.5000000E+00 Diff = 0.748E-05

Time = 0.6000000E+00 Diff = 0.883E-05

Time = 0.7000000E+00 Diff = 0.101E-04

Time = 0.8000000E+00 Diff = 0.113E-04

Time = 0.9000000E+00 Diff = 0.124E-04

Time = 0.1000000E+01 Diff = 0.135E-04

4.4. Тест 4

*Цель теста:* проверка работы на большом тесте

*Параметры теста:* , , , 

*Параметры области:* , , , , ,  (9261 узел)

*Параметры времени:* , 

*Краевые условия:* везде первые

*Результаты:*

***Временной слой Относительная погрешность***

Time = 0.1000000E+01 Diff = 0.228E-10

Time = 0.2000000E+01 Diff = 0.635E-11

Time = 0.3000000E+01 Diff = 0.322E-11

Time = 0.4000000E+01 Diff = 0.251E-11

Time = 0.5000000E+01 Diff = 0.206E-11

Time = 0.6000000E+01 Diff = 0.182E-11

Time = 0.7000000E+01 Diff = 0.161E-11

Time = 0.8000000E+01 Diff = 0.698E-12

Time = 0.9000000E+01 Diff = 0.379E-12

Time = 0.1000000E+02 Diff = 0.263E-12

Time = 0.1100000E+02 Diff = 0.198E-12

Time = 0.1200000E+02 Diff = 0.179E-12

Time = 0.1300000E+02 Diff = 0.161E-12

Time = 0.1400000E+02 Diff = 0.157E-12

Time = 0.1500000E+02 Diff = 0.151E-12

Time = 0.1600000E+02 Diff = 0.150E-12

Time = 0.1700000E+02 Diff = 0.146E-12

Time = 0.1800000E+02 Diff = 0.146E-12

Time = 0.1900000E+02 Diff = 0.144E-12

Time = 0.2000000E+02 Diff = 0.144E-12

Time = 0.2100000E+02 Diff = 0.143E-12

Time = 0.2200000E+02 Diff = 0.143E-12

Time = 0.2300000E+02 Diff = 0.143E-12

Time = 0.2400000E+02 Diff = 0.142E-12

Time = 0.2500000E+02 Diff = 0.142E-12

Time = 0.2600000E+02 Diff = 0.570E-13

Time = 0.2700000E+02 Diff = 0.289E-13

Time = 0.2800000E+02 Diff = 0.209E-13

Time = 0.2900000E+02 Diff = 0.157E-13

Time = 0.3000000E+02 Diff = 0.153E-13

Time = 0.3100000E+02 Diff = 0.136E-13

Time = 0.3200000E+02 Diff = 0.139E-13

Time = 0.3300000E+02 Diff = 0.131E-13

Time = 0.3400000E+02 Diff = 0.135E-13

Time = 0.3500000E+02 Diff = 0.131E-13

Time = 0.3600000E+02 Diff = 0.133E-13

Time = 0.3700000E+02 Diff = 0.130E-13

Time = 0.3800000E+02 Diff = 0.132E-13

Time = 0.3900000E+02 Diff = 0.130E-13

Time = 0.4000000E+02 Diff = 0.132E-13

Time = 0.4100000E+02 Diff = 0.130E-13

Time = 0.4200000E+02 Diff = 0.131E-13

Time = 0.4300000E+02 Diff = 0.131E-13

Time = 0.4400000E+02 Diff = 0.131E-13

Time = 0.4500000E+02 Diff = 0.131E-13

Time = 0.4600000E+02 Diff = 0.131E-13

Time = 0.4700000E+02 Diff = 0.131E-13

Time = 0.4800000E+02 Diff = 0.131E-13

Time = 0.4900000E+02 Diff = 0.131E-13

Time = 0.5000000E+02 Diff = 0.131E-13

Time = 0.5100000E+02 Diff = 0.131E-13

Time = 0.5200000E+02 Diff = 0.131E-13

Time = 0.5300000E+02 Diff = 0.131E-13

Time = 0.5400000E+02 Diff = 0.131E-13

Time = 0.5500000E+02 Diff = 0.131E-13

Time = 0.5600000E+02 Diff = 0.131E-13

Time = 0.5700000E+02 Diff = 0.131E-13

Time = 0.5800000E+02 Diff = 0.131E-13

Time = 0.5900000E+02 Diff = 0.131E-13

Time = 0.6000000E+02 Diff = 0.131E-13

Time = 0.6100000E+02 Diff = 0.131E-13

Time = 0.6200000E+02 Diff = 0.131E-13

***Временной слой Относительная погрешность***

Time = 0.6300000E+02 Diff = 0.131E-13

Time = 0.6400000E+02 Diff = 0.131E-13

Time = 0.6500000E+02 Diff = 0.131E-13

Time = 0.6600000E+02 Diff = 0.131E-13

Time = 0.6700000E+02 Diff = 0.131E-13

Time = 0.6800000E+02 Diff = 0.131E-13

Time = 0.6900000E+02 Diff = 0.131E-13

Time = 0.7000000E+02 Diff = 0.131E-13

Time = 0.7100000E+02 Diff = 0.131E-13

Time = 0.7200000E+02 Diff = 0.131E-13

Time = 0.7300000E+02 Diff = 0.131E-13

Time = 0.7400000E+02 Diff = 0.131E-13

Time = 0.7500000E+02 Diff = 0.131E-13

Time = 0.7600000E+02 Diff = 0.131E-13

Time = 0.7700000E+02 Diff = 0.131E-13

Time = 0.7800000E+02 Diff = 0.131E-13

Time = 0.7900000E+02 Diff = 0.131E-13

Time = 0.8000000E+02 Diff = 0.520E-14

Time = 0.8100000E+02 Diff = 0.267E-14

Time = 0.8200000E+02 Diff = 0.194E-14

Time = 0.8300000E+02 Diff = 0.147E-14

Time = 0.8400000E+02 Diff = 0.142E-14

Time = 0.8500000E+02 Diff = 0.124E-14

