Метод контрольных объемов для численного решения задач механики сплошных сред

Подготовил: Рассадин А.А. студент группы 601-01
email: a.a.rassadin@yandex.ru

В работе описаны основные аспекты численного решения задач механики сплошных сред методом контрольных объемов. Сначала рассмотрены простейшие схемы дискретизации дифференциальных уравнений. Затем описаны наиболее распространенные итерационные алгоритмы решения систем линейных уравнений и методы численного решения уравнений гидродинамики. Многие алгоритмы описаны с учетом их реализации в программном комплексе OpenFOAM.

Оглавление

1 Метод контрольных объемов для численного решения задач гидродинамики

Метод контрольных объемов [1], [2] применяется для численного решения задач механики сплошных сред, в частности, для задач гидродинамики.

Непрерывная расчетная область заменяется дискретным (конечным) набором ячеек (контрольных объемов). Непрерывные векторные и скалярные поля тоже заменяются на дискретные. Значения дискретных полей определены в центрах контрольных объемов.

Дифференциальные уравнения интегрируются по каждой ячейке. Объемные интегралы преобразуются в поверхностные на основании теоремы Гаусса - Остроградского [3]. Каждый поверхностный интеграл аппроксимируется через сумму потоков по всем граням интегрируемой ячейки. Величины на гранях вычисляются через значения центрах ячеек. Конкретный способ вычислений определяется выбранной *схемой дискретизации*.

В результате для каждой ячейки получается *линейное уравнение*, содержащее значения искомой величины как в самой ячейке, так и в соседних с ней. Совокупность линейных уравнений по всем ячейкам расчетной сетки формирует систему линейных алгебраических уравнений (СЛАУ). Решение системы уравнений дает значения искомой величины в центрах контрольных объемов.

1.1 Дискретизация расчетной сетки

В общем случае расчетная сетка в методе контрольных объемов представляет собой конечный набор многогранников, каждый из них является ячейкой сетки[1]. На рис. 1 показан пример ячейки. Есть два вида ячеек: **внутренние** и **граничные**. Центры ячеек (центроиды) называют **узлами**.

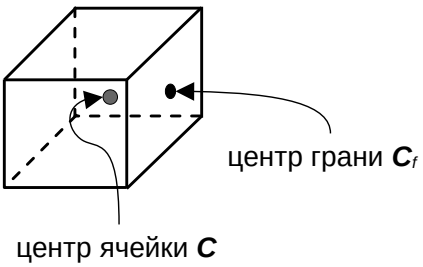


Рис. 1. Пример контрольного объема в трехмерной расчетной сетке, $\boldsymbol{C}$ – радиус-вектор центра ячейки, $\boldsymbol{C_f}$ – радиус-вектор центр одной из граней.

Важно сделать небольшое пояснение относительно центров ячеек и центров граней. Координаты вектора $\boldsymbol{C}$ – центра очередного контрольного объема определяются в *глобальной системе координат*, заданной для всей расчетной сетки. Координаты центров граней выбранной ячейки – векторов $\boldsymbol{C_f}$ записываются в *локальной системе координат*, определенной в каждой ячейке, причем начало отсчета локальной системы координат расположено в центре ячейки (в точке $\boldsymbol{C}$).

Грани внутренних ячеек являются общими для двух контрольных объемов. При интегрировании уравнения по внутренней ячейке в конечном счете рассчитываются потоки на всех её гранях. Для расчета этих потоков используются как текущая ячейка, так и смежные с ней. В результате на величину в центре данной ячейки влияют все ячейки, смежные с данной.

У граничных контрольных объемов есть грани несмежные ни с одним другим контрольным объемом расчетной сетки. В центрах таких граней задается **граничное условие** (отдельное условие для каждой искомой величины). Причем множество граничных ячеек может быть разделено на несколько частей, в каждой из них можно задать граничные условия разных типов.

В процессе расчетов важно знать следующую информацию о расчетной сетке:

- Расположение узловых точек (центров ячеек)
- Объем каждой ячейки
- Площадь каждой грани
- Вектор внешней нормали к каждой грани

Ниже приведен способ определения данных расчетной сетки, который реализован в платформе OpenFOAM [2].

[1]Для двумерной задачи ячейки это многоугольники, а в одномерном случае отрезки.

