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

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 λ – 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 u определяется дифференциальным уравнением

(1) $\sigma \dfrac{\partial u}{\partial t} - div(\lambda \operatorname{grad} u) = f$,

заданным в некоторой области Ω с границей $S = S_1 \bigcup S_2 \bigcup S_3$, начальным условием

(2) $u\big|_{t=t_0} = u^0$

и краевыми условиями

(3) $u\big|_{S_1} = u_g$, (4) $\lambda \dfrac{\partial u}{\partial n}\Big|_{S_2} = \theta$, (5) $\lambda \dfrac{\partial u}{\partial n}\Big|_{S_3} + \beta\left(u\big|_{S_3} - u_\beta\right) = 0$,

в которых $u\big|_{S_i}$ - значение искомой функции u на границе S_i, а $\dfrac{\partial u}{\partial n}\Big|_{S_i}$ - значение на S_i производной функции u по направлению внешней нормали к поверхности S_i, $i = 1,2,3$.

Коэффициент λ - кусочно-постоянная функция, коэффициент $\sigma(x,y,z)$ - некоторая известная функция.

Вид конечных элементов – параллелепипеды.
Вид базисных функций – трилинейные.

2. Теоретическая часть

2.1. Дискретизация по времени

Положим, что ось времени t разбита на так называемые временные слои со значениями t_j, $j = 1...J$, а значения искомой функции u и параметров λ, σ и f уравнения (1) будем обозначать соответственно через λ^j, σ^j и f^j, которые уже не зависят от времени t, но остаются функциями пространственных координат.

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

(6) $\sigma \dfrac{u^j - u^{j-1}}{\Delta t} - \operatorname{div}\left(\lambda \operatorname{grad} \dfrac{u^j - u^{j-1}}{2}\right) = \dfrac{f^j - f^{j-1}}{2}$, $j = 1...J$, $\Delta t = t^j - t^{j-1}$

2.2. Вариационная постановка

Пусть v – некоторая пробная функция из пространства H_0^1, тогда:

(7) $\int\limits_\Omega \sigma \dfrac{u^j - u^{j-1}}{\Delta t} v\, d\Omega + \int\limits_\Omega - \operatorname{div}\left(\lambda \operatorname{grad} \dfrac{u^j - u^{j-1}}{2}\right) v\, d\Omega = \int\limits_\Omega \dfrac{f^j - f^{j-1}}{2} v\, d\Omega$.

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

$$\frac{1}{2}\int\limits_\Omega \lambda^j \operatorname{grad}\left(u^j\right)\operatorname{grad}(v)\, d\Omega + \frac{1}{2}\int\limits_\Omega \lambda^j \operatorname{grad}\left(u^{j-1}\right)\operatorname{grad}(v)\, d\Omega + \frac{1}{2}\int\limits_{S_3} \beta^j u^j v\, dS_3 + \frac{1}{2}\int\limits_{S_3} \beta^{j-1} u^{j-1} v\, dS_3 +$$

(8) $$+\frac{1}{\Delta t}\int\limits_\Omega \sigma^j u^j v\, d\Omega - \frac{1}{\Delta t}\int\limits_\Omega \sigma^j u^{j-1} v\, d\Omega = \frac{1}{2}\int\limits_\Omega f^j v\, d\Omega + \frac{1}{2}\int\limits_\Omega f^{j-1} v\, d\Omega + \frac{1}{2}\int\limits_{S_2} \theta^j v\, dS_2 + \frac{1}{2}\int\limits_{S_2} \theta^{j-1} v\, dS_2 +$$

$$+\frac{1}{2}\int\limits_{S_3} \beta^j u_\beta^j v\, dS_3 + \frac{1}{2}\int\limits_{S_3} \beta^{j-1} u_\beta^{j-1} v\, dS_3$$

2.3. Конечноэлементная дискретизация

Заменим пространство H_0^1 на конечномерное пространство V^h, которое определим как линейное пространство, натянутое на базисные функции ψ_i, $i=1...n$. Заменим в (8) функцию u аппроксимирующей ее функцией u^h, а функцию v - функцией v^h.