Time = 0.8600000E+02 Diff = 0.127E-14

Time = 0.8700000E+02 Diff = 0.115E-14

Time = 0.8800000E+02 Diff = 0.119E-14

Time = 0.8900000E+02 Diff = 0.113E-14

Time = 0.9000000E+02 Diff = 0.116E-14

Time = 0.9100000E+02 Diff = 0.111E-14

Time = 0.9200000E+02 Diff = 0.115E-14

Time = 0.9300000E+02 Diff = 0.111E-14

Time = 0.9400000E+02 Diff = 0.113E-14

Time = 0.9500000E+02 Diff = 0.111E-14

Time = 0.9600000E+02 Diff = 0.113E-14

Time = 0.9700000E+02 Diff = 0.111E-14

Time = 0.9800000E+02 Diff = 0.112E-14

Time = 0.9900000E+02 Diff = 0.111E-14

Time = 0.1000000E+03 Diff = 0.111E-14

5. Исследования

5.1. Определение порядка аппроксимации по времени

*Параметры теста:* , , , 

*Параметры области:* , , , , ,  (1331 узел)

*Параметры времени:* , 

*Краевые условия:* везде первые

*Результаты:*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 0.05 | 2.730E-06 | 7.640E-07 | 2.720E-07 | 1.837 | 1.490 |
| 0.10 | 1.070E-05 | 2.840E-06 | 8.720E-07 | 1.914 | 1.703 |
| 0.15 | 2.390E-05 | 6.230E-06 | 1.800E-06 | 1.940 | 1.791 |
| 0.20 | 4.240E-05 | 1.090E-05 | 3.060E-06 | 1.960 | 1.833 |
| 0.25 | 6.610E-05 | 1.690E-05 | 4.640E-06 | 1.968 | 1.865 |
| 0.30 | 9.510E-05 | 2.430E-05 | 6.550E-06 | 1.968 | 1.891 |
| 0.35 | 1.290E-04 | 3.290E-05 | 8.780E-06 | 1.971 | 1.906 |
| 0.40 | 1.690E-04 | 4.280E-05 | 1.130E-05 | 1.981 | 1.921 |
| 0.45 | 2.130E-04 | 5.400E-05 | 1.420E-05 | 1.980 | 1.927 |
| 0.50 | 2.630E-04 | 6.650E-05 | 1.740E-05 | 1.984 | 1.934 |
| 0.55 | 3.170E-04 | 8.020E-05 | 2.090E-05 | 1.983 | 1.940 |
| 0.60 | 3.770E-04 | 9.520E-05 | 2.470E-05 | 1.986 | 1.946 |
| 0.65 | 4.410E-04 | 1.110E-04 | 2.890E-05 | 1.990 | 1.941 |
| 0.70 | 5.100E-04 | 1.290E-04 | 3.330E-05 | 1.983 | 1.954 |
| 0.75 | 5.840E-04 | 1.470E-04 | 3.790E-05 | 1.990 | 1.956 |
| 0.80 | 6.620E-04 | 1.670E-04 | 4.290E-05 | 1.987 | 1.961 |
| 0.85 | 7.430E-04 | 1.870E-04 | 4.800E-05 | 1.990 | 1.962 |
| 0.90 | 8.290E-04 | 2.080E-04 | 5.340E-05 | 1.995 | 1.962 |
| 0.95 | 9.170E-04 | 2.310E-04 | 5.900E-05 | 1.989 | 1.969 |
| 1.00 | 1.010E-03 | 2.530E-04 | 6.470E-05 | 1.997 | 1.967 |

6. Выводы

Фактический порядок аппроксимации по времени у реализованного метода получился второй. Однако, в зависимости от коэффициентов уравнения, вида функции и временной сетки, схема может давать как худшую, так и лучшую аппроксимацию.

Следует отметить, что при малых значениях  или при недостаточной мелкости разбиения временной сетки, схема может давать значительную погрешность в виде осцилляций решения во времени.

7. Тексты основных модулей программы

7.1. fem\_module

#if \_\_GFORTRAN\_\_ == 1 && \_\_GNUC\_\_ == 4 && \_\_GNUC\_MINOR\_\_ < 5

#define class type

#endif

module fem\_module

use cgm\_module

use param\_module

implicit none

type :: local\_area ! конечные элементы

integer :: mas\_(8) ! номера узлов

double precision :: lambda\_ ! lambda в области

end type

type :: cross ! координаты узлов

double precision :: mas\_(3) ! x, y, z соотв.

end type

type :: limits ! границы, для вторых-третьих краевых

double precision :: min\_(3), max\_(3)

end type

type :: local\_matrix ! локльные матрицы

double precision :: m\_1(8,8)=reshape(source=(/&

8d0,4d0,4d0,2d0,4d0,2d0,2d0,1d0,&

4d0,8d0,2d0,4d0,2d0,4d0,1d0,2d0,&

4d0,2d0,8d0,4d0,2d0,1d0,4d0,2d0,&

2d0,4d0,4d0,8d0,1d0,2d0,2d0,4d0,&

4d0,2d0,2d0,1d0,8d0,4d0,4d0,2d0,&

2d0,4d0,1d0,2d0,4d0,8d0,2d0,4d0,&

2d0,1d0,4d0,2d0,4d0,2d0,8d0,4d0,&

1d0,2d0,2d0,4d0,2d0,4d0,4d0,8d0 &

/),shape=(/8,8/)) ! матрица массы

double precision :: g\_x(8,8)=reshape(source=(/&

4d0,-4d0,2d0,-2d0,2d0,-2d0,1d0,-1d0,&

-4d0,4d0,-2d0,2d0,-2d0,2d0,-1d0,1d0,&

2d0,-2d0,4d0,-4d0,1d0,-1d0,2d0,-2d0,&

-2d0,2d0,-4d0,4d0,-1d0,1d0,-2d0,2d0,&

2d0,-2d0,1d0,-1d0,4d0,-4d0,2d0,-2d0,&

-2d0,2d0,-1d0,1d0,-4d0,4d0,-2d0,2d0,&

1d0,-1d0,2d0,-2d0,2d0,-2d0,4d0,-4d0,&

-1d0,1d0,-2d0,2d0,-2d0,2d0,-4d0,4d0 &

/),shape=(/8,8/)) ! матрица жесткости (x)

