Anes Team

ANES/20XE : Код для численного моделирования процессов гидродинамики и тепломассообмена

Версия 2.24

Описание численных алгоритмов кода

Москва 2019 г.

Оглавление.

Обозначения

Описание моделей пакета - это основной текст документа.

ⓘ *Описание в проекте прикладной задачи :*
Использование Ф-переменной в задаче определяется оператором - эта вставка используется для ссылки на AIL-операторы файла описания проекта.

Введение

В данном документе описываются конечно-разностные алгоритмы кода Anes. Подробности математических моделей изложены в документе [1].

Для численного решения уравнений переноса Ф-переменных в коде Anes используется метод контрольного объема (МКО). В этом методе расчетная область (РО) разбивается на множество ячеек или контрольных объемов (КО) и с каждым КО связывается значение Ф-переменной. В итоге непрерывное трехмерное поле Ф-переменной заменяется конечным массивом Ф-значений. Способ построения ячеек зависит от типа используемых сеток КО. В текущей версии кода можно использовать *четыре типа* сеток:

1) структурные сетки КО (СС) в декартовой системе координат ($x = x_c$, $y = y_c$, $z = z_c$),
2) структурные сетки КО в цилиндрической системе координат ($x = r$, $y = \theta$, $z = z_c$),
3) неструктурные сетки (НС) с локальным дроблением в декартовой системе координат ($x = x_c$, $y = y_c$, $z = z_c$),
4) неструктурные сетки с локальным дроблением в *двумерной* цилиндрической системе координат ($x = r$, $z = z_c$).

Здесь (x,y,z) - обобщенные координаты Anes, (x_c ,y_c, z_c) - декартовые координаты, (r, θ, z_c) – трехмерные цилиндрические координаты (см. документ [1]).

Код Anes позволяет решать стационарные или нестационарные задачи в одномерной (1D) двухмерной (2D)- или трехмерной (3D) постановке. *Размерность задачи* «фиксируется» при разбивке расчетной области на контрольные объемы (КО). Для 2D постановки по одной из координат число КО должно быть равным единице. Для 1D задач число КО по двум осям равно единице.

Для решения конкретной прикладной задачи можно использовать и структурные и неструктурные сетки. Выбор типа сетки зависит от многих параметров. Перечислим основные особенности сеток:

1) при использовании СС поля переменных хранятся в виде трехмерных массивов, что удобно для обработки результатов (в НС поля хранятся в виде одномерных списков);
2) при использовании СС для моделирования компонент скорости используются сдвинутые шахматные сетки, что дает более точное описание течения при наличии пористых зон (в НС используется модель совмещенных скоростей, которая требует специальных алгоритмов расчета массовых потоков на гранях ячеек);
3) в НС для описания сложных границ расчетной области (РО) используется модель дробных ячеек, а ячейки расположенные вне РО удаляются; в СС для описания границ используется модель целых ячеек, а ячейки вне расчетной области переводятся в категорию блокированных ячеек и остаются в РО;
4) параллельные алгоритмы расчета более эффективны при использовании неструктурных сеток;

ⓘ *Описание в проекте прикладной задачи :*
Выбор сетки КО определяется оператором
TypeMesh = TM_CARTES / TM_CYLIND / TM_UnCartes / TM_UnCylind
секции [Main] проекта.

1. Дискретизация РО. Структурные сетки

Структурные сетки являются наиболее простым и «очевидным» способом разбиения РО на ячейки. В этом подходе массив значений Ф-переменных описывается в виде трехмерного массива, а сами контрольные объемы нумеруются с помощью трех одномерных индексов (ix,iy,iz).

1.1 Сетка контрольных объемов

В Anes используется *регулярная прямоугольная ортогональная* сетка КО. Для построения 3D сетки КО используется три одномерные системы сеток по трем обобщенным координатным осям. Рассмотрим построение одномерной сетки для оси x.

На первом этапе строится сетка граней КО (см. рис. 1.1), координаты которых хранятся в одномерном массива пакета

xFace(ix) , ix = 2 … NX

здесь NX - число КО + 2.

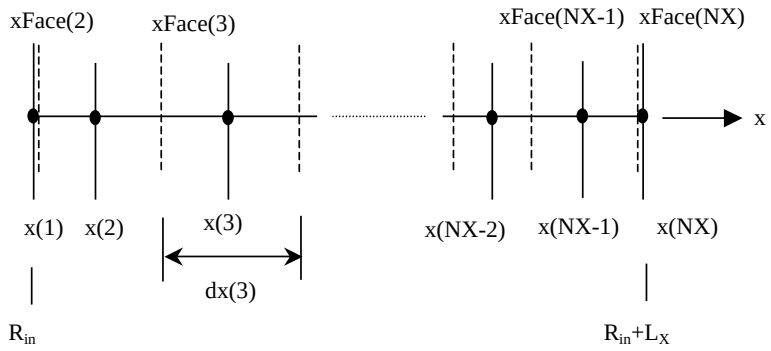


Рисунок 1.1 - Семейство сеток КО по оси x

На втором этапе строится сетка «основных» узлов, расположенных в центрах КО, с координатами

x(ix) = [(xFace(ix) + xFace(ix+1)]/2 , ix = 2 … NX-1

На границе расчетной области помещаются дополнительные основные узлы (*граничные узлы*), имеющие координаты x(1) и x(NX) (можно считать, что на границах расположены два дополнительных КО нулевой толщины). Поэтому переменная NX равна числу основных узлов, а число КО по оси x равно NX-2. Аналогичные построения выполняются для координат y и z. Граничные узлы не участвуют в процессе расчета, они используются только для *однотипного* описания границы расчетной области, образованной границей базовой РО и поверхностями Block-патчей, и *хранения* в них граничных значений Ф-переменных.

Типичный основной КО со своими КО - соседями изображен на рис. 1.2. На этом же рисунке приводятся принятые ниже обозначения для граней КО и узлов сетки. Центральный узел обозначен символом P, соседние узлы - символами N, S, E, W, H, L. Точки, лежащие в геометрических центрах граней, обозначенные также символами n, s, e, w, h, l, являются центральными узловыми точками смещенных (вдоль одной из координат) КО, используемых, в частности, для расчета соответствующих компонент вектора скорости.

Для контрольного объема с индексами (ix,iy,iz) , указанные на рис. 1.2 точки имеют следующие координаты и индексы:

P - (x(ix),y(iy),z(iz)),
e - (xFace(ix+1),y(iy),z(iz)),
w - (xFace(ix),y(iy),z(iz))
n - (x(ix),yFace(iy+1),z(iz)),
s - (x(ix),yFace(iy),z(iz))
h - (x(ix),y(iy),zFace(iz+1)),
l - (x(ix),y(iy),zFace(iz)).

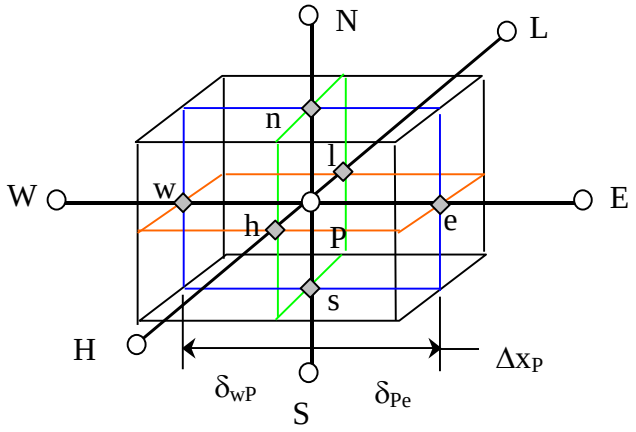


Рисунок 1.2 - Типичный структурный основной КО

Отсутствующие в математическом описании координаты в 2D и 1D задачах будем называть *вырожденными*. По вырожденной координате сетка строится иначе (см. рис. 1.3).

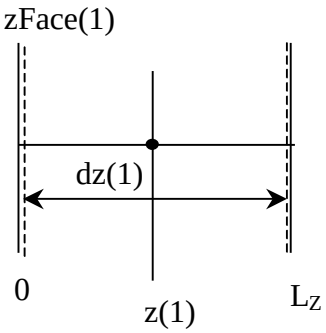


Рисунок 1.3 - Сетка с одним КО по вырожденной координате z

В этом случае, как уже упоминалось выше, имеется всего один КО по этой координате, все сеточные массивы содержат один элемент, а NZ = 1.

1.2 Построение сеток КО в коде Anes

Сетка КО в Anes строится для *базовой расчетной области* (БРО, см. [1]), которая всегда представляет собой параллелепипед в обобщенных координатах. Для построения сеток граней КО в коде используется следующий алгоритм. Вся зона БРО вдоль оси x = 0 ... L_x (или осей y или z) разбивается на произвольные последовательные зоны - ΔL_m:

$$L_X = \sum \Delta L_m \qquad (1.1)$$

Для каждой зоны строится своя сетка КО с использованием одного из четырех алгоритмов:

1) "Sym" - логарифмическая симметричная сетка *граней* КО;
2) "ASym" - логарифмическая несимметричная сетка *граней* КО;
3) "DNS_COS" - несимметричная сетка *граней* КО, распределенных по закону косинуса;
4) "DNS_TANH" - несимметричная сетка *граней* КО, распределенных по закону гиперболического тангенса.

Для первых двух алгоритмов для каждой зоны задается число КО в зоне N_{CV} и параметр логарифмичности q_L. Если $q_L = 1$, то в зоне строится равномерная сетка КО с постоянной шириной КО:

$$\Delta x_{ix} = \frac{\Delta L_m}{N_{CV}}.$$

Если используется *ассиметричный* алгоритм построения сетки "ASym", то создаются КО разной ширины, удовлетворяющие следующему соотношению:

$$\Delta x_{ix+1} = q_L \cdot \Delta x_{ix}$$

Это позволяет «сгустить» (уменьшить ширину КО) ячейки сетки к началу ($q_L > 1$) или к концу ($q_L < 1$) зоны. При использовании *симметричного* алгоритма построения "Sym" сгущение при $q_L > 1$ осуществляется симметрично к границам зоны (при $q_L < 1$ к ее центру).

Алгоритмы "DNS_COS" и "DNS_TANH" строят сильно неоднородную сетку граней КО со сгущением к началу зоны ($q_L > 0$) или к концу ($q_L < 0$) по следующим соотношениям. Для "DNS_COS":

$$xFace_i = \begin{cases} x_{beg} + \Delta L_m \left[1 - \cos(\alpha_i)\right], & q_L > 0 \\ x_{beg} + \Delta L_m \sin(\alpha_i), & q_L < 0 \end{cases}$$

$$\alpha_i = \frac{\pi}{2} \frac{i-1}{N_{cv}-1}, \quad i = 1, \ldots N_{cv}+1$$

Здесь x_{beg} - координата начала зоны вдоль оси, i - относительный индекс грани в зоне (i=1 соответствует грани с координатой x_{beg}).

Для алгоритма "DNS_TANH" используется аналогичная зависимость:

$$xFace_i = \begin{cases} x_{beg} + \Delta L_m \left[1 - \dfrac{\tanh(\alpha_i)}{\tanh(q_L)}\right], & q_L > 0 \\ x_{beg} + \Delta L_m \dfrac{\tanh(\alpha_i)}{\tanh(|q_L|)}, & q_L < 0 \end{cases}$$

$$\alpha_i = q_L \frac{i-1}{N_{cv}-1}, \quad i = 1, \ldots N_{cv}+1$$

Эти алгоритмы в отличие от логарифмических сеток позволяют создать вблизи левой границы зоны КО с большими значениями коэффициента логарифмичности, при этом это коэффициент вблизи правой границы будет близок к 1. Основное назначение таких алгоритмов - LES и DNS моделирование турбулентности.

Заметим, что работе с "DNS_COS" используется только знак q_L.

При построении сетки по оси x в цилиндрической системе координат к координатам граней КО зон прибавляется значение, заданное параметром R_{in}.

☞Замечание #1. В коде Anes размеры базовой расчетной области явно не задаются! Размеры рассчитываются по соотношениям (1.1).

☞Замечание #2. При решении ряда задач, например, при использовании в задаче гибридных моделей турбулентности, требуется построить сетку так, чтобы в центре канала она была равномерная, а у стенки «быстро» сгущалась. При этом необходимо, чтобы на границе центральной зоны и зоны сгущения размеры КО были одинаковые. Это легко сделать, если задать число КО в центре - N_c, число КО в пристенной зоне - N_w и степень сгущения в близи стенки - q_w. В этом случае легко рассчитать размер пристенной зоны по соотношению:

$$L_w = L\frac{1}{N_c q^{N_w-1}\dfrac{1-q}{1-q^{N_w}}+1},$$

где L - общая длина центральной и пристенной зон. Зная этот размер можно построить сетку, используя два оператора Zone.

ⓘ *Описание в проекте прикладной задачи :*
Зоны сеткок КО описываются в секциях [XR Grid] , [YFI Grid] и [ZZ Grid] с помощью операторов
Zone(ΔL,NoCV,qL,ASym/ Sym/ DNS_COS/ DNS_TANH)
Значения начала БРО по оси x для цилиндрической системы координат задается оператором
Rin = 0.0
секции [XR Grid].

1.3 Формирование расчетной области

Главной особенностью структурной сетки (и ее главным «минусом») является использование модели блокированных КО для формирования сложной геометрии расчетной области. При использовании общего алгоритма построения (см. [1]):

$$РО = БРО - \sum_m (Block - патчи)_m,$$

контрольные объемы, «попавшие» внутрь Block-патчей, *не удаляются* из РО, они просто помечаются как блокированные с помощью специального флага IsBlock. Установка этого флага и есть собственно построение РО для структурной сетки.

В коде для структурных сеток используется алгоритм «целых ячеек», заключающийся в следующем:

1) все КО, расположенные внутри объема патча Block/BlockWall (все восемь вершин КО находятся внутри объема) помечаются флагом IsBlock;
2) КО, которые «разрезаются» поверхностями, ограничивающими патч (часть вершин КО лежит вне фигуры), включаются в состав объекта и помечаются как блокированные, если объем объекта в КО превышает 50% объема КО (см. рис. 1.4),
3) границей РО становится совокупность граней КО, соединяющих ячейки с флагом и без флага IsBlock.

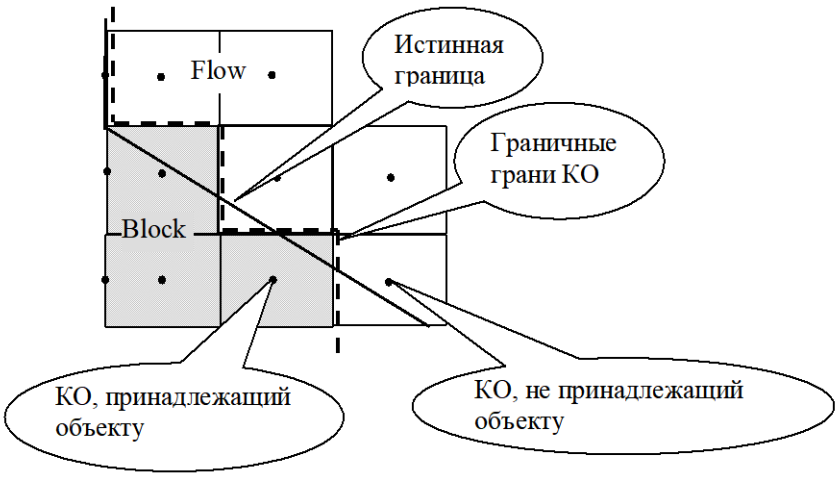


Рисунок 1.4 - Моделирование границ расчетной области для алгоритма «целых КО» структурных сеток

Аналогичным образом обрабатываются все остальные патчи (Struct, Flow и Porous). В итоге объемные патчи в Решателе будут преобразованы в совокупность КО, а поверхностные - в совокупность граней КО. Заметим, что патчи в файле проекта и патчи в Решателе могут отличаться, особенно при наличии «ошибок» преобразования генератором сетки Anes. Обе группы патчей можно просмотреть в Постпроцессоре кода Anes. В нем патчи, описанные в файле проекта называются *патчами Компилятора (или патчами Проекта)*, а патчи во внутреннем сеточном представлении (совокупности либо КО, либо граней КО) называются *патчами Решателя*.

1.4 Определение Ф-переменных

Зависимая переменная $\Phi(x,y,z,\tau)$ (давление, компоненты вектора скорости, температура и т.п.) представляет собой 3D поле. При переходе к структурным сеткам этому полю соответствует дискретный набор значений Ф-переменной, которые можно представить трехмерным массивом F(ix,iy,iz). В коде Anes все Ф-переменные хранятся в виде одного четырехмерного массива Фортрана F8(1:NX,1:NY,1:NZ,iPHI), где iPHI – индекс Ф-переменной.

Все Ф-переменные, *кроме скоростей*, и все вспомогательные переменные (например, плотность, вязкость и т.п.) определены в основных узлах (центрах основного КО), т.е. в точке с координатами {x(ix), y(iy), z(iz)}. Компоненты вектора скорости G-фазы ($\mathbf{U}_g$) определены в узловых точках, лежащих *на гранях* основного КО:

```
UgX(ix,iy,iz) = F8(ix,iy,iz,P_UGX)      - в точке { xFace(ix),y(iy),Z(iz) },
UgY(ix,iy,iz) = F8(ix,iy,iz,P_UGY)      - в точке { x (ix),yFace(iy),Z(iz) },
UgZ(ix,iy,iz) = F8(ix,iy,iz,P_UGZ)      - в точке { x (ix),y(iy),ZFace (iz) }.
```

Исключением из этого правила является случай вырожденных координат. Если одна из координат вырождена, то компонента скорости в направлении этой координаты определена в основном узле с индексом 1, как и обычная Ф-переменная.

Для скоростей начальные индексы, равные 1 не используются (в элементах массива расположены нули). Например, для скорости UgX скорость на левой грани задана для ix = 2.

При использовании нерегулярных расчетных областей значения Ф-переменной в КО, принадлежащих Block-зонах (КО, помеченные флагом IsBlock) лишены смысла и полагаются равными нулю.

2. Дискретизация РО. Неструктурные сетки

Для детального описания сложной геометрии РО наиболее подходящим является разбиение РО на ячейки произвольной формы. Такие ячейки уже нельзя пронумеровать с помощью структурных индексов, поэтому такие сетки называются неструктурными. Главный «недостаток» неструктурных сеток с ячейками произвольной формы (далее такие сетки будут называться *полностью* неструктурными) заключается в проблеме генерации эффективной сетки для произвольной геометрии РО.

В коде Anes реализованы неструктурные сетки специальной структуры, которые будут называться неструктурными сетками с локальным дроблением (НСЛД). В этих сетках ячейки имеют форму параллелепипеда в обобщенной системе координат (за исключением приграничных ячеек), однако размеры ячеек могут отличаться на степень двойки.

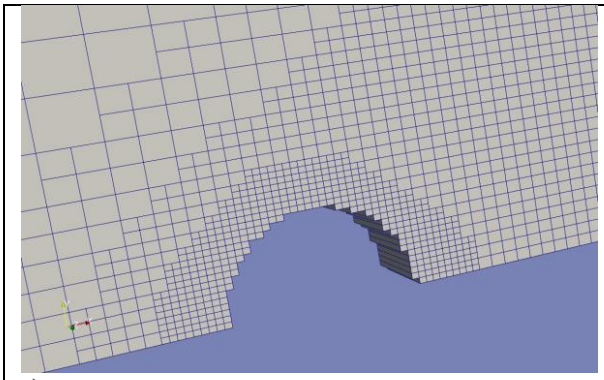
Важной особенностью НСЛД сетки является то, что она представляет собой эффективный компромисс между двумя крайностями: структурными сетками и полностью неструктурными. Это позволяет с одной стороны работать с сеткой ячеек, используя только два неструктурных списка – ячеек и их граней. С другой стороны, с ячейками можно связать структурные индексы самой мелкой сетки (FineMesh), что существенно ускоряет получение геометрических характеристик ячеек и граней. Другое важное отличие НСЛД сеток от полностью неструктурных – возможность «автоматизации» построения сетки. Для построения такой сетки пользователь должен задать исходную грубую (CoarseMesh) сетку, расположить в расчетной области геометрические объекты, задать максимальный уровень разбиения MaxLevel и все. После этого генератор сетки компилятора Anes автоматически создаст сетку. Время генерации сетки даже для уровней разбиения 6-8 не превышает нескольких минут, причем это время практически не зависит от сложности геометрии расчетной области.

Идея построения НСЛД сетки основана на локальном измельчении ячеек первоначально структурной декартовой сетки КО. Суть этого подхода достаточно проста. Сначала строится грубая структурная декартовая сетка (иногда состоящая из одного КО), потом производится дробление выбранных КО на 4 КО (в двухмерном случае) или на 8 КО (в трехмерном). Процедура дробления повторяется до тех пор, пока не будет достигнут заданный уровень разбиения. Типичный пример такой сетки показан на рисунке 2.1.

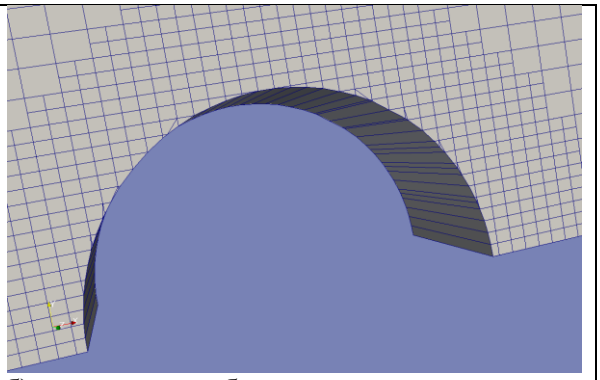
Для описания границы РО при построении сетки наряду с алгоритмом целых ячеек (являющимся базовым для структурных сеток) в НСЛД реализован алгоритм дробных ячеек (см. рисунок 2.1б).

Неструктурные сетки можно использовать для описания 2D и 3D задач в декартовой системе координат и для описания осесимметричных 2D (r,z) задач в цилиндрической системе. Трехмерные задачи в цилиндрической системе координат не реализованы, поскольку их легко реализовать в 3D декартовой системе.

Построение неструктурной сетки осуществляет генератор сетки, который является частью Компилятора проекта кода.



а) алгоритм целых ячеек



б) алгоритм дробных ячеек

Рисунок 2.1 – Неструктурная сетка кода Anes

2.1 Алгоритм построения неструктурных сеток

Создание неструктурной сетки начинается с построения обычной структурной сетки БРО. Эта сетка в дальнейшем рассматривается в качестве начальной *грубой* сетки (CoarseMesh). Отметим сразу, что эта сетка представляет собой обычную структурную сетку и для ее построения используются алгоритмы, описанные в предыдущем разделе.

В качестве исходной информации для генерации неструктурной сетки используется эта грубая структурная сетка и набор патчей, влияющих на процесс дробления сетки. В качестве таких патчей в текущей версии кода используются:

1) Block, Struct, Flow и Porous патчи (далее эти патчи будут называться Blockage-патчи);
2) 2D BSwall поверхностные патчи;
3) 2D BS поверхностные патчи;
4) Volume объемные патчи, для которых указан дополнительный параметр FixLevel.

Пользователь может управлять процессом построения сетки, задав следующие параметры:

1) MaxLevel – максимальный уровень дробления сетки (исходная грубая сетка соответствует нулевому уровню) вблизи Blockage-патчей; если этот уровень равен нулю, то процесса дробления не производится и фактически неструктурная сетка геометрически эквивалентна структурной.
2) NoLayers – число слоев ячеек одного уровня для сглаживания перехода между уровнями (по умолчанию = 2);
3) BSLevel – *минимальный* уровень дробления ячеек вблизи BS-патчей;
4) BSWallLevel – *минимальный* уровень дробления ячеек вблизи BSWall-патчей;
5) значение FixLevel для VOLUME патча (если необходимо),
6) флаг IsNoRefZ, отключающий процесс дробления ячеек в направлении оси z для трехмерных задач.

Для построения неструктурной сетки используются следующие достаточно простые правила:

1) вблизи Blockage-патчей по окончанию дробления должны располагаться ячейки с MaxLevel уровнем дробления;
2) вблизи BS-патчей должны быть ячейки с уровнем дробления не меньшим, чем BSLevel;
3) вблизи BSWAll-патчей должны быть ячейки с уровнем дробления не меньшим, чем BSWallLevel;
4) ячейки, расположенные внутри Block-патчей *удаляются* из сгенерированной сетки;

5) ячейки одного уровня должны располагаться слоями: для любой ячейки в любом направлении должны располагаться NoLayers ячеек ее уровня;
6) поверхность раздела ячеек разного уровня должна быть достаточно гладкой: для любой ячейки число соседей *другого* уровня не должно превышать 3 в двумерных задачах и 4 в трехмерных;
7) не допускается соседство ячеек, уровни которых отличаются больше чем на единицу.
8) ячейки, попадающие в Volume-патчи с заданным FixLevel, должны иметь уровень дробления не меньший чем FixLevel,
9) при использовании алгоритма дробных ячеек КО вблизи Blockage-границ «усекаются» поверхностью границы.

Процесс построения сетки состоит из несколько шагов. Информация о каждом шаге выводится в консоль выполнения Компилятора проекта и в его листинг. При построении возможно появление ошибок. Ниже эти шаги рассмотрены более подробно, чтобы пояснить смысл возможных ошибок и способы их ликвидации. При описании этих шагов будут приводиться «кусок» листинга компилятора проекта, связанного с шагом.

Шаги работы генератора неструктурной сетки следующие:
1) построение грубой структурной сетки;
2) сканирование всех патчей с помощью алгоритма лучей;
3) построение дерева ячеек с помощью алгоритма дробления;
4) обработка дробных ячеек;
5) создание списка неструктурных ячеек;
6) создание списка неструктурных граней;
7) проверка правильности построения ячеек и граней;
8) создание списка неструктурных вершин;
9) создание списка неструктурных патчей Решателя;
10) создание выходных файлов для Решателя;
11) создание VTK-файлов для анализа сетки в ParaView,
12) Проверка качества построенной неструктурной сетки.

2.2 Построение грубой сети

Начальная грубая сетка CoarseMesh строится так же, как и структурная сетка. Эта сетка является вспомогательной, в процессе построения неструктурной сетки ячеек и внутри Решателя используется другая сетка, которая называется мелкой сеткой – FineMesh. Эта сетка получается из CoarseMesh путем разбиения всех ее КО на $2^{MaxLevel}$ одинаковых ячеек. В Решателе мелкие сетки описываются набором трех одномерных массивов, например для оси x:

usX(1:NFX) - центр КО,
usXF(1:NFX) - координата левой грани КО,
usXL(1:NFX) - координата правой грани КО.

здесь NFX – число ячеек FineMesh по оси x. Заметим, что в неструктурных сетках используется немного другой принцип нумерации, чем в структурных: на границе БРО нет граничных узлов, поэтому индекс левой приграничной ячейки равен 1 (в структурной сетке он равен 2).

В дальнейшем тройка индексов FineMesh (ifx, ify,ifz) будут называться структурными Fine-индексами.

2.3 Сканирование патчей с помощью алгоритма лучей

Для работы процесса дробления ячеек необходимо быстро получать следующую информацию для ячейки любого уровня:

1) содержит или нет ячейка пересечение с поверхностью одного из патчей;

2) если пересечение есть, то необходимо запомнить имя этого патча, чтобы в дальнейшем можно было связать с ячейкой граничные условия (или источники);
3) если пересечения нет, то необходимо запомнить имя материала G- или S- фазы для данной ячейки; если ячейка расположена внутри Blockage-патча, то ее нужно пометить для дальнейшего удаления.

Для быстрой проверки прохождения поверхности объема патча через любую ячейку любого уровня в генераторе сетки используется алгоритм *лучевого сканирования* патчей, участвующих в построении сетки. Кратко суть этого алгоритма сводится к следующему.

Луч – это линия, направленная вдоль одной из осей x,y,z и проходящая через центр КО FineMesh (для модели целых ячеек) или через ребро КО (для модели дробных ячеек). При сканировании на луче рассчитываются и запоминаются точки пересечения (и их характеристики) для всех объектов-патчей, участвующих в построении сетки.

В процессе сканирования лучи проводятся через *все* КО FineMesh в трех направлениях. Патчи сканируются последовательно и для каждого луча проводится суммирование точек пересечения с использованием алгоритма перекрытия. Идея этого алгоритма демонстрируется на рисунке 1.6 на примере двух Block-патчей.

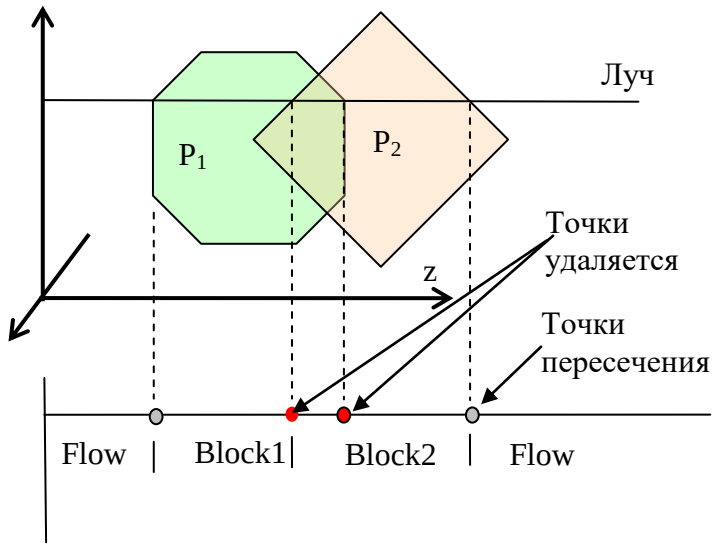


Рисунок 2.2 – Алгоритм перекрытия объектов

При сканировании патча P_1 на луче были отмечены две точки пересечения. После сканирования второго патча P_2 на луче остаются две новые точки пересечения, поскольку две внутренние точки удаляются алгоритмом перекрытия.

Список патчей, используемых на этапе сканирования, приводится в листинге Компилятора проекта:

```
------------(Патчи компилятора)------------
#   Name          Type   TypeObject    NoElement
  1 ROD0          Block(Wall)  2DBody    63
  2 ROD1          Block(Wall)  2DBody    63
  3 ROD2          Block(Wall)  2DBody    63
  4 TRI           Block(Wall)  2DBody     3
  5 In            BS           Plate      0
  6 Out           BS           Plate      0
  7 Wall          BS(Wall)     Plate      0
```

Если процесс сканирования прошел удачно, то Компилятор запишет в свой листинг (и на консоль) следующие сообщения:

```
.Начинаем: Сканирование лучей ...
Сканирование в Z-направлении, патч= ROD0
Сканирование в Y-направлении, патч= ROD0
Сканирование в X-направлении, патч= ROD0
Сканирование в Z-направлении, патч= ROD1
Сканирование в Y-направлении, патч= ROD1
Сканирование в X-направлении, патч= ROD1
Сканирование в Z-направлении, патч= ROD2
Сканирование в Y-направлении, патч= ROD2
Сканирование в X-направлении, патч= ROD2
Сканирование в Z-направлении, патч= TRI
Сканирование в Y-направлении, патч= TRI
Сканирование в X-направлении, патч= TRI
Сканирование в Z-направлении, патч= In
Сканирование в Z-направлении, патч= Out
Сканирование в Y-направлении, патч= Wall
.Закончили: Сканирование лучей, time=000.00.00:640
```

После окончания сканирования всех патчей производится проверка правильности сканирования. Если при сканировании были ошибки, то будут выведены сообщение об их числе.