Поскольку любая функция v^h может быть представлена в виде линейной комбинации $v^h=\sum_i q_i^v\psi_i$, получим:

$$\frac{1}{2}\int_\Omega \lambda^j \operatorname{grad}\left(u^{hj}\right)\operatorname{grad}\left(\psi_i\right)d\Omega+\frac{1}{2}\int_\Omega \lambda^j \operatorname{grad}\left(u^{hj-1}\right)\operatorname{grad}\left(\psi_i\right)d\Omega+\frac{1}{2}\int_{S_3}\beta^j u^{hj}\psi_i dS_3+\frac{1}{2}\int_{S_3}\beta^{j-1}u^{hj-1}\psi_i dS_3+$$

$$(9)\quad +\frac{1}{\Delta t}\int_\Omega \sigma^j u^{hj}\psi_i d\Omega-\frac{1}{\Delta t}\int_\Omega \sigma^j u^{hj-1}\psi_i d\Omega=\frac{1}{2}\int_\Omega f^j\psi_i d\Omega+\frac{1}{2}\int_\Omega f^{j-1}\psi_i d\Omega+\frac{1}{2}\int_{S_2}\theta^j\psi_i dS_2+\frac{1}{2}\int_{S_2}\theta^{j-1}\psi_i dS_2+$$

$$+\frac{1}{2}\int_{S_3}\beta^j u_\beta^j\psi_i dS_3+\frac{1}{2}\int_{S_3}\beta^{j-1}u_\beta^{j-1}\psi_i dS_3$$

Решение u^h может быть представлено в виде: (10) $u^h=\sum_{k=1}^{n} q_k\psi_k$, причем $n-n_0$ компонент вектора весов q могут быть фиксированы и определены из условия $u^h\big|_{S_1}=u_g$. Подставляя (10) в (9) получим СЛАУ для вектора весов q:

$$\sum_{k=1}^{n}\left(\frac{1}{2}\int_\Omega \lambda^j \operatorname{grad}\left(\psi_k\right)\operatorname{grad}\left(\psi_i\right)d\Omega+\frac{1}{\Delta t}\int_\Omega \sigma^j\psi_k\psi_i d\Omega+\frac{1}{2}\int_{S_3}\beta^j\psi_k\psi_i dS_3\right)\cdot q_k^j=\frac{1}{2}\int_\Omega f^j\psi_i d\Omega+$$

$$(11)\quad +\frac{1}{2}\int_\Omega f^{j-1}\psi_i d\Omega+\frac{1}{2}\int_{S_2}\theta^j\psi_i dS_2+\frac{1}{2}\int_{S_2}\theta^{j-1}\psi_i dS_2+\frac{1}{2}\int_{S_3}\beta^j u_\beta^j\psi_i dS_3+\frac{1}{2}\int_{S_3}\beta^{j-1}u_\beta^{j-1}\psi_i dS_3+$$

$$+\sum_{k=1}^{n}\left(\frac{1}{\Delta t}\int_\Omega \sigma^j\psi_k\psi_i d\Omega-\frac{1}{2}\int_\Omega \lambda^j \operatorname{grad}\left(\psi_k\right)\operatorname{grad}\left(\psi_i\right)d\Omega-\frac{1}{2}\int_{S_3}\beta^{j-1}\psi_k\psi_i dS_3\right)\cdot q_k^{j-1}$$

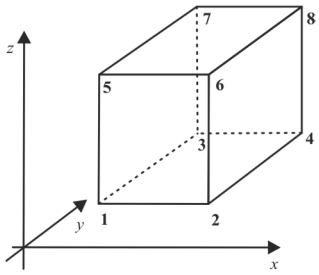
2.4. Переход к локальным матрицам

Введем локальную нумерацию узлов в соответствии с рисунком и определим базисные функции на конечном элементе. На отрезках $\left[x_i,x_{i+1}\right]$, $\left[y_j,y_{j+1}\right]$, $\left[z_k,z_{k+1}\right]$ зададим по две одномерные линейные функции:

$$(12.1)\qquad X_1(x)=\frac{x_{i+1}-x}{h_x},\ X_2(x)=\frac{x-x_i}{h_x},\ h_x=x_{i+1}-x_i,$$

$$(12.2)\qquad Y_1(y)=\frac{y_{j+1}-y}{h_y},\ Y_2(y)=\frac{y-y_j}{h_y},\ h_y=y_{j+1}-y_j,$$

$$(12.3)\qquad Z_1(z)=\frac{z_{k+1}-z}{h_z},\ Z_2(z)=\frac{z-z_k}{h_z},\ h_z=z_{k+1}-z_k.$$