double precision :: g\_y(8,8)=reshape(source=(/&

4d0,2d0,-4d0,-2d0,2d0,1d0,-2d0,-1d0,&

2d0,4d0,-2d0,-4d0,1d0,2d0,-1d0,-2d0,&

-4d0,-2d0,4d0,2d0,-2d0,-1d0,2d0,1d0,&

-2d0,-4d0,2d0,4d0,-1d0,-2d0,1d0,2d0,&

2d0,1d0,-2d0,-1d0,4d0,2d0,-4d0,-2d0,&

1d0,2d0,-1d0,-2d0,2d0,4d0,-2d0,-4d0,&

-2d0,-1d0,2d0,1d0,-4d0,-2d0,4d0,2d0,&

-1d0,-2d0,1d0,2d0,-2d0,-4d0,2d0,4d0 &

/),shape=(/8,8/)) ! матрица жесткости (y)

double precision :: g\_z(8,8)=reshape(source=(/&

4d0,2d0,2d0,1d0,-4d0,-2d0,-2d0,-1d0,&

2d0,4d0,1d0,2d0,-2d0,-4d0,-1d0,-2d0,&

2d0,1d0,4d0,2d0,-2d0,-1d0,-4d0,-2d0,&

1d0,2d0,2d0,4d0,-1d0,-2d0,-2d0,-4d0,&

-4d0,-2d0,-2d0,-1d0,4d0,2d0,2d0,1d0,&

-2d0,-4d0,-1d0,-2d0,2d0,4d0,1d0,2d0,&

-2d0,-1d0,-4d0,-2d0,2d0,1d0,4d0,2d0,&

-1d0,-2d0,-2d0,-4d0,1d0,2d0,2d0,4d0 &

/),shape=(/8,8/)) ! матрица жесткости (z)

double precision :: c\_1(4,4)=reshape(source=(/&

4d0,2d0,2d0,1d0,&

2d0,4d0,1d0,2d0,&

2d0,1d0,4d0,2d0,&

1d0,2d0,2d0,4d0 &

/),shape=(/4,4/)) ! матрица c\_1

end type

type :: fem

type(slae) :: gl\_matr ! СЛАУ

type(local\_area), private, allocatable :: matr(:) ! конечные элементы

type(cross), private, allocatable :: set(:) ! координаты узлов

type(local\_matrix), private :: lc\_matr ! локальные матрицы

double precision :: t\_curr = 0d0, t\_old = 0d0 ! значение времени

double precision, allocatable :: q\_curr(:), q\_old(:) ! вектора весов

type(limits), private :: lim ! границы, для вторых-третьих краевых

contains

procedure :: calc\_ ! функция запуска вычислений

procedure :: read\_ ! функция считывания исходных данных (кроме краевых)

procedure, private :: do\_make\_matrix ! формирование матрицы

procedure, private :: do\_make\_bound ! учет краевых условий

procedure, private :: add ! добавлениие эл -та x к (i, j) эл -ту матрицы

procedure :: dealloc\_ ! освобождение памяти

procedure :: grid\_gen ! генератор сеток

end type

contains

subroutine grid\_gen(this)

implicit none

class(fem) :: this

integer, allocatable :: grid(:,:,:)

integer :: num\_x, num\_y, num\_z, i, j, k, number\_loc

num\_x = ceiling((x\_end - x\_start) / x\_delta)

num\_y = ceiling((y\_end - y\_start) / y\_delta)

num\_z = ceiling((z\_end - y\_start) / z\_delta)

allocate(grid(num\_x+1, num\_y+1, num\_z+1))

! Координаты

open(100, file = 'cross.txt', status = 'unknown')

write(100,\*) (num\_x+1) \* (num\_y+1) \* (num\_z+1)

number\_loc = 1

do i = 0, num\_z, 1

do j = 0, num\_y, 1

do k = 0, num\_x, 1

write(100,\*) x\_start + dble(k) \* x\_delta, y\_start + dble(j) \* y\_delta, z\_start + dble(i) \* z\_delta

grid(k+1, j+1, i+1) = number\_loc

number\_loc = number\_loc + 1

end do

end do

end do

write(100,\*)

close(100)

! КЭ

open(100, file = 'local.txt', status = 'unknown')

write(100,\*) (num\_x) \* (num\_y) \* (num\_z)

do i = 1, num\_z, 1

do j = 1, num\_y, 1

do k = 1, num\_x, 1

write(100,\*) grid(k, j, i), grid(k+1, j, i), grid(k, j+1, i), grid(k+1, j+1, i), &

grid(k, j, i+1), grid(k+1, j, i+1), grid(k, j+1, i+1), grid(k+1, j+1, i+1)

write(100,\*) lambda\_grid

end do

end do

end do

write(100,\*)

close(100)

! Краевые

open(100, file = 'bound1.txt', status = 'unknown')

do j = 1, num\_y+1, 1

do k = 1, num\_x+1, 1

write(100,\*) grid(k, j, 1)

write(100,\*) grid(k, j, num\_z+1)

end do

end do

do i = 2, num\_z, 1

do k = 1, num\_x+1, 1

write(100,\*) grid(k, 1, i)

write(100,\*) grid(k, num\_y+1, i)

end do

end do

do i = 2, num\_z, 1

do j = 2, num\_y, 1

write(100,\*) grid(1, j, i)

write(100,\*) grid(num\_x+1, j, i)

end do

end do

write(100,\*)

close(100)

deallocate(grid)

end subroutine

subroutine read\_(this) ! функция считывания исходных данных (кроме краевых)

use list\_module

implicit none

common / counters / c\_local, c\_cross

class(fem) :: this

integer :: i, j, c\_local, c\_cross, gg\_size, i1, j1, k, tmp

double precision :: q\_l, x\_l, y\_l, z\_l

class(list), allocatable :: profile(:)