Чаще всего причиной ошибок является «неправильная» структура описания 2DBody (файлы *.2gr) или 3DBody объектов (файлы *.tri и *.stl). Точки многоугольника 2DBody должны обходиться *против часовой стрелки*, а треугольники-фасеты Body должны иметь нормаль, направленную вне объекта (нормали фасетов stl-файлов можно «просмотреть» в постпроцессоре ParaView) . Например, типичные ошибки для двумерного объекта показаны на рисунке 2.3:

1) фасеты объекта не образуют замкнутую фигуру: в этом случае число точек пересечения может быть нечетно (рис. 1.7(а));
2) внутри или вне объекта имеются одиночные фасеты (рис. 1.7(б));
3) нормаль одной из грани направлена неправильно (точки фасета указаны не против часовой стрелки) – рис. 1.7 (с);

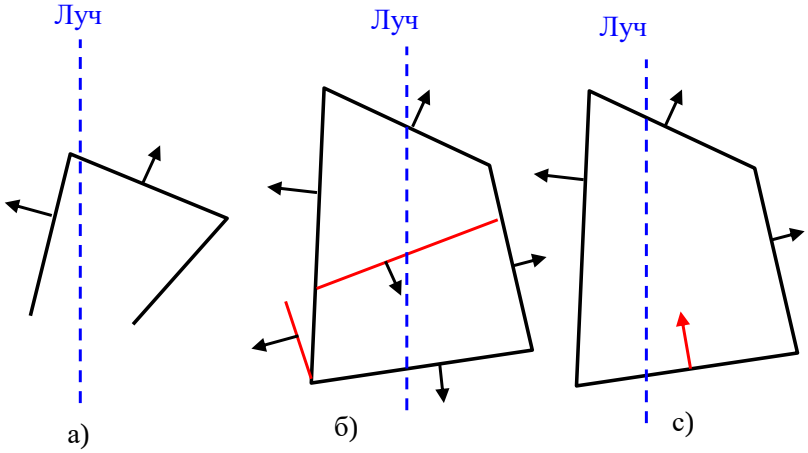


Рисунок 2.3 – Ошибки сканирования

Вторая группа ошибок сканирования связана с ошибками округления в случае, когда объекты *касаются* друг друга или внешней границы БРО. В этом случае решение проблемы – смещение одного из объектов на малую величину (для внешней границы – смещение объекта *вне* БРО).

Проблему касания объектов можно решить, если поварьировать еще одним параметром генератора сеток

```
EQUTOL = 1.e-5
```

Этот оператор задает относительную точность сравнения точек пространства в алгоритме сканирования лучей.

Для выяснения причины ошибок сканирования можно использовать специальные операторы файла проекта, описанные в [2]. Например, если в процессе сканирования Blockage-патча были выданы ошибки, то можно «визуализировать» фасеты объекта, которые вызвали ошибки. Для этого нужно в секции [Unstructure Cartesian Grid] указать оператор

```
DbgTestScan = .TRUE.
```

Информация об первых трех ошибках сканирования в каждом направлении будет выдана в LOG-файл Компилятора и будут созданы stl-файлы с ошибочными фасетами объекта:

```
rays_<направление луча>_<номер ошибки>.stl
```

В LOG-файл Компилятора будет выведена информация о текущем луче и его наложении в конечный луч, а в stl-файле будут записаны «неправильные» фасеты (этот файл можно посмотреть в ParaView, в частности можно посмотреть нормали фасета).

При использовании модели дробных ячеек производится дополнительная коррекция пересечений для уменьшения числа слишком маленьких дробных ячеек. Для этого положение точек пересечений на всех лучах сравнивается с вершинами параллелепипедов ячеек FineMesh. Если точки пересечения расположены «вблизи» вершины, то они переносятся в эту вершину. В качестве критерия «близости» используется условие

$$\xi_c = \frac{|S_c - S_v|}{\Delta S} < \xi_{cri}$$

Здесь

S_c – координата точки пересечения,

S_v – координата вершины Fine-ячейки,

ΔS – ширина ячейки в направлении луча,

ξ_{cri} – критическое значение. Это значение задается пользователем в файле проекта через переменную CriFracEdge. По умолчанию $\xi_{cri} = 0.1$.

Информация об этом этапе выводится в листинг Компилятора и в консоль выполнения в виде:

```
.Начинаем: Проверка правильности сканирования лучей ...
.Закончили: Проверка правильности сканирования лучей, time=000.00.00:047

.Начинаем: Коррекция лучей для дробных ячеек ...
No. of moving intersections=295797
.Закончили: Коррекция лучей для дробных ячеек, time=000.00.00:203
```

2.4 Построение дерева ячеек с помощью алгоритма дробления

На третьем шаге производится процесс дробления исходных CoarseMesh ячеек. Процесс дробления производится по следующему алгоритму:

1. При разбиении ячейки создаются восемь (для 3D задачи) или четыре (2D задачи) одинаковые ячейки. Если флаг IsNoRefZ = .TRUE., то ячейка разбивается на 4 ячейки (дробление проводится только в направлении осей x и y, так же, как и в случае 2D задачи с NZ = 1).
2. В процессе разбиения создается дерево ячеек, начиная с ячеек грубой сетки. Для этого при разбиении ячейки ее индекс запоминается в ячейках-потомках. Глубина дерева

равна MaxLevel. Дерево внутри генератора сетки используется для быстрого нахождения соседей для любой ячейки при просмотре всех низовых ячеек.

3. Ячейка требует дробления, если:
   - поверхность любого объекта проходит через ячейку и еще не достигнут нужный уровень разбиения (MaxLevel, BSLevel, BSWallLevel,FixLevel);
   - ячейка помечена на разбиение алгоритмами сглаживания и создание слоев;
4. Процесс дробления осуществляется в цикле, начиная с ячеек грубой сетки. Процесс прекращается, когда ячейки, помеченные для разбиения, отсутствуют.

По окончанию процесса дробления получается дерево ячеек, в котором низовые ячейки с уровнем дробления MaxLevel содержат пересечения с объектами (точнее в этих ячейках находятся точки пересечения лучей). Это так называемые «приграничные» ячейки.

Для обработки этих «приграничных» ячеек используются два алгоритма:

1) алгоритм целых ячеек,
2) алгоритм дробных ячеек.

Если используется алгоритм целых ячеек, то «приграничные» ячейки остаются прямоугольными и процесс дробления заканчивается. Если используется модель дробных ячеек, то запускается шаг создания дробных ячеек.

2.5 Обработка дробных ячеек

После процесса дробления имеется список «приграничных» ячеек, которые содержат точки пересечения с границами объектов. Эти точки пересечения сохранены в виде точек пересечения с ребрами ячейки. Текущий генератор сеток может обрабатывать ячейки, которые содержат только по *одному пересечению* на ребре (не считая пересечений в вершинах ячейки) – см. рисунок 2.4.

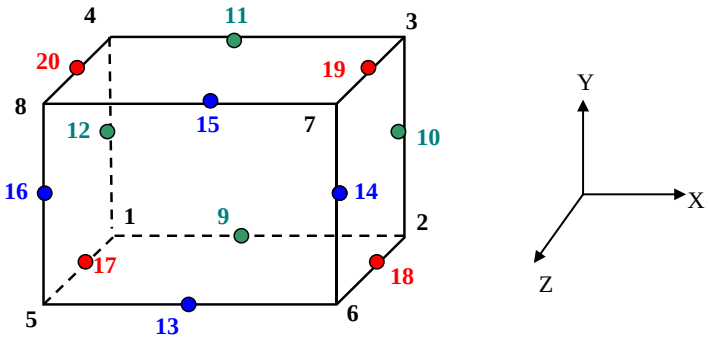


Рисунок 2.4 – Допустимые точки пересечения в дробной ячейке

Если геометрия пересечений удовлетворяет этому условию, то из целой ячейки создается одна (для объектов типа Block) или две (для объектов типа Flow или Struct) дробные ячейки типа Cut-Cell (типичные примеры показаны на рисунке 2.5).

Если генератор не может построить дробную ячейку (типичная причина - многократные пересечения на ребрах), то ячейка преобразуется в целую Whole-cell ячейку. Такие «преобразованные» Whole-cell ячейки можно увидеть в постпроцессоре ParaView, загрузив файл с описанием ячеек

<Префикс файлов результатов>_cells.vtk.

Отчет о создании дробных ячеек выдается в листинг компиляции в виде:

```
.Начинаем: Процесс дробления КО ...
 No. CUT Cells.....= 568
!WARNING: No. converted CUT to WHOLE cells=372
.Закончили: Процесс дробления КО, time=000.00.00:028

 ----------(Результат дробления)----------
 1.Число Coarse структурных КО.....= 4000
 2.Число Fine структурных КО.....= 64000
 3.Число неструктурных КО.....= 6352
 4.Число блокированных КО.....= 2646
```

В отчете указывается число дробных ячеек, число «конвертированных» дробных ячеек в целые ячейки (Cut to Whole cells), число структурных ячеек на Coarse и Fine уровнях, окончательное число неструктурных ячеек и число блокированных ячеек, которые в дальнейшем будут удалены из списков неструктурных ячеек.

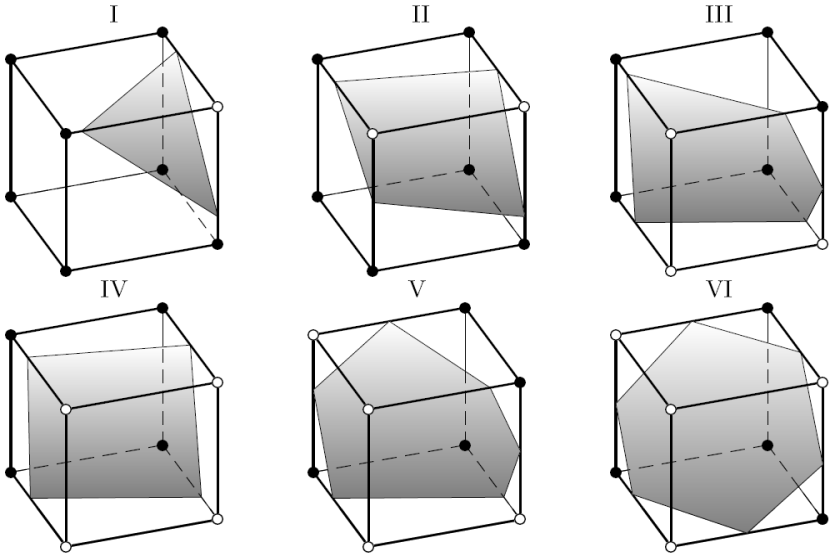


Рисунок 2.5 – Типичные примеры дробных ячеек

2.6 Создание неструктурной сетки

Древовидная структура ячеек, используемая в процессе построения сетки, удобна для ее построения, но не удобна для работы Решателя. Поэтому на следующем шаге генератор строит из дерева неструктурную сетку Решателя.

Для описания этой сетки используется три объекта – одномерные списки ячеек и внутренних и граничных граней (ниже для описания используются операторы Фортрана):

```
Type TCell
  integer (2) IX,IY,IZ       ! Fine индексы данного КО
  integer (1) level          ! уровень разбиения (счет с 0)
  real (4) Volume            ! объем ячейки
  real (4) Centre(3)         ! координаты центра тяжести
  ……
End type
Type (TCell), pointer ::     Cells(1:NoCells)

Type TFace
  integer(4) posCellID, negCellID     ! индексы КО объемов
  real (4)   Area                     ! площадь грани
  real (4)   Centre(3)                ! координаты центра тяжести грани
  real(4)   PN(3)                     ! единичная нормаль
  …….
End type
```

```
Type (TFace), pointer::     FaceXYZ (1:NoFaceXYZ)

Type TFaceBS
  integer(4) CellID      ! индексы приграничного КО
  real (4)   Area                ! площадь грани
  real (4)   Centre(3)             ! координаты центра тяжести грани
  real(4)   PN(3)                ! единичная нормаль
  …….
End type
Type (TFaceBS), pointer::     FaceBS (1:NoFaceBS)
```

Ячейка описывается объектом TCell, а информация о ячейках хранится в одномерном массиве Cells(1:NoCells). Для описания граней ячеек используются два массива: массив внутренних ячеек FaceXYZ(1:NoFaceXYZ) и массив граничных ячеек FaceBS(1:NoFaceBS).

Этот подход для описания сетки составляет суть неструктурных сеток с локальным дроблением. С одной стороны, ячейки расположены в произвольном порядке, а связь между ячейками производится через их грани (см. рисунок 2.6):

Faces()%PosCellID - это индекс ячейки в массиве Cells(), для которой грань является «левой» (расположенной в направлении уменьшения координат),

Faces()%NegCellID - это индекс ячейки в массиве Cells(), для которой грань является «правой» (расположенной в направлении увеличения координат).

С другой стороны, с каждой ячейкой связаны структурные Fine-индексы

ifx = Cells()%ix; ify = Cells()%iy; ifz = Cells()%iz,

которые можно использовать для визуализации в постпроцессоре.

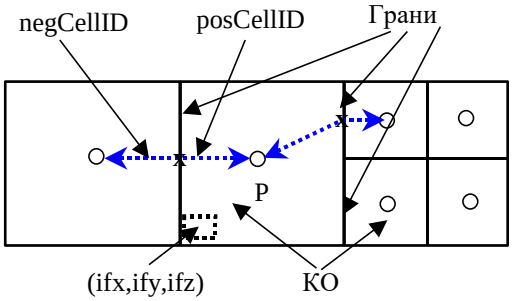


Рисунок 2.6 – ячейки (КО) и грани неструктурной сетки.

Отличие внутренней грани от граничной грани заключается в том, что для граничной грани имеется только одна связь с ячейкой (FaceBS()%CellID), содержащей данную грань.

Отчет о создании неструктурных объектов выдается в листинг и на консоль:

```
.Начинаем: Создание UNCells ...
.Закончили: Создание UNCells, time=000.00.00:001

.Начинаем: Создание UNFaces ...
.Закончили: Создание UNFaces, time=000.00.00:005

 ----------(Результат дробления)----------
 1.Число Faces.....= 13280
 2.Число внутренних Faces.....= 12562
 3.Из них число FS Faces.....= 0
 4.Число граничных Faces.....= 718
 5. No. Cut Faces.....= 1136
```

2.7 Перенумерация ячеек

В принципе при работе с неструктурными сетками КО порядок ячеек в списке Cell() не важен. Но как показали численные эксперименты, время расчета существенным образом зависит от порядка нумерации КО в массиве Cells(). Еще более существенную роль этот порядок играет при использовании параллельных алгоритмов.

При «идеальном» варианте нумерации соседние по пространству ячейки должны быть расположены как можно *ближе* друг к другу в массиве Cells.

Существует много алгоритмов перенумерации ячеек. Наиболее простым является алгоритм, основанный на использовании построения SFC-линии (Space Filling Curve). SFC-линия – это линия *непрерывно* соединяющая центры всех КО. Порядок прохождения этой линии и определяет порядок расположения КО в Cells(). Наиболее известны два алгоритма построения такой линии: Гильберта-Пеано (Hilbert) и Мортона (Morton). Для простой геометрии эти линии показаны на рисунке 2.7.

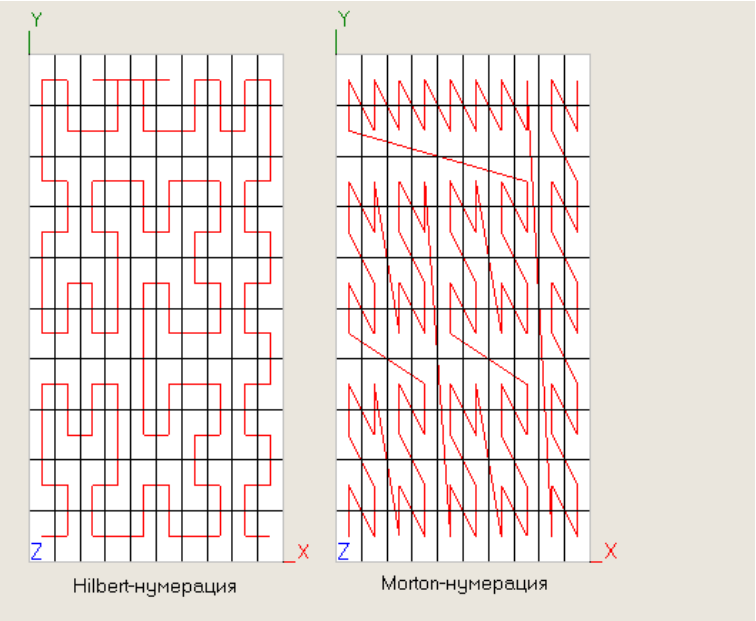


Рисунок 2.7 - Порядок нумерации КО с использованием SFC-алгоритмов.

Более сложные алгоритмы перенумерации используются в независимом пакете MeTis.

В коде генератора сеток Anes реализованы следующие алгоритмы перенумерации (ниже они обозначаются значениями переменной файла проекта AlgoSFC):

SFC_NONE - ячейки расположены в том же порядке, как и в дереве дробления.

SFC_byIZ - ячейки нумеруются в порядке возрастания одномерного индекса
ifxyz = ifx + (ify-1)*NFY + (ifz-1)*NFX*NFY.

SFC_byIY - ячейки нумеруются в порядке возрастания одномерного индекса
ifxyz = ifz + NFZ*(ifx-1)+NFZ*NFX*(ify-1).

SFC_byIX - ячейки нумеруются в порядке возрастания одномерного индекса
ifxyz = ify + NFY*(izf-1)+NFZ*NFY*(ixf-1).

SFC_Morten - ячейки нумеруются в соответствии с алгоритмом Мортена.

SFC_Metis - ячейки нумеруются в соответствии с алгоритмами пакета MeTis.

SFC_IXYZ_Metis - это наиболее эффективный алгоритм перенумерации для проведения параллельных вычислений. В этом режиме ячейки перенумеруются в два этапа. На первом этапе используется перенумерация SFC_byIX/ SFC_byIY / SFC_byIZ (направление выбирается по максимальному из значений NFX,NFY,NFZ), на втором этапе используется алгоритм MeTis.

Эффективность перенумерации легко оценить при проведении параллельных расчетов. Критерий – минимальность числа HALO-ячеек. Для визуального анализа перенумерации можно использовать постпроцессор ParaView и файл Компилятора проекта

<Префикс файлов результатов>_cells.vtk.

В последовательном расчете индекс ячейки содержится в поле CellID, в параллельном нужно анализировать поле SubDomID.

2.8 Определение Ф-переменных

Зависимая переменная Ф(x,y,z,τ) (давление, компоненты вектора скорости, температура и т.п.) представляет собой 3D поле. Как и в структурном представлении, при использовании неструктурных сеток этому полю соответствует дискретный набор значений Ф-переменной, который в Решателе хранится в виде двух одномерных массивов-списков:

```
usF8(idCV,iPHI),   idCV = 1 .. NoCells;   iPHI = 1 .. NoPHI,
usFBV(idFACE,iPHI), idFACE = 1 .. NoFaceBS; iPHI = 1 .. NoPHI,
```

где NoPHI – число Ф-переменных. В первом массиве хранятся значения Ф-переменных в центрах ячеек Cells()%Centre, во втором хранятся значения Ф-переменных на границе РО (в центрах граничных граней FaceBS()%Centre).

В отличие от структурных сеток, в неструктурной модели все Ф-переменные связываются с центрами ячеек (для скоростей используется модель совмещенных скоростей).

2.9 Создание вершин ячеек и граней

Для визуализации результатов расчета в постпроцессорах Anes и ParaView неудобно использовать значения Ф-переменных в центрах ячеек. Более подходящими значениями являются значения в вершинах ячеек. Поэтому генератор неструктурных сеток после создания неструктурных ячеек и граней создает список неструктурных вершин. Вершины в Решателе описываются объектом TNodes:

```
Type TNode
  real      X,Y,Z
end type
Type (TNode), pointer::    Nodes(1:NoNodes)
```

Для связи ячеек и граней с вершинами используются массивы:

```
integer(4), pointer:: Cell_Nodes(1:10,1:NoCells)
integer(4), pointer:: Face_BS_Nodes(1:6,1:NoFaceBS)
```

которые содержат индексы вершин для данной ячейки и данной граничной грани (в силу принятой модели дробных ячеек число вершин ячеек не больше 10, а число вершин граней не больше 6).

Для расчета значений в вершинах в коде Решателя реализованы два алгоритма:

1) на основе градиента Ф-переменной,
2) с использованием объемного осреднения.

Выбор алгоритма определяется переменной ModelCellToNodes секции [Main]:

ctnBYGRAD - на основе градиента,

ctnBYVOLUME - с использованием объемного осреднения.

При использовании градиента значение в вершине “in” рассчитывается по формуле:

$$\Phi_{in} = \frac{1}{N_{nb}} \sum_{i=1}^{N_{nb}} \left[\Phi_i + (\nabla\Phi)_i (\mathbf{r}_i - \mathbf{r}_{in}) \right]$$

где

N_{nb} - число ячеек, содержащих вершину in (для трехмерной целой ячейки таких вершин 8),

$\mathbf{r}_i$ - радиус-вектор центра i-ой ячейки Cells(i)%Centre(3),

$\mathbf{r}_{in}$ - радиус-вектор центра in-ой вершины - [Nodes(in)%x, Nodes(in)%x Nodes(in)%x)].

В модели объемного осреднения используется другое соотношение:

$$\Phi_{in} = \frac{\sum_{i=1}^{N_{nb}} \frac{\Phi_i}{V_i}}{\sum_{i=1}^{N_{nb}} \frac{1}{V_i}}$$

где V_i - объем i-от ячейки (Cells(i)%Volume).

При гладких решениях оба алгоритма дают близкие распределения полей в постпроцесорах. При сильных градиентах Ф-переменных, например, при работе с турбулентными пристенными функциями, более удобным является объемное осреднение. Следует отметить, что значения в вершинах не используются в процессе расчета Ф-переменных, они нужны только для визуализации результатов.

2.10 Обработка патчей

После создания неструктурной сетки ячеек, граней и вершин производится обработка всех патчей (как тех, которые учувствовали в процессе построения, так и оставшихся). Все объемные патчи переводятся в списки ячеек, а все поверхностные – в списки либо внутренних граней (из FaceXYZ()), либо граничных граней (из FaceBS()).

Количество ячеек и граней в списках патчей выводится в листинг Компилятора в виде:

```
.Начинаем: Создание неструктурных патчей солвера ...

  -----(НЕструктурные патчи компилятора)-----

 ------- Параметры патча -------

#   Name          Type     USType   NoCells    NoFaces
  1 In            BS       Faces      0          38
  2 Out           BS       Faces      0          38
  3 BOT           BS(Wall) Faces      0         320
  4 UP            BS(Wall)  Faces     0         320

 ------- Патчи (Объем/поверхность) -------

#   Name          Type     Vol(True) Vol(Solver) Area(True) Area(Solver)
  1 In            BS         0.000      0.000     0.1497E-01  0.8010E-02
  2 Out           BS         0.000      0.000     0.1497E-01  0.8010E-02
  3 BOT           BS(Wall)   0.000      0.000     0.1673      0.8013E-01
  4 UP            BS(Wall)   0.000      0.000     0.1673      0.8013E-01
.Закончили: Создание неструктурных патчей солвера, time=000.00.00:000
```

Просмотреть патчи Решателя можно в постпроцессоре Anes и ParaView. Для просмотра патчей в ParaView используется поле, которое называется ObjID. ObjID – это порядковый номер патча из списка патчей. Для просмотра ячеек, принадлежащих к ObjID-патчу нужно использовать файл

<Префикс файлов результатов>_cells.vtk.

Для просмотра поверхностных патчей необходимо использовать файл

<Префикс файлов результатов>_obj_faces.vtk.

2.11 Проверка правильности построения ячеек и граней

В конце работы генератора неструктурной сетки проводится проверка правильности построения списков ячеек, граней и вершин. Для этого используется разностный аналог теоремы Остроградского-Гаусса: если грани ячейки построены правильно и они образуют замкнутую поверхность, то векторная сумма поверхностей (вектор поверхности – это вектор, направленный по нормали грани и модуль которого численно равен площади грани) всех граней ячейки должна быть равна нулю! Генератор сетки проверяет это условие для трех его компонент в направлении осей x,y,z.

Для каждой ячейки вычисляется величина

$$A_{sum,k} = \sum_{nb} \Delta A_{nb} n_k$$

где

ΔA_{nb} - площадь грани, вычисленная по координатам ее вершин,

n_k - проекция внешней нормали к грани в направлении k=x,y,z.

Если значения $A_{sum,k}$ для всех ячеек равны нулю, то в листинг Компилятора будут выданы следующие сообщения

```
.Начинаем: Проверка связей Ячейки-ИхГрани ...
 >Sum of Area Faces along X is OK!
 >Sum of Area Faces along Y is OK!
 >Sum of Area Faces along Z is OK!
.Закончили: Проверка связей Ячейки-ИхГрани, time=000.00.00:707
```

Если при проверке были ошибки, то будет выдано сообщение, в котором будут перечислены индексы первых трех ошибочных ячеек. Для выяснения причины появления таких ячеек нужно найти их в постпроцессоре Anes или ParaView. Здесь правда есть два нюанса:

1) Нумерация ячеек в Решателе начинается с 1, в ParaView – с нуля.
2) В силу особенностей ParaView требует, чтобы ячейки имели определенную форму. При создании дробных ячеек возможно появление довольно «кривых» ячеек, поэтому при выводе в vtk-файлы Компилятор разбивает такие ячейки на под-ячейки простой формы. Это приводит к тому, что индекс ячейки в Компиляторе и vtk-файле будут различны. При анализе ошибок можно отключить механизм разбиения на под-ячейки (при этом некоторые ячейки будут высвечиваться необычно) Для включения/выключения этого механизма используется оператор IsSplitPH секции [Unstructured Cartesian Grid]. Если IsSplitPH = .false., то разбиения на под-ячейки не производится.

2.12 Проверка качества сетки ячеек. «Плохие» ячейки

Главным «недостатком» сетки с дробными ячейками является появление «плохих» ячеек: очень маленьких ячеек и ячеек «неправильной» формы. Такие ячейки могут приводить к большим погрешностям при расчете турбулентных характеристик при использовании модели пристенных функций, а иногда к расходимости итерационного процесса. Это связано в первую очередь с ошибками расчета градиента Ф-переменной в таких ячейках.

Для борьбы с этим явлением в коде предусмотрено два механизма:

1) пользователь может уменьшить число таких ячеек, используя переменную CriFracEdge секции [Unstructured Grid] (см. раздел 1.5.3).
2) можно использовать алгоритм коррекции для «плохих» ячеек, реализованный в Решателе (см. раздел 4.4).

Для тестирования сетки на наличие «плохих» ячеек в конце работы генератора сеток проводится проверка следующих характеристик сетки:

1) относительных объемов дробных ячеек,

2) относительных расстояний до BS- и FS-граней для дробных ячеек,
3) углов нормалей для внутренних граней.
4) наличия «изолированных» ячеек.

Информация о «плохих» ячейках записывается в файлы *.agr и *_cells.vtk,.что позволяет просмотреть плохие ячейки в постпроцессорах Anes и ParaView. В постпроцессоре Anes для этого используется диалог «Геометрия расчетной области» и выпадающий список «Раскраска ячеек». В постпроцессоре ParaView нужно использовать поле POOR_CELL файла *_cells.vtk.

2.12.1 Относительные объемы дробных ячеек

Для каждой дробной ячейки рассчитывается отношение ее объема к объему ее целой ячейки (Frac of Volume Cell). Отчет о работе выводится в листинг компилятора в виде:

```
========== Test Volume of CELLS ===============
 >Frac of Volume Cell = > 0.5     , No. Cells=4492645, Cell ID=25781
 >Frac of Volume Cell = 0.5 - 0.1 , No. Cells=259524, Cell ID=2421265
 >Frac of Volume Cell = 0.1 - 0.01, No. Cells=42246, Cell ID=1098024
 >Frac of Volume Cell = < 0.01    , No. Cells=997, Cell ID=4222650
 !WARNING: Frac of Volume Cell < 0.01 , No. Cells=997
```

Если имеются очень маленькие ячейки (с долей < 0.01), то последнее сообщение выводится и на консоль. Для удобства просмотра ячеек для каждой группы выводится индекс ячейки.

Если доля объема ячейки меньше 0.1, то такая ячейка помечается флагом POC_TINY_VOL, который используется в Решателе для реализации алгоритма коррекции «плохих» ячеек. В постпроцессоре Anes таким ячейкам соответствует раскраска «Плохие КО: объем < 0.1».

2.12.2 Относительные расстояния до BS- и FS-граней для дробных ячеек

При использовании «вытянутых» ячеек (например, при моделировании канальных течений) возможны ошибки и медленная сходимость, связанные с существенным отличием расстояний от центра ячейки к грани, направленных по нормали к грани и вектору из центра ячейки к центру грани. Для тестирования такой ситуации для приграничных дробных ячеек рассчитываются две характеристики:

1) относительное расстояние от центра ячейки до граничной BS- или FS- грани - d_w,
2) угол между нормалью к грани и вектором, соединяющим центр ячейки и центр грани θ_w (см. рисунок 2.8):

$$d_w = \frac{(\mathbf{r}_{PF} \bullet \mathbf{n})}{(\mathbf{r}_{cv} \bullet \mathbf{n})}, \qquad \mathbf{r}_{cv} = \left\{ \frac{\Delta x_P}{2}, \frac{\Delta y_P}{2}, \frac{\Delta z_P}{2} \right\},$$

$$\cos(\theta_w) = \frac{(\mathbf{r}_{PF} \bullet \mathbf{n})}{|\mathbf{r}_{PF}|}$$

Для масштабирования расстояния используется «псевдовектор», компоненты которого равны половине длин целой ячейки. В случае целой ячейки относительная длина равна единице, а угол равен нулю.

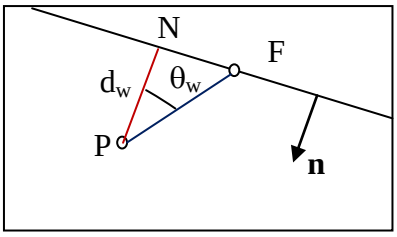


Рисунок 2.8 – Дробная ячейка и дробная FS-грань

Отчет о работе выводится в листинг компилятора в виде:

```
========== Test Wall Distance ===============
 >Frac of Wall Distance = > 0.5     , No. Cells=135466, Cell ID=3568676
 >Frac of Wall Distance = 0.5 - 0.1 , No. Cells=83628, Cell ID=0
 >Frac of Wall Distance = 0.1 - 0.01, No. Cells=3227, Cell ID=2472900
========== Test Angle Distance ===============
 >Angle Distance = < 10 град , No. Cells=62612, Cell ID=2773071
 >Angle Distance = 10-30 град, No. Cells=56860, Cell ID=4201874
 >Angle Distance = 30-50 град, No. Cells=24276, Cell ID=1498429
 >Angle Distance = > 50 град , No. Cells=78573, Cell ID=4215695
 !WARNING: Angle Distance > 50 град , No. Cells=78573
```

Если относительное расстояние до грани меньше 0.1, то такая ячейка помечается флагом POC_TINY_DWALL. Если угол между двумя расстояниями больше 60°, то такая ячейка помечается флагом POC_ANGLE_WALL. Эти флаги используется в Решателе для реализации алгоритма коррекции «плохих» ячеек. В постпроцессоре Anes таким ячейкам соответствуют раскраски «Плохие КО: Dist.Wall < 0.1» и «Плохие КО: Angle.Wall > 60».