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:

$$\psi_1=\frac{x_{i+1}-x}{h_x}\cdot\frac{y_{j+1}-y}{h_y}\cdot\frac{z_{k+1}-z}{h_z},\quad \psi_2=\frac{x-x_i}{h_x}\cdot\frac{y_{j+1}-y}{h_y}\cdot\frac{z_{k+1}-z}{h_z},\quad \psi_3=\frac{x_{i+1}-x}{h_x}\cdot\frac{y-y_j}{h_y}\cdot\frac{z_{k+1}-z}{h_z}$$

$$\psi_4=\frac{x-x_i}{h_x}\cdot\frac{y-y_j}{h_y}\cdot\frac{z_{k+1}-z}{h_z},\quad \psi_5=\frac{x_{i+1}-x}{h_x}\cdot\frac{y_{j+1}-y}{h_y}\cdot\frac{z-z_k}{h_z},\quad \psi_6=\frac{x-x_i}{h_x}\cdot\frac{y_{j+1}-y}{h_y}\cdot\frac{z-z_k}{h_z}$$

$$\psi_7=\frac{x_{i+1}-x}{h_x}\cdot\frac{y-y_j}{h_y}\cdot\frac{z-z_k}{h_z},\quad \psi_8=\frac{x-x_i}{h_x}\cdot\frac{y-y_j}{h_y}\cdot\frac{z-z_k}{h_z}\qquad (13)$$

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

(14) $\hat{G}_{ij} = \int\limits_{\Omega_k} \lambda_k \left(\frac{\partial \hat{\psi}_i}{\partial x}\frac{\partial \hat{\psi}_j}{\partial x} + \frac{\partial \hat{\psi}_i}{\partial y}\frac{\partial \hat{\psi}_j}{\partial y} + \frac{\partial \hat{\psi}_i}{\partial z}\frac{\partial \hat{\psi}_j}{\partial z} \right) dxdydz$ – матрица жесткости

(15) $\hat{M}_{ij} = \int\limits_{\Omega_k} \gamma_k \hat{\psi}_i \hat{\psi}_j dxdydx$ – матрица массы

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

$$\hat{G}_{ij} = \lambda_k \int_0^1\int_0^1\int_0^1 \left(\frac{\partial \hat{\psi}_i}{\partial x}\frac{\partial \hat{\psi}_j}{\partial x} + \frac{\partial \hat{\psi}_i}{\partial y}\frac{\partial \hat{\psi}_j}{\partial y} + \frac{\partial \hat{\psi}_i}{\partial z}\frac{\partial \hat{\psi}_j}{\partial z} \right) dxdydz \qquad \hat{M}_{ij} = \gamma_k \int_0^1\int_0^1\int_0^1 \hat{\psi}_i \hat{\psi}_j dxdydz$$

После вычислений итоговые матрицы примут вид:

(16) $\hat{G} = \frac{\lambda_k h_x h_y h_z}{36}\left(\frac{1}{h_x^2}\hat{G}^x + \frac{1}{h_y^2}\hat{G}^y + \frac{1}{h_z^2}\hat{G}^z \right)$ (17) $\hat{M} = \frac{\gamma_k h_x h_y h_z}{216}\hat{M}^1$