! Считывание координат узлов

open(10, file = 'cross.txt', status = 'old')

rewind 10

read(10, \* ) c\_cross

this%gl\_matr%n = c\_cross

allocate(this%set(c\_cross))

do i = 1, c\_cross

read(10, \*) (this%set(i)%mas\_(j), j = 1, 3)

do j = 1, 3

this%lim%max\_(j) = max(this%set(i)%mas\_(j), this%lim%max\_(j))

this%lim%min\_(j) = min(this%set(i)%mas\_(j), this%lim%min\_(j))

end do

end do

close(10)

! Считывание нумерации

open(20, file = 'local.txt', status = 'old')

rewind 20

read(20, \* ) c\_local

allocate(this%matr(c\_local))

do i = 1, c\_local

read(20, \*) (this%matr(i)%mas\_(j), j = 1, 8)

read(20, \*) this%matr(i)%lambda\_

end do

close(20)

! Формирование профиля

gg\_size = 0

allocate(profile(this%gl\_matr%n))

do k = 1, c\_local

do i = 1, 8

do j = i + 1, 8, 1

i1 = this%matr(k)%mas\_(i)

j1 = this%matr(k)%mas\_(j)

if(i1 .lt. j1)then

tmp = i1

i1 = j1

j1 = tmp

end if

if(profile(i1 -1)%elem\_count(j1) .eq. 0) then

call profile(i1 -1)%add(j1)

gg\_size = gg\_size + 1

end if

end do

end do

end do

do i = 1, this%gl\_matr%n, 1

call profile(i)%sort()

end do

! Выделение памяти

call this%gl\_matr%alloc\_all(gg\_size)

! Формирование ig и jg

this%gl\_matr%ig(1) = 1

this%gl\_matr%ig(2) = 1

tmp = 1

do i = 1, this%gl\_matr%n - 1

k = 0

do j = 1, i

if(profile(i)%elem\_count(j) .ne. 0) then

this%gl\_matr%jg(tmp) = j

tmp = tmp + 1

k = k + 1

end if

end do

this%gl\_matr%ig(i + 2) = this%gl\_matr%ig(i + 1) + k

end do

deallocate(profile)

! И еще немного памяти

allocate(this%q\_curr(this%gl\_matr%n))

allocate(this%q\_old(this%gl\_matr%n))

this%t\_curr = t\_start + t\_delta

this%t\_old = t\_start

! Начальные условия

do i = 1, c\_cross

x\_l = this%set(i)%mas\_(1)

y\_l = this%set(i)%mas\_(2)

z\_l = this%set(i)%mas\_(3)

q\_l = u\_(x\_l, y\_l, z\_l, t\_start)

this%q\_curr(i) = q\_l

end do

end subroutine

subroutine calc\_(this) ! функция запуска вычислений

implicit none

class(fem) :: this

integer i, j, num\_t

double precision :: diff, x\_l, y\_l, z\_l, u\_l, norm

open(90, file = 'solution.txt', status = 'unknown')

open(95, file = 'diff.txt', status = 'unknown')

num\_t = ceiling((t\_end - t\_start) / t\_delta)

write(90, \*) '======= t = ', t\_start, '========='

do j = 1, this%gl\_matr%n

write(90, \*) this%q\_curr(j)

end do

do i = 1, num\_t, 1

this%q\_old = this%q\_curr

this%gl\_matr%di = 0d0

this%gl\_matr%f = 0d0

this%gl\_matr%gg = 0d0

this%gl\_matr%x = 0d0

call this%do\_make\_matrix()

call this%do\_make\_bound()

call this%gl\_matr%calc\_cgm()

this%q\_curr = this%gl\_matr%x

diff = 0d0

norm = 0d0

do j = 1, this%gl\_matr%n

x\_l = this%set(j)%mas\_(1)

y\_l = this%set(j)%mas\_(2)

z\_l = this%set(j)%mas\_(3)

u\_l = u\_(x\_l, y\_l, z\_l, this%t\_curr)

diff = diff + (this%q\_curr(j) - u\_l) \*\* 2

norm = norm + u\_l \*\* 2

end do

diff = dsqrt(diff) / dsqrt(norm)

write(\*, '(a7,e14.7,a9,e10.3)') 'Time = ', this%t\_curr, ' Diff = ', diff

write(95, '(a7,e14.7,a9,e10.3)') 'Time = ', this%t\_curr, ' Diff = ', diff

write(90, \*) '======= t = ', this%t\_curr, '========='

do j = 1, this%gl\_matr%n

x\_l = this%set(j)%mas\_(1)

y\_l = this%set(j)%mas\_(2)

z\_l = this%set(j)%mas\_(3)

u\_l = u\_(x\_l, y\_l, z\_l, this%t\_curr)

write(90, \*) this%q\_curr(j), u\_l

end do

this%t\_old = this%t\_curr

this%t\_curr = t\_start + t\_delta \* dble(i+1)

end do

close(90)

close(95)

end subroutine

subroutine add(this, i, j, x) ! добавлениие эл -та x к (i, j) эл -ту матрицы

implicit none

class(fem) :: this

double precision :: x

integer :: i, j, k

do k = this%gl\_matr%ig(i), this%gl\_matr%ig(i + 1) -1

if(this%gl\_matr%jg(k) .eq. j) goto 100

end do

100 continue

this%gl\_matr%gg(k) = this%gl\_matr%gg(k) + x

end subroutine

subroutine do\_make\_matrix(this) ! формирование матрицы

implicit none

common / counters / c\_local, c\_cross

class(fem) :: this

integer :: i, j, c\_local, c\_cross, k, i1, j1, l, flag

double precision :: h(3), b\_k, c\_k, fr(8), tmp\_vector\_1(8), tmp\_vector\_2(8)

double precision :: tmpmatr(8,8), sigma\_l

! Вычисление шага h

do i = 1, this%gl\_matr%n

this%gl\_matr%di(i) = 0d0

this%gl\_matr%f(i) = 0d0

end do

do i = 1, this%gl\_matr%ig(this%gl\_matr%n)

this%gl\_matr%gg(i) = 0d0

end do

do k = 1, c\_local

do i = 1, 3

flag = 1

do j = 2, 8

if(flag .eq. 1)then

flag = 0

do l = 1, 3 ! проверяем, лежат ли точки на нужном ребре

if(i .ne. l .and. this%set(this%matr(k)%mas\_(1))%mas\_(l) .ne. this%set(this%matr(k)%mas\_(j))%mas\_(l))then