2.12.3 Углы нормалей для внутренних граней

Если построить вектор из центра Neg ячейки в Pos ячейку и найти угол между этим вектором и нормалью грани, то для *кубических* целых ячеек одного уровня дробления этот угол будет равен нулю, а для ячеек разного уровня он будет равен 24 градусов (см. рисунок 2.6). При работе с дробными ячейками или сильно вытянутыми ячейками этот угол может быть > 24 градусов, что может также приводить к ухудшению сходимости итерационного процесса.

Для контроля таких ячеек проводится вычисление этого угла для всех внутренних граней и отчет об углах выводится в листинг компилятора в виде:

```
========== Test Angle XYZ faces ===============
 >Angle XYZ faces = 27-40 град, No. Cells=49187, Cell ID=3568676
 >Angle XYZ faces = 40-50 град, No. Cells=17460, Cell ID=3568676
 >Angle XYZ faces = 50-60 град, No. Cells=18645, Cell ID=3568676
 >Angle XYZ faces = > 60 град , No. Cells=239285, Cell ID=3568676
 !WARNING: Angle Distance > 60 град , No. Cells=239285
```

Если угол больше 60°, то обе ячейки грани помечаются флагом POC_ANGLE_POS и POC_ANGLE_NEG. Эти флаги используется в Решателе для реализации алгоритма коррекции «плохих» ячеек. В постпроцессоре Anes таким ячейкам соответствуют раскраски «Плохие КО: Angle(POS).FaceXYZ > 60» и «Плохие КО: Angle(NEG).FaceXYZ > 60».

2.12.4 «Изолированные» ячейки

При построении сложной геометрии РО возможно появление ячеек, у которых имеются только одна или ни одной внутренней грани, а все остальные грани являются граничными («изолированные» ячейки). Для таких граней расчет гидродинамических уравнений будет приводить к развалу итерационного процесса. Для поиска таких ячеек производится подсчет внутренних граней ячеек. Если число граней меньше двух, то в листинг Компиятора выводятся сообщения об изолированных ячейках:

```
========== Test Isolated cells ==============='
 >Isolated cells are found ! No. Cells= 2, Cell ID= 78654
```

2.13 Модель CBL для сглаживания дробных ячеек

Главным недостатком модели дробных ячеек ANES является появление «плохих» дробных ячеек на границе. Такие ячейки приводят к «пульсациям» граничных значений (трения, температуры стенки, давления), особенно при использовании турбулентных моделей.

Для уменьшения таких эффектов в коде реализован алгоритм «граничных слоев» CBL (Cut Boundary Layers). В текущей версии этот алгоритм разработан для РО в двумерной (x,y) и трехмерной (x,y,z) *декартовой* геометрии, границы которой созданы с помощью 2Dpatch объектов типа BlockWall, Block, Flow и Struct с направлением фигуры *вдоль оси* z. В текущей версии при решении трехмерных задач нельзя использовать дробление вдоль оси z. Для этого необходимо в секции [Unstructed Cartesian Grid] включить флаг:

```
IsNoRefZ = .TRUE.
```

Идея алгоритма достаточно простая: из граничных граней граничного патча создаются шестигранные новые ячейки, а граница переносится на новые грани этих ячеек. Можно создавать до 4 слоев новых ячеек. Общий размер новых ячеек в направлении границы H_{bc} задается пользователем для каждого патча. Размеры ячеек слоев создаются с использованием коэффициента логарифмичности q_{bc}:

$H_{k+1} = q_{bc} * h_k$

Если $q_{bc} = 1$ (это значение по умолчанию), то создаются слои ячеек одинаковой высоты. Если $q_{bc} < 1$, то высота ячеек уменьшается по мере приближения к «новой» границе.

Для сохранение геометрических пропорций объект 2D патча корректируется так (расширяется или сужается на величину H_{bc}), чтобы новые граничные грани совпадали с поверхностью исходного объекта.

Работа алгоритма с одним слоем CBL ячеек демонстрируется на рисунке 2.9.

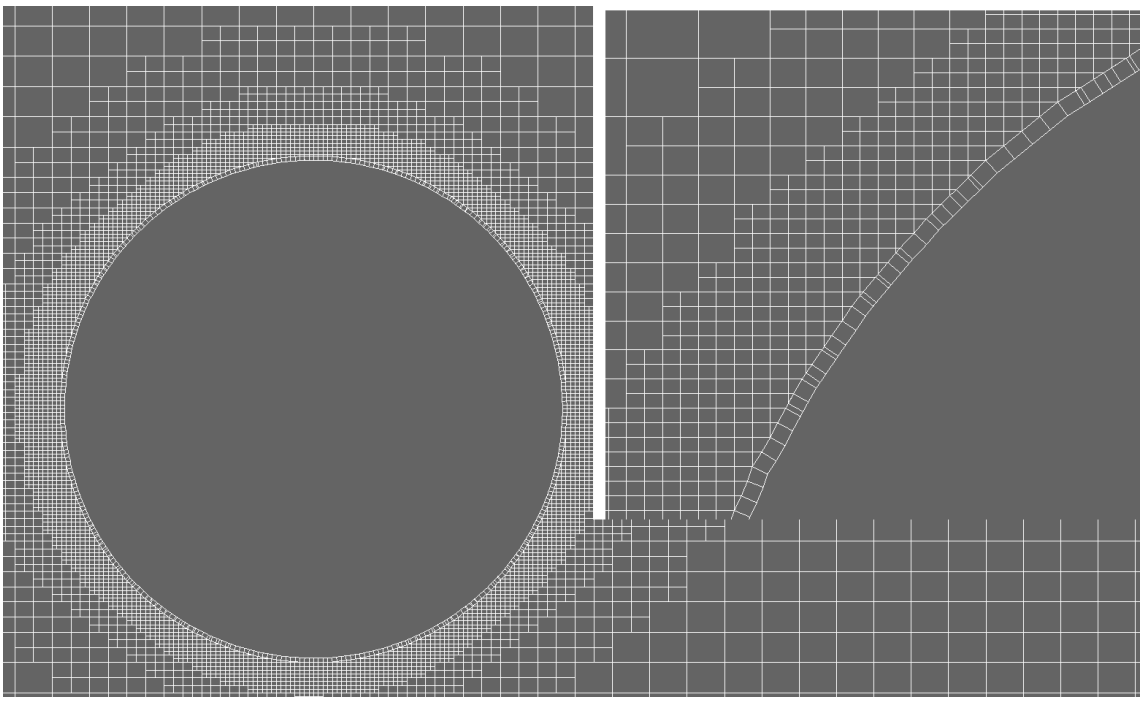


Рисунок 2.9 - Приграничный слой ячеек модели CBL

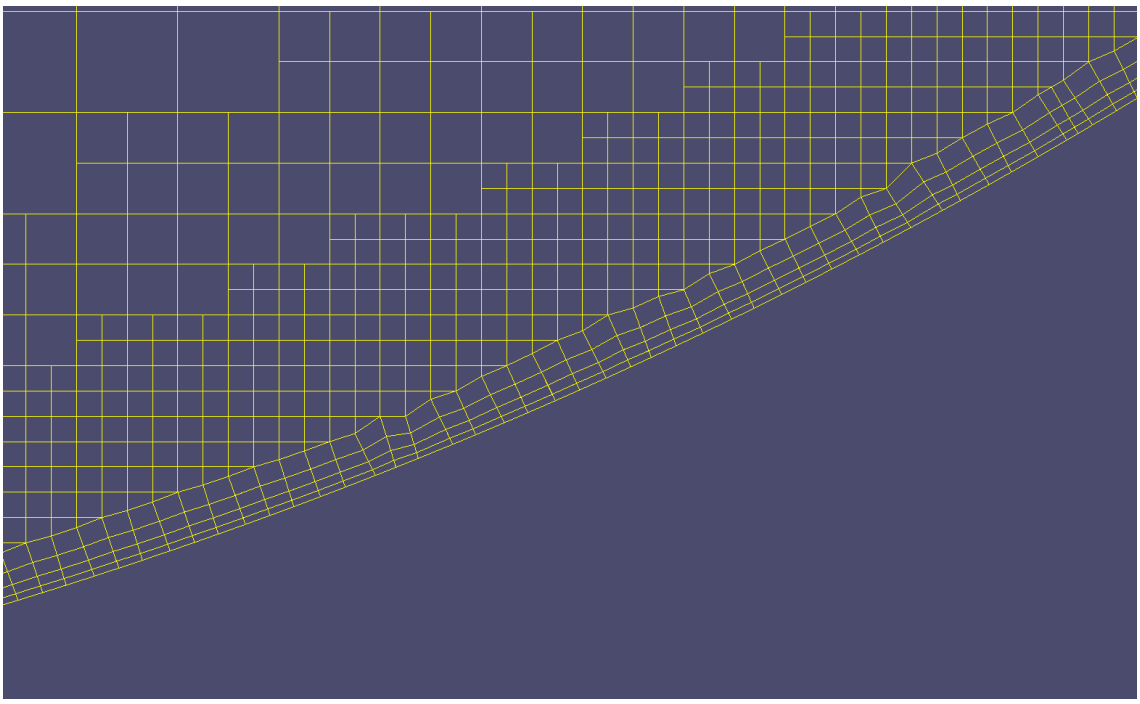
Для активации модели для патча с именем <ИмяПатча> необходимо в секцию [Special Data] поместить следующие операторы:

```
C("CBLpatch") = <ИмяПатча>
R("CBLsize. <ИмяПатча>") =  Hbc
I("CBLNoSubCell. <ИмяПатча>") =  1 / <Число слоев ячеек>
R("CBLqMesh. <ИмяПатча>") =  1/ qbc
```

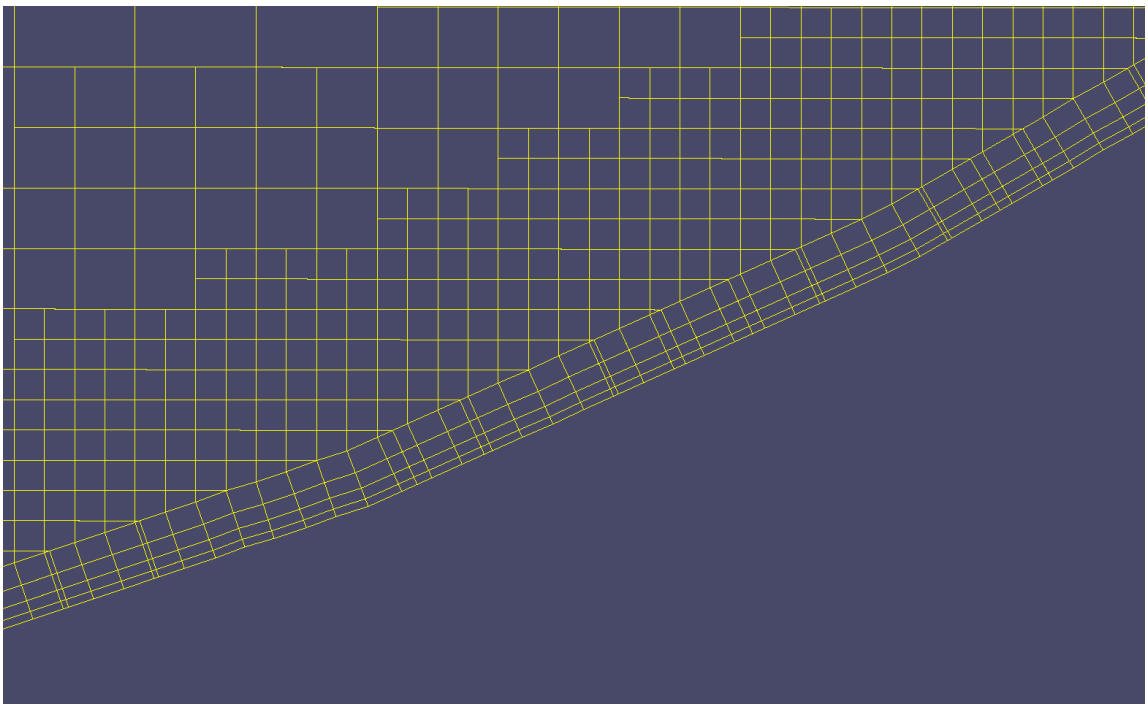
Для моделирования течений в круглой трубе можно еще уменьшить число «плохих» ячеек. Для этого нужно воспользоваться моделью CBL/Pipe. В этой модели на границе на первом этапе создается достаточно «грубая» граница дробных ячеек (CriFracEDge=0.25). А новая внешняя граница корректируется на основании радиуса трубы и положения ее центра (XC,YC). Эта модель активируется, если заданы операторы секции [Special Data]:

```
R("CBLradiusPIPE. <ИмяПатча>") =  <радиус трубы>
I("CBLcenterPIPE. <ИмяПатча>") =  vec(XC,YC,0)
```

Пример использования этой модели показан на рисунке 2.10.



Модель CBL/PIPE



Стандартная модель CBL

Рисунок 2.10 - Сравнение сетки стандартной CBL и CBL/PIPE

3. Конечно-разностный алгоритм для структурной сетки

Для дискретизации уравнений переноса в коде Anes используется метод контрольного объема (МКО). С подробностями этого алгоритма можно познакомиться в монографиях [3,4]. При использовании структурных сеток метод контрольных объемов (МКО) имеет ряд особенностей.

КО представляют собой параллелепипеды в обобщенной системе координат, а грани – прямоугольники. Все Ф-переменные, за исключением вектора скорости, помещаются в центры КО (точки P, E, W, N, S, H, L), а компоненты вектора скорости – в центры граней КО (U_{gx} в точках "w, e", U_{gy} в точках "n, s", U_{gz} в точках "h, l") (см. рисунок 1.2). Для скоростей контрольные объемы формируются из двух половинок КО, связанных с гранью, на которой определена компонента вектора скорости.

Рассмотрим получения дискретных уравнений на примере уравнения переноса для обобщенной Ф-переменной G-фазы (см. раздел 3.2 документа [1]). Для удобства ниже будет опускаться подстрочные индексы "g" и "Ф":

$$\frac{\partial(\rho\varphi\Phi)}{\partial\tau}+\mathrm{div}(\varphi\mathbf{J})=\varphi S, \qquad (3.1)$$

$$\mathbf{J}=\mathbf{m}\Phi-\Gamma\nabla\Phi,\quad \mathbf{m}=\rho\mathbf{U}$$

Здесь **U** - вектор скорости G-фазы, ρ - плотность G-фазы, Γ - эффективный коэффициент «диффузии», **J** - вектор плотности полного потока Ф-переменной, S - объемный источник порождения Ф-переменной, φ - пористость. Вектор плотности потока массы **m** удовлетворяет закону сохранения массы

$$\frac{\partial(\varphi\rho)}{\partial\tau}+\mathrm{div}(\varphi\mathbf{m})=\varphi M \qquad (3.2)$$

Для получения дискретных уравнений в методе МКО используется теорема Гаусса-Остроградского:

$$\int_V \mathrm{div}(\mathbf{F})dV=\oint_A(\mathbf{F}\cdot\mathbf{n})dA, \qquad (3.3)$$

которая связывает интеграл по произвольному объему V от дивергенции вектора **F** с поверхностным интегралом этого вектора по замкнутой поверхности A объема V. В поверхностном интеграле (3.3) **n** - это единичная *внешняя* нормаль поверхности.

3.1 Дискретное уравнение неразрывности

Если проинтегрировать уравнение (3.2) по контрольному объему P и по времени от τ - Δτ до τ, то с использованием (3.3) дискретное уравнение неразрывности можно записать в виде:

$$\frac{\rho_P\varphi_P-\rho_P^0\varphi_P^0}{\Delta\tau}\Delta V_P+m_e\varphi_e\Delta A_e-m_w\varphi_w\Delta A_w+m_n\varphi_n\Delta A_n-m_s\varphi_s\Delta A_s\ +$$
$$m_h\varphi_h\Delta A_h-m_l\varphi_l\Delta A_l\ =\ \varphi_P M_P\Delta V_P, \qquad (3.4)$$

здесь ΔV_P - объем КО, ΔA_k - площадь k-ой грани (k = w, e, s, n, l, h), Δτ - шаг по времени. Верхний индекс «0» означает, что значение величин берется на предыдущем шаге по времени τ - Δτ. Величины, у которых в обозначениях отсутствует верхний индекс «0», определены на новом шаге по времени (τ). Отметим сразу, что в коде Anes для интегрирования по времени используется полностью неявная схема [2].

Уравнение (3.4) удобно переписать в более *универсальном* виде:

$$\frac{\rho_P\varphi_P-\rho_P^0\varphi_P^0}{\Delta\tau}\Delta V_P+\sum_k m_{n,k}\varphi_k\Delta A_e\ =\ \varphi_P M_P\Delta V_P, \qquad (3.5)$$

$$m_{n,k}=\rho_k U_{n,k},\qquad U_{n,k}=(\mathbf{U}\cdot\mathbf{n})_k=U_k\cdot n_k$$

Во втором члене слева (3.5) суммирование производится по всем граням КО P, $U_{n,k}$ - это проекция вектора скорости в центре грани k на ее единичную *внешнюю* нормаль $\mathbf{n}_k$ (говоря проще, нормальная компонента скорости $U_{n,k}$ положительна, если поток *покидает* КО через эту грань). Для структурных сеток вместо вектора нормали $\mathbf{n}_k$ удобнее использовать ее компоненты n_k:
$n_k = +1$: для граней k = e, n, h;
$n_k = -1$: для граней k = w, s, l.

Для аппроксимации плотности массового потока на грани
$$m_k = \rho_k U_k n_k = \rho_k U_{n,k}$$
необходимо рассчитать только значение плотности и пористости на грани, поскольку нормальная компонента скорости U_k уже определена в центре грани (это первое преимущество структурных сеток). В коде Anes можно использовать два алгоритма аппроксимации плотности на k-грани:

1) модель против потока:
$$\rho_k = \rho_k^{UDS} = \begin{cases} \rho_P, U_{n,k} \ge 0 \\ \rho_{nb}, U_{n,k} < 0 \end{cases} \qquad (3.6)$$
2) модель линейной интерполяции:
$$\rho_k = \rho_k^{CDS} = \rho_P f_{p,k} + \rho_{nb}(1 - f_{p,k}), \quad f_{p,k} = \frac{\Delta x_{nb,k}}{\Delta x_{p,k} + \Delta x_{nb,k}}. \qquad (3.7)$$

Здесь nb - это индекс КО - соседа (E, W, S, N, L, H) через k-грань, $\Delta x_{p,k}$, $\Delta x_{nb,k}$ - размеры параллелепипеда КО P в направлении нормали k-грани. Пористость k-грани в коде всегда рассчитывается по соотношению
$$\varphi_k = \begin{cases} \dfrac{\varphi_P + \varphi_{nb}}{2}, \text{ если } 0 < \varphi_p < 1 \text{ и } 0 < \varphi_{nb} < 1 \\ \varphi_P, \text{ если } \varphi_{nb} = 0 \text{ или } \varphi_{nb} = 1 \\ \varphi_{nb}, \text{ если } \varphi_P = 0 \text{ или } \varphi_P = 1 \end{cases} \qquad (3.8)$$

По умолчанию в коде используется линейная интерполяция плотности. Противопоточная схема для плотности применяется при моделировании сжимаемых трансзвуковых и сверхзвуковых течений.

ⓘ *Описание в проекте прикладной задачи :*

Способ вычисления плотности на грани КО задается оператором секции [Solver]:
RhoFaceModel = RF_UPWIND / RF_LINEAR

3.2 Дискретное уравнение для Ф-переменной

Интегрируя уравнение (3.1) по контрольному объему P и по времени от τ - Δτ до τ, можно получить дискретное уравнение для Ф-переменной в «развернутом» виде:
$$\frac{\rho_P \varphi_P \Phi_P - \rho_P^0 \varphi_P^0 \Phi_P^0}{\Delta\tau} \Delta V_P + J_e \varphi_e \Delta A_e - J_w \varphi_w \Delta A_w + J_n \varphi_n \Delta A_n - J_s \varphi_s \Delta A_s + \qquad (3.9)$$
$$J_h \varphi_h \Delta A_h - J_l \varphi_l \Delta A_l = \varphi_P S_{\Phi,P} \Delta V_P,$$
или в универсальном виде, аналогичном дискретному уравнению неразрывности (3.8):
$$\frac{\rho_P \varphi_P \Phi_P - \rho_P^0 \varphi_P^0 \Phi_P^0}{\Delta\tau} \Delta V_P + \sum_k J_{n,k} \varphi_k \Delta A_e = \varphi_P S_P \Delta V_P, \qquad (3.10)$$
$$J_{n,k} = J_{c,k} + J_{d,k} = m_k \Phi_k - \Gamma_k \left(\mathbf{n} \cdot \nabla\Phi\right)_k$$

Здесь $J_{n,k}$ - нормальная компонента вектора плотности полного потока Ф на k-ой грани КО (отнесенного к единице поверхности, занятой G-фазой). Этот поток состоит из двух составляющих : $J_{c,k}$ - конвективной и $J_{d,k}$ - диффузионной.

3.3 Аппроксимация потоков Ф-переменной

Для «замыкания» уравнения (3.10) необходимо в первую очередь аппроксимировать обе составляющих плотности полного потока на грани КО.

3.3.1 Диффузионный поток на грани

В коде Anes (как и в большинстве CFD кодов) для аппроксимации *диффузионной части потока* используется центрально-разностная схема и среднегеометрическая интерполяция для коэффициента диффузии [3]:

$$J_{d,k} = d_k\left(\Phi_P - \Phi_{nb}\right),\ d_k = \frac{1}{\frac{1}{\alpha_P}+\frac{1}{\alpha_{nb}}},\ \alpha_p = \frac{\Gamma_P}{\delta_{P-k}},\ \alpha_{nb} = \frac{\Gamma_{nb}}{\delta_{nb-k}} \tag{3.11}$$

здесь nb - это индекс КО-соседа (E, W, S, N, L, H) через k-грань, $\delta_{p\text{-}k}$, $\delta_{nb\text{-}k}$ – расстояния от центра контрольных объемов P и nb до центра k-грани.

3.3.2 Конвективный поток на грани

Для расчета *конвективной составляющей* потока $J_{c,k}$ необходимо рассчитать значение Ф-переменной на грани. Для этого можно использовать различные аппроксимации [3,4,12]. Все эти аппроксимации лежат между двумя предельными значениями Φ_P и Φ_{nb}. Типичными представителями аппроксимаций являются две схемы: *центрально-разностная* схема:

$$\Phi_k = \Phi_k^{CDs} = \Phi_P f_P + \Phi_{nb}(1-f_P), \tag{3.12}$$

и схема *против потока*

$$\Phi_k = \Phi_k^{UDS} = \begin{cases} \Phi_P, U_{n,k} \ge 0 \\ \Phi_{nb}, U_{n,k} < 0 \end{cases} \tag{3.13}$$

Противопоточная аппроксимация позволяет записать конвективную составляющую полного потока в «компактном» виде

$$J_{c,k} = \max(m_k,0)\cdot\Phi_P - \max(-m_k,0)\cdot\Phi_{nb} = \max(-m_k,0)\cdot(\Phi_P - \Phi_{nb}) + m_k\Phi_P, \tag{3.14}$$

Схема против потока является наиболее простой и физически понятной [3], однако она обладает очень большой схемной диффузией. Для борьбы со схемной диффузией в коде используется два подхода при аппроксимации полного потока Ф-переменной на грани КО.

В первом подходе, предложенном Патанкаром [3], конвективная составляющая потока рассчитывается по противопоточной схеме, а диффузионная составляющая *корректируется* с помощью специальной корректирующей функции.

Во втором подходе [4,12] диффузионная составляющая не меняется, а коррекции подвергается конвективная составляющая потока, точнее - значение Ф на грани.

3.3.3 Степенная схема Патанкара

В подходе Патанкара полные плотности на грани рассчитываются по соотношениям:

$$\begin{aligned} J_{n,k}\varphi_k\Delta A_k &= a_{nb}\left(\Phi_P - \Phi_{nb}\right) + m_k\varphi_k\Delta A_k\Phi_P, \\ a_{nb} &= \left[\max(-m_k,0) + d_k A\left(P_k\right)\right]\varphi_k\Delta A_k \end{aligned} \tag{3.15}$$

Функция $A(P_k)$ определяет *тип численной схемы* для коррекции схемной диффузии [3]. В коде Anes для структурной сетки используются две модификации этой функции:
схема *против потока*

$$A(P_k) = 1 \tag{3.16}$$

и *степенная* схема

$$A(P_k) = \max\left(0\ ,\ \left(1 - \frac{P_k}{10}\right)^5\right),\ P_k = \frac{d_k}{|m_{n,k}|}, \tag{3.17}$$

где безразмерный параметр P_k представляет собой своеобразное сеточное число Пекле. Согласно этому соотношению при увеличении конвективной составляющей потока вклад диффузионной составляющей уменьшается и при $P_k > 10$ диффузионный перенос на грани полностью «удаляется». Это означает, что для описания переноса Ф-переменной используется параболическое приближение, при котором возмущения распространяются только «вниз по потоку».

Сделаем важное замечание. Несмотря на «физичность» степенной схемы, она в 2D и 3D задачах может приводить к большой схемной диффузии при решении задач с «эффективным» числом Пекле (или числом Рейнольдса), рассчитанным по размерам РО и характерным параметрам, больше 100 ... 300. Это может приводить к большим ошибкам при моделировании течений с большими числами Пекле. Типичный пример - DNS и LES модели турбулентного переноса.

3.3.4 Конвективные схемы второго порядка

Второй подход, реализованный в коде, основан на использовании схем второго порядка точности [4,12] для конвективной части потока.

Типичным примером такой схемы является схема QUICK (Quadratic Upstream Interpolation of Convective Kinematic). В этой схеме используется квадратичная интерполяция значений Ф-переменной по трем КО: двум КО-соседям грани и третий КО, расположенный в направлении *против* потока (если скорость U_e на грани "e" больше нуля, то используются КО E, P и W). При использовании второго подхода значение Ф-переменной на грани при $U_e > 0$ рассчитывается по специальным безразмерным соотношениям, удовлетворяющим условию монотонности распространения невязких возмущений. Существует, по крайней мере, три таких безразмерных представлений [12], вид безразмерных функций которых отличаются. В коде ANES используется соотношение:

$$\Phi_e = \Phi_P + \frac{\psi(r)}{2}(\Phi_E - \Phi_P),\quad r = \frac{\Phi_P - \Phi_W}{\Phi_E - \Phi_P} \tag{3.18}$$

Параметр r представляет собой отношение производных Ф-переменной на *предыдущей* по потоку грани и на *текущей* грани, а функция $\psi(r)$ играет ту же роль, что и функция $A(P_e)$ в подходе Патанкара.

Такой подход расширяет количество соседей в дискретном уравнении и, что самое главное, - «нарушает» модель «соседи - только через грань». Поскольку в основе численного решения дискретных уравнений в коде Anes лежат итерационные алгоритмы, то можно не расширять количество соседей, а для расчета значений в «дополнительных» КО использовать градиенты Ф-переменных, рассчитанных по полям с предыдущей итерации. В этом случае параметр r можно рассчитать по значению градиента в КО P:

$$\Phi_W = \Phi_E - 2\left(\frac{\partial\Phi}{\partial x}\right)_P \delta x_{E-P},\quad r = \frac{2\left(\frac{\partial\Phi}{\partial x}\right)_P \delta x_{E-P}}{\Phi_E - \Phi_P} - 1, \tag{3.19}$$

$$\delta x_{E-P} = \frac{1}{2}(\Delta x_P + \Delta x_E)$$

Заметим, что такой способ расчета значения в узле W связан с тем, что параметр r в (3.18) получен для равномерной сетки!

Различные схемы второго порядка различаются видом функции ψ(r). В текущей версии Anes реализованы шесть схем, функции которых приведены в таблице 3.1.

Таблица 3.1 Численные схемы 2 порядка ANES

Схема	ψ(r)
Quick	$\psi(r)=2(1-f_p)\left(\left(1-\frac{f_p}{2}\right)+r\frac{f_p}{2}\right)$
SuperBee	$\psi(r)=\max(0,\ \min(2,\ 2(1-f_p)\cdot r),\ \min(2(1-f_p),\ 2\cdot r))$
MisCL	$\psi(r)=\max(0,\ \min(2,\ 2\cdot r,\ (1-f_p)(r+1)))$
Gamma	$\Phi_p^*=\frac{r}{r+1},\quad \gamma=\frac{\Phi_p^*}{\beta_m},$ $\Psi(r)=\begin{cases}0, & \Phi_p^*<0 \text{ или } \Phi_p^*>1\\ 2(1-f_p), & \Phi_p^*>=\beta_m\\ 2(1-f_p)\gamma, & 0<\Phi_p^*<\beta_m\end{cases}$
BoundCD	$\Psi(r)=\begin{cases}0, & r<0\\ 2(1-f_p), & r>=0\end{cases}$
TrueCD	$\psi(r)=2(1-f_p)$

Используя (3.19) полный поток на k-грани можно записать в виде:

$$J_{n,k}\varphi_k\Delta A_k = a_{nb}(\Phi_P-\Phi_{nb})+m_k\varphi_k\Delta A_k\Phi_P,$$

$$a_{nb}=\left[\max(-m_k,0)-\frac{1}{2}|m_k|\psi(r_k)+d_k\right]\varphi_k\Delta A_k, \qquad (3.20)$$

$$r_k=\frac{2\left(\frac{\partial\Phi}{\partial x_k}\right)^*\delta x_{k,nb-P}}{\Phi_{nb}-\Phi_P}-1,\quad \left(\frac{\partial\Phi}{\partial x_k}\right)^*=\begin{cases}\left(\frac{\partial\Phi}{\partial x_k}\right)_{\mathbf{P}}, & m_k>0\\ \left(\frac{\partial\Phi}{\partial x_k}\right)_{\mathbf{nb}}, & m_k<0\end{cases}$$

Как видно из (3.20) возможно появление ситуации, когда коэффициент $a_{nb} < 0$, что обычно приводит к развалу процесса решения системы дискретных уравнений. Для борьбы с такими ситуациями в коде используется подход, основанный на введении поправки Ф-переменной (см. ниже).

ⓘ *Описание в проекте прикладной задачи :*

Тип численной схемы для структурной сетки определяется оператором секции [Solver]:

FVESheme = FVES_UPWIND / FVES _POWER / FVES_QUICK / FVES_SUPERBEE
FVES_MISCL / FVES_GAMMA, FVES_BoundCD, FVES_TrueCD

Для схем GAMMA/BoundCD/TrueCD параметр β_m задается оператором секции [Special Data]:

R("BetaFVES") = 0.1

Если оператор не задан, то используется значение $\beta_m = 0.1$.