$$\hat{G}^x = \begin{pmatrix} 4 & -4 & 2 & -2 & 2 & -2 & 1 & -1 \\ -4 & 4 & -2 & 2 & -2 & 2 & -1 & 1 \\ 2 & -2 & 4 & -4 & 1 & -1 & 2 & -2 \\ -2 & 2 & -4 & 4 & -1 & 1 & -2 & 2 \\ 2 & -2 & 1 & -1 & 4 & -4 & 2 & -2 \\ -2 & 2 & -1 & 1 & -4 & 4 & -2 & 2 \\ 1 & -1 & 2 & -2 & 2 & -2 & 4 & -4 \\ -1 & 1 & -2 & 2 & -2 & 2 & -4 & 4 \end{pmatrix} \qquad \hat{G}^y = \begin{pmatrix} 4 & 2 & -4 & -2 & 2 & 1 & -2 & -1 \\ 2 & 4 & -2 & -4 & 1 & 2 & -1 & -2 \\ -4 & -2 & 4 & 2 & -2 & -1 & 2 & 1 \\ -2 & -4 & 2 & 4 & -1 & -2 & 1 & 2 \\ 2 & 1 & -2 & -1 & 4 & 2 & -4 & -2 \\ 1 & 2 & -1 & -2 & 2 & 4 & -2 & -4 \\ -2 & -1 & 2 & 1 & -4 & -2 & 4 & 2 \\ -1 & -2 & 1 & 2 & -2 & -4 & 2 & 4 \end{pmatrix}$$

$$\hat{G}^z = \begin{pmatrix} 4 & 2 & 2 & 1 & -4 & -2 & -2 & -1 \\ 2 & 4 & 1 & 2 & -2 & -4 & -1 & -2 \\ 2 & 1 & 4 & 2 & -2 & -1 & -4 & -2 \\ 1 & 2 & 2 & 4 & -1 & -2 & -2 & -4 \\ -4 & -2 & -2 & -1 & 4 & 2 & 2 & 1 \\ -2 & -4 & -1 & -2 & 2 & 4 & 1 & 2 \\ -2 & -1 & -4 & -2 & 2 & 1 & 4 & 2 \\ -1 & -2 & -2 & -4 & 1 & 2 & 2 & 4 \end{pmatrix} \qquad \hat{M}^1 = \begin{pmatrix} 8 & 4 & 4 & 2 & 4 & 2 & 2 & 1 \\ 4 & 8 & 2 & 4 & 2 & 4 & 1 & 2 \\ 4 & 2 & 8 & 4 & 2 & 1 & 4 & 2 \\ 2 & 4 & 4 & 8 & 1 & 2 & 2 & 4 \\ 4 & 2 & 2 & 1 & 8 & 4 & 4 & 2 \\ 2 & 4 & 1 & 2 & 4 & 8 & 2 & 4 \\ 2 & 1 & 4 & 2 & 4 & 2 & 8 & 4 \\ 1 & 2 & 2 & 4 & 2 & 4 & 4 & 8 \end{pmatrix}$$

f также раскладываем по базису: $f = \sum_{i=1}^{n} f_i \hat{\psi}_i$, тогда вектор правой части имеет вид (18) $F = \frac{h_x h_y h_z}{216}\hat{M}^1 \begin{bmatrix} f_1 \\ \cdots \\ f_n \end{bmatrix}$.

Глобальная матрица собирается путем добавления к ней локальных матриц в соответствующие места элементов. Профиль матрицы формируется следующим образом: $A_{ij} \neq 0$, если i и j 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 $\int\limits_{S_3} \beta u_\beta \psi_i dS$ и $\int\limits_{S_3} \beta \psi_j \psi_i dS$.

Положим параметр β константой на каждом конечном элементе, а функцию u_β разложим по базису $u_\beta = \sum_{i=1}^{n} u_{\beta i}\psi_i$.

В данном случае, базис получится вырожденным в билинейный, так как краевые условия заданы на гранях, т.е. одна из переменных фиксирована.

Вычислив интеграл $\int_{S_3} \beta \psi_j \psi_i dS$, получим локальную матрицу: $\hat{C} = \beta \frac{h_1 h_2}{36} \hat{C}^1$, где:

$$\hat{C}^1 = \begin{pmatrix} 4 & 2 & 2 & 1 \\ 2 & 4 & 1 & 2 \\ 2 & 1 & 4 & 2 \\ 1 & 2 & 2 & 4 \end{pmatrix}$$

Умножив $\hat{C}$ на $\begin{bmatrix} u_{\beta 1} \\ \cdots \\ u_{\beta n} \end{bmatrix}$ получим добавку в вектор правой части, а сама матрица $\hat{C}$ будет добавкой в глобальную матрицу (или правую часть, необходимо учесть (11)).