flag = 1

end if

end do

end if

end do

if(flag .eq. 0)then

h(i) = dabs(this%set(this%matr(k)%mas\_(1))%mas\_(i) - this%set(this%matr(k)%mas\_(j -1))%mas\_(i))

end if

end do

! заполнение массива gg

b\_k = this%matr(k)%lambda\_ \* h(1) \* h(2) \* h(3) / 36d0

c\_k = h(1) \* h(2) \* h(3) / 216d0

! вектор правой части для локал. матрицы

do i = 1, 8

tmp\_vector\_1(i) = f\_(this%set(this%matr(k)%mas\_(i))%mas\_(1), &

this%set(this%matr(k)%mas\_(i))%mas\_(2), &

this%set(this%matr(k)%mas\_(i))%mas\_(3), this%t\_curr)

tmp\_vector\_2(i) = f\_(this%set(this%matr(k)%mas\_(i))%mas\_(1), &

this%set(this%matr(k)%mas\_(i))%mas\_(2), &

this%set(this%matr(k)%mas\_(i))%mas\_(3), this%t\_old)

end do

fr = c\_k \* (matmul(this%lc\_matr%m\_1, tmp\_vector\_1) + matmul(this%lc\_matr%m\_1, tmp\_vector\_2)) / 2d0

! Сигма на текущем КЭ

sigma\_l = 0d0

do i = 1, 8, 1

sigma\_l = sigma\_l + sigma\_(this%set(this%matr(k)%mas\_(i))%mas\_(1), &

this%set(this%matr(k)%mas\_(i))%mas\_(2), &

this%set(this%matr(k)%mas\_(i))%mas\_(3), &

this%t\_curr)

end do

sigma\_l = sigma\_l / 8d0

! Матрица для добавки в правую часть

tmpmatr = 1d0 / (this%t\_curr - this%t\_old) \* c\_k \* sigma\_l \* this%lc\_matr%m\_1 - &

1d0 / 2d0 \* b\_k \* (1d0 / (h(1) \*\* 2) \* this%lc\_matr%g\_x + &

1d0 / (h(2) \*\* 2) \* this%lc\_matr%g\_y + &

1d0 / (h(3) \*\* 2) \* this%lc\_matr%g\_z)

! (1/dt\*M-1/2\*G)\*q^(j-1)

do i = 1, 8, 1

tmp\_vector\_1(i) = this%q\_old(this%matr(k)%mas\_(i))

end do

fr = fr + matmul(tmpmatr, tmp\_vector\_1)

! Локальная матрица

tmpmatr = 1d0 / (this%t\_curr - this%t\_old) \* c\_k \* sigma\_l \* this%lc\_matr%m\_1 + &

1d0 / 2d0 \* b\_k \* (1d0 / (h(1) \*\* 2) \* this%lc\_matr%g\_x + &

1d0 / (h(2) \*\* 2) \* this%lc\_matr%g\_y + &

1d0 / (h(3) \*\* 2) \* this%lc\_matr%g\_z)

do i = 1, 8

do j = 1, i -1, 1

i1 = this%matr(k)%mas\_(i)

j1 = this%matr(k)%mas\_(j)

! добавка в gg

if(i1 .lt. j1)then

call this%add(j1, i1, tmpmatr(i, j))

else

call this%add(i1, j1, tmpmatr(i, j))

end if

end do

! добавка в диагональ

this%gl\_matr%di(this%matr(k)%mas\_(i)) = this%gl\_matr%di(this%matr(k)%mas\_(i)) + tmpmatr(i, i)

! добавка в правую часть

this%gl\_matr%f(this%matr(k)%mas\_(i)) = this%gl\_matr%f(this%matr(k)%mas\_(i)) + fr(i)

end do

end do

end subroutine

subroutine do\_make\_bound(this) ! учет краевых условий

implicit none

common / counters / c\_local, c\_cross

class(fem) :: this

integer :: c\_local, c\_cross, i, j, cr(4)

double precision :: teta(4), beta, vec\_tet(4), hplane, h1, h2, tmp, grad(4), teta\_old(4), grad\_old(4)

integer :: i\_s, i\_e, cur\_row, p

! учет вторых краевых условий

open(30, file = 'bound2.txt', status = 'old', err = 100)

rewind 30

!print \*, '2 bound detected!'

do while(.true.)

read(30, \*, err = 100, end = 100) (cr(i), i = 1, 4)

! 2-4-6-8

if(this%set(cr(1))%mas\_(1) .eq. this%lim%max\_(1) .and. &

this%set(cr(2))%mas\_(1) .eq. this%lim%max\_(1) .and. &

this%set(cr(3))%mas\_(1) .eq. this%lim%max\_(1) .and. &

this%set(cr(4))%mas\_(1) .eq. this%lim%max\_(1)) then

do i = 1, 4

grad = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_curr)

grad\_old = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_old)

do j = 1, c\_local

if(this%matr(j)%mas\_(2) .eq. cr(1) .and. &

this%matr(j)%mas\_(4) .eq. cr(2) .and. &

this%matr(j)%mas\_(6) .eq. cr(3) .and. &

this%matr(j)%mas\_(8) .eq. cr(4)) goto 40

end do

40 continue

teta(i) = grad(1) \* this%matr(j)%lambda\_

teta\_old(i) = grad\_old(1) \* this%matr(j)%lambda\_

end do

end if

! 1-3-5-7

if(this%set(cr(1))%mas\_(1) .eq. this%lim%min\_(1) .and. &

this%set(cr(2))%mas\_(1) .eq. this%lim%min\_(1) .and. &

this%set(cr(3))%mas\_(1) .eq. this%lim%min\_(1) .and. &

this%set(cr(4))%mas\_(1) .eq. this%lim%min\_(1)) then

do i = 1, 4

grad = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_curr)

grad\_old = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_old)