3.3.5 Нестационарный член

Нестационарный член в уравнении (3.5) и (3.10) можно рассматривать как своеобразный «поток» в направлении оси времени. По *умолчанию* в коде ANES для аппроксимации нестационарного члена используется полностью неявная схема Эйлера (TTS_IMPLICIT).

В этом случае временной поток записывается в виде:

$$\frac{\rho_P\varphi_P\Phi_P-\rho_P^0\varphi_P^0\Phi_P^0}{\Delta\tau}\Delta V_P = a_T\left(\Phi_P-\Phi_P^0\right)+\Phi_P\left[\frac{\partial(\rho\varphi)}{\partial\tau}\right]_P\Delta V_P,\quad a_T=\frac{\varphi_P^0\rho_P^0}{\Delta\tau}\Delta V_P\ ,$$
$$\left[\frac{\partial(\rho\varphi)}{\partial\tau}\right]_P=\frac{\varphi_P\rho_P-\varphi_P^0\rho_P^0}{\Delta\tau},\quad \Phi_P^{0*}=\Phi_P^0 \qquad (3.21)$$

Такая аппроксимация является эффективной для решения большинства нестационарных задач. Однако при решении задач с большими числами Рейнольдса (или Пекле) эта схема может приводить к появлению дополнительной схемной временной диффузии. Типичный пример влияния такой схемной диффузии - подавление пульсаций при LES моделировании турбулентности.

Для решения таких задач в коде ANES реализована схема второго порядка по времени - схема TTS_3LEVEL. В этой схеме нестационарный член для переменной Ф аппроксимируется с использованием значения на пред-предыдущем шаге по времени с помощью параболической интерполяции:

$$\left(\frac{\partial\Phi}{\partial\tau}\right)_p=-\frac{(\xi_2-\xi_3)}{\Delta\tau}\left[\frac{\xi_2\Phi_p^0-\xi_3\Phi_p^{00}}{\xi_2-\xi_3}-\Phi_p\right],$$
$$\xi=\frac{\Delta\tau_0}{\Delta\tau},\quad \xi_2=\frac{1+\xi}{\xi},\ \xi_3=\frac{1}{\xi(1+\xi)},\quad \xi_1=\xi_2-\xi_3=\frac{2+\xi}{1+\xi}$$

Здесь индекс "0" как обычно означает поля на шаге по времени τ - $\Delta\tau$, а индекс "00" уровень τ - $\Delta\tau$ - $\Delta\tau_0$. Если шаг по времени постоянен, то : $\xi_1 = 3/2,\ \xi_2 = 2,\ \xi_3 = 1$.

В этом случае аналог уравнения имеет вид:

$$\left(\frac{\partial\rho\varphi\Phi}{\partial\tau}\right)_p\Delta V_p = a_T\left[\Phi_p-\Phi_p^{0*}\right]+\Phi_p\left[\frac{\partial\rho\varphi}{\partial\tau}\right]\Delta V_p,$$
$$\Phi_p^{0*}=\frac{\xi_2\rho_p^0\varphi_p^0\Phi_p^0-\xi_3\rho_p^{00}\varphi_p^{00}\Phi_p^{00}}{\xi_2\rho_p^0\varphi_p^0-\xi_3\rho_p^{00}\varphi_p^{00}},\qquad a_T=\frac{\left(\xi_2\rho_p^0\varphi_p^0-\xi_3\rho_p^{00}\varphi_p^{00}\right)}{\Delta\tau}$$

Заметим, что данная схема при сильно переменной плотности может приводить к появлению отрицательных значений a_T. Эту проблема частично решается использованием модели отложенной коррекции и поправки Ф-переменной.

ⓘ *Описание в проекте прикладной задачи :*
Тип численной схемы для нестационарного члена определяется оператором секции [Solver]:
TemporalSheme = TTS_IMPLICIT / TTS_3LEVEL

3.3.6 Окончательный вид дискретного уравнения

Собирая все члены, уравнение (3.10) можно переписать в виде:

$$a_T\left(\Phi_P-\Phi_P^{0*}\right)+D_P\Phi_P+\sum_{nb}a_{nb}\left(\Phi_P-\Phi_{nb}\right)=\varphi_P\left(S_P-M_P\Phi_P\right)\Delta V_P,$$
$$D_P=\left\{\frac{\rho_P\varphi_P-\rho_P^0\varphi_P^0}{\Delta\tau}\Delta V_P+\sum_k m_{n,k}\varphi_k\Delta A_e\ -\ \varphi_P M_P\Delta V_P\right\} \qquad (3.22)$$

Заметим, что член D_P представляет собой «невязку» уравнения неразрывности (3.8). Если мы получили численное решение дискретных уравнений, то этот член будет равен нулю, и его можно не учитывать в уравнении (3.22). В коде Anes в процессе итерационного решения дискретных уравнений этот член «не отбрасывается», он добавляется к дискретному уравнению, если он больше нуля: $D_P = \max(D_P, 0)$. Это позволяет «улучшить» процесс сходимости итерационного решения.

Окончательно дискретное уравнение для Ф-переменной можно представить в «стандартном» для кода Anes виде:

Окончательно дискретное уравнение для Ф-переменной можно представить в «стандартном» для кода Anes виде:

$$\boxed{\begin{aligned} &a_P\Phi_P = \sum_{nb} a_{nb}\Phi_{nb} + b_P, \\ &b_P = a_T\Phi_P^0 + \left(S_P - M_P\Phi_P\right)\varphi_P\Delta V_P, \quad a_P = a_T + \max(D_P,0) + \sum_{nb} a_{nb} \end{aligned}} \qquad (3.23)$$

В (3.23) суммирование проводится по всем КО-соседям "nb" текущего КО P.

3.4 Аппроксимация источниковых членов

В коде Anes для задания источниковых членов используется линеаризация источника следующего вида (см. документ [1]):

$$S_\Phi = \max(M_P,0)\cdot V_\Phi - \max(-M_P,0)\cdot\Phi + C_\Phi\left(V_\Phi - \Phi\right) \qquad (3.24)$$

Подставляя (3.21) в (3.23) получим:

$$\begin{aligned} &b_P = a_T\Phi_P^0 + \left\{\max(M_P,0) + C_\Phi\right\}V_{\Phi,P}\varphi_P\Delta V_P, \\ &a_P = a_T + \max(D_P,0) + \sum_{nb} a_{nb} + \left\{\max(M_P,0) + C_\Phi\right\}\varphi_P\Delta V_P \end{aligned} \qquad (3.25)$$

Соотношения (3.25) показывают, что линеаризация источника в виде (3.24) соответствует аппроксимации потоков на грани КО (3.15) и (3.20). При этом коэффициент источника C_Φ выполняет роль «диффузионной» составляющей потока. Разные знаки массовой части в коэффициенте a_{nb} и в (3.25) связаны с «направлением» потока и источника массы. В потоке на грани член m_k положителен, если он «вытекает» из КО. В источнике (3.24) M_P положительно, если источник массы «вдувается» в КО.

В коде Anes пользователь связывает источниковые члены (3.24) с объемными патчами и таких источников может быть много. В этом случае члены, связанные с источниками просто суммируются:

$$\begin{aligned} &b_P = a_T\Phi_P^0 + \varphi_P\Delta V_P\sum_{is}\left\{\max(M_P,0) + C_\Phi\right\}V_\Phi, \\ &a_P = a_T + \max(D_P,0) + \sum_{nb} a_{nb} + \varphi_P\Delta V_P\sum_{is}\left\{\max(M_P,0) + C_\Phi\right\} \end{aligned}$$

Если в качестве значения коэффициента источника C_Φ пользователь задал специальное значение SR_FIXFLUX, то в значении источника V_Φ задается непосредственно *сам источник*. В этом случае массовая составляющая источника просто *игнорируется* в (3.25):

$$b_P = a_T\Phi_P^0 + \left(V_{\Phi,P} - M_P\Phi_P\right)\varphi_P\Delta V_P, \qquad a_P = a_T + \max(D_P,0) + \sum_{nb} a_{nb} \qquad (3.26)$$

Приведенные выше соотношения справедливы для объемных источников. Если источник связан с *поверхностным патчем*, то в дискретном уравнении (3.25) он умножается не на объем КО, а на площадь грани патча ΔA_{pt}:

$$\begin{aligned} &b_P = a_T\Phi_P^0 + \varphi_P\sum_{is}\left\{\max(M_P,0) + C_\Phi\right\}V_\Phi\Delta A_{pt}, \\ &a_P = a_T + \max(D_P,0) + \sum_{nb} a_{nb} + \varphi_P\sum_{is}\left\{\max(M_P,0) + C_\Phi\right\}\Delta A_{pt} \end{aligned} \qquad (3.27)$$

Если источник связан с *точечным патчем*, то используются соотношения

$$\begin{aligned} &b_P = a_T\Phi_P^0 + \varphi_P\sum_{is}\left\{\max(M_P,0) + C_\Phi\right\}V_\Phi, \\ &a_P = a_T + \max(D_P,0) + \sum_{nb} a_{nb} + \varphi_P\sum_{is}\left\{\max(M_P,0) + C_\Phi\right\} \end{aligned}$$

☞Замечание. Может быть ситуация, когда для какой-то Ф-переменной *не задан* источник S_Φ. Если массовый источник для этого патча отличен от нуля, то в дискретном уравнении

член с источником для данной Ф-переменной будет равен нулю (при формировании дискретных коэффициентов для Ф-переменной в Решателе этого члена просто не будет):

$$\left(S_P - M_P\Phi_P\right)\varphi_P\Delta V_P = 0$$

Это эквивалентно заданию «скрытого» источника:

$$S_P = M_P\Phi_P .$$

3.5 Аппроксимация граничных условий

Разностное уравнение (3.23) несправедливо для *приграничных* КО, имеющих грани, совпадающие с границей расчетной области (ниже такие грани будут называться *граничными* гранями). Это связано с тем, что для этих граней неприменимы соотношения для потоков на грани КО (3.15) и (3.20), поскольку на границе нет «соседа». Для таких граней потоки должны быть заданы пользователем через *задание граничных условий* (ГУ).

В коде Anes ГУ задаются как *поверхностные источники*, связанные с поверхностными граничными патчами (см. раздел 3.5.2 [1]):

$$\begin{aligned} J_{\Phi,b} &= \max(m_b,0)\cdot V_{\Phi,b} - \max(-m_b,0)\cdot\Phi_P + C_{\Phi,b}\left(V_{\Phi,b} - \Phi_w\right), \\ J_{\Phi,b} &= \max(m_b,0)\cdot V_{\Phi,b} - \max(-m_b,0)\cdot\Phi_P + \left|C_{\Phi,b}\right|\left(V_{\Phi,b} - \Phi_P\right) \end{aligned} \qquad (3.28)$$

Здесь $J_{\Phi,b}$ - нормальная компонента вектора плотности потока Ф-переменной, при этом нормаль к границе направлена внутрь РО, Φ_P - значение Ф-переменной в приграничном КО, Φ_w - значение Ф-переменной в центре граничной грани. Пользователь, как и для источника, оперирует двумя V-переменными: $C_{\Phi,b}$ - коэффициент ГУ, $V_{\Phi,b}$ - значение ГУ. Две формы (3.28) отличаются значениями Ф-переменной, используемыми в «диффузионной» части источника. По умолчанию используется первая форма. Вторая форма используется для задания «хитрых» ГУ пользователя. Для ее использования нужно задать значение коэффициента источника $C_{\Phi,b}$ отрицательным.

Рассмотрим алгоритм реализации ГУ на примере внешних и внутренних границ РО, изображенных на рисунке 3.1.

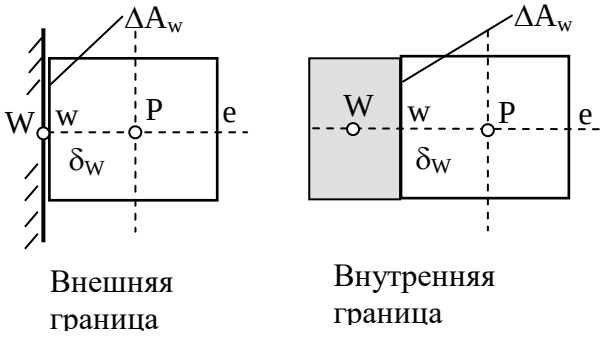


Рисунок 3.1 - Граничная грань приграничного КО

При переходе к дискретному аналогу *вторая* форма обрабатывается как поверхностный источник (3.27)

$$\left(J_{\Phi,b} - m_{b,w}\Phi_P\right)\varphi_P\Delta A_w = \left(\max(m_{b,w},0) + \tilde{C}_b\right)\left(V_{\Phi,b} - \Phi_P\right)\varphi_P\Delta A_w, \quad \tilde{C}_b = \left|C_{\Phi,b}\right| \qquad (3.29)$$

При формировании источника для первой «стандартной» формы ГУ (3.28) изменяется значение коэффициента:

$$\tilde{C}_b = \frac{1}{\frac{1}{C_{\Phi,b}} + \frac{1}{\alpha_w}}, \quad \alpha_w = \frac{\Gamma_P^*}{\delta_w} \tag{3.30}$$

Здесь δ_w - расстояние от центра КО до центра грани, Γ_P^* - *эффективное граничное* значение коэффициента диффузии в КО, (по умолчанию - это значение коэффициента диффузии в КО с узловой точкой P, при использовании модели турбулентности с пристенными функциями - это значение может существенно отличаться от Γ_P, см. раздел 9.2 документа [1]).

Как и для источников, если граничное условие для Ф-переменной *не задано* для участка проницаемой (входной или выходной) границы, Решатель все равно задает поток

$$J_{\Phi,b} = m_b \Phi_P . \tag{3.31}$$

Для удобства пользователя в коде Anes предусмотрен набор частных граничных условий, которые упрощают процесс задания ГУ. Для этого используются специальные символьные значения коэффициента ГУ $C_{\Phi,b}$. Рассмотрим дискретизацию этих специальных ГУ.

3.5.1 Граничное условие первого рода

Это граничное условие задается коэффициентом ГУ $C_{\Phi,b}$ = BC_FIXVAL. Компилятор Anes это символьное значение передает Решателю в виде $C_{\Phi,b} = -10^{30}$. Для этого условия коэффициент дискретного источника (3.29) задается в виде:

$$\tilde{C}_b = \alpha_w, \quad \alpha_w = \frac{\Gamma_P^*}{\delta_w}$$

3.5.2 Граничное условие второго рода

Это граничное условие задается коэффициентом ГУ $C_{\Phi,b}$ = BC_FIXFLUX. Компилятор Anes это символьное значение передает Решателю в виде $C_{\Phi,b} = -2 \cdot 10^{30}$. В этом случае массовая составляющая источника игнорируется, а сам источник задается в виде (при этом ГУ $V_{\Phi,b}$ - это плотность потока):

$$\left(J_{\Phi,b} - m_{b,w}\Phi_P\right)\varphi_P \Delta A_w = V_{\Phi,b}\varphi_P \Delta A_w$$

3.5.3 Граничное условие на входной границе

Это граничное условие задается коэффициентом ГУ $C_{\Phi,b}$ = BC_MASSONLY. Компилятор Anes это символьное значение передает Решателю в виде $C_{\Phi,b} = -3 \cdot 10^{30}$. При использовании этого ГУ игнорируется диффузионная составляющая источника:

$$\left(J_{\Phi,b} - m_{b,w}\Phi_P\right)\varphi_P \Delta A_w = \max(m_{b,w}, 0) \cdot \left(V_{\Phi,b} - \Phi_P\right)\varphi_P \Delta A_w$$

3.5.4 Блокированные КО

Как уже отмечалось, при использовании структурных сеток КО, расположенные внутри Block-патчей, не удаляются из расчетной области. Они просто помечаются специальным флагом IsBlock.

В таких блокированных КО значение Ф-переменной не имеет смысла и обычно полагается равным нулю. Это достигается введением для этих КО следующего разностного уравнения

$$a_P \Phi_P = 0, \; a_P = V_{big},$$

где V_{big} – большое значение ($= 10^{30}$).

Для удобства *обработки* результатов расчета Решатель вычисляет граничные значения Ф-переменных Φ_w (которые в самом процессе вычислений не участвуют). При использовании структурной сетки эти значения помещаются в узлы *приграничных блокированных* КО (узлы "W" на рисунке 3.1). Это, кстати, является основной причиной введения граничных узлов на границе базовой расчетной области. При использовании граничных значений Ф-переменных следует помнить, что для внутренних границ граничное значение хранится в узле блокированного КО "W", однако пространственно оно относится к грани "w". Само граничное значение вычисляется из балансов потоков на границе. Например, в случае непроницаемой границы и граничного условия первого рода

$$\Phi_w = \frac{C_{\Phi,b}}{C_{\Phi,b}+\alpha_w} V_{\Phi,b} + \frac{\alpha_w}{C_{\Phi,b}+\alpha_w}\Phi_P, \quad \alpha_w = \frac{\Gamma^*_P}{\delta_w}$$

Приграничные блокированные КО (КО W на рисунке 3.1) в коде Anes используются для сохранения значений Ф-переменных на границе РО.

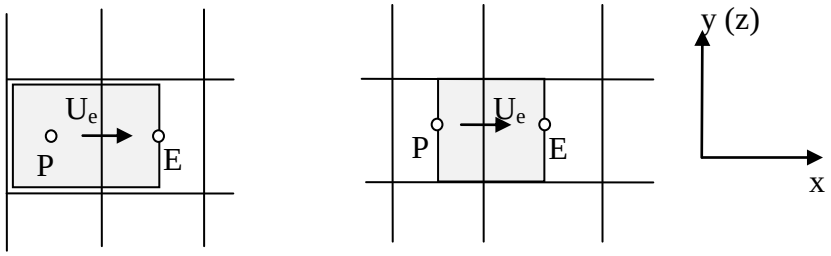
3.5.5 Периодические граничные условия

При использовании структурных сеток периодические условия (ПГУ) реализуются помощью следующего алгоритма (для определенности рассмотрим ПГУ вдоль оси X).

Граничные внешние грани xFace(2) и xFace(NX) не помечаются как границы РО и при вычислении дискретных коэффициентов a_{nb} для КО с ix = 2 и NX-1 они просто связываются друг с другом через грань xFace(2).

3.6 Уравнения движения

Как уже отмечалось, КО для скоростей создаются из двух половинок основных КО (см. рисунок 3.2). Особый случай - это приграничные КО. В этом случае для скорости создается полуторный КО, состоящий из «целого» пристенного КО и половинки КО-соседа.



Приграничный КО Внутренний КО

Рисунок 3.2 - КО для компоненты скорости U_x

Дискретные уравнения для компонентов вектора скорости можно записать в виде, аналогичном (3.23):

$$\boxed{\begin{aligned} a_{P,k}U_k &= \sum_{nb} a_{nb}U_{k,nb} + C_k n_k\left(p_P - p_{nb}\right) + b_k, \\ C_k &= \frac{\varphi_P \Delta V_P + \varphi_{k,nb}\Delta V_{k,nb}}{\Delta x_{k,P} + \Delta x_{k,nb}} \end{aligned}} \qquad (3.32)$$

Здесь p_{nb} - давление в КО-соседе через k-грань, $\Delta x_{k,P}$, $\Delta x_{k,nb}$ - размеры КО в направлении нормали k-грани.

4. Конечно-разностный алгоритм для неструктурной сетки

Алгоритм МКО для неструктурных сеток очень похож на структурный алгоритм, но имеются и несколько отличий:

1) компоненты вектора скорости (U_{gx},U_{gy},U_{gz}) определены не на гранях КО, а в его центре, поэтому возникает проблема расчета массовых потоков m_k на гранях КО;
2) грань может соединять КО разных уровней, уровни которых отличаются на единицу (рисунок 4.1(а)); в этом случае вектор P-nb не перпендикулярен грани и не проходит через ее центр «e»; аналогичная ситуация возникает при использовании дробных ячеек (рисунок 4.1(б));
3) для реализации алгоритма SIMPLE (об этом ниже в главе 6) необходима связь между компонентами скорости и градиента давления на грани; при использовании структурной сетки эта проблема решается автоматически, поскольку скорости заданы на грани, для неструктурной сетки необходимо использовать «хитрую» интерполяцию.

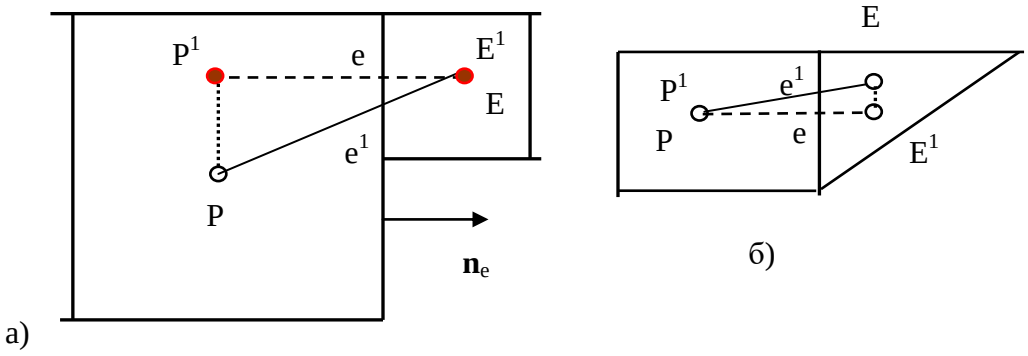


Рисунок 4.1 – Типичные связи ячеек неструктурной сетки

Подробности реализации МКО для неструктурных сеток можно найти в монографии [4], ниже будут изложены только особенности неструктурного МКО, необходимые для понимания кода.

4.1 Аппроксимация диффузионного члена

Для вычисления диффузионного потока в центре грани «e» используется следующая идея: для вычисления потока будем использовать не центры КО P и E, а точки P^1 и E^1, которые получаются путем проекции центров на линию нормали грани (P^1-E^1). Если для расчета значений в точках P^1 и E^1 использовать значения градиентов в ячейках, то диффузионную составляющую потока можно записать в виде:

$$J_{d,e} = d_e\left(\Phi_P - \Phi_E\right) + d_e\delta\Phi_{PE},$$

$$\delta\Phi_{PE} = \left[\left(\nabla\Phi\right)_P \bullet \left(\mathbf{r}_{P^1} - \mathbf{r}_P\right) - \left(\nabla\Phi\right)_E \bullet \left(\mathbf{r}_{E^1} - \mathbf{r}_E\right)\right], \qquad (4.1)$$

$$d_e = \frac{1}{\dfrac{\delta_{e-E}}{\Gamma_{\Phi,E}} + \dfrac{\delta_{P-e}}{\Gamma_{\Phi,P}}},\ \delta_{P-e} = \left(\mathbf{r}_e - \mathbf{r}_P\right)\bullet\mathbf{n}_e,\ \delta_{e-E} = \left(\mathbf{r}_E - \mathbf{r}_e\right)\bullet\mathbf{n}_e$$

Вторая составляющая потока в (4.1) связана с неортогональностью ячеек и для своего вычисления требует расчета градиента Ф-переменной в центре ячейки.

Заметим, что учет «неортогональной» составляющей потока может существенно повысить точность расчета, особенно при использовании дробных ячеек. Однако при наличии «плохих» ячеек (см. пункт 2.12) «неортогональные» члены могут приводить к расхождению итерационного процесса. В этом случае можно использовать специальный алгоритм коррекции, который описан ниже.

4.2 Расчет массового потока на грани. Поправка Рие-Чоу

Для вычисления конвективной составляющей полного потока $J_{n,k}$ на грани с индексом k необходимо знать нормальную к грани плотность массового потока

$$m_k = \rho_k U_{n,k}, \quad U_{n,k} = (\mathbf{U}\cdot\mathbf{n}_f)_k,$$

для расчета которой необходима нормальная компонента скорости $U_{n,k}$ в центре грани.

Как показали численные эксперименты, использование линейной интерполяции скорости на грань:

$$U_{n,k} = \overline{(\mathbf{U}\cdot\mathbf{n}_k)}_f = f_P\left(\mathbf{U}_P\cdot\mathbf{n}_k\right) + \left(1-f_P\right)\left(\mathbf{U}_E\cdot\mathbf{n}_k\right), \qquad f_P = \frac{\left|\mathbf{r}_E-\mathbf{r}_{e^1}\right|}{\left|\mathbf{r}_E-\mathbf{r}_P\right|} = \frac{x_E-x_e}{x_E-x_P}, \tag{4.2}$$

приводит к появлению численной «шахматной» неустойчивости [3,4]. Здесь и далее функция $\overline{(f)}_k$ будет обозначать линейную интерполяцию значения поля на k-грань.

Для решения проблемы «шахматной» неустойчивости были предложены различные алгоритмы, которые основываются на идее, впервые высказанной Рие и Чоу (Rhie and Chow) [4].

Рассмотрим суть этого подхода на примере грани "e" между целыми ячейками "P" и "E" одного уровня дробления. В этом случае нормальная компонента скорости на грани совпадает с компонентой скорости $U = U_{gx}$. Уравнение для скорости в КО "P" можно записать в стандартном дискретном виде:

$$a_P U_P = \sum_{nb} a_{nb} U_{nb} + b_P - \Delta V_P \left(\frac{\partial p}{\partial x}\right)_P,$$

из которого можно «определить» U_P:

$$U_P = H_P - \frac{\Delta V_P}{a_P}\left(\frac{\partial p}{\partial x}\right)_P, \quad H_P = \frac{\sum_{nb} a_{nb} U_{nb} + b_P}{a_P} \tag{4.3}$$

Для вывода соотношения для скорости на грани будем использовать подход, предложенный Х. Ясаком (H. Jasak) [9]: для получения скорости на грани просто проинтерполируем уравнение (4.3) на грань:

$$U_e = \overline{(H)}_e - \overline{\left(\frac{\Delta V}{a_P}\left(\frac{\partial p}{\partial x}\right)\right)}_e \approx \overline{(H)}_e - \left(\overline{\frac{\Delta V}{a_P}}\right)_e \left(\frac{\partial p}{\partial x}\right)_e \tag{4.4}$$

Выражая из уравнения (4.3) H_P и подставляя его в (4.4), можно получить соотношение для скорости:

$$U_e = \overline{(U)}_e + \overline{\left(\frac{\Delta V}{a_P}\cdot\frac{\partial P}{\partial x}\right)}_e - \overline{\left(\frac{\Delta V}{a_P}\right)}_e \frac{P_E - P_P}{\delta_{PE}} \tag{4.4}$$

Скорость на грани вычисляется с помощью линейной интерполяции, к которой добавляется поправка, зависящая от разности градиентов давления, рассчитанных по разным аппроксимациям (эта поправка в литературе и называется поправкой Рие-Чоу). Второй градиент давления – это градиент давления, рассчитанный относительно грани КО (он аналогичен градиенту давления для шахматной структурной сетки), первый градиент – это интерполяция значений градиентов в ячейках P и E.

Соотношение (4.4) позволяет ликвидировать «шахматную» неустойчивость и, самое главное, связать нормальную компоненту скорости на грани с градиентом давления на грани, что является основой реализации метода SIMPLE для совмещенных скоростей.

Соотношение (4.4) записано для простой «структурной» сетки. Аналогичные соотношения можно получить для произвольной геометрии ячеек. В этом случае соотношение (4.4) имеет следующий вид:

$$U_{n,k} = \left(\mathbf{U}\cdot\mathbf{n}_f\right)_k = \overline{\left(\mathbf{U}\cdot\mathbf{n}_f\right)_k} - \overline{\left(\frac{\Delta V}{a_p}\right)}_f \frac{\left(p_{nb}-p_P\right)}{\delta_k} + \overline{\left(\frac{\Delta V}{a_p}\nabla p\right)}_f \cdot \frac{\mathbf{r}_{P-nb}}{\delta_k}, \quad \delta_k = \left|\mathbf{r}_{P-nb}\cdot\mathbf{n}_f\right| \quad (4.5)$$

Здесь $\mathbf{n}_f$ - внешняя к ячейке P нормаль грани, направленная в сторону nb-ячейки, $\mathbf{r}_{P\text{-}nb}$ - вектор, направленный из центра P ячейки в центр E ячейки.

При получении соотношений (4.4) и (4.5) из источникового члена b_p был явно выделен член, связанный с градиентом давления. Все остальные силовые источники были оставлены в b_p и включены в H_P (4.3). В большинстве задач такой подход срабатывает. Однако при наличии «сильно неоднородных» сил возникает необходимость особой обработки этих силовых источников. Типичные примеры таких источников:

1) межфазные силовые источники в VOF-алгоритме (см. документ [10]);
2) поверхностные источники с заданными скачками давления (различные модели вентилятора и винтов);
3) объемные источники в пористых зонах.

В этом случае эти источники необходимо «выделить» из b_p и обработать так же, как и градиент давления. Это приводит к появлению в соотношении (4.5) дополнительных членов. Более подробно вид этих членов описан в документе [10] и ниже в разделе 4.8.

В связи с этим в коде Anes реализованы два алгоритма Рие-Чоу. В первом алгоритме - «Ferziger-Peric» - для расчета скорости на грани использует соотношение (4.5). Во втором алгоритме - «BodyForce Model» - в соотношение (4.5) добавляются поправки связанные с источниками, которые аппроксимируются с использованием BodyForce модели (см. раздел 4.8). Выбор алгоритма производится автоматически: если хотя бы один источник описывается с помощью BodyForce модели, то используется алгоритм «BodyForce Model».

4.3 Расчет градиента Ф-переменной

Для аппроксимации членов дискретных уравнений необходимо рассчитать в центрах ячеек три компоненты вектора градиента Ф-переменной

$$\left(\nabla\Phi\right)_P = \left[\frac{\partial\Phi}{\partial x}, \frac{\partial\Phi}{\partial y}, \frac{\partial\Phi}{\partial z}\right]_P$$

В Решателе для расчета градиента используется два алгоритма.

Первый основан на использовании теоремы Остроградского - Гаусса, которую для ячейки P можно записать в виде

$$\left(\frac{\partial\Phi}{\partial x_k}\right)_P = \frac{\sum_f \Phi_f \Delta A_f n_{f,k}}{\Delta V_P}, \; k = x, y, z \quad (4.6)$$

Здесь $n_{f,k}$ – компоненты вектора внешней нормали грани, связывающей ячейку P и ее соседа (nb). Суммирование в (4.6) производится по всем граням ячейки. Если грань является граничной, то значение Ф-переменной на ней Φ_f известно из граничных условий. Для внутренней грани значение в центре грани рассчитывается с помощью линейной интерполяции второго порядка точности. Для этого сначала вычисляется с помощью интерполяции значение Φ_f в точке «e^1» (см. рисунок 4.1), а затем с использованием градиента Ф-переменной на грани вычисляется значение в точке «e». Сам градиент на грани вычисляется с помощью линейной интерполяции градиентов в узлах P и nb:

$$\Phi_f = f_p\Phi_P + \left(1-f_p\right)\Phi_{nb} + \overline{\left(\nabla\Phi\right)}_f \cdot \mathbf{s}_e,$$
$$\overline{\left(\nabla\Phi\right)}_f = f_p\left(\nabla\Phi\right)_P + \left(1-f_p\right)\left(\nabla\Phi\right)_{nb}, \quad \mathbf{s}_e = \mathbf{r}_e - \mathbf{r}_{e1}$$

При такой интерполяции в соотношение (4.6) в правую часть также входят значения градиентов, поэтому для расчета градиента необходимо использовать итерации. Максимальное число итераций задается значением целой переменной секции [Special Data] проекта

```
I("MaxGradIter")      = 10
```

Если этот оператор не указан, то используется значение по умолчанию, равное 10.