Рассмотрим краевые условия второго рода (4). Необходимо вычислить интеграл $\int_{S_2} \theta \psi_i dS$. 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 (LL^T -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 λ 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 x, y и z.

```
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 λ.
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 β

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 х 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 (LL^T -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 u
function f_(x, y, z, t)                     функция правой части уравнения
function sigma_(x, y, z, t)                 функция параметра σ
function grad_u_(x, y, z, t)                функция градиента u
```

4. Описание тестирования программы

4.1. Тест 1

Цель теста: проверка вклада $\mathrm{div}(\mathrm{grad})$

Параметры теста: $u = x^2 + y^2 + z^2 + t$, $f = -6$, $\lambda = 1$, $\sigma = 0$

Параметры области: $x \in [0,2]$, $h_x = 1$, $y \in [0,2]$, $h_y = 1$, $z \in [0,2]$, $h_z = 1$ (27 узлов)

Параметры времени: $t \in [0,1]$, $h_t = 0.1$ *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 σ

Параметры теста: $u = 2x + yt + z + 3t^2 + t - 2$, $f = \sigma(y + 6t + 1)$, $\lambda = 0$, $\sigma = 12$

Параметры области: $x \in [0,2]$, $h_x = 1$, $y \in [0,2]$, $h_y = 1$, $z \in [0,2]$, $h_z = 1$ (27 узлов)

Параметры времени: $t \in [0,1]$, $h_t = 0.1$ *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

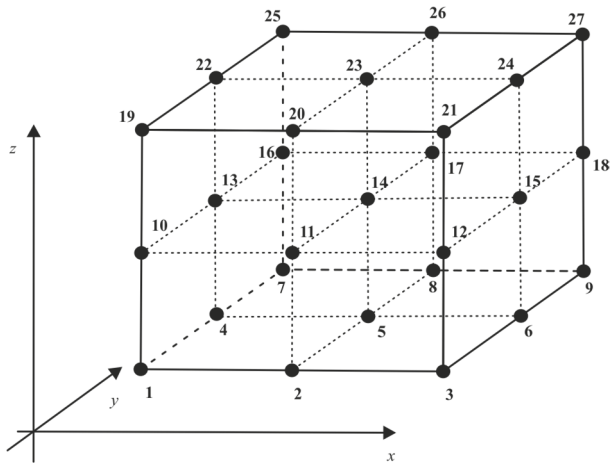
Параметры теста: $u = x^2 + y^2 + z^2 + t^2$, $f = 2\sigma t - 6$, $\lambda = 1$, $\sigma = 10^5$

Параметры области: $x \in [0,2]$, $h_x = 1$, $y \in [0,2]$, $h_y = 1$, $z \in [0,2]$, $h_z = 1$ (27 узлов)

Параметры времени: $t \in [0,1]$, $h_t = 0.1$

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: $u = 2x + yt + z^2 + x^2t^2 + 3t^2 + t - 2$, $f = \sigma\left(2tx^2 + y + 6t + 1\right) - 2t^2 - 2$, $\lambda = 1$, $\sigma = 1$

Параметры области: $x \in [0,10]$, $h_x = 0.5$, $y \in [0,10]$, $h_y = 0.5$, $z \in [0,10]$, $h_z = 0.5$ (9261 узел)

Параметры времени: $t \in [0,100]$, $h_t = 1$

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
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: $u = \sin\left(x + y + z + t^2\right)$, $f = 2\sigma t \cos\left(t^2 + x + y + z\right) + 3\sin\left(t^2 + x + y + z\right)$, $\lambda = 1$, $\sigma = 10^5$

Параметры области: $x \in [0,10]$, $h_x = 1$, $y \in [0,10]$, $h_y = 1$, $z \in [0,10]$, $h_z = 1$ (1331 узел)

Параметры времени: $t \in [0,1]$, $h_t = 0.05$

Краевые условия: везде первые

Результаты:

t	$\|u^* - u_{ht}\| / \|u^*\|$	$\|u^* - u_{ht/2}\| / \|u^*\|$	$\|u^* - u_{ht/4}\| / \|u^*\|$	$\log_2\left(\Delta_{ht} / \Delta_{ht/2}\right)$	$\log_2\left(\Delta_{ht/2} / \Delta_{ht/4}\right)$
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 σ 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 х 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 C1*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
```