do j = 1, c\_local

if(this%matr(j)%mas\_(1) .eq. cr(1) .and. &

this%matr(j)%mas\_(3) .eq. cr(2) .and. &

this%matr(j)%mas\_(5) .eq. cr(3) .and. &

this%matr(j)%mas\_(7) .eq. cr(4)) goto 41

end do

41 continue

teta(i) = - grad(1) \* this%matr(j)%lambda\_

teta\_old(i) = - grad\_old(1) \* this%matr(j)%lambda\_

end do

end if

! 3-4-7-8

if(this%set(cr(1))%mas\_(2) .eq. this%lim%max\_(2) .and. &

this%set(cr(2))%mas\_(2) .eq. this%lim%max\_(2) .and. &

this%set(cr(3))%mas\_(2) .eq. this%lim%max\_(2) .and. &

this%set(cr(4))%mas\_(2) .eq. this%lim%max\_(2)) then

do i = 1, 4

grad = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_curr)

grad\_old = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_old)

do j = 1, c\_local

if(this%matr(j)%mas\_(3) .eq. cr(1) .and. &

this%matr(j)%mas\_(4) .eq. cr(2) .and. &

this%matr(j)%mas\_(7) .eq. cr(3) .and. &

this%matr(j)%mas\_(8) .eq. cr(4)) goto 42

end do

42 continue

teta(i) = grad(2) \* this%matr(j)%lambda\_

teta\_old(i) = grad\_old(2) \* this%matr(j)%lambda\_

end do

end if

! 1-2-5-6

if(this%set(cr(1))%mas\_(2) .eq. this%lim%min\_(2) .and. &

this%set(cr(2))%mas\_(2) .eq. this%lim%min\_(2) .and. &

this%set(cr(3))%mas\_(2) .eq. this%lim%min\_(2) .and. &

this%set(cr(4))%mas\_(2) .eq. this%lim%min\_(2)) then

do i = 1, 4

grad = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_curr)

grad\_old = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_old)

do j = 1, c\_local

if(this%matr(j)%mas\_(1) .eq. cr(1) .and. &

this%matr(j)%mas\_(2) .eq. cr(2) .and. &

this%matr(j)%mas\_(5) .eq. cr(3) .and. &

this%matr(j)%mas\_(6) .eq. cr(4)) goto 43

end do

43 continue

teta(i) = - grad(2) \* this%matr(j)%lambda\_

teta\_old(i) = - grad\_old(2) \* this%matr(j)%lambda\_

end do

end if

! 5-6-7-8

if(this%set(cr(1))%mas\_(3) .eq. this%lim%max\_(3) .and. &

this%set(cr(2))%mas\_(3) .eq. this%lim%max\_(3) .and. &

this%set(cr(3))%mas\_(3) .eq. this%lim%max\_(3) .and. &

this%set(cr(4))%mas\_(3) .eq. this%lim%max\_(3)) then

do i = 1, 4

grad = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_curr)

grad\_old = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_old)

do j = 1, c\_local

if(this%matr(j)%mas\_(5) .eq. cr(1) .and. &

this%matr(j)%mas\_(6) .eq. cr(2) .and. &

this%matr(j)%mas\_(7) .eq. cr(3) .and. &

this%matr(j)%mas\_(8) .eq. cr(4)) goto 44

end do

44 continue

teta(i) = grad(3) \* this%matr(j)%lambda\_

teta\_old(i) = grad\_old(3) \* this%matr(j)%lambda\_

end do

end if

! 1-2-3-4

if(this%set(cr(1))%mas\_(3) .eq. this%lim%min\_(3) .and. &

this%set(cr(2))%mas\_(3) .eq. this%lim%min\_(3) .and. &

this%set(cr(3))%mas\_(3) .eq. this%lim%min\_(3) .and. &

this%set(cr(4))%mas\_(3) .eq. this%lim%min\_(3)) then

do i = 1, 4

grad = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_curr)

grad\_old = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_old)

do j = 1, c\_local

if(this%matr(j)%mas\_(1) .eq. cr(1) .and. &

this%matr(j)%mas\_(2) .eq. cr(2) .and. &

this%matr(j)%mas\_(3) .eq. cr(3) .and. &

this%matr(j)%mas\_(4) .eq. cr(4)) goto 45

end do

45 continue

teta(i) = - grad(3) \* this%matr(j)%lambda\_

teta\_old(i) = - grad\_old(3) \* this%matr(j)%lambda\_

end do

end if

h1 = 0d0

h2 = 0d0

do i = 1, 3

tmp = dabs(this%set(cr(1))%mas\_(i) - this%set(cr(2))%mas\_(i))

if(tmp .gt. eps)then

h1 = h1 + tmp

end if

tmp = dabs(this%set(cr(1))%mas\_(i) - this%set(cr(3))%mas\_(i))

if(tmp .gt. eps)then

h2 = h2 + tmp

end if

end do

! добавка в правую часть

hplane = h1 \* h2 / 36d0

vec\_tet = (matmul(this%lc\_matr%c\_1, teta) + matmul(this%lc\_matr%c\_1, teta\_old)) / 2d0

do i = 1, 4

this%gl\_matr%f(cr(i)) = this%gl\_matr%f(cr(i)) + hplane \* vec\_tet(i)

end do

end do

close(30)

100 continue

! учет третьих краевых условий

open(40, file = 'bound3.txt', status = 'old', err = 200)

rewind 40

!print \* , '3 bound detected!'

do while(.true.)