Второй алгоритм основан на использовании метода наименьших квадратов. Рассмотрим функционал

$$L[(\nabla\Phi)_P] = \sum_{nb} w_{nb}^2 \left\{\Phi_{nb} - \left[\Phi_P + (\nabla\Phi)_P \cdot \mathbf{r}_{P-nb}\right]\right\}^2,$$
$$\mathbf{r}_{P-nb} = \mathbf{r}_{nb} - \mathbf{r}_P$$

Здесь w_{nb}^2 - вес грани, связывающей ячейки "P" и "nb". Суммирование производится по всем ячейкам-соседям. Для граничных граней в качестве Φ_{nb} используются граничные значения. Если потребовать, чтобы значение функционала было минимально, то легко получить *три линейных* уравнения для трех компонент градиента в центре ячейки P (ниже индекс k - это номер уравнения, индекс i – индекс суммирования по координатам x,y,z):

$$\frac{\partial L}{\partial\left\{\dfrac{\partial\Phi}{\partial x_k}\right\}} = 0$$

или

$$\sum_{nb} x_{P-nb,k} \cdot x_{P-nb,i} \cdot w_{nb}^2 \left(\frac{\partial\Phi}{\partial x_i}\right)_P = \sum_{nb} x_{P-nb,k} \cdot w_{nb}^2 \left(\Phi_{nb} - \Phi_P\right), \qquad (4.7)$$

Эту система трех уравнений (k = x,y,z) для каждой ячейки P легко решается прямым методом Крамера. В коде Anes используется три модели для весов граней w_{nb}^2:

1) безвесовой алгоритм W_UNIT: $w_{nb}^2 = 1$,
2) нормированные веса W_NORM: $w_{nb}^2 = 1/\left|\mathbf{r}_{P-nb}\right|^2$,
3) гауссовский вес W_GAUSS:

$$w_{nb}^2 = \frac{\left(1-f_P\right)\Delta A_{f,nb}}{\Delta V_P \left|\mathbf{r}_{P-nb}\right|},$$

здесь f_P – коэффициент линейной интерполяции (4.2).

В Решателе для расчета градиента по умолчанию используется алгоритм метода наименьших квадратов с моделью гауссовского веса, поскольку он не использует итерации.

Выбор алгоритма расчета градиента зависит от прикладной задачи и проще всего это определить предварительными расчетами.

ⓘ *Описание в проекте прикладной задачи :*
Для выбора алгоритма расчета градиента используется символьный оператор секции [Special Data]:
C("ModelCALC_GRAD ") = "LSM" / "Gauss"

По умолчанию используется модель LSM

Выбор алгоритма для веса грани определяется символьным оператором
C("ModelLSM_WEIGHT ") = " W_GAUSS" / "W_NORM" / "W_UNIT "

По умолчанию используется модель W_GAUSS для веса грани.

4.4 Алгоритм коррекции «плохих» ячеек

При наличии в расчетной области «плохих» ячеек возможно появление расходимости итерационного процесса. Чаще всего признаком расходимости является «взрывной» рост давления в одной из «плохих» ячеек. Для нахождения таких ячеек можно остановить расчет до «развала» итерационного процесса (выбором максимального числа SWEEP-итераций) и просмотреть поля в ячейках, используя режим сохранения полей в центрах ячеек в ARS- или VTK-файлов. Для активации такого режима сохранения используются логические операторы секции [Special Data]:

```
L("ARS.OUTCELL") = .TRUE.
L("VTK.OUTCELL") = .TRUE.
```

Причинами расходимости итерационного процесса могут быть:

1) использование второго порядка аппроксимации при расчете диффузионного потока на грани (см. раздел 4.1);
2) использование второго порядка аппроксимации при расчете массового потока на грани (см. раздел 4.2);
3) ошибки расчета градиента в маленьких вытянутых вдоль одной из координат ячейках.

Для подавления этих причин расходимости в коде Решателя реализован алгоритм коррекции «плохих» ячеек, выявленных на этапе генерации сетки (см. раздел 2.12):

1. Если при расчете диффузионного или массового потоков одна из ячеек грани является плохой (установлены флаги POC_TINY_VOL, POC_TINY_DWALL, POC_ANGLE_WALL), то отключается второй порядок аппроксимации для потоков.
2. Если для ячейки установлен флаг POC_TINY_VOL, то градиент в этой ячейки не рассчитывается с использованием моделей LSM и Gauss. Он вычисляется как среднее по соседним «хорошим» ячейкам.

В коде этот алгоритм «вЫключен» по умолчанию. Для его включения необходимо использовать оператор секции [Special Data]:

```
L("IsPOOR_MESH") = .True.
```

Как показали расчеты тепловых задач в каналах, модель коррекции для температуры дают большие пульсации потока тепла или температуры (в зависимости от ГУ) на внешних границах или границе FS. Отключение модели коррекции для TG, TS и HG позволяет уменьшить эти пульсации. Для этого добавлен флаг

```
L("IsPOOR_MESH_TGS ") = .False. / .True.
```

который по умолчанию отключает модель коррекции для уравнения энергии для TG и TS.

4.5 Численые схемы второго порядка точности

Для неструктурных сеток, как и для структурных, реализованы два алгоритма расчета полной плотности потока на грани (см. раздел 3.3).

При использовании подхода Патанкара (схемы против потока и степенная схема) нормальная компонента плотности полного потока рассчитывается по соотношению, аналогичному (3.15) (с учетом «неортогональной» составляющей):

$$\begin{aligned} J_{n,k}\varphi_k\Delta A_k &= a_{nb}\left(\Phi_P-\Phi_{nb}\right)+m_k\varphi_k\Delta A_k\Phi_P+\delta\Phi_{PE}\psi\left(P_k\right)\varphi_k\Delta A_k, \\ a_{nb} &= \left[\max(-m_k,0)+d_k\psi\left(P_k\right)\right]\varphi_k\Delta A_k \end{aligned} \tag{4.8}$$

При использовании схем высокого порядка используется соотношение (3.20):

$$J_{n,k}\varphi_k\Delta A_k = a_{nb}\left(\Phi_P - \Phi_{nb}\right) + m_k\varphi_k\Delta A_k\Phi_P,$$
$$a_{nb} = \left[\max(-m_k, 0) - \frac{1}{2}|m_k|\psi\left(r_k\right) + d_k\right]\varphi_k\Delta A_k, \qquad (4.9)$$

в которой параметр r_k рассчитывается по соотношению:

$$r_k = \frac{2\left(\nabla\Phi\right)^* \cdot \mathbf{r}_{P-nb}}{\Phi_{nb} - \Phi_P} - 1, \quad \left(\nabla\Phi\right)^* = \begin{cases} \left(\nabla\Phi\right)_P, & m_k > 0 \\ \left(\nabla\Phi\right)_{nb}, & m_k < 0 \end{cases} \qquad (4.10)$$

4.6 Уравнения движения

При работе с неструктурными сетками используется модель совмещенных скоростей, поэтому дискретные уравнения для компонент вектора скорости являются обычными уравнениями переноса (3.23) (k = x,y,z):

$$a_p U_{p,k} = \sum_{nb} a_{nb} U_{nb,k} + b_k - \Delta V_p \left(\nabla p\right)_p \qquad (4.11)$$

4.7 Источники и граничные условия

Источниковые члены и граничные условия, как поверхностные источники, обрабатываются также, как и для структурных сеток (см. разделы 3.4 и 3.5). Единственное отличие - это расчет расстояния от центра приграничной ячейки до граничной грани δ_w в соотношении (3.30). Для ячейки произвольной формы:

$$\delta_w = \left|\mathbf{n}_f \cdot \left(\mathbf{r}_f - \mathbf{r}_P\right)\right|$$

4.8 Body Force модель источниковых членов

При использовании силовых источников с большими градиентами в зоне резкого изменения источника возникают нефизичные пульсации скорости и давления при использовании стандартного алгоритма Рие-Чоу (см. пункт 4.2). Типичным примером является постоянный источник, заданный в подобласти расчетной области. В этом случае на границе подобласти возникает скачек источника. Как показали различные численные эксперименты, появление паразитных возмущений скоростей и давления связаны с «несогласованностью» значений источника на гранях ячейки и в ее центре. Для базового источника уравнений движения - градиента давления - такое согласование производится в поправке Рие-Чоу при расчете скорости на грани.

В коде Anes реализована модель BodyForce для аппроксимации таких источниковых членов, основанная на модели работы [11], в которой производится сглаживание источников вблизи «разрывных» границ.

Рассмотрим грань и две соседних ячейки, изображенных на рисунке 4.2.

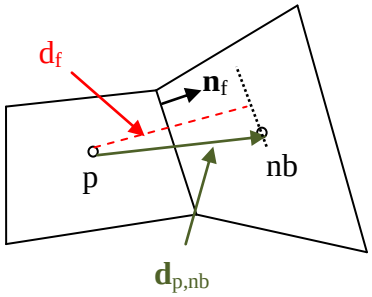


Рисунок 4.2 - Грань и ячейки-соседи

Рассмотрим ситуацию, когда скорость потока равна нулю. В этом случае уравнения движения сводятся к уравнениям гидростатики:

$$\nabla p = \mathbf{F} \tag{4.12}$$

Здесь **F** - вектор объемного силового источника. Проинтегрируем это уравнение вдоль вектора $\mathbf{d}_{p,nb}$:

$$\left(p_{nb} - p_p\right) = \left(\mathbf{F}_f \cdot \mathbf{d}_{p,nb}\right) \quad \text{или} \quad p_{nb} = p_p + \left(\mathbf{F}_f \cdot \mathbf{d}_{p,nb}\right) \tag{4.13}$$

Значение источника на грани $\mathbf{F}_f$ можно рассчитать либо из известных соотношений для грани, либо рассчитать с помощью линейной интерполяции. В алгоритме VOF используется первый вариант [10], в общем случае используется специальная линейная интерполяция. Пусть $\mathbf{F}^0$ - это значения, рассчитанные из исходного поля силы в центрах ячеек. Тогда значения источника на грани можно определить с помощью линейной интерполяции:

$$\mathbf{F}_f = \mathbf{F}_P^0 \cdot f_{p,nb} + \mathbf{F}_E^0 \cdot \left(1 - f_{p,nb}\right) \tag{4.14}$$

Здесь $f_{p,nb}$ - это коэффициент интерполяции f_P (4.2) для грани с ячейкой "nb".

Для того чтобы не возникало паразитных пульсаций давления необходимо, чтобы уравнение (4.12) выполнялось и для самой ячейки "P":

$$\mathbf{F}_p = \frac{1}{\Delta V_p}\int_V \nabla p\, dV = \frac{1}{\Delta V_p}\oint p\mathbf{n}_f\, dA = \frac{1}{\Delta V_p}\sum_{nb} p_f \mathbf{n}_f \Delta A_f \tag{4.15}$$

В коде Anes для расчета градиента давления в ячейке используется именно соотношение (4.15), в котором давление на грани рассчитывается с помощью линейной интерполяции

Для вычисления давления на грани используется та же самая линейная интерполяция:

$$p_f = p_p f_{p,nb} + p_{nb}\left(1 - f_{p,nb}\right) = p_p + \left(1 - f_{p,nb}\right)\cdot\left(\mathbf{F}_f \cdot \mathbf{d}_{p,nb}\right)$$

Подставляя это соотношение в (4.15) получим выражение для расчета источника в ячейке:

$$\mathbf{F}_p = \frac{p_p}{\Delta V_p}\oint \mathbf{n}_f\, dA + \frac{1}{\Delta V_p}\sum_{nb}\left(1 - f_{p,nb}\right)\cdot\left(\mathbf{F}_f \cdot \mathbf{d}_{p,nb}\right)\mathbf{n}_f \Delta A_f$$

Поскольку первый член справа равен нулю (ячейка замкнута), то:

$$\mathbf{F}_p = \frac{1}{\Delta V_p}\sum_{nb}\left(1 - f_{p,nb}\right)\cdot\left(\mathbf{F}_f \cdot \mathbf{d}_{p,nb}\right)\mathbf{n}_f \Delta A_f \tag{4.16}$$

Таким образом, если значения источника на гранях $\mathbf{F}_f$ и в центрах ячеек $\mathbf{F}_p$ связаны соотношениями (4.14) и (4.16), то силы будут согласованы с давлением и паразитные возмущения давления будут отсутствовать. Именно эта идея и лежит в основе BodyForce модели:

1) по стандартному алгоритму Anes рассчитываются «точные» значения источника - $\mathbf{F}^0$;
2) по значениям в ячейках рассчитываются силы на гранях по соотношению (4.14);
3) рассчитываются «размазанные» силовые источники в ячейках по соотношениям (4.16) и именно эти источники в дискретных уравнениях (4.11).

При использовании BodyForce алгоритма производится коррекция и соотношения Рие-Чоу для скорости на грани (4.5). Силовой BodyForce источник «вычленяется» из члена H_p и переносится в выражение для скорости на грани (4.4) так же, как и силовой источник, связанный с градиентом давления. В этом случае в правую часть (4.5) добавляется член:

$$\delta u_f = \overline{\left(\frac{\Delta V_p}{a_p}\right)}_f \mathbf{F}_f \cdot \mathbf{n}_f - \overline{\left(\frac{\Delta V_p}{a_p}\mathbf{F}_p \cdot \mathbf{n}_f\right)}_f \qquad (4.17)$$

В коде Anes источниковые члены задаются в линеаризованном виде (Коэффициент и Значение источника). Для векторных силовых источников эта линеаризация имеет вид:

$$S_k = C_k(V_k - U_k), \quad \mathbf{S} = \mathbf{F}^{(+)} - \mathbf{F}^{(-)},$$
$$\mathbf{F}^{(+)} = \left(F_x^{(+)}, F_y^{(+)}, F_z^{(+)}\right), \quad \mathbf{F}^{(-)} = \left(C_x U_x, C_y U_y, C_z U_z\right) \qquad (4.18)$$

Источники типа $\mathbf{F}^{(+)}$ появляются при моделировании вентиляторов и винтов, в которых вентилятор описывается поверхностным источником, на котором задан скачек давления. Источники $\mathbf{F}^{(-)}$ возникают при описании пористого трения в зонах Porous.

В текущей версии кода BodyForce модель реализована как для $\mathbf{F}^{(+)}$, так и для $\mathbf{F}^{(-)}$. По умолчанию BodyForce модель не используется. Для использования BodyForce для источников, заданных на патче, необходимо в секции [Special Data] включить оператор:

```
L("BodyForce.<ИмяПатча>") = .True.
```

Если силовые источники связаны со всей расчетной областью необходимо указать оператор

```
L("BodyForce.*") = .True.
```

Для моделирования пористого трения в зонах Porous с помощью BodyForce необходимо использовать оператор

```
L("IsBodyForcePorous") = .True.
```

Для моделирования силового источника в VOF-модели двухфазного потока используется оператор

```
L("IsBodyForceVOF") = .True.
```

☞Замечание. В текущей версии BodyForce алгоритма могут возникнуть проблемы при моделировании пористого трения $\mathbf{F}^{(-)}$. Поэтому пока рекомендуется не использовать оператор IsBodyForcePorous.

4.9 Периодические граничные условия

При использовании неструктурных сеток периодические условия (ПГУ) реализуются также, как и для структурных: граничные грани просто объединяются и превращаются в одну внутреннюю грань.

4.10 Наклонные периодические граничные условия

При использовании неструктурных сеток допускаются особые периодические граничные условия – ПГУ на «наклонных» границах. Этот тип геометрии можно использовать для моделирования теплообменных блоков, в основе которых лежат элементы в треуголь-

ной упаковке (см. рисунок 4.3). При расположении элементов в треугольной решетке возникают три геометрических размеров-шагов:

S_y – шаг в вертикальном направлении (основной шаг),

S_x – шаг в горизонтальном направлении,

S_{xy} – шаг в «наклонном» направлении, связанный с предыдущими шагами очевидным соотношением:

$$S_{xy} = \sqrt{\left(\frac{S_y}{2}\right)^2 + S_x^2}$$

Обычно используется симметричная треугольная упаковка, в которой $S_{xy} = S_y$ и сами элементы образуют равносторонний треугольник.

Типичный пример использования наклонных ПГУ – задача о течении в элементе симметрии активной зоны реактора, состоящего из твэлов с проволочной навивкой.

Этот тип ПГУ можно применять, если:

1) рассматривается трехмерная задача и неструктурные сетки,
2) нет дробления вдоль оси z или нет дробления вообще (максимальный уровень дробления равен нулю).

Геометрия расчетной области (РО) для случая наклонных периодических условий показана на рисунке 4.3. В этом геометрии используются три периодические границы:

$Y_1 – Y_2$ – периодические границы вдоль оси y,

$XY_a – XY_c$ – первые наклонные периодические границы,

$XY_b – XY_d$ – вторые наклонные периодические границы.

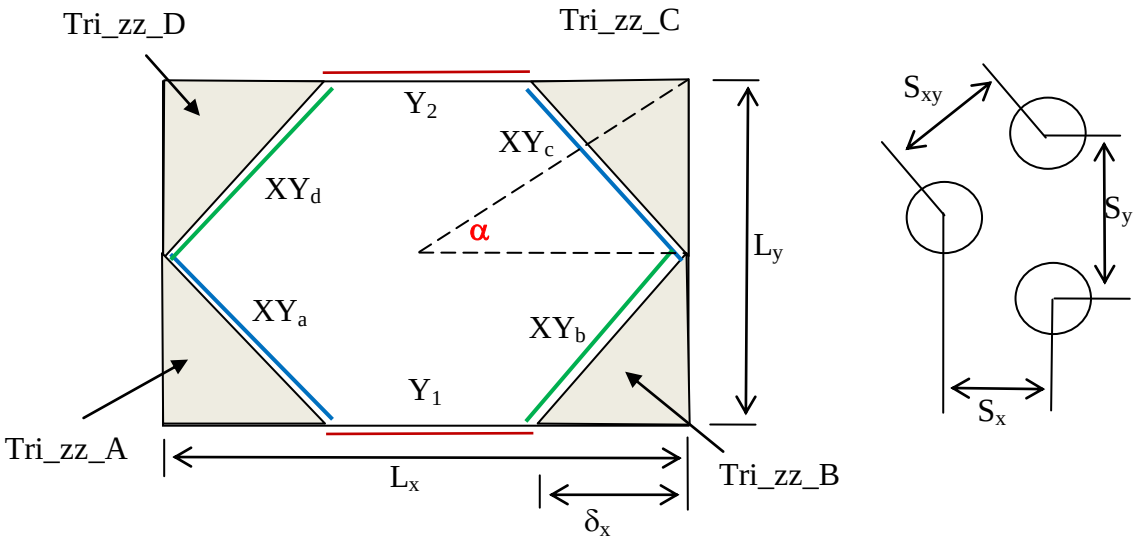


Рисунок 4.3 – Реализация наклонных РГУ

Расчетная область с наклонными периодическими границами создается следующим образом.

Пользователь задает размеры РО в плоскости (x,y) – L_x и L_y и включает флаги активации «плоских» ПГУ в секции [Main] проекта:

```
IsCycleBCX = .True.
IsCycleBCY = .True.
```

и флаг активации наклонных ПГУ в секции описания неструктурной сетки [Unstructed Cartesian Grid]:

```
IsCycleBC_XY = .TRUE.
```

Все остальное делается компилятором Anes автоматически. Генератор сетки создает четыре патча - Tri_ZZ_A, Tri_ZZ_B, Tri_ZZ_C и Tri_ZZ_D – треугольной формы, используя для расчета δ_x соотношение:

$$\sin(2\alpha)=\frac{L_y}{L_x},\quad \delta_x=\frac{L_y}{2}\mathrm{tg}(\alpha)$$

При «идеальном» построении неструктурной сетки эти патчи удаляются, а их наклонные грани превращаются в периодические границы. Для визуального представления периодических границ генератор создает три поверхностных патча:

CYCLE_BS_Y - для представления границ $Y_1 - Y_2$,

CYCLE_BS_AXY - для представления границ $XY_a - XY_c$,

CYCLE_BS_BXY - для представления границ $XY_b - XY_d$.

Для некоторых геометрий объектов внутри РО генератор не может из-за ошибок округления «замкнуть» все периодические грани. В этом случае в РО могут остаться единичные грани патчей Tri_ZZ_A, Tri_ZZ_B, Tri_ZZ_C и Tri_ZZ_D и CYCLE_BS_X. На этих гранях в расчете будут заданы не периодические, а «адиабатические» граничные условия. В этом нет ничего страшного, поскольку таких граней будет очень мало.

Пример сетки с наклонными ПГУ показан на рисунке 4.4.

Приведем некоторые соотношения для выбора размеров РО. Если известны шаги расположения элементов S_y и S_x, то размеры РО легко получить из соотношений:

$$\mathrm{tg}(\alpha)=\frac{S_y}{2S_x},\quad S_{xy}=\sqrt{\left(\frac{S_y}{2}\right)^2+S_x^2},$$

$$L_x=\frac{S_{xy}}{\cos(\alpha)},\quad L_y=S_y$$

Для наиболее типичного случая равносторонней треугольной упаковки

$$S_{xy}=S_y,\quad L_x=\frac{2}{\sqrt{3}}S_y,\ L_y=S_y$$

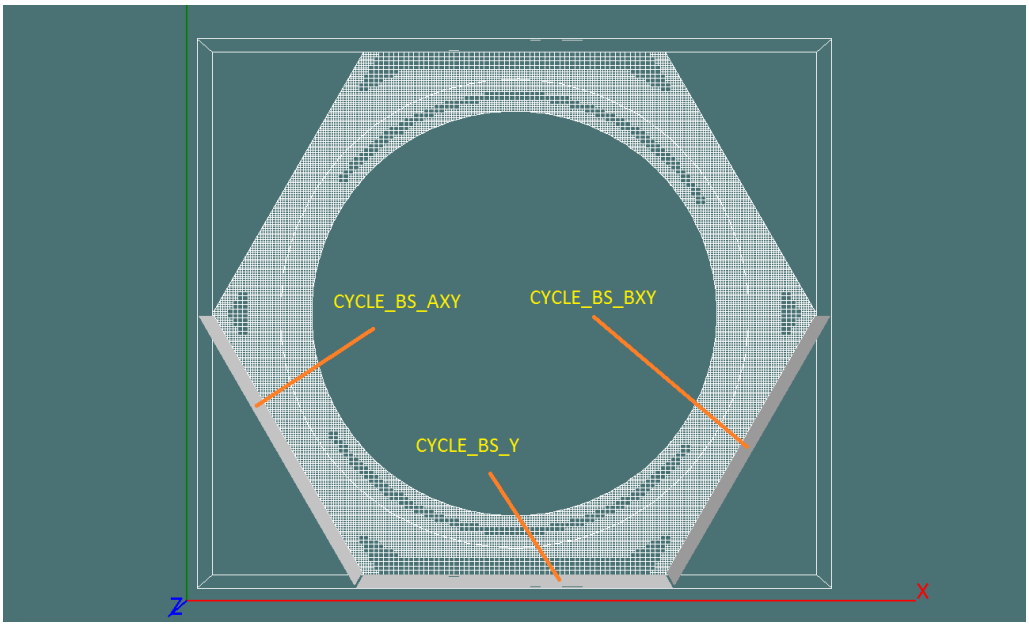


Рисунок 4.4 – Пример реализации наклонных РГУ

Генератор сетки в листинг компилятора выводит информацию о шагах элементов, «восстановленных» из размеров РО L_x и L_y. Эту информацию можно использовать Для проверки правильности задания размеров:

---------(Параметры наклонных ПГУ)---------
1.Угол Alfa.....= 29.9999
2.StepX.....= 1.229756E-02
3.StepY.....= 1.420000E-02
4.StepXY.....= 1.420000E-02
5.DeltaX.....= 4.099187E-03

5. Общие конечно-разностные алгоритмы

5.1 Модель поправки для дискретных уравнений

Уравнения сохранения для Ф-переменной на дискретном уровне для структурных и неструктурных сеток сводится к системе нелинейных алгебраических уравнений вида (3.20):

$$a_P\Phi_P = \sum_{nb} a_{nb}\Phi_{nb} + b_P \qquad (5.1)$$

где P - текущий КО, nb - индекс КО-соседа. Суммирование в (5.1) производится по всем КО-соседям. При использовании структурной сетки в трехмерном случае для внутренней ячейки число соседей равно 6. При решении нестационарной задачи в число соседей включается «копия» данного КО с предыдущего шага по времени (член a_T в уравнении (3.20)).

Система уравнений (5.1) состоит из N_C = (NX-2)*(NY-2)*(NZ-2) для структурной сетки и N_C = NoCells для неструктурной сетки. Эта система уравнений в общем случае является нелинейной, поскольку коэффициенты a_P,a_{nb}, b_P могут зависеть от самих Ф-переменных. Для решения такой системы необходимо использовать итерационный процесс, при котором на каждой итерации система уравнений (5.1) приводится к системе линейных уравнений с постоянными коэффициентами. Сам процесс линеаризации сводится просто к вычислению коэффициентов уравнений по значениям Ф-переменных, взятых с предыдущей итерации. Линейная система уравнений (5.1) на каждой итерации решается с помощью одного из «линейных» LES-солверов (Linear Equation Solver), реализованных в коде. Все LES-солверы для решения используют свой *внутренний* итерационный цикл, поскольку эффективных «прямых» методов решения линейных систем такой большой размерности не существует. Для отличия внешних и внутренних итераций в коде Anes внешний цикл итераций называется SWEEP-итерациями.

При таком двухуровневом итерационном процессе не имеет смысла добиваться полной сходимости итерационного процесса LES-солвера, поскольку на следующей итерации это решение будет снова «испорчено» пересчетом коэффициентов уравнений. Поэтому в пакете предусмотрены определенные средства оптимизации итерационных циклов LES-солверов.

Линеаризованную форму уравнений (3.29) можно представить в другой форме, более удобной для LES-солвера. Введем *поправку* (коррекцию) для Ф-переменной на текущей итерации

$$\Phi_P = \Phi_P^* + \Phi'_P \qquad (5.2)$$

где Φ_P - искомое поле на текущей итерации, Φ_P^* - значение с предыдущей итерации. В этом случае дискретное уравнение можно записать в следующем виде

$$\begin{aligned} a_P\Phi'_P &= \sum_{nb} a_{nb}\Phi'_{nb} + r_P^*, \\ r_P &= b_P^* + \sum_{nb} a_{nb}^*\Phi_{nb}^* - a_P^*\Phi_P^* \end{aligned} \qquad (5.3)$$

В этом уравнении r_P - это невязка решения дискретного уравнения, верхний символ * означает, что коэффициенты рассчитываются по полям с предыдущей итерации. Несмотря на то, что с точки зрения математики уравнения (5.1) и (5.3) идентичны, с точки зрения численных алгоритмов форма (5.3) более предпочтительна:

1. Система (5.3) является системой *линейных* уравнений с *постоянными* коэффициентами, а система (5.3) нелинейная в общем случае.

2. Если итерационный процесс решения системы (5.3) получил «точное» решение (с учетом ошибок округления), то поля Ф' и r_P будут равны нулю. Поэтому интегральные характеристики этих полей можно использовать для оценки процесса сходимости.
3. Коэффициенты a_P и a_{nb} в дискретном уравнении и и коэффициенты a_P^*, a_{nb}^* в невязке r_P могут быть *различными*. Главное, чтобы было получено сошедшееся решение. Этот подход лежит в основе алгоритма «отложенной» коррекции (deferred correction) [4], который используется при работе с неструктурными сетками и схемами второго порядка. В этом случае при расчете невязки используется выбранная численная схема для коррекции схемной диффузии (например, степенная схема или схема 2-го порядка), а для расчета коэффициентов a_{nb} используется схема *против потока*.

В коде Решателя используется три критерия окончания внешнего итерационного процесса и ограничения внутренних итераций LES-солвера.
Первый критерий - максимальное значение невязки:

$$e_\Phi = \frac{\max\left(|r_P|\right)}{M_r} N_C \qquad (5.4)$$

Второй критерий - среднеарифметическое значение невязки:

$$e_\Phi = \frac{\sum |r_P|}{M_r} \qquad (5.5)$$

Третий критерий – максимальное значение поправки:

$$e_\Phi = \frac{\max\left(\delta\Phi_P\right)}{M_\Phi} \qquad (5.6)$$

В коде в качестве масштаба невязки используется сумма абсолютных величин всех полных потоков на всех гранях

$$M_r = \sum \left|a_{nb}(\Phi_P - \Phi_{nb})\right|$$

Итерационный процесс прекращается (считается *сошедшимся*), если безразмерные невязки для всех Ф-переменных становятся меньше заданной точности

$$e_\Phi < \varepsilon \qquad (5.7)$$

Обычно значение точности ε лежит в пределах 10^{-4} .. 10^{-5}.

Для дополнительного контроля над процессом сходимости Решатель рассчитывает интегральные балансы для каждой Ф-переменной. Для этого рассчитывается алгебраическая сумма невязок

$$B_\Phi = \frac{\sum r_P}{M_r} N_C$$

Поскольку потоки через внутренние грани КО компенсируются при суммировании невязок, то в B_Φ останутся составляющие, связанные только с источниками и граничными потоками.

Замечание: при использовании степенной схемы Патанкара можно использовать два подхода для расчета коэффициентов дискретного уравнения для поправки:

1) для коэффициентов a_{nb} используется степенная схема (как и для «точных» коэффициентов a_{nb}*),
2) для коэффициентов a_{nb} используется схема против потока.

Использование первого или второго варианта зависит от задачи. Первый вариант обычно дает более быструю сходимость, однако в ряде задач он может приводить к расходимости

итерационного потока. Чаще всего это связано с занулением коэффициентов a_{nb} в пристенных ячейках.

По умолчанию используется первый вариант. Для выбора варианта используется оператор IsUpCoefFVE секции [Special Data].

ⓘ *Описание в проекте прикладной задачи :*

Точность окончания итераций e_Φ для Ф-переменной с именем "Ф-имя" задается оператором секции [PHI Variables]:

Eps("Ф-имя") = 1.0e-5

Если это значение не задано, то используется «значение по умолчанию» из оператора секции [Solvers]:

Eps = 1.0e-5

Для выбора алгоритма (3.32) - (3.34) используется оператор секции [Solvers]:

TypeEndSweep = TES_MAXRES / TES_TOTRES / TES_MAXCOR

Для использования противопоточной схемы для расчета коэффициентов a_{nb} используется оператор секции [Special Data]:

IsUpCoefFVE =.TRUE.

5.2 Управление сходимостью итераций. Релаксация

Часто итерационный процесс расходится при использовании любых линейных солверов. Есть две причины такой расходимости:

1. сильная нелинейность коэффициентов дискретного уравнения,
2. ограничения алгоритмов решения уравнений гидродинамики (см. следующую главу).

Для улучшения сходимости итерационного процесса в пакете предусмотрена возможность использования двух типов релаксации: линейной нижней релаксации и FTS-релаксации (False Time Step - псевдо нестационарная релаксация).

При использовании формы дискретных уравнений через поправку (3.31) значение на внешней итерации при использовании линейной нижней релаксации получается из соотношения

$$\Phi_P = \Phi_P^* + \alpha_\Phi \Phi'_P \qquad (5.8)$$

Здесь α_Φ - коэффициент линейной нижней релаксации ($0 < \alpha_\Phi \le 1$). Использование нижней релаксации позволяет ограничить изменение переменной на внешних итерациях и тем самым предотвратить расходимость (или «болтание») итерационного процесса. В коде Решателя соотношение (5.8) не используется напрямую. Следуя идее Патанкара, коэффициент релаксации добавляется в линеаризованное дискретное уравнение

$$a_P \to \frac{a_P}{\alpha_\Phi}$$

В коде механизм нижней релаксации предусмотрен не только для Ф-переменных, но и для всех вспомогательных полей (плотности, вязкости и т.д.).

Иногда улучшение сходимости итерационного процесса можно добиться, ограничив изменение Ф-переменной. Для этого в пакете предусмотрены ограничения на минимальное и максимальное значение переменной.

Наряду с нижней релаксацией при решении *стационарной задачи* можно использовать второй подход (FTS-релаксация), основанный на добавлении в дискретное уравнение (5.1) псевдо-нестационарного члена

$$a_P \Phi_P = \sum_{nb} a_{nb} \Phi_{nb} + b_{nb} + \rho_P \frac{\Delta V_P}{\Delta \tau_F} \left(\Phi_P - \Phi_P^* \right)$$

Здесь $\Delta\tau_F$ – псевдо-шаг по времени. При переходе к поправке:

$$a_P \to a_P + \rho_P \frac{\Delta V_P}{\Delta \tau_F}$$

Заметим, что коэффициент нижней релаксации безразмерен, а коэффициент FTS-релаксации имеет размерность времени.

Какую релаксацию использовать – зависит от прикладной задачи. Например, при решении задач с вязкостью равной нулю (невязкие течения и уравнения гидродинамики в форме уравнений Эйлера) коэффициент a_P может быть равен нулю в областях с нулевой скоростью. В этом случае только FTS-релаксация может обеспечить получение решения.

ⓘ *Описание в проекте прикладной задачи :*
Для Ф-переменных коэффициент релаксации задается оператором секции [PHI Variables]
Relax("<Ф-имя>") = <значение>

Если значение положительно, то используется механизм линейной нижней релаксации. Если значение отрицательно, то используется алгоритм FTS-релаксации. В этом случае |<значение>| - это $\Delta\tau_F$.

Для задания релаксации для вспомогательных переменных необходимо использовать операторы секции [Special Data]
R("Relax.<V-Имя>") = <значение>

Ограничение изменения значения Ф-переменной описывается операторами
MinValue("<Ф-имя>") = <значение>
MaxValue ("<Ф-имя>") = <значение>
секции [PHI Variables].

5.3 Алгоритм PEA

При моделировании двухтемпературных пористых систем возникает проблема сходимости решений при больших коэффициентах межфазной теплоотдачи. При больших значениях этого коэффициента температура G-фазы T_g и температура S-фазы T_s асимптотически стремятся друг к другу и двухтемпературная модель переходит в однотемпературную. При последовательном решении на итерации уравнений (а именно такой алгоритм используется в коде) такой переход приводит к расходимости итераций. Для получения сходимости в коде Anes используется оригинальный алгоритм, предложенный Б. Сполдингом - алгоритм PEA (Partial Elimination Algorithm).

Рассмотрим дискретные уравнения для T_g (прописные символы) и T_s (строчные символы), в которых явно выделим межфазный теплообмен:

$$\begin{aligned} A_P T_P &= \sum_{nb} A_{nb} T_{nb} + B_P + C_P(t_P - T_P), \\ a_P t_P &= \sum_{nb} a_{nb} t_{nb} + b_P + c_P(T_P - t_P) \end{aligned} \qquad (5.9)$$

Выразим из второго уравнения температуру t_P и подставим ее в первое уравнение:

$$(C_P + A_P)T_P = \sum_{nb} A_{nb} T_{nb} + B_P + \frac{C_P}{a_P + c_P}\left(\sum_{nb} a_{nb} t_{nb} + b_P + c_P T_P\right)$$

Проделав элементарные преобразования, уравнение для T_g можно записать в виде:

$$\begin{aligned} \tilde{A}_P T_P &= \sum_{nb} A_{nb} T_{nb} + \tilde{B}_P, \\ \tilde{A}_P = A_P + \frac{C_P}{C_P + c_P} a_P, &\quad \tilde{B}_P = B_P + \frac{C_P}{C_P + c_P}\left[\sum_{nb} a_{nb} t_{nb} + b_P\right] \end{aligned} \qquad (5.10)$$

Аналогичное «симметричное» уравнение можно получить и для температуры T_s.

При $C_P = c_P \to \infty$ уравнение (5.10) переходит в уравнение однотемпературной модели:

$$(A_P + a_P)T_P = \sum_{nb}(A_{nb} + a_{nb})T_{nb} + B_P + b_P$$

Использование PEA алгоритма позволяет решать уравнения для температур последовательно на итерации и получать хорошую сходимость для любых значений межфазного обмена.

Заметим, что подобный алгоритм используется и при решении уравнений для концентраций. Он гарантирует, что сумма концентраций в любом КО равна единице.

5.4 Особенности отдельных Ф-переменных

5.4.1 Уравнение энергии для G-фазы

Уравнение энергии для однокомпонентной (уравнение (4.14) документа [1]) и многокомпонентной G-фазы (уравнение (4.38) документа [1]) записаны с использованием двух полей: энтальпии H_g и температуры T_g. При получении дискретного уравнения энтальпия в конвективной части плотности потока тепла на грани КО исключается из дискретного уравнения.

Продемонстрируем этот алгоритм на примере расчета *конвективной* составляющей потока $J_{c,e}$ на грани КО "e", который входит в дискретное уравнение (3.23) в виде (3.20):

$$\left(J_e - m_e H_P\right) = \max(-m_e, 0)\cdot(H_P - H_E) \qquad (5.11)$$

При использовании итерационных алгоритмов решения дискретных уравнений конвективный член (5.11) можно переписать в виде:

$$\left(J_e - m_e H_P\right) = \max(-m_e, 0)\cdot c_e \cdot (T_P - T_E), \qquad c_e = \frac{H_E^* - H_P^*}{T_E^* - T_P^*} \qquad (5.12)$$

Здесь c_e – эффективная средняя теплоемкость на грани КО, вычисляемая по полям энтальпии и температуры, взятыми с предыдущей итерации. В коде во избежание ошибок округления, при малых перепадах температур

$$\left|T_E^* - T_P^*\right| < dT_{min}$$

в качестве значения c_e используется теплоемкость c_{pg}. Величина dT_{min} задается пользователем в файле описания проекта. Отметим, что использование такого алгоритма требует, чтобы энтальпия h_g и теплоемкость c_{pg} (определяемые пользователем) должны быть согласованными между собой, т.е.

$$c_{pg} = \left(\frac{\partial h_g}{\partial T_g}\right)_p .$$

Описанный подход исключения энтальпии используется также при описании источниковых членов и граничных условий.

ⓘ *Описание в проекте прикладной задачи :*

Значение dT_{min} задается оператором секции [Properties]:

DelTMIN = <значение>

5.4.2 Уравнение энергии S-фазы

В коде Anes предусмотрена возможность задания термического сопротивления на границе раздела S-зон с разными материалами. Для этого используется автоматические патчи с именем "SS_..." и псевдо-источник с параметрами C_{Ts} = SR_CONTRES, $V_{Ts} = R_T$. В этом случае для грани, принадлежавшей патчу, коэффициент диффузионного переноса (3.11) рассчитывается по соотношению:

$$d_k = \frac{1}{\frac{1}{\alpha_P} + \frac{1}{\alpha_{nb}} + R_T} \qquad (5.13)$$

5.4.3 Межфазное трение на границах раздела G- и S-фаз

На явно выделенной G-S межфазной границе, которая описывается патчами с именами "FS_..", для компонент вектора скорости автоматически задаются условия прилипания. На гранях патчей (см. рисунок 5.1) задаются поверхностные источники для КО с G-фазой:

$$S_{Ugi}^{(gs)}\Delta A_w = -\varphi_P \frac{\mu_P^*}{\delta_g} U_{gi}\Delta A_w \qquad (5.14)$$

где δ_g - расстояние от центра КО до грани патча "FS_…", μ_P^* - *эффективное приграничное* значение коэффициента эффективной вязкости в КО. По умолчанию - это значение коэффициента эффективной вязкости в КО с узловой точкой P, при использовании модели турбулентности с пристенными функциями, это значение может существенно отличаться от нее.

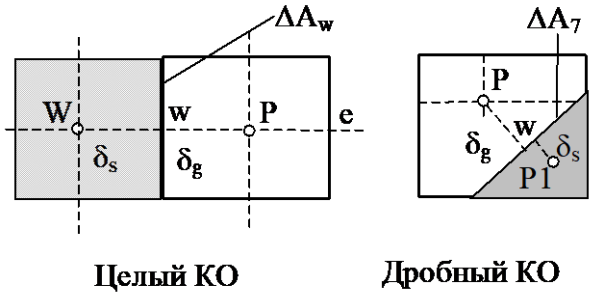


Рисунок 5.1 - Граница раздела G-S фаз

5.4.4 Межфазный теплообмен на границах раздела G- и S-фаз

Межфазный теплообмен между G и S фазами на явно выделенных границах раздела G- и S-фаз сводится к сопряжению на границе плотностей тепловых потоков [1]:

$$\left(\lambda_{eff}\nabla T_g\right)\cdot \mathbf{n}_{sg} = \left(\lambda_s \nabla T_s\right)\cdot \mathbf{n}_{sg} \qquad (5.15)$$

где $\mathbf{n}_{sg}$ - единичная нормаль к межфазной границе, направленная в сторону G-фазы, λ_s и T_s - коэффициент теплопроводности и температура S-фазы.

Соотношение (5.15) моделируется двумя *поверхностными источниками* для КО с G- и S- фазами, которые в дискретном виде можно записать в виде (см. рисунок 5.1):

$$S_{Tg}^{(gs)}\Delta A_w = \varphi_P \alpha_{sg}\left(T_{s,W} - T_{g,P}\right)\Delta A_w, \qquad S_{Ts}^{(gs)}\Delta A_w = \alpha_{sg}\left(T_{g,P} - T_{s,W}\right)\Delta A_w,$$

$$\alpha_{sg} = \frac{1}{\frac{\delta_g}{\lambda_{g,P}^*} + \frac{\delta_s}{\lambda_{s,W}}} \qquad (5.16)$$

где δ_g , δ_s - расстояния от центра КО до грани патча "FS_…", $\lambda_{g,P}^*$ - *эффективное приграничное* значение коэффициента эффективной теплопроводности в КО, (по умолчанию - это значение коэффициента эффективной вязкости в КО с узловой точкой P, при использовании модели турбулентности с пристенными функциями, это значение может существенно отличаться от нее), $\lambda_{s,W}$ - коэффициент теплопроводности S-фазы в приграничном КО с S-фазой.

Пользователь может изменить эти автоматические условия «сопряжения». Для этого необходимо задать *источники пользователя*, связанные с патчами "FS_...". В этом случае FS-патчи рассматриваются как обычные поверхностные патчи, а источники пользователя – как обычные поверхностные источники. Есть только две особенности:

1. Источники пользователя *не заменяют* источники (3.54), а *добавляются* к ним.

2. При задании источника пользователя для T_g автоматически формируется сопряженный источник для T_s.

Поясним последний пункт.

Пусть пользователь задает *только* источник для T_g в виде (см. разделы 4.3.2 и 4.5.2 документа [1]):

$$S_{Tg} = \max(M_P,0)\cdot H_0 - \max(-M_P,0)\cdot H_g + C_T\left(V_T - T_g\right) + \max(M_P,0)\cdot \Delta H_0$$

или при наличии концентраций:

$$S_{Tg} = \sum \max(M_{Ck},0)\cdot H_{k,0} - \sum \max(-M_{Ck},0)\cdot H_k + C_T\left(V_T - T_g\right) + \sum \max(M_{Ck},0)\cdot \Delta H_{0k}$$

задав при этом три параметра:

C_T - коэффициент источника,

V_T - значение источника,

ΔH_0 (или ΔH_{0k}) - тепловые эффекты G-фазы при переходе через границу.

При обработке этого источника Решатель кода Anes добавит источники (5.16) к окончательному источнику. Т.е. Решатель сформирует следующий *окончательный* источник для T_g (для простоты приведен источник для однокомпонентной G-фазы):

$$S_{Tg}^{(gs)} = \max(M_P,0)\cdot H_g\left(T_{s,W}\right) - \max(-M_P,0)\cdot H_g\left(T_{g,P}\right) + \left(C_S + \alpha_{gs}\right)\left(T_{s,W} - T_{g,P}\right) + \max(M_P,0)\cdot \Delta H_0 \qquad (5.17)$$

Одновременно будет автоматически сформирован *окончательный* источник для T_s:

$$S_{Ts}^{(gs)} = \left(\max(-M_P,0)\cdot \bar{c} + C_S + \alpha_{sg}\right)\left(T_{g,P} - T_{s,W}\right) - \max(-M_P,0)\cdot \Delta H_0,$$

$$\bar{c} = \frac{H_g(T_{g,P}) - H_{g,W}(T_s)}{T_{g,P} - T_{s,W}} \qquad (5.18)$$

Отметим особенности этих источников:

1. Значение источника T_0 не используется, вместо нее используется температура S-фазы.
2. T_g и T_s в этих источниках – это температуры в приграничных к FS-границе КО.
3. Коэффициент источника C_S добавляется в оба источника, чтобы выполнить равенство полных потоков. Для большинства задач проще всего положить этот коэффициент равным нулю. Для этого нужно указать C_S = SR_MASSONLY.

5.4.5 Описание химических реакций

При протекании в G-фазы гомогенных химических реакций возникают объемные массовые потоки компонентов, связанных законом сохранения массы

$$\sum_k M_{Ck} = 0$$

При этом в уравнении энергии смеси, записанном в балансовом «дивергентном» виде (уравнение (4.38) в документе [1]), *не возникает ни какого* теплового источника, если отсчеты энтальпий компонентов H_k^0 *согласованы* (т.е. они представляют собой теплоты образования компонентов при $T=T_0$ или $T_g = 0$).

В связи с этим для корректного описания тепловых эффектов химических реакций в коде Anes необходимо:

1) согласовать переменные H_k^0,
2) для объемного патча с именем «*» (т.е. для всей расчетной области) задать постоянный *нулевой* источник: C_{Tg} = SR_FIXSOURCE, T0 = 0

Важно отметить, что нулевой источник обязательно должен быть описан. Это связано с численным алгоритмом кода: если для патча с массовыми источниками *не задан явно* источник для T_g, то будет сформирован источник по умолчанию

$$S_{Tg} = \sum_k S_{Ck} H_{gk} \quad ,$$

чтобы в дискретном аналоге появился нулевой *приведенный* источник для данного патча

$$\tilde{S}_{Tg,P} = \left(S_{Tg,P} - \sum_k S_{Ck,P} H_{gk,P} \right) = 0$$

Если нулевой источник будет *явно задан* с C_{Tg} = SR_FIXSOURCE, то будет сформирован действительно нулевой источник

$S_{Tg} = 0$,

которому в дискретном аналоге соответствует член (3.23):

$$\varphi_P \left(0 - \sum_k S_{Ck,P} \left(H_{gk} + H_k^0 \right)_P \right) \Delta V_P \, .$$

Заметим, что вторая составляющая в этом члене на *дискретном уровне* и является тепловым эффектом реакции.

5.5 Расчет градиента давления в стабилизированных задачах

В задачах моделирования течения на стабилизированном участке канала (см. разделы 4.2.6 и 4.2.7 документа [1]) в уравнении для UgZ появляется постоянный градиент давления $(dP/dz)_0$. Если пользователь задает полный расход через канал G_0

$$G_0 = \iint_{xy} \rho_g \varphi U_{gz} dxdy \, , \qquad (5.19)$$

то градиент давления неизвестен и его необходимо определить. В коде ANES реализовано несколько алгоритмов расчета градиента, выбор которых осуществляется оператором секции [Special Data]:

C("ModelMassRate") = "DivDP" / "Newton" / "DBS" / "SIMPLE"

5.5.1 Алгоритм DivDP

В этом алгоритме решается дискретное уравнение не для скорости U_{gz} , а для переменной $V = Ugz/(dP/dz)_0$:

$$a_P V_P = \sum_{nb} a_{nb} V_{nb} + \frac{b_P}{\left(dP \big/ dz \right)_0^*} + 1 \cdot \Delta V_p \, ,$$

в котором градиент давления берется с предыдущей итерации. Новое значение градиента получается из соотношения:

$$\left(\frac{dp}{dz} \right)_0 = \frac{G_0}{\iint_{xy} \rho_g \varphi V dxdy} \qquad (5.20)$$

Этот алгоритм является эффективным для большинства одномерных и двумерных стабилизированных задач и для задач с периодическими граничными условиями. Однако при сильно нелинейных источников b_p может наблюдаться расходимость итерационного процесса. В этом случае можно попробовать использовать другие алгоритмы.

5.5.2 Алгоритм DBS

Этот алгоритм (предложен профессором Б. Сполдингом) основан на соотношении для коэффициента гидравлического сопротивления:

$$\xi = \left(\frac{dP}{dz} \right) \frac{2 d_h}{\rho U_m^2}, \quad U_m = \frac{G_0}{\rho S_c}, \quad \xi = f(G_0) \qquad (5.21)$$

Здесь S_c - площадь поперечного сечения канала, d_h - гидравлический диаметр, U_m - средняя скорость.

Их этого соотношения можно получить итерационное соотношение для расчета градиента:

$$\left(\frac{dP}{dz}\right)=\left(\frac{dP}{dz}\right)^{*}(1-\alpha_{dp})+\alpha_{dp}\left(\frac{dP}{dz}\right)^{*}\left[\frac{G_0}{G^{*}}\right]^{m}, \qquad (5.22)$$

где символ * означает значения с предыдущей итерации, α_{dp} - коэффициент релаксации.

Показатель степени "m" зависит от модели турбулентности. Для ламинарных течений m = 1 (коэффициент сопротивления обратно пропорционален расходу), для турбулентных течений m = 2.

Перерасчет градиента давления производится не на каждой итерации, а с шагом StepDPDZ. Значение этого шага для Sweep - итераций может меняться в широких пределах (StepDPDZ = 10 ... 1000) и подбирается для конкретной задачи экспериментально. На скорость сходимости оказывает существенное влияние начальное значение градиента и коэффициент релаксации α_{dp}.

Для ограничения интервала изменения градиента давления можно задать допустимые границы его изменения (минимальное и максимальное значения).

Эти параметры задаются операторами секций:

```
[Internal Source]
  DEV_DPDZ = <начальное значение градиента>
.......
[Special Data]
  R("Relax.DPDZ") = <коэффициент релаксации>
  R("Min.DPDZ") = <минимальное значение>
  R("Max.DPDZ") = <максимальное значение>
  I("StepDPDZ") = <шаг смены градиента>
  C("ModelMassRate") = "DBS"
```

5.5.3 Алгоритм Newton

Этот алгоритм основан на решении нелинейного уравнения G=f(dP/dz) методом Ньютона. В этом случае итерационное соотношение для расчета градиента имеет вид:

$$\left(\frac{dP}{dz}\right)=\left(\frac{dP}{dz}\right)_1+\alpha_{dp}\frac{G_0-G_2}{\left(\frac{dG}{dP}\right)},\quad \frac{dG}{dP}=\frac{G_2-G_1}{\left(\frac{dP}{dz}\right)_2-\left(\frac{dP}{dz}\right)_1} \qquad (5.23)$$

Здесь индексы 2 и 1 означают значения на двух «соседних» итерациях пересчета градиента давления (с шагом StepDPDZ). Данный алгоритм, как и DBS-алгоритм, использует релаксацию, начальное значение градиента и шаг пересчета на итерациях.

5.5.4 Алгоритм SIMPLE

Этот алгоритм расчета градиента используется в нестационарных задачах LES моделирования. В настоящее время он находится в процессе разработки.

6. Алгоритмы решения уравнений гидродинамики

Как уже отмечалось в документе [1] важной особенностью системы уравнений Навье-Стокса является отсутствие «явного» уравнения для давления и присутствие «лишнего» уравнения - уравнения неразрывности. Это противоречие можно разрешить, если получить уравнение для давления из уравнения неразрывности. Неформально этот путь можно продемонстрировать с помощью следующих соображений (идея взята из описания алгоритмов кода OpenFoam). Если отбросить непринципиальные нюансы, то дискретное уравнение для вектора скорости (4.11) можно записать в векторном виде:

$$\mathbf{U} = \frac{\mathbf{H}'(\mathrm{U})}{a_P} - \left(\frac{\Delta V_P}{a_P}\right)(\nabla p), \quad \mathbf{H}'(\mathrm{U}) = \sum_{nb} a_{nb}\mathbf{U}_{nb} + \mathbf{b}$$

Подставляя это выражение в уравнение неразрывности:

$$\frac{\partial \rho}{\partial \tau} + \mathrm{div}(\rho \mathbf{U}) = 0,$$

можно получить «явное» уравнение для давления:

$$\mathrm{div}\left[\left(\frac{\Delta V_P}{a_P}\right)(\nabla p)\right] = \frac{\partial \rho}{\partial \tau} + \mathrm{div}\left[\frac{\mathbf{H}'(\mathrm{U})}{a_P}\right]$$

Эта идея послужила основой большого числа алгоритмов для совместного решения уравнений движения и уравнения неразрывности [3,4]: SIMPLE, SIMPLER, SIMPLEST, PISO и т.д.

В текущей версии кода Anes реализованы два алгоритма:

1. алгоритм SIMPLE для решения стационарных и нестационарных задач,
2. алгоритм PIMPLE - комбинация алгоритмов SIMPLE и PISO - для решения нестационарных задач.

6.1 Алгоритм SIMPLE

Для решения уравнений гидродинамики (Ф-переменные P_f, U_x, U_y, U_z) в коде используются алгоритм SIMPLE (для простоты ниже опускается индекс "g" у переменных G-фазы и полагается что пористость равна 1)..

Рассмотрим случай структурных сеток, при использовании которых дискретные уравнения для компонент вектора скорости содержат значения скоростей на гранях КО (здесь k - индекс грани, n_k - внешняя нормаль грани, nb - индекс КО-соседа через k-грань):

$$U_k = \frac{H_k(U_k)}{a_{p,k}} + \frac{n_k C_k}{a_{p,k}}(p_p - p_{nb}) + b_k, \quad H_k = \sum_{nb} a_{k,nb} U_{k,nb} \tag{6.1}$$

Уравнение неразрывности в дискретном виде можно записать в виде:

$$\frac{\rho_P - \rho_P^0}{\Delta \tau}\Delta V_P + \sum_k \rho_k U_k n_k \Delta A_k = 0 \tag{6.2}$$

В алгоритме SIMPLE поля Ф-переменных рассчитываются во внешнем итерационном цикле. Ниже значения переменных, рассчитанные по полям на предыдущей итерации, будут помечаться верхним индексом "m".

На каждой итерации уравнения гидродинамики содержит два шага: предиктор и корректор.

На *предиктор-шаге* с помощью LES солвера решаются уравнения движения для компонент скорости U_k*:

$$U_k^* = \frac{H_k^m(U_k^*)}{a_{p,k}^m} + \frac{n_k C_k}{a_{p,k}^m}(p_p^m - p_{nb}^m) + b_k^m \tag{6.3}$$

На первой итерации (m = 0) в качестве полей для расчета коэффициентов используются поля с предыдущего шага по времени (для нестационарных задач) или из начального распределения полей (для стационарных задач).

Полученное поле скорости U_k^* не удовлетворяет уравнению неразрывности (6.2). Для выполнения уравнения неразрывности используется корректор-шаг.

Рассчитаем новое поле скорости U_k^{**} из *полу-неявного* аналога уравнения (3.6):

$$U_k^{**} = \frac{H_k^m(U_k^*)}{a_{p,k}^m} + \frac{n_k C_k}{a_{p,k}^m}(p_p^* - p_{nb}^*) + b_k^m \qquad (6.4)$$

так, чтобы поле скорости U_k^{**} удовлетворяло уравнению баланса массы (6.2). Подставляя уравнение (6.4) в уравнение неразрывности, можно получить уравнение для давления p*. В коде Anes используется модель поправки Ф-переменной, поэтому уравнение для давления удобнее записать через поправку. Для перехода к поправкам, вычтем уравнение (6.3) из (6.4):

$$U_k^{**} = U_k^* + U_k', \quad U_k' = \frac{n_k C_k}{a_{p,k}^m}(p_p' - p_k'), \quad p^* = p^m + p' \qquad (6.5)$$

В итоге получаем уравнение для поправки давления в виде:

$$\sum_k \rho_k^m \frac{\Delta A_k C_k}{a_{p,k}^m}\left(p_p' - p_k'\right) = -\left\{\frac{\rho_P^m - \rho_P^0}{\Delta\tau}\Delta V_P + \sum_k \rho_k^m U_k^* n_k \Delta A_k\right\} \qquad (6.7)$$

Уравнение (6.7) можно привести к «стандартному» виду Anes:

$$a_P p_P' = \sum_{nb} a_{nb} p_{nb}' + r_P \ ,$$

$$a_{nb} = \rho_k \frac{C_k}{a_{p,k}^m} C_k, \quad a_P = \sum_{nb} a_{nb}, \qquad (6.8)$$

$$r_P = -D_P = -\left\{\frac{\rho_P^m - \rho_P^0}{\Delta\tau}\Delta V_P + \sum_k U_k^* n_k \Delta A_e\right\}$$

Член r_P в уравнении для поправки давления - это не что иное, как *невязка* уравнения неразрывности, взятая со знаком минус. В коде она используется в соотношениях (5.4) - (5.6) для контроля сходимости процесса решения уравнения для Ф-переменной PF .

На этом SIMPLE-итерация закончена. Поскольку используется полунеявная форма уравнения (6.4), то для получения сходимости итераций необходимо использовать нижнюю релаксацию:

1. для скоростей - на предиктор шаге (при решении уравнений (6.3)),
2. для давления - на корректор-шаге:

$$p^* = p^m + \alpha_p \cdot p' \qquad (6.8)$$

Уравнение для поправки давления для неструктурной сетки получатся аналогичным способом. Единственное отличие - для связи скоростей на грани КО и перепада давления используется соотношение (4.5), основанное на поправке Рие-Чоу.

При использовании алгоритма SIMPLE важен порядок решения Ф-переменных:

1) расчет всех коэффициентов дискретных уравнений по текущим полям,
2) расчет U_x^*, U_y^*, U_z^* по полю давления с предыдущей итерации p^m,
3) решение уравнения для поправки давления (6.8),
4) коррекция скоростей по соотношениям (6.5),
5) коррекция поля давления по соотношению (6.8),
6) решение уравнений для оставшихся Ф-переменных,
7) если не выполнены критерии сходимости для всех Ф-переменных, переход к шагу 1.

При решении большинства стационарных задач коэффициенты релаксации должны удовлетворять следующим условиям:

1) для структурных сеток: $\alpha_p \le 0.8$, $\alpha_U \le 0.5$,

2) для неструктурных сеток: $\alpha_p \le 0.2$, $\alpha_U \le 0.8$

При решении нестационарных задач также требуется использование нижней релаксации. В ряде задач можно отказаться от релаксации, поскольку член a_T в дискретных уравнениях (3.22) играет роль своеобразной релаксации. Экспериментально было замечено, что от релаксации можно отказаться в алгоритме SIMPLE, если число Куранта

$$Co = \max\left(\frac{U_k \Delta\tau}{\Delta x_k}\right),$$

рассчитанное для всех КО, не превышает 1. Заметим, что это условие обычно используется в явных дискретных схемах.

В связи с этим, в коде Anes для решения нестационарных задач был реализован еще один алгоритм, объединяющий алгоритм SIMPLE с некоторыми элементами алгоритма PISO. Следуя за кодом OpenFoam этот алгоритм в Anes называется PIMPLE (заметим, что сами алгоритмы PIMPLE в Anes и OpenFoam различаются).

6.2 Алгоритм PIMPLE

Главное отличие исходного алгоритма PISO, предложенного Исса (R.I. Issa) [8], от алгоритма SIMPLE (и его «клонов») заключается в отсутствии итераций на шаге по времени. Для этого в PISO используется схема «расщепления» (Spliting) на шаге по времени.

В большинстве «современных» реализаций алгоритма PISO итерации сохраняются, поэтому современные реализации PISO более близки к SIMPLE, чем к «исходному» алгоритму PISO Исса.

В коде Anes PISO-шаги просто добавляются в SIMPLE-корректору. Как показывают экспериментальные расчеты, это позволяет отказаться от использования нижней релаксации и, при этом «расширить» ограничение по числу Куранта.

В алгоритме PIMPLE после корректора-шага SIMPLE выполняется еще один корректор-шаг. На второй - PISO коррекции - ищем новые поля скоростей u*** и новое поле давления p**:

$$U_k^{***} = U_k^{**} + H_k^m(U_k^{**} - U_k^*) + \frac{n_k C_k}{a_{p,k}^m}(p_p'' - p_k''), \quad p^{**} = p^* + p'',$$

$$a_P^* p_P'' = \sum_{nb} a_{nb}^* p_{nb}'' + b_P^{**}, \qquad (6.9)$$

$$b_P^{**} = -\sum_k \rho_k^m \frac{H_k^m(U_k^{**} - U_k^*)}{a_{p,k}^m} n_k \Delta A_k = -\sum_k \rho_k^m \frac{H_k^m(U_k')}{a_{p,k}^m} n_k \Delta A_k$$

Эти вычисления можно повторить - это и есть PISO-цикл. Заметим, что в этом цикле коэффициенты уравнения для поправки не меняются, меняется только источник.

Как показывают наши эксперименты, новые PISO-коррекции не улучшают ситуацию, поэтому по умолчанию коррекция (6.9) выполняется только *один раз*. Но пользователь может изменить число PISO-циклов с помощью оператора секции [SpeDat]:

```
I("NoPISOiter") = <Число PISO-итераций>
```

☞Замечание #1: При использовании алгоритма PIMPLE можно использовать для улучшения сходимости коэффициенты релаксации для скорости. Но для давления не нужно использовать релаксацию: $\alpha_p = 1$.

☞Замечание #2: При проведении тестовых расчетов с VOF- алгоритмом, в котором используется PIMPLE, был обнаружен интересный результат. После нескольких итераций на шаге по времени, на всех последующих итерациях число внутренних итераций любого линейного солвера было равно единице. Более детальный анализ показал, что на этих итерациях поля Ф-переменных практически не изменяются, хотя невязки их уравнений не достигают «стандартных» значений 10^{-5}. Фактически это является «оправданием» безитерационности исходного алгоритма PISO. Этот факт лег в основу StopOneIterLES окончания SWEEP-итераций шага по времени: если для всех решаемых Ф-переменных число LES-итераций равно единице, то итерации прекращаются.