read(40, \*, err = 200, end = 200) (cr(i), i = 1, 4)

read(40, \*, err = 200, end = 200) beta

! 2-4-6-8

if(this%set(cr(1))%mas\_(1) .eq. this%lim%max\_(1) .and. &

this%set(cr(2))%mas\_(1) .eq. this%lim%max\_(1) .and. &

this%set(cr(3))%mas\_(1) .eq. this%lim%max\_(1) .and. &

this%set(cr(4))%mas\_(1) .eq. this%lim%max\_(1)) then

do i = 1, 4

grad = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_curr)

grad\_old = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_old)

do j = 1, c\_local

if(this%matr(j)%mas\_(2) .eq. cr(1) .and. &

this%matr(j)%mas\_(4) .eq. cr(2) .and. &

this%matr(j)%mas\_(6) .eq. cr(3) .and. &

this%matr(j)%mas\_(8) .eq. cr(4)) goto 60

end do

60 continue

teta(i) = grad(1) \* this%matr(j)%lambda\_ + beta \* &

u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_curr)

teta\_old(i) = grad\_old(1) \* this%matr(j)%lambda\_ + beta \* &

u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_old)

end do

end if

! 1-3-5-7

if(this%set(cr(1))%mas\_(1) .eq. this%lim%min\_(1) .and. &

this%set(cr(2))%mas\_(1) .eq. this%lim%min\_(1) .and. &

this%set(cr(3))%mas\_(1) .eq. this%lim%min\_(1) .and. &

this%set(cr(4))%mas\_(1) .eq. this%lim%min\_(1)) then

do i = 1, 4

grad = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_curr)

grad\_old = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_old)

do j = 1, c\_local

if(this%matr(j)%mas\_(1) .eq. cr(1) .and. &

this%matr(j)%mas\_(3) .eq. cr(2) .and. &

this%matr(j)%mas\_(5) .eq. cr(3) .and. &

this%matr(j)%mas\_(7) .eq. cr(4)) goto 61

end do

61 continue

teta(i) = - grad(1) \* this%matr(j)%lambda\_ + beta \* &

u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_curr)

teta\_old(i) = - grad\_old(1) \* this%matr(j)%lambda\_ + beta \* &

u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_old)

end do

end if

! 3-4-7-8

if(this%set(cr(1))%mas\_(2) .eq. this%lim%max\_(2) .and. &

this%set(cr(2))%mas\_(2) .eq. this%lim%max\_(2) .and. &

this%set(cr(3))%mas\_(2) .eq. this%lim%max\_(2) .and. &

this%set(cr(4))%mas\_(2) .eq. this%lim%max\_(2)) then

do i = 1, 4

grad = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_curr)

grad\_old = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_old)

do j = 1, c\_local

if(this%matr(j)%mas\_(3) .eq. cr(1) .and. &

this%matr(j)%mas\_(4) .eq. cr(2) .and. &

this%matr(j)%mas\_(7) .eq. cr(3) .and. &

this%matr(j)%mas\_(8) .eq. cr(4)) goto 62

end do

62 continue

teta(i) = grad(2) \* this%matr(j)%lambda\_ + beta \* &

u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_curr)

teta\_old(i) = grad\_old(2) \* this%matr(j)%lambda\_ + beta \* &

u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_old)

end do

end if

! 1-2-5-6

if(this%set(cr(1))%mas\_(2) .eq. this%lim%min\_(2) .and. &

this%set(cr(2))%mas\_(2) .eq. this%lim%min\_(2) .and. &

this%set(cr(3))%mas\_(2) .eq. this%lim%min\_(2) .and. &

this%set(cr(4))%mas\_(2) .eq. this%lim%min\_(2)) then

do i = 1, 4

grad = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_curr)

grad\_old = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_old)

do j = 1, c\_local

if(this%matr(j)%mas\_(1) .eq. cr(1) .and. &

this%matr(j)%mas\_(2) .eq. cr(2) .and. &

this%matr(j)%mas\_(5) .eq. cr(3) .and. &

this%matr(j)%mas\_(6) .eq. cr(4)) goto 63

end do

63 continue

teta(i) = - grad(2) \* this%matr(j)%lambda\_ + beta \* &

u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_curr)

teta\_old(i) = - grad\_old(2) \* this%matr(j)%lambda\_ + beta \* &

u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_old)

end do

end if

! 5-6-7-8

if(this%set(cr(1))%mas\_(3) .eq. this%lim%max\_(3) .and. &

this%set(cr(2))%mas\_(3) .eq. this%lim%max\_(3) .and. &

this%set(cr(3))%mas\_(3) .eq. this%lim%max\_(3) .and. &

this%set(cr(4))%mas\_(3) .eq. this%lim%max\_(3)) then

do i = 1, 4

grad = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_curr)

grad\_old = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_old)

do j = 1, c\_local

if(this%matr(j)%mas\_(5) .eq. cr(1) .and. &

this%matr(j)%mas\_(6) .eq. cr(2) .and. &

this%matr(j)%mas\_(7) .eq. cr(3) .and. &

this%matr(j)%mas\_(8) .eq. cr(4)) goto 64

end do

64 continue

teta(i) = grad(3) \* this%matr(j)%lambda\_ + beta \* &

u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_curr)

teta\_old(i) = grad\_old(3) \* this%matr(j)%lambda\_ + beta \* &

u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_old)

end do

end if

! 1-2-3-4

if(this%set(cr(1))%mas\_(3) .eq. this%lim%min\_(3) .and. &

this%set(cr(2))%mas\_(3) .eq. this%lim%min\_(3) .and. &

this%set(cr(3))%mas\_(3) .eq. this%lim%min\_(3) .and. &

this%set(cr(4))%mas\_(3) .eq. this%lim%min\_(3)) then

do i = 1, 4

grad = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_curr)

grad\_old = grad\_u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_old)

do j = 1, c\_local

if(this%matr(j)%mas\_(1) .eq. cr(1) .and. &

this%matr(j)%mas\_(2) .eq. cr(2) .and. &

this%matr(j)%mas\_(3) .eq. cr(3) .and. &

this%matr(j)%mas\_(4) .eq. cr(4)) goto 65

end do

65 continue

teta(i) = - grad(3) \* this%matr(j)%lambda\_ + beta \* &

u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_curr)

teta\_old(i) = - grad\_old(3) \* this%matr(j)%lambda\_ + beta \* &

u\_(this%set(cr(i))%mas\_(1), this%set(cr(i))%mas\_(2), this%set(cr(i))%mas\_(3), this%t\_old)

end do

end if

h1 = 0d0

h2 = 0d0

do i = 1, 3

tmp = dabs(this%set(cr(1))%mas\_(i) - this%set(cr(2))%mas\_(i))

if(tmp .gt. eps)then

h1 = h1 + tmp

end if

tmp = dabs(this%set(cr(1))%mas\_(i) - this%set(cr(3))%mas\_(i))