6.3 Нестационарные задачи: SIMPLE или PIMPLE

Как показывают численные эксперименты, выбор алгоритма зависит от задачи и лучше всего провести предварительные расчеты. Для большинства задач более предпочтителен алгоритм PIMPLE без релаксации ($\alpha_p = \alpha_U = 1$) и ограничением по числу Куранта

$Co < Co_{cr}$, $Co_{cr} = 1 .. 8$

ⓘ *Описание в проекте прикладной задачи :*
Выбор метода решения определяется оператором секции [Main]
WayOfSolve = WS_LAPLAS / WS_SIMPLE / WS_PIMPLE

Критическое число Куранта по умолчанию равно 1000. Для его изменения необходимо использовать оператор секции [Special Data]:
R("CriCourant") = 1

Активация StopOneIterLES алгоритма окончания SWEEP-итераций осуществляется оператором
L("IsStopOneIterLES") = .FALSE. / .TRUE.
По умолчанию этот флаг активируется только для задач с VOF-алгоритмом.

6.4 Источники и граничные условия для давления

Источниковые члены и поверхностные источники ГУ описываются соотношениями (см. соотношения (3.9) и (3.14) документа [1]):

$$\begin{aligned} M_P &= C_p\left(V_p - p\right) \\ m_b &= C_{p,b}\left(V_{p,b} - p_P\right) \end{aligned} \qquad (6.10)$$

Эти источники легко добавляются в уравнение неразрывности, записанное через поправку давления SIMPLE и PIMPLE:

$$\begin{aligned} a_P &= \sum_{nb} a_{nb} + \varphi_P C_{p,P}\Delta V_P + \varphi_P C_{p,b}\Delta A_w, \\ r_P &= -D_P + \varphi_P C_{p,P}\left(V_{p,P} - p_P^*\right)\Delta V_P + \varphi_P C_{p,b}\left(V_{p,b} - p_P^*\right)\Delta A_w \end{aligned} \qquad (6.11)$$

где ΔA_w - площадь граничной грани.

При использовании специального ГУ с $C_{p,b}$ = BC_FIXMASS в Решателе подменяются значения коэффициента и значения граничного условия на следующие:

$$C_{p,b} \to V_{tiny}, \quad V_{p,b} \to \frac{V_{p,b}}{V_{tiny}}$$

здесь $V_{tiny} = 10^{-20}$ - «очень малое» значение.

При задании давления на границе ($C_{p,b}$ = BC_FIXPRESS) в Решателе значение коэффициента заменяется на «достаточно большое» число, равное 1000. Как показывают численные эксперименты, это значение дает хорошие результаты для большинства задач. Однако, если сходимость итерационного процесса плохая, то можно попробовать изменить значение коэффициента в пределах 10^2 - 10^5.

6.4.1 Граничное условие для давления BC_OUTPRESS

В коде Anes для задания *давления* на границе реализованы два граничных условия. Первое - BC_FIXPRESS - основано на использовании общей линеаризованной формы источника (6.10), в котором задается большой (но не бесконечный) коэффициент источника $C_{p,b}$. Физически это означает, что на выходе формируется дроссель, который «экранирует» поля внутри расчетной области от влияния выходной границы. В большинстве задач такое экранирование работает хорошо. Но в ряде задач такое граничное условие может приводить к искажению полей в ячейках вблизи выходной границы. Типичный пример - течение в обогреваемом постоянной плотностью теплового потока канале или трубе. Если канал длинный, то в области выходной границы реализуется стабилизированный участок течения, на котором профиль скорости не меняется, а давление и температура стенки изменяется линейно. При использовании граничного условия BC_FIXPRESS в одном или двух слоях КО вблизи выходной границы эти профили могут существенно искажаться (особенно при моделировании турбулентных течений). Это «искажения» не распространяется внутрь РО, и в принципе можно просто не учитывать их в обработке результатов расчета. Однако в ряде задач они недопустимы. Для борьбы с такими искажениями полей вблизи выходной границы был реализован еще один тип ГУ для заданного давления - BC_OUTPRESS.

Это граничное условие основано на двух идеях:

1) «классическом» граничном условии $\partial u_n / \partial n = 0$ для нормальной компоненты скорости,
2) идее расчета массового потока для совмещенных скоростей (поправки Рие-Чоу).

Граничное условие $\partial u_n / \partial n = 0$ означает «перенос» профиля скорости из приграничных КО на выходную границу, что хорошо вписывается в модель стабилизированного течения. Однако такое ГУ не позволяет «связать» поле давления внутри РО с граничным значением заданного давления на выходе - p_{out}. Кроме того, если на выходе нет стабилизированного течения (например, на участке границы нормальный градиент давления переменный), то это ГУ не позволяет выполнить баланс массы в РО.

В связи с этим были реализованы новые ГУ типа BC_OUTPRESS. Рассмотрим эти условия для случая использования структурных сеток.

Рассмотрим приграничный КО "P", изображенный на рисунке 6.1, и пусть грань "e" является граничной гранью выходной границы. Контрольный объем для приграничной скорости u_w показан на рисунке пунктиром и состоит из целого основного КО "P" и половинки КО "W".

При использовании граничного условия $\partial u_n / \partial n = 0$ значение скорости на грани равно:

$$u_e = u_w, \tag{6.12}$$

и это условие никак не связано с давлением $p_e = p_{out}$. Воспользуемся идеей поправки Рие-Чоу [4]. Уравнение для скорости в узле "w" перепишем в виде

$$U_w = \hat{U}_w - \frac{\Delta V_w}{a_w}\frac{(p_P - p_W)}{\delta x_{WP}}, \quad \hat{U}_w = \frac{\sum_{nb} a_{nb} U_{x,nb} + b_w}{a_w}, \quad \Delta V_w = \Delta V_P + \frac{\Delta V_W}{2} \tag{6.13}$$

и проинтерполируем его на правую половинку контрольного объема "P"(КО для скорости "e"):

$$U_e = \overline{\left(\hat{U}\right)}_e - \overline{\left(\frac{\Delta V}{a_w}\right)}_e \frac{(p_{out} - p_P)}{\delta x_{Pe}}.$$

Здесь функция $\overline{(f)}_e$ означает экстраполяцию поля f на границу РО.

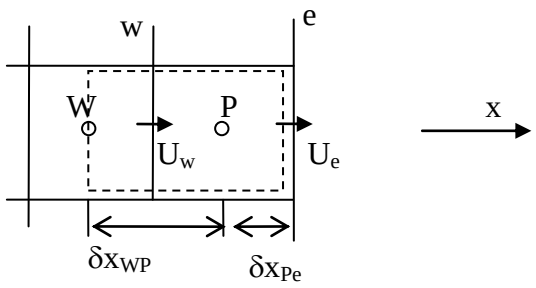


Рисунок 6.1 - Приграничный скоростной КО

Используя первый порядок для экстраполяции, соотношение для скорости на грани можно записать в окончательном виде граничного условия BC_OUTPRESS:

$$U_e = U_w - \frac{\Delta V_w}{a_w}\left\{\frac{(p_{out} - p_P)}{\delta x_{Pe}} - \frac{(p_P - p_W)}{\delta x_{WP}}\right\} \tag{6.14}$$

Это соотношение позволяет не только рассчитать скорость на грани (и плотность массового потока), но и связать поправку скорости на грани с поправкой давления:

$$U'_e = \frac{\Delta V_w}{a_w}\frac{1}{\delta x_{Pe}} p'_P \tag{6.15}$$

Если давление меняется линейно в направлении нормали к границе, то соотношение (6.14) переходит в (6.12). Если поле давления меняется нелинейно, то поправка, связанная с различием градиентов давления, позволяет правильно рассчитать плотность массового потока на границе и выполнить баланс массы во всей РО. Но самое главное, соотношение (6.14) позволяет привязать заданное давление на выходе к полю давления в РО.

Соотношение (6.14) справедливо только для сошедшегося решения. Для ускорения сходимости необходимо ввести в уравнение (6.14) невязку уравнения движения r_w:

$$U_b = U_w + \frac{r_w}{a_w} - \frac{\Delta V_w}{a_w}\left\{\frac{(p_{out} - p_P)}{\delta x_{Pe}} - \frac{(p_P - p_W)}{\delta x_{WP}}\right\},$$

$$U'_e = \frac{\Delta V_w}{a_w} p'_P$$

Для неструктурных сеток (в которых используется модель совмещенных скоростей) данное ГУ реализуется аналогично. Скорость на граничной выходной грани "b" рассчитывается по соотношению:

$$U_b = \mathbf{U}_b \cdot \mathbf{n}_b = \mathbf{U}_P \cdot \mathbf{n}_b + \frac{\mathbf{r}_P \cdot \mathbf{n}_b}{a_p} - \frac{\Delta V_p}{a_p}\left\{\frac{(p_{out} - p_P)}{(\mathbf{r}_b - \mathbf{r}_p)\cdot \mathbf{n}_b} - (\nabla p)_p \cdot \mathbf{n}_b\right\},$$

$$U'_b = \frac{\Delta V_p}{a_p}\frac{1}{(\mathbf{r}_b - \mathbf{r}_p)\cdot \mathbf{n}_b} p'_P \tag{6.16}$$

Здесь $\mathbf{r}_P$ - вектор невязки, компоненты которого равны невязкам дискретных уравнений для компонент скорости.

6.4.2 Граничные условия для давления торможения

Наряду со «стандартными» граничными условиями BC_FIXMASS, BC_FIXPRESS и BC_OUTPRESS только для *неструктурных сеток* реализованы дополнительные условия (см. раздел 4.1.1 документ [1]) заданного давления торможения на входной границе.

Граничное условие для *давления торможения* на входной границе "in" можно записать в виде:

$$m_{in} = \rho_{in} u_{in} = \sqrt{2\rho_0 (p_0 - p_b)} \tag{6.17}$$

Здесь ρ_0, p_0 - давление и плотность «заторможенного» потока, p_b - неизвестное значение давления на границе. Дискретный аналог (6.17) можно записать в виде:

$$m_b = \sqrt{2\rho_0 |p_0 - p_b|}, \quad p_b = p_P + (\nabla p)_P \cdot (\mathbf{r}_b - \mathbf{r}_P),$$
$$m'_b = \frac{\rho_0}{m_b} p'_P \tag{6.18}$$

где индексы "b" и "P" обозначают центры граничной грани и приграничной ячейки. Второе уравнение (6.18) используется в алгоритме SIMPLE для расчета поправки плотности массового потока на входной границе.

6.5 Моделирование сжимаемых течений

В текущей версии Решателя реализованы алгоритмы, позволяющие рассчитывать течения с плотностью, зависящей от давления p. К этой категории относятся как трансзвуковые и сверхзвуковые течения, так и существенно дозвуковые течения при наличии объемных источников массы и импульса (типичный пример – процессы адсорбции водорода в пористых засыпках металлогидридов).

Если плотность G-фазы зависит от давления, в коде используется *сжимаемый* вариант уравнения для поправки давления алгоритмов SIMPLE и PIMPLE. При его получении, наряду с поправкой скорости необходимо учитывать и поправка плотности

$$\rho = \rho^m + \beta_P p', \quad \beta_P = \left(\frac{\partial \rho}{\partial p}\right)^m \tag{6.19}$$

где β_P – коэффициент сжимаемости G-фазы (V-переменная).

В этом случае в уравнение для поправки давления появятся дополнительные члены, связанные с поправками плотности:

$$\left[a_P + (\beta_P)_P \frac{\Delta V_P}{\Delta \tau}\right] p'_P = \sum_{nb} a_{nb} p'_{nb} + r_P - \sum_{nb} \rho_k U_k^* n_k (\beta_P)_k p'_k \tag{6.20}$$

Для аппроксимации дополнительных последних «красных» членов в коде Anes используется два алгоритма.

Первый алгоритм использует противопоточную схему и реализован для обоих типов сеток. Второй алгоритм, предназначенный для более точного моделирования сверхзвуковых течений, реализован только для неструктурных сеток.

Рассмотрим эти алгоритмы на примере аппроксимации дополнительного члена для k-грани. В *первом* алгоритме для аппроксимации используется соотношение:

$$\rho_k U_k^* n_k (\beta_P)_k p'_k = m_k \left(\frac{\beta_P}{\rho}\right)_k p'_k = \max(m_k, 0) \left(\frac{\beta_P}{\rho}\right)_P p'_P - \max(-m_k, 0) \left(\frac{\beta_P}{\rho}\right)_{nb} p'_{nb} \tag{6.21}$$

Заметим, что эти члены принципиально меняют тип уравнения (6.20). В коэффициентах a_{nb} уравнения наряду с «диффузионной» составляющей появляется «конвективная» составляющая и коэффициент a_P уже не равняется сумме коэффициентов a_{nb}:

$$a_{nb} = \left\{ \max(-m_k, 0) \left(\frac{\beta_P}{\rho}\right)_{nb} + \frac{\rho_k \Delta A_k}{a_{p,k}} C_k \right\},$$
$$a_P = \ldots + \left(\frac{\beta_P}{\rho}\right)_P \sum_k \max(m_k, 0) + (\beta_P)_P \frac{\Delta V_P}{\Delta \tau}$$

В случае, когда «конвективная» составляющая превышает «диффузионную», тип уравнения из эллиптического начинает трансформироваться в параболический.

Для более детального «выделения» областей ударных волн для неструктурных сеток реализован второй алгоритм, предложенный Периком [4]. В этом алгоритме для аппроксимации членов с пульсацией плотности используется аналог схемы второго порядка:

$$\rho_k = \rho_k^{UDS} + \alpha_R \left(\rho_k^{CDS} - \rho_k^{UDS}\right),$$

$$U_k\rho'_k = \left(U_k\rho'_k\right)^{UDS} + \alpha_R \left\{\left(U_k\rho'_k\right)^{CDS} - \left(U_k\rho'_k\right)^{UDS}\right\}$$

Здесь верхний индекс UDS соответствует противопоточной аппроксимации, CDS - линейной аппроксимации (для плотности эти аппроксимации приведены в (3.5) и (3.6)). Коэффициент α_R играет роль функции $\psi()$ для схем второго порядка (см. пункт 3.3). Как показывают численные эксперименты, наиболее приемлемым для этого коэффициента являются значения 0.8 … 0.9.

При использовании второго алгоритма для учета членов с α_R используется модель отложенной коррекции [4], примененной к уравнению для поправки. В связи с этим при решении уравнения для поправки давления используется еще один цикл внутренних итераций. Число итераций такого цикла пользователь должен задать при описании проекта.
Примеры использования этих алгоритмов можно найти в библиотеке примеров в секции «Сжимаемые течения».

ⓘ *Описание в проекте прикладной задачи :*
Использование «сжимаемой» или «несжимаемой» формы уравнения для поправки давления определяется значением свойства β_P, если оно равно нулю или не задано для материала G-фазы, то используется «несжимаемая» форма.
Для активации второго алгоритма «выделения» ударных волн необходимо задать оператор в секции [Special Data]:
L(“ComprsPF”) = .TRUE.
Для задания коэффициента α_R используется оператор
R(“AlfaPF_CDS”) = 0.8
Для задания числа дополнительных внутренних итераций используется оператор:
I(“NoSecondPStep”) = 10

6.6 Нестационарные сжимаемаемые течения и P0(time)

В коде Anes полное статическое давление p рассчитывается как сумма

$$p = p_0 + P_f,$$

где p_0 – отсчет давления, P_f – поле гидродинамической составляющей давления. Такое разбиение повышает точность вычисления, поскольку для «типичных» задач p порядка 10^5 Па, а изменение P_f – порядка 10^2 Па. Гидродинамическое давление P_f в коде рассматривается как Ф-переменная для уравнения неразрывности (сохранения массы). Если в граничных условиях и источниках для P_f не задано *явно* давление (заданы только плотности потоков массы), то Решатель автоматически и *принудительно* в одной точке расчетной области зануляет давление P_f.

В файле проекта эта точка задается переменными

(XpfZERO, YpfZERO, ZpfZERO)

секции [Solver] файла проекта.

При решении нестационарных задач с плотностью, зависящей от давления, принудительное зануление гидродинамического давления в одной точке может приводить к конфликту с интегральным балансом массы. Для решения этой проблемы необходимо предположить, что отсчет давления p_0 начинает зависеть от времени. Для расчета зависимости $p_0(\tau)$ в Решателе используется следующий алгоритм.

На шаге по времени при расчете давления P_f не производится зануления давления. Это зануление производится в конце расчета шага по времени путем вычитания давления в

точке зануления из поля давления. Одновременно это давление в точке зануления *прибавляется* к отсчету давления p_0.

Заметим, что в случае «явного» задания давления в ГУ или источниках этот алгоритм не используется и в этом случае лучше всего не использовать p_0 и положить его значение равным нулю. В этом случае давление P_f является одновременно и статическим давлением.

Для работы алгоритма расчета $p_0(\tau)$ необходимо чтобы плотность зависела от давления, т.е. должно быть задано свойство $\beta_{Pg} = \frac{\partial \rho_g}{\partial p}$.

В составе библиотеки примеров кода в секции нестационарных задач есть несколько примеров расчета сжимаемых задач.

6.7 Coupled- алгоритм для сопряженных задач теплообмена

При решении сопряженных задач (только зоны Flow и Struct в расчетной области) по умолчанию уравнения энергии для Ts и Tg решаются последовательно в своих зонах. В ряде задач, например, длинном канале с тонкими стенками и граничным условием q_w = const, это приводит к медленной сходимости уравнений энергии.

Для ускорения сходимости в текущей версии реализован алгоритм совместного решения этих уравнений. Его идея заключается в следующем:

1) дискретные уравнения для Tg и Ts объединяются в одну систему дискретных уравнений для Tg (оно становится уравнением типа SPACE);
2) эта система уравнений решается одним из линейных солверов;
3) температура Tg копируется в поле Ts.

В текущей версии кода этот подход для *структурной* сетки можно использовать с любым линейным солвером. Для *неструктурной* сетки Coupled-алгоритм можно использовать только с солвером HYPRE для Tg или Ts.

ⓘ *Описание в проекте прикладной задачи :*
Для активации алгоритма используется оператор в секции [Special Data]:

```
L("IsCoupledSolveTGS") = .TRUE. / .FALSE.
```

По умолчанию алгоритм активирован:

7. Алгоритмы Решателя Anes

7.1 Алгоритм итерационного процесса

На рисунках 7.1 и 7.2 показаны алгоритмы решения стационарной и нестационарной задач.

```
<Setup>     ! Инициализация переменных по умолчанию,
<Input>     ! Чтение файлов исходных данных Решателя и выделение памяти под массивы полей
 *Вызов «события пользователя» OnInput
<Init>      ! Инициализация PHI-полей (по умолчанию и из файла результатов)
 *Вызов «события пользователя» OnInit
! ---------- Итерационный процесс ------------------
<BeforeSolve>   ! Расчет специальных Ф-переменных до основного решения
! ---------- Итерационный процесс ------------------
iSWEEP  = 0
Liter = .TRUE.
DO WHILE(LITER)
    iSWEEP = iSWEEP + 1
    *Вызов «события пользователя» BeforeSweep
    <Расчет одной итерации>   *Вызов события OnLESolver
    *Вызов «события пользователя» AfterSweep
! --- Проверка сходимости ------
    IF(Все errФ(iPHI) < Eps)  Выход из SWEEP-цикла
ENDDO
<Отчет Решателя>
*Вызов «события пользователя» OnReport
<Создание файлов результатов >

STOP:     ->   *Вызов «события пользователя» OnStop
```

Рисунок 7.1 - Блок-схема решения стационарной задачи

```
<Setup>     ! Инициализация переменных по умолчанию,
<Input>     ! Чтение файлов исходных данных Решателя и выделение памяти под массивы полей
 *Вызов «события пользователя» OnInput
<Init>      ! Инициализация PHI-полей (по умолчанию и из файла результатов)
 *Вызов «события пользователя» OnInit
! ---------- Итерационный процесс ------------------
<BeforeSolve>   ! Расчет специальных Ф-переменных до основного решения
! ======== Шаги по времени ===============
iTIME = 0
Time = TimeBeg
DO WHILE(Time < TimeEnd)
   <Переприсвоение   Φk^0 = Φk >
   Time = Time + dTime
   *Вызов «события пользователя» BeforeTime
! ---------- Итерационный процесс на шаге по времени ------------------
   iSWEEP  = 0
   Liter = .TRUE.
   DO WHILE(LITER)
     iSWEEP = iSWEEP + 1
     *Вызов «события пользователя» BeforeSweep
     <Расчет одной итерации>                  : *Вызов события OnLESolver
     *Вызов «события пользователя» AfterSweep
! --- Проверка сходимости ------
      IF(Все errФ(iPHI) < Eps)  Выход из SWEEP-цикла
   ENDDO
   *Вызов «события пользователя» AfterTime
   <Отчет на шаге по времени>
   *Вызов «события пользователя» OnReport
   <Создание файлов результатов на шаге по времени>
ENDDO
*Вызов «события пользователя» OnResume
STOP:
```

Рисунок 7.2 - Блок-схема решения нестационарной задачи

Главным «элементом» алгоритмов 7.1 и 7.2 является блок расчета одной итерации. Его блок схема показана на рисунке 7.3.

```
DO iPHI = 1, NoPHI          !  Цикл по всем расчитываетмых Ф-переменных
  <Расчет коэффициентов дискретного уравнения aP, anb, rP>
  <Вызов LES-солвера и расчет δФ>
  <Коррекция поля    Ф = Ф* + δФ >

   Если (iPHI = PF и PIMPLE) <Вторая PISO коррекция PF и U>

  <Расчет критерия сходимости для текущей Ф - errФ(iPHI) >
ENDDO
```

Рисунок 7.3 - Расчет одной итерации

Дадим некоторые комментарии.

Перед началом итерационного процесса производится инициализация полей Ф-переменных. В пакете для этого предусмотрено две возможности:

1) распределения полей задается с помощью специальных V-переменных Φ_{ini},
2) поля считываются из уже существующего файла результатов или файла контрольной точки.

Как уже отмечалось, для итерационного процесса в пакете предусмотрено два уровня итераций:

1) внешний уровень - уровень SWEEP-итераций, на котором производится последовательное решение всех Ф-уравнений,
2) внутренний уровень – итерации внутри конкретного LES-солвера.

Внешний итерационный процесс прекращается, если:

1) *достигнуты критерии сходимости* (5.4) - (5.6) для *всех* решаемых Ф-переменных,
2) число внешних SWEEP-итераций превысило допустимое значение MaxSWEEP, задаваемое пользователем,
3) превышено время решения TimeSolution, задаваемое пользователем.

Для разных LES-солверов используются свои собственные критерии прекращения внутренних итерации. Пользователь может влиять на эти критерии, задав максимальное число внутренних итераций для каждой Ф-переменной.

В коде Anes предусмотрена возможность провести решение для определенных Ф-переменных до основного процесса решения. Это позволяет не решать уравнения, которые не «зависят» от других Ф-переменных, в основном SWEEP-цикле. В текущей версии только одна Ф-переменная решается до основного SWEEP-цикла - это переменная «wDIS», которая используется для расчета расстояния до ближайшей твердой стенки.

При решении нестационарной задачи в дискретных уравнениях появляется два поля Ф-переменной:

Φ_k^0 - значение на предыдущем шаге по времени Time - dTime,

Φ_k - значение на текущем шаге по времени Time.

Сетка по времени в пакете строится по аналогии с сеткой по структурной пространственной координате. Интервал решения (TimeBeg .. TimeEnd) разбивается на зоны, в пределах которых задается свой постоянный шаг по времени dTime. Значение этого шага определяется как физикой задачи, так и точностью решения.

Как показывает опыт решения нестационарных задач наиболее оптимальный алгоритм выбора шага по времени при использовании алгоритма SIMPLE - число внешних SWEEP-итераций на шаге по времени, при которых достигается сходимость, должно быть не более 50 - 100. При использовании алгоритма PIMPLE, в котором можно отключить линейную релаксацию, необходимо указать критическое значение числа Куранта Co_{cr} = 1 .. 8.. Во

втором случае шаг по времени будет определяться не шагом, заданным в сетке по времени, а шагом, рассчитанным из допустимого максимального значения Co.

Схема расчета одной итерации показана на рисунке 7.3. На SWEEP-итерации уравнения для Ф-переменных решаются последовательно. Само решение сводится к последовательному выполнению следующих этапов:

1. Расчет дискретных коэффициентов уравнения a_P , a_{nb}, r_P.
2. Вызов линейного LES-солвера и расчет нового поля.
3. Расчет критериев сходимости (5.4) - (5.6).

На рисунках 7.1 и 7.2 коричневым цветом показаны места вызова «событий пользователя». Эти «точки» кода Решателя используются для расчета полей и скалярных переменных пользователя (см. документ [2]).

ⓘ *Описание в проекте прикладной задачи :*

Параметры итерационного процесса описываются следующими операторами секции [Solver] и [PHI Variables]:

1) Максимальное число внешних SWEEP-итераций задается оператором секции [Solver]:
MaxSweep = <значение>

2) Точность решения по умолчанию для всех Ф-переменных задается оператором секции [Solver]:
Eps = <значение>

2) Критерий сходимости задается оператором секции [Solver]:
TypeEndSweep = TES_MAXRES / TES_TOTRES / TES_MAXCOR

2) Число внутренних итераций LES-солвера для Ф-переменной задается оператором секции [PHI Variables]:
MaxIter("Ф-имя") = <значение>

3) Для инициализации Ф-переменных используется операторы секции [PHI Variables]:
Init ("Ф-имя") = <V-переменная>

4) Тип задачи (стационарная или нет) определяется оператором секции [Main]
IsSteady = .TRUE./.FALSE.

5) Сетка шагов по времени описывается в секции [Time Grid] операторами
TZone(TimeBeg,TimeEnd,DTime,….)

6) Критическое число Куранта по умолчанию равно 1000. Для его изменения необходимо использовать оператор секции [Special Data]:
R("CriCourant") = 1

7.2 Алгоритм обработки источниковых членов

Важной частью расчета дискретных коэффициентов является обработка источников пользователя. Все источниковые члены в Решателе хранятся в виде массива структур:

Source(Ф-имя, Имя_патча, $C_{SФ}$, $V_{SФ}$)

При подготовки коэффициентов дискретного уравнения для данной Ф-переменной просматривается массив Source, отбираются все элементы, связанные с Ф-переменной, и обрабатываются с помощью следующего алгоритма:

1) находится патч, связанный с источником, из которого определяется тип источника (объемный, поверхностный или точечный),
2) по имени патча находится массовый источник (Ф-переменная P_f) для этого патча,
3) перебираются все КО патча и для каждого КО рассчитываются массовый источник и коэффициенты и значения источника для Ф-переменной,
4) эти составляющие в соответствии с соотношением (3.25) *суммируются* в разностные коэффициенты a_P и r_P.

Таким образом, в принципе для одного патча можно задать несколько источников для конкретной Ф-переменной (хотя делать это не рекомендуется).

7.3 Алгоритм обработки ГУ

Граничные условия, представляющие собой поверхностные источники, обрабатываются немного по-другому. Все граничные источники в Решателе хранятся в виде массива структур, аналогичным источникам:

BCSource(Ф-имя, Имя_BS_патча, СФ, VФ)

В отличие от источниковых членов, при подготовке коэффициентов дискретного уравнения для данной Ф-переменной просматривается не этот массив, а массив всех граничных патчей (тип BS):

1) для данной Ф-переменной находится *первое* ГУ в массиве BCSource, связанное с данным патчем,
2) по имени патча находится массовый источник (Ф-переменная P_f) для этого патча,
3) перебираются все КО патча и для каждой грани КО патча рассчитываются массовый источник и коэффициенты и значения ГУ для Ф-переменной,
4) Для Ф-переменных, связанных с S-фазой или типа PD_SPACE, *массовый источник всегда игнорируется* !
5) эти составляющие в соответствии с соотношением (3.29) *суммируются* в разностные коэффициенты a_P и r_P.
6) независимо от того, есть или нет описание ГУ, рассчитывается информация для определения граничного значения Ф-переменной для данного патча.

8. Параллельные вычисления

Текущая версия кода поддерживает проведение параллельных расчетов. Для реализации параллельности используется подсистема MPI.

MPI-интерфейс предназначен для работы на многопроцессорных системах различной архитектуры (как с общей памятью, так и на кластерных системах). При использовании MPI компьютер с точки зрения пользователя представляет собой набор логических процессоров (один процессор и его оперативная память), связанных друг с другом. При этом топология этой связи определяется самим пользователем (одномерная, двумерная или трехмерная сетка процессоров) и напрямую никак не связана с истинной физической связью процессоров. Алгоритм работы MPI-интерфейса достаточно прост:

1) На всех логических процессорах запускается *одна* копия параллельной программы (далее эта копия будет называться процессом), которые начинают независимое выполнение; для идентификации процесса в него передается специальный уникальный код процесса – MyID;
2) Для синхронизации работы процессов используются специальные коммуникационные подпрограммы MPI. Эти подпрограммы позволяют осуществлять обмен сообщениями между копиями программ. MPI сообщение – это одномерный массив любых данных (целых, действительных, логических и т.д.);
3) Все операции обмена условно можно разделить на следующие категории
   - *парные обмены*: один процесс посылает (принимает) сообщение другому процессу, для чего используется его MyID,
   - *рассылка*: один процесс рассылает сообщение всем остальным процессам,
   - *суммирование*: для элементов массива сообщений проводится агрегатная операция по все процессам и это сообщение рассылается всем процессам. Например, производится суммирование (или вычисление среднего, минимального или максимального значения) значения переменной программы, после чего сумма пересылается в эту же переменную во всех процессах.

Стратегия распараллеливания в первую очередь зависит от математических моделей CFD-кода. Для метода контрольного объема, который используется в Anes, алгоритм распараллеливания очевиден – это *декомпозиция* расчетной области (РО). Необходимо разбить РО на подобласти (субдомены), число которых равно числу логических процессоров, и рассчитывать каждый субдомен в отдельном процессе.

Такое разбиение зависит от типа сетки КО и оно различно для структурных и неструктурных сеток.

8.1 Распараллеливание структурной сетки

Для распараллеливания кода, работающего со структурной сеткой, используется подход, основанный на «прямоугольной» декомпозиции.

Расчетная область разбивается на прямоугольные субдомены в 2D случае или параллелепипеды в 3D случае (рисунок 8.1). Число субдоменов должно быть равно числу используемых процессоров. Область субдомена расширяется во все стороны (за исключением границ РО) на один слой КО, которые называются HALO КО или ячейками обмена. Эти КО необходимы для решения дискретных аналогов уравнений для граничных КО субдомена. Для этих граничных КО необходимы значения Ф в соседних узлах близлежащего субдомена.

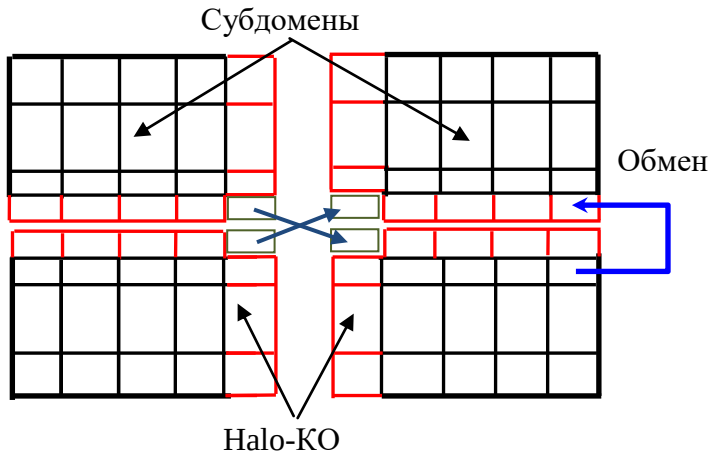


Рисунок 7.1 - Разбиение РО на четыре субдомена.