if(tmp .gt. eps)then

h2 = h2 + tmp

end if

end do

! добавка в правую часть

hplane = h1 \* h2 / 36d0

vec\_tet = (matmul(this%lc\_matr%c\_1, teta) + matmul(this%lc\_matr%c\_1, teta\_old)) / 2d0

do i = 1, 4

teta\_old(i) = this%q\_old(cr(i)) ! Это у нас такой q\_old

end do

teta = (matmul(this%lc\_matr%c\_1, teta\_old)) / 2d0 ! А это у нас такой С1\*q\_old

do i = 1, 4

this%gl\_matr%f(cr(i)) = this%gl\_matr%f(cr(i)) + hplane \* vec\_tet(i) - hplane \* teta(i)

end do

! добавка в глобальную матрицу

hplane = hplane \* beta / 2d0

do i = 1, 4

this%gl\_matr%di(cr(i)) = this%gl\_matr%di(cr(i)) + hplane \* this%lc\_matr%c\_1(i, i)

do j = 1, i -1, 1

call this%add(cr(i), cr(j), hplane \* this%lc\_matr%c\_1(i, j))

end do

end do

end do

close(40)

200 continue

! учет первых краевых условий

open(50, file = 'bound1.txt', status = 'old', err = 300)

rewind 50

!print\*, '1 bound detected!'

do while(.true.)

read(50, \*, err = 300, end = 300) cur\_row

tmp = u\_(this%set(cur\_row)%mas\_(1), this%set(cur\_row)%mas\_(2), this%set(cur\_row)%mas\_(3), this%t\_curr)

this%gl\_matr%di(cur\_row) = 1d0

this%gl\_matr%f(cur\_row) = tmp

i\_s = this%gl\_matr%ig(cur\_row)

i\_e = this%gl\_matr%ig(cur\_row + 1) - 1

do i = i\_s, i\_e, 1

this%gl\_matr%f(this%gl\_matr%jg(i)) = this%gl\_matr%f(this%gl\_matr%jg(i)) - this%gl\_matr%gg(i) \* tmp

this%gl\_matr%gg(i) = 0d0

end do

do p = cur\_row + 1, this%gl\_matr%n, 1

i\_s = this%gl\_matr%ig(p)

i\_e = this%gl\_matr%ig(p + 1) - 1

do i = i\_s, i\_e, 1

if(this%gl\_matr%jg(i) .eq. cur\_row) then

this%gl\_matr%f(p) = this%gl\_matr%f(p) - this%gl\_matr%gg(i) \* tmp

this%gl\_matr%gg(i) = 0d0

end if

end do

end do

end do

close(50)

300 continue

end subroutine

subroutine dealloc\_(this) ! освобождение памяти

implicit none

class(fem) :: this

call this%gl\_matr%dealloc\_all()

deallocate(this%set)

deallocate(this%matr)

deallocate(this%q\_curr)

deallocate(this%q\_old)

end subroutine

end module

7.2. param\_module

#if \_\_GFORTRAN\_\_ == 1 && \_\_GNUC\_\_ == 4 && \_\_GNUC\_MINOR\_\_ < 5

#define class type

#endif

module param\_module

double precision, parameter :: derivative = 1d-8

double precision, parameter :: t\_start = 0d0, t\_delta = 0.0125d0, t\_end = 1d0

double precision, parameter :: x\_start = 0d0, x\_delta = 1d0, x\_end = 10d0

double precision, parameter :: y\_start = 0d0, y\_delta = 1d0, y\_end = 10d0

double precision, parameter :: z\_start = 0d0, z\_delta = 1d0, z\_end = 10d0

double precision, parameter :: lambda\_grid = 1d0

contains

function u\_(x, y, z, t)

implicit none

double precision :: x, y, z, t, u\_

!u\_ = x\*\*2 + t\*\*2 + y\*\*2 + z\*\*2

!u\_ = x\*\*2 + y\*\*2 + z\*\*2 + t

!u\_ = 2d0\*x + y\*t + z - 2d0 + t + 3d0\*t\*\*2

!u\_ = 2d0\*x + y\*t + z\*\*2 - 2d0 + t + 3d0\*t\*\*2 + x\*\*2\*t\*\*2

u\_ = sin(x + y + z + t\*\*2)

end function

function f\_(x, y, z, t) ! функция правой части уравнения

implicit none

double precision :: x, y, z, t, f\_

!f\_ = -6d0 + sigma\_(x, y, z, t) \* 2d0 \* t

!f\_ = -6d0 + sigma\_(x, y, z, t)

!f\_ = sigma\_(x, y, z, t) \* (y + 1d0 + 6d0\*t)

!f\_ = sigma\_(x, y, z, t) \* (y + 1d0 + 6d0\*t + 2d0\*t\*x\*\*2) - 2 - 2d0\*t\*\*2

f\_ = sigma\_(x, y, z, t) \* 2d0 \* t \* cos(t\*\*2 + x + y + z) + 3d0\*sin(t\*\*2 + x + y + z)

end function

function sigma\_(x, y, z, t)

implicit none

double precision :: x, y, z, t, sigma\_

sigma\_ = 1d5

end function

function grad\_u\_(x, y, z, t)

implicit none

double precision :: x, y, z, t

double precision :: grad\_u\_(4)

!grad\_u\_(1) = (u\_(x + derivative, y, z, t) - u\_(x - derivative, y, z, t)) / (2d0 \* derivative)

!grad\_u\_(2) = (u\_(x, y + derivative, z, t) - u\_(x, y - derivative, z, t)) / (2d0 \* derivative)

!grad\_u\_(3) = (u\_(x, y, z + derivative, t) - u\_(x, y, z - derivative, t)) / (2d0 \* derivative)

!grad\_u\_(4) = (u\_(x, y, z, t + derivative) - u\_(x, y, z, t - derivative)) / (2d0 \* derivative)

grad\_u\_(1) = 2d0\*x

grad\_u\_(2) = 2d0\*y

grad\_u\_(3) = 2d0\*z

grad\_u\_(4) = 2d0\*t

end function

end module