Значения в HALO-КО обновляются внутри линейного солвера путем посылки-приема сообщений между субдоменами. Для избежания «зависания» параллельного расчета все субдомены «раскрашиваются» в черно-белые цвета (как шахматная доска) и используется следующее правило: «белый» субдомен сначала посылает свои данные соседу, а затем принимает данные от него, «черный» субдомен – напротив, сначала принимает данные, а затем посылает данные соседнему субдомену. Для обмена значений в «угловых» HALO-ячейках используется более сложный алгоритм, основанных на неблокирующих обменах.

Как показали численные эксперименты, использование структурной сетки для решения задач со сложной геометрией обладает рядом недостатков, главный из которых – сложность равномерной *загрузки* процессоров кластера. Как уже отмечалось выше, в Anes для моделирования сложной геометрии используется модель блокированных КО и концепция «сквозного» счета. Для оптимальной работы программы необходимо провести декомпозицию на субдомены так, чтобы в них было одинаковое число *неблокированных* КО. В общем случае сделать это можно только при одномерной декомпозиции по одной из осей. С другой стороны, этот вариант наиболее неудачный, поскольку для него число ячеек обмена максимально.

Поэтому для проведения расчетов для сложной геометрии более эффективным является использование неструктурных, для которых равномерная загрузка процессоров реализуется гораздо проще.

Для управлением процесса разбиения структурной сетки на субдомены в коде предусмотрены два механизма: автоматическое разбиение и ручное разбиение.

При ручной декомпозиции пользователь задает три переменные NoSubDomX, NoSubDomY и NoSubDomZ файла проекта, которые задают число субдоменов вдоль осей координат. Естественно, что их произведение должно быть равно числу логических процессоров. При автоматической декомпозиции Компилятор и Решатель сами вычисляют значение этих переменных. Выбор типа декомпозиции задается переменной файла проекта TypeSplit.

```
TypeSplit = TS_AUTO / TS_MANUAL
```

ⓘ *Описание в проекте прикладной задачи :*

Для разбиения РО на субдомены используются операторы секции [Parall].

```
TypeSplit = TS_AUTO / TS_MANUAL
NoSubDomX = <число субдоменов вдоль оси x>
NoSubDomY = <число субдоменов вдоль оси y>
NoSubDomZ = <число субдоменов вдоль оси z>
```

Если выбран режим TS_AUTO, то операторы NoSubDomX, NoSubDomY, NoSubDomZ игнорируются.

8.2 Распараллеливание неструктурной декартовой сетки

При использовании неструктурных сеток процесс декомпозиции всегда проводится автоматически, пользователь должен выбрать только число логических процессоров.

Сам процесс декомпозиции сводится к разбиению массива Cell(:) на подмассивы – субдомены SubCells(:), число которых равно числу используемых процессоров. Обычно число ячеек в субдомене одинаковое. Одновременно производится декомпозиция и массива граней FaceXYZ(:) и FaceBS(:). Массив граней Faces() для каждого процесса разбивается на два массива: SubFaces(:) - массив внутренних граней, их posCellID и negCellID ссылаются на КО из своего SubCells() и HaloFaces() – массив граней, один из posCellID или negCellID ссылается на КО в *чужом* субдомене. Последний массив HaloFaces используется для осуществления обменов между субдоменами.

При использовании неструктурной декомпозиции ее эффективность определяется числом ячеек обмена. При работе в параллельном режиме Компилятор выдает в листинг информацию о процессе декомпозиции в свой листинг в виде двух таблиц:

```
.Начинаем: Декомпозиция на неструктурные СубДомены ...

   -----------(Параметры СубДоменов)-----------

 ID   IDCF         IDCL         NoCell    NoFaceXYZ  NoFaceBS
---- ---------- ----------  ---------- ---------- ----------
   1          1          294632     294632      879201      51349
   2     294633     589847     295215      881444      51535
   3     589848     885169     295322      879965      48265
   4     885170    1181949     296780     885573      48404
   5    1181950    1473800     291851     870215      50857
   6    1473801    1771057     297257     884790      45691
   7    1771058    2065773     294716     877105      46255
   8    2065774    2361597     295824     881166      45853
   9    2361598    2656724     295127     879094      52645
  10    2656725    2952714     295990     883812      52329
  11    2952715    3243686     290972     868821      52425
  12    3243687    3544147     300461     894362      51754
  13    3544148    3838277     294130     877854      50011
  14    3838278    4135193     296916     883598      46990
  15    4135194    4429363     294170     876037      46769
  16    4429364    4725120     295757     881359      49160

 ID  NoCellHALO NoFaceHALO  NoSubDomNB   Halo/Cell,%
---- ---------- ----------  ---------- ----------
   1       3913       3913          3   1.33
   2       4144       4144          3   1.40
   3       7270       7270          4   2.46
   4       2665       2665          2   0.90
   5       6877       6877          3   2.36
   6       6746       6746          3   2.27
   7       8373       8373          3   2.84
   8       7642       7642          3   2.58
   9       8676       8676          4   2.94
  10       4796       4796          4   1.62
  11       3064       3064          1   1.05
  12       6526       6526          3   2.17
  13       4781       4781          3   1.63
  14       9431       9431          4   3.18
  15       7909       7909          2   2.69
  16       3743       3743          1   1.27
.Закончили: Декомпозиция на неструктурные СубДомены, time=000.00.07:267
```

В первой таблице выдаются индексы ячеек в субдоменах (индексация КО всегда абсолютная) и число ячеек и граней субдомена. Во второй таблице выводится информация о HALO-ячейках. Самая главная колонка – последняя, в которой печатается отношение HALO-ячеек к числу ячеек субдомена в процентах. Чем меньше это число, тем эффективнее параллельные вычисления.

Эффективность декомпозиции зависит от способа перенумерации ячеек РО. Выбрать наиболее правильную перенумерацию можно только экспериментально. Однако, как показывают эксперименты, наиболее приемлемой является алгоритм SFC_IXYZ_Metis (подробнее – смотрите пункт 2.7).

Визуально декомпозицию можно проанализировать в постпроцессоре ParaView. Для этого нужно загрузить поле SubDomID из файла
<Префикс файлов результатов>_cells.vtk.

Пример этого поля для 4-ех логических процессоров показан на рисунке 8.2.

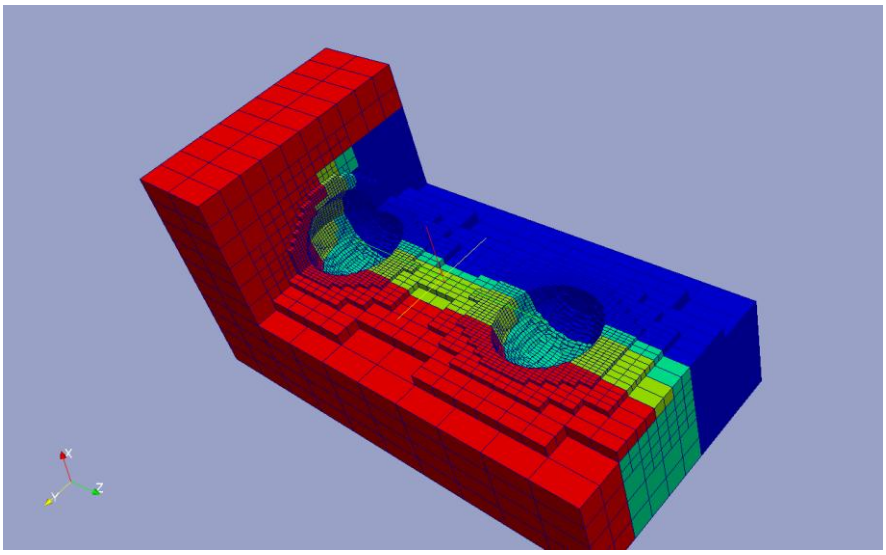


Рисунок 8.2 - Декомпозиция 3D неструктурной сетки на 4 субдомена.

9. Линейные солверы Решателя

Решатель находит решение системы уравнений с помощью итерационного процесса, при этом на внешней SWEEP-итерации уравнения для Ф-переменных решается последовательно. При этом нелинейная система уравнений (5.1) превращается в линейную систему уравнений (5.3) путем вычисления ее коэффициентов a_P, a_{nb}, r_P по значениям с предыдущей SWEEP-итерации (в данной главе значения на предыдущей итерации будет отмечаться верхним символом "*"):

$$a_P\Phi'_P = \sum_{nb} a_{nb}\Phi'_{nb} + r_P,$$
$$r_P = b^*_P + \sum_{nb} a_{nb}\Phi^*_{nb} - a_P\Phi^*_P \qquad (9.1)$$

Для решения этой системы используется подсистема Решателя, которая называется линейным солвером (LES-солвером). Что из себя представляет LES-солвер, можно понять из его простейшей реализации, которая называется P2P-солвер (Point-to-Point). Если пренебречь коэффициентами a_{nb}, то решение (9.1) легко получить:

$$\Phi'_P = \frac{r_P}{a_P} \qquad (9.2)$$

Это и есть P2P-солвер. Кстати, этот солвер легко «подключить» для решения дискретных уравнений для любой Ф-переменной. Для этого в секции [Special Data] проекта нужно добавить оператор

```
L("P2P.<Ф-имя>")      = .TRUE.
```

Для «объединения» структурных и неструктурных сеток и возможности подключения к коду внешних линейных солверов в коде Решателя используется подход, основанный на преобразовании системы (9.1) к *матричному* представлению. Если для структурной сетки пронумеровать КО с помощью одномерного индекса, например

```
iCELL = ix + (iy-1)*NX + (iz-1)*NX*NY,
```

то структурную сетку можно свести к неструктурной сетке с числом ячеек (NX-2)*(NY-2)*(NZ-2). При использовании одномерного индекса КО можно рассматривать поле Ф-переменной как вектор **Ф** (ниже для обозначения вектора используется жирный шрифт):

$\mathbf{\Phi} = (\Phi_1, \ldots \Phi_{NC})$

где N_C – это общее число ячеек: N_C = NoCells для неструктурной сетки и N_C = (NX-2)*(NY-2)*(NZ-2). для структурной. Если записать (9.1) в виде

$$a_P\Phi'_P - \sum_{nb} a_{nb}\Phi'_{nb} = r_P,$$

то можно заметить, что левая часть этого уравнения представляет собой умножение квадратной матрицы [A] на вектор Ф':

$$[A]\cdot\mathbf{\Phi}' = \mathbf{r} \qquad (9.3)$$

Матрица [A] имеет размерность N_C*N_C. Подавляющее большинство элементов матрицы равны нулю, а ненулевые элементы – это коэффициенты a_P и a_{nb}. Отметим, что элементы a_P расположены на диагонали матрицы. Типичное расположение ненулевых элементов в матрице [A] показано на рисунке 9.1. Для неструктурной сетки расположение ненулевых элементов в матрице (рисунок 9.1) можно получить, если в секции [Unstructured Cartesian Grid] задать оператор

```
DbgGraphFile = .TRUE.
```

В этом случае по окончанию работы Компилятора Anes будет создан файл
<Префикс файлов результатов>.graph,
который можно просмотреть в утилиты Anes aMatView.exe, расположенной в подкаталоге bin кода.

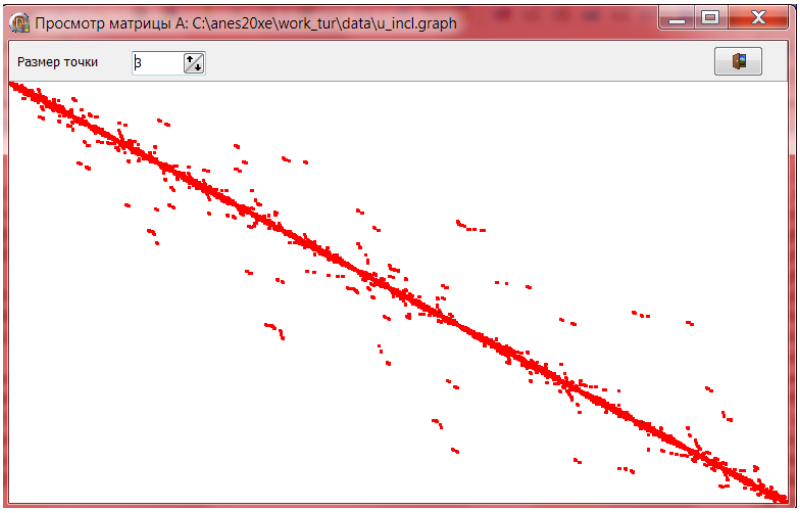


Рисунок 9.1 – Ненулевые элементы матрицы решения [A]

Для хранения в памяти Решателя разреженной матрицы очень большой размерности используется CSR-формат (Compressed Sparse Row формат), который упаковывает матрицу по строкам. В этом формате для хранения матрицы используется три одномерных массива, один действительный A и два целых массива IA и JA.

В массиве A(1:NoASize) хранятся все ненулевые элементы матрицы [A] «развернутые» по строкам (NoASize – число ненулевых элементов матрицы [A]). Этот массив можно разбить на N_C-блоков переменной длины. Каждый такой блок будет содержать ненулевые коэффициенты для *строки* матрицы с номером iCELL.

Для определения начала и конца ненулевых элементов для любой строки матрицы с номером iCELL используется первый целый массив IA(1 : NC+1) .Для строки iCELL ее элементы располагаются в массиве A в элементах: A(IA(iCELL) : IA(iCELL+1)-1. Для однотипности вычисления элементов последней строки используется элемент IA(NC+1).

И, наконец, для определения номера *столбца* матрицы для каждого ненулевого элемента используется второй целый массив JA(1:NoASize), который содержит номер столбца для элемента массива A(I).

В коде Решателя предусмотрены два интерфейса для вызова линеного солвера:

1) солвер может использовать «исходные» коэффициенты a_P, a_{nb} и r_P ,
2) солвер может использовать представление коэффициентов в виде A(:), IA(:), JA(:).

Важной особенностей Решателя является «простота» создания своего линейного солвера (подробности – в документе [5]).

Ниже кратко описаны линейные солверы, «подключенные» к коду Решателя. Отметим сразу «наиболее эффективные» линейные солверы.

Если используются *структурные* сетки и *последовательный* режим расчета, то наиболее эффективными являются два LES-солвера: aTDMA и KIVA_ILU. Для *неструктурных* сеток для *последовательного* режима лучше всего использовать линейный солвер KIVA_ILU.

При проведении параллельных вычислений для структурных и неструктурных сеток можно использовать только два солвера – KIVA и KIVA_ILU.

Все линейные солверы кода являются итерационными. В качестве критерия окончания внутренних итераций в большинстве солверов используются те же критерии, что и для внешних SWEEP-итераций, дополненные условием:

Число итераций ≤ MaxIter

причем для различных Ф-переменных можно задавать свое максимальное число итераций.

В солверах SparsKit2 используется еще одно условие – степень уменьшения невязки линейного уравнения:

$$R_P = r_P + \sum_{nb} a_{nb}\delta\Phi_{nb} - a_P\delta\Phi_P$$

в процессе решения

$$\frac{\sum|R_P|}{\sum|r_P|} < \text{Speed Res}$$

ⓘ *Описание в проекте прикладной задачи :*

Для всех Ф-переменных задается линейный солвер по умолчанию. Его имя указывается в операторе файла проекта в секции [Solver]:

```
LESSolver    = KIVA / KIVA_ILU / ATDMA / SIP_PHO / P.SIPSOL / P.CGS
               SPKT.CS / SPKT.CG / SPKT.BCGSTAB / SPKT.GMRES /
               SPKT. FGMRES / SPKT.FOM
```

Для конкретной Ф-переменной можно указать свой линейный солвер с помощью оператора секции [PHI Variables]:

```
LESSolver("Ф-имя")    = ....
```

Максимальное число итераций описывается оператором секции [PHI Variables]:

```
MaxIter("Ф-имя")    = 20
```

Параметр SpeedRes описывается оператором секции [Solver]:

```
SpeedRes  = 0.5
```

9.1 Последовательные линейные солверы

В составе текущей версии пакета имеется несколько LES-солверов, которые можно использовать в последовательном режиме расчета. Эти солверы перечислены в таблице 9.1.

Таблица 9.1 Последовательные линейные солверы

Название	Характеристика
aTDMA	- метод продольно-поперечной прогонки, применим только для структурной сетки;
SIP_PHO	- метод трехмерной прогонки, применим только для структурной сетки;
P.SIPSOL	- метод LU-декомпозиции для структурной сетки;
P.CGS	- итерационный алгоритм метода сопряженных градиентов для структурной сетки,
KIVA	- трехшаговый итерационный алгоритм пакета KIVA,
KIVA_ILU	- алгоритм KIVA с ILU – прекондиционером,
SPKT.XXX	- солверы из пакета SparsKit2

Для структурной сетки в последовательном режиме наилучший результат дают солверы aTDMA и KIVA_ILU.

9.1.1 aTDMA солвер

В этом солвере реализован алгоритм продольно-поперечной прогонки. На каждой внутренней итерации последовательно выполняется одномерный алгоритм прогонки по трем направлениям: вдоль оси x , y и z.

Для каждого направления, например для x, производится последовательный перебор одномерной «колонки» КО в перпендикулярном сечении:

iy = 2 .. NY-1 , iz = 2 .. NZ-1

Для каждой «колонки» КО с помощью *одномерной* прогонки решается одномерная в направлении оси x система уравнений

$$a_P\Phi'_P = a_W\Phi'_W + a_E\Phi'_E + \left\{ r_P + \sum_{N,S,H,L} a_k\Phi'_k \right\} \qquad (9.4)$$

9.1.2 KIVA и KIVA_ILU солверы

Эти солверы основаны на итерационных алгоритмах решения системы линейных уравнений в матричном виде. Подробно эти алгоритмы описаны в [6,7]. Суть этих алгоритмов заключается в следующем. Вместо системы (9.3) методом итераций решается система

$$[I]\bullet\left(\delta\boldsymbol{\Phi}^{k+1} - \delta\boldsymbol{\Phi}^{k}\right) = \alpha_k\mathbf{R}^k, \quad \mathbf{R}^k = \mathbf{r}^k - [A]\delta\boldsymbol{\Phi}^k \qquad (9.5)$$

В этом уравнении k- номер внутренней итерации, , [I] – единичная матрица, α_k – параметр коррекции невязки. Собственно алгоритмом расчета этого параметра α_k и отличаются различные линейные солверы.

Для расчета этого параметра в KIVA- солвере используется идея одновременной минимизации невязки и коррекции (этот солвер является основой CFD кода KIVA).

Для ускорения процесса сходимости системы (9.5) можно использовать алгоритм предобусловливания. Суть этого алгоритма основана на переходе от системы (9.3) к эквивалентной системе

$$[M]^{-1}[A]\delta\boldsymbol{\Phi} = [M]^{-1}\mathbf{r} \qquad (9.6)$$

где [M] – невырожденная матрица, которая называется предобусловлевателем или прекондиционером. Матрица M должна удовлетворять трем условиям:

1) она должна быть «близка» к матрице A;
2) она должна быть легко вычислима
3) она должна быть легко обратима.

Заметим, что если матрица M совпадает с матрицей A и мы нашли обратную матрицу, то решение находится мгновенно.

В солвере KIVA в качестве прекондиционера используется диаганальная часть матрицы A (фактически это означает «отсутствие» предобуславливания). В солвере KIVA_ILU в качестве матрицы M используется *приближенное* разложение матрицы A на «нижне-треугольную» и «верхне-треугольную» матрицы

$$[M] \approx [A_{low}]\bullet[A_{upper}]$$

Для нахождения «нижнее-треугольной» матрицы (в ней все элементы выше диагонали равны нулю) и «верхне-треугольную» матрицы (в ней все элементы ниже диагонали равны нулю) используется алгоритм ILU-факторизации, реализованный в пакете SparsKit2.

9.1.3 Остальные линейные солверы

Другие солверы, реализованные в Anes, носят «экспериментальный» характер. Однако их можно использовать для проведения расчетов. Эффективность того, или иного солвера можно определить только экспериментально.

9.2 Параллельные линейные солверы

В текущей версии для проведения параллельных расчетов можно использовать солверы KIVA, KIVA_ILU, SPKT.XX для неструктурных сеток и KIVA для структурных.
Дополнительно реализована экспериментальная версия внешнего параллельного солвера HYPRE, разработанного в «Center for Applied Scientic Computing Lawrence Livermore National Laboratory».

9.2.1 Солверы HYPRE

В текущей версии для неструктурной сетки реализован один multy-grid солвер BoomerAMG, для структурных - два солвера SMG и PFMG (полную документацию по солверам можно получить по ссылкам:

https://computing.llnl.gov/sites/default/files/public/hypre-2.11.2_usr_manual.pdf
https://computing.llnl.gov/sites/default/files/public/hypre-2.11.2_ref_manual.pdf

).

Для использования солверов HYPRE для всех Ф-переменных необходимо в секции [Solver] указать оператор:

```
LESsolver = HYPRE
```

Однако, как показал первый опыт, в силу того, что он требует большие ресурсы компьютера, правильнее использовать этот солвер для отдельной переменной (например, TS или PF для длинных каналов). В этом случае в секции [PHI Variables] указывается оператор:

```
LESsolver("Ф-имя") = HYPRE
```

Настройка параметров солверов HYPRE (а их там много) определяется операторами секции [Special Data]. Для выбора HYPRE солвера по умолчанию для конкретной Ф:

```
C("HYPRE.Solver") = "Имя HYPRE солвера"
C("HYPRE.Ф-имя.Solver") = "Имя HYPRE солвера"
```

В текущей версии используется только один HYPRE-солвер "BoomerAMG" для неструктурной сетки, поэтому этот оператор можно не указывать (солвер BoomerAMG является солвером по умолчанию). Для структурных сеток по умолчанию используется солвер SMG. Для настройки целых и действительных параметров конкретного HYPRE-солвера используются операторы вида:

```
I("BoomerAMG.<ИмяПараметра>") = <целое значение >
R("BoomerAMG.<ИмяПараметра>") = <действительное значение >
```

или

```
I("BoomerAMG.Ф-имя.<ИмяПараметра>") = <целое значение >
R("BoomerAMG.Ф-имя.<ИмяПараметра>") = < действительное значение >
```

9.2.2 Неструктурный солвер BoomerAMG

Для удобства работы с документацией HYPRE ниже будет указываться функция HYPRE, которая устанавливает параметр:

1. <ИмяПараметра> = PrintLevel : Выдача информации о процессе решения на консоль

Значения параметра:

Значение	Действие
0	no printout (default)
1	print setup information
2	print solve information
3	print both setup and solve information

Функция HYPRE: HYPRE_BoomerAMGSetPrintLevel()

По умолчанию PrintLevel = 0

2. <ИмяПараметра> = CoarsenType : Выбор Алгоритма Огрубления

Значения параметра:

Значение	Действие
0	- CLJP-coarsening (a parallel coarsening algorithm using independent sets.
1	- classical Ruge-Stueben coarsening on each processor, no boundary treatment (not recommended!)
3	- classical Ruge-Stueben coarsening on each processor, followed by a third pass, which adds coarse points on the boundaries
6	- Falgout coarsening (uses 1 first, followed by CLJP using the interior coarse points generated by 1 as its first independent set)
7	- CLJP-coarsening (using a fixed random vector, for debugging purposes only)
8	- PMIS-coarsening (a parallel coarsening algorithm using independent sets, generating lower complexities than CLJP, might also lead to slower convergence)
9	- PMIS-coarsening (using a fixed random vector, for debugging purposes only)
10	- HMIS-coarsening (uses one pass Ruge-Stueben on each processor independently, followed by PMIS using the interior C-points generated as its first independent set)
11	- one-pass Ruge-Stueben coarsening on each processor, no boundary treatment (not recommended!)
21	- CGC coarsening by M. Griebel, B. Metsch and A. Schweitzer
22	- CGC-E coarsening by M. Griebel, B. Metsch and A.Schweitzer

Функция HYPRE: HYPRE_BoomerAMGSetCoarsenType ()
По умолчанию используется значение CoarsenType = 6.

3. <ИмяПараметра> = RelaxType : Выбор типа гибридной релаксации для алгоритиов G-S/Jacobi
Значения параметра:

Значение	Действие
1	- Gauss-Seidel, sequential (very slow!)
2	- Gauss-Seidel, interior points in parallel, boundary sequential (slow!)
3	- hybrid Gauss-Seidel or SOR, forward solve
4	- hybrid Gauss-Seidel or SOR, backward solve
5	- hybrid chaotic Gauss-Seidel (works only with OpenMP)
6	- hybrid symmetric Gauss-Seidel or SSOR
8	- L1-scaled hybrid symmetric Gauss-Seidel
9	- Gaussian elimination (only on coarsest level)
15	- CG (warning - not a fixed smoother - may require FGMRES)
16	- Chebyshev
17	- FCF-Jacobi
18	- L1-scaled jacobi

Функция HYPRE: HYPRE_BoomerAMGSetRelaxType ()
По умолчанию используется значение RelaxType = 3.

4. <ИмяПараметра> = MaxLevels : Максимальное число уровней multigrid огрубления
Значения параметра - целое число больше нуля.
Функция HYPRE: HYPRE_BoomerAMGSetMaxLevels ()
По умолчанию используется значение MaxLevels = 20.

5. <ИмяПараметра> = MaxIter : Максимальное число внешних итераций.
Значения параметра устанавливается равным значению числу LES- итераций для каждой Ф-переменной в секции [PHI Variables]:
MaxIter("Ф-имя") = <число LES итераций>

По умолчанию используется значение MaxIter = 20.
Функция HYPRE: HYPRE_BoomerAMGSetMaxIter ()

6. <ИмяПараметра> = Tol : Точность решения на HYPRE-итерации.
Авторы не очень поясняют, что это такое. Но вроде бы из кода следует, что это относительное потребное уменьшение L2-нормы невязки по сравнению с входной L2-нормой (L2-норма - это SQRT(Sum($|r_p|$)). В коде по умолчанию предлагается значение Tol = 10^{-7}. В коде ANES по умолчанию (если этот параметр не задан) предлагается динамически рассчитывать эту величину по соотношению:

$$\mathrm{Tol} = 0.001 \cdot \frac{\varepsilon_\Phi}{e_\Phi}$$

Где ε_Φ - заданная точность решения для данной Ф-переменной, e_Φ - ошибка решения на текущей sweep-итерации ANES.
Функция HYPRE: HYPRE_BoomerAMGSetTol ()

Дополнительные параметры настройки были предложены К. Минко.

7. <ИмяПараметра> = AggNumLevel : Defines the number of levels of aggressive coarsening.
По умолчанию используется значение AggNumLevel = 0.
Функция HYPRE: HYPRE_BoomerAMGSetAggNumLevels ()

8. <ИмяПараметра> = InterpType : Defines which parallel interpolation operator is used.

Значение	Действие
0	- classical modified interpolation (default !)
1	- LS interpolation (for use with GSMG)
2	- classical modified interpolation for hyperbolic PDEs
3	- direct interpolation (with separation of weights)
4	- multipass interpolation
5	- multipass interpolation (with separation of weights)
6	- extended+i interpolation
7	- extended+i (if no common C neighbor) interpolation
8	- standard interpolation
9	- standard interpolation (with separation of weights)
10	- classical block interpolation (for use with nodal systems version only)
11	- classical block interpolation (for use with nodal systems version only) with diagonalized diagonal blocks
12	- FF interpolation
13	- FF1 interpolation
14	- extended interpolation

По умолчанию используется значение InterpType = 0.
Функция HYPRE: HYPRE_BoomerAMGSetInterpType ()

9. <ИмяПараметра> = PMaxElmts : Defines the maximal number of elements per row for the interpolation.
По умолчанию используется значение PMaxElmts = 0.
Функция HYPRE HYPRE_BoomerAMGSetPMaxElmts ()

9.2.3 Структурные солверы SMG и PFMG

Для структурных сеток реализованы два мульти-грид солвера SMG и PFMG. Их алгоритмы близки друг другу (см. документацию HYPRE), а для настройки используются следующие параметры:

1. <ИмяПараметра> = Tol : Точность решения на HYPRE-итерации.
То же самое, что и для BoomerAMG.

Дополнительные параметры для солвера PFMG:

2. <ИмяПараметра> = RAPtype : Coarse grid operator
Значения параметра:

Значение	Действие
0	Galerkin (default)
1	non-Galerkin ParFlow operators
2	Galerkin, general operators

Функция HYPRE: HYPRE_StructPFMGSetRAPType()

3. <ИмяПараметра> = RelaxType : Relaxation type
Значения параметра:

Значение	Действие
0	Jacobi
1	Weighted Jacobi (default)
2	R/B Gauss-Seidel
3	R/B Gauss-Seidel (nonsymmetric)

Функция HYPRE: HYPRE_StructPFMGSetRelaxType ()

4. <ИмяПараметра> = SkipRelax : Skip levels in PFMG (0 or 1)

Функция HYPRE: HYPRE_StructPFMGSetSkipRelax ()

Замечание: При использовании периодических граничных условий (например, по Y для задач в цилиндрической СК) число КО в направлении периодичности должно быть степенью двойки Ncv = 2^m!

9.2.4 Эксперименты с солверами HYPRE

Солвер BoomerAMG показал хорошую работоспособность для задач теплопроводности с сильно переменной ступенчатой теплопроводностью и для расчета PF в длинных каналах. Как показали первые эксперименты К. Минко время расчета можно сократить процентов на 40, если установить следующие значения, вместо значений по умолчанию:

```
I("BoomerAMG.CoarsenType")      = 10
I("BoomerAMG.AggNumLevels")   = 1
I("BoomerAMG.InterpType")   = 7
I("BoomerAMG. PMaxElmts")   = 5
```

Литература

1. Код Anes20xe. «Описание математических моделей кода», версия 2.24, 2019.
2. Anes20xe. «Работа с проектом пользователя», версия 2.24, 2019.
3. С. Патанкар. Численные методы решения задач тепломассообмена и динамики жидкости: Пер. с англ. - М.: Энергоатомиздат, 1984. - 152 с.
4. Ferziger, J.H. and Peric, M.: Computational methods for fluid dynamics: Springer Verlag, Berlin-New York, 1995.
5. Код Anes20xe. «Фортран-интерфейс пользователя», версия 2.24, 2019.
6. Баландин М.Ю., Шурина М.П. Методы решения СЛАУ большой размерности.// Новосибирск: НГТУ, 2000. – 79 стр.
7. Yousef Saad. Iterative methods for Sparse Linear Systems// January 3RD, 2000.
8. R.I. Issa. Solution of the implicitly discretised fluid flow equations by operator-splitting. // J.Comput. Phys., 1986, Vol. 62(1), p. 40-65;
9. H. Jasak, Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows, Ph.D. thesis, Imperial College, University of London, London, UK, 1996.
10. Anes20xe. «VOF алгоритм кода», версия 2.20, 2016.
11. J. Mencinger and I. Zun. On the finite volume discretization of discontinuous body force field on collocated grid: Application to vof method. Journal of Computational Physics, 221:524–538, 2007.
12. C. Hirsh. Numerical Computation of Internal and External Flows. Volume 1. Fundamentals of Computational Fluid Dynamics // Second Edition, 2007.