Объем ячейки – величина V вычисляется по формуле:

$$V = \int\limits_V dV = \frac{1}{3}\int\limits_V \nabla\cdot(x_1,x_2,x_3)\,dV = \frac{1}{3}\oint\limits_{\partial V}(x_1,x_2,x_3)\cdot d\boldsymbol{S} \approx \frac{1}{3}\sum_f \boldsymbol{S}_f\cdot\boldsymbol{C}_f \tag{1}$$

f – индекс грани

$\boldsymbol{S}_f$ – вектор нормали к грани с индексом f, $|\boldsymbol{S}_f|$ равен площади грани

$\boldsymbol{C}_f$ – вектор центра грани

Центр ячейки – вектор $\boldsymbol{C}$ вычисляется по формуле:

$\boldsymbol{x} = (x_1, x_2, x_3)$, где координаты локальные (от центра ячейки)

вспомогательный вектор $\boldsymbol{x}$ обладает свойством $\nabla(\boldsymbol{x}\cdot\boldsymbol{x}) \equiv 2\boldsymbol{x}$

$$\boldsymbol{C} = \frac{1}{V}\int\limits_V \boldsymbol{x}\,dV = \frac{1}{2V}\int\limits_V \nabla(\boldsymbol{x}\cdot\boldsymbol{x})\,dV = \frac{1}{2V}\oint\limits_{\partial V}(\boldsymbol{x}\cdot\boldsymbol{x})\,d\boldsymbol{S} \approx \tag{2}$$

$$\approx \frac{1}{2V}\sum_f (\boldsymbol{C}_f\cdot\boldsymbol{C}_f)\boldsymbol{S}_f$$

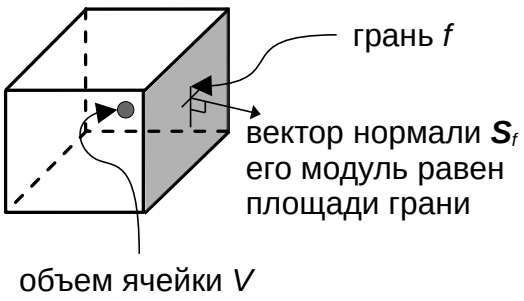
Для вычисления площади грани – S_f и её центра – $\boldsymbol{C}_f$ многоугольник, которым является грань, разбивается на N треугольников ($N = M-2$, где M – число вершин многоугольника). Пример показан на рис. 2. Для каждого из треугольников определяются площадь S_t и его центр $\boldsymbol{C}_t$, где t – номер треугольника. Формулы для расчета вспомогательных величин S_t, $\boldsymbol{C}_t$, $\boldsymbol{a}$:

$\boldsymbol{p}_i$ – вектор вершины грани, $\qquad 1 \le i \le N$

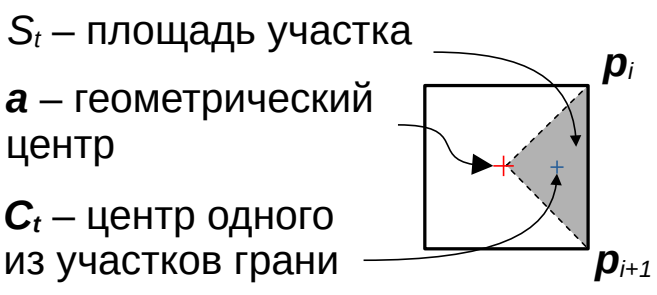
$$\boldsymbol{a} = \frac{1}{N}\sum_i \boldsymbol{p}_i \text{ – геометрический центр грани}$$

$$S_t = \frac{(\boldsymbol{p}_{i+1}-\boldsymbol{p}_i)\times(\boldsymbol{a}-\boldsymbol{p}_i)}{2} \qquad (\text{«}\times\text{»} \text{ – векторное произведение}) \tag{3}$$

$$\boldsymbol{C}_t = \frac{\boldsymbol{p}_{i+1}+\boldsymbol{p}_i+\boldsymbol{a}}{3}$$



(a) Общий вид контрольного объема



(b) Одна из граней контрольного объема

Рис. 2. Пояснение к обозначениям, используемым в формуле (3)

Площадь грани – величина S_f вычисляется по формуле:

$$S_f = \sum_{t=1}^{N} S_t \tag{4}$$

Центр грани – вектор $\boldsymbol{C}_f$ вычисляется по формуле:

$$\boldsymbol{C}_f = \frac{1}{S_f} \sum_{t=1}^{N} S_t \boldsymbol{C}_t \tag{5}$$

При определении вектора $\boldsymbol{C}_f$ – центра грани используется *взвешенная* векторная сумма центров треугольников, на которые разбита грань. Вес определяется в зависимости от величины площади частичного треугольника (см. рис. 2).

1.2 Дискретизация дифференциальных уравнений

Под *дискретизацией* в методе контрольных понимается процесс замены дифференциальных уравнений для *непрерывных* величин на линейные уравнения для *дискретных* полей.

Схема дискретизации влияет на систему линейных уравнений. Для каждой из искомых величин формируется отдельная СЛАУ. Например, для величины Ψ соответствующая СЛАУ в матричной форме записывается как: $\mathrm{A}\boldsymbol{\Psi} = \boldsymbol{b}$, где A – квадратная матрица коэффициентов, $\boldsymbol{b}$ – вектор свободных членов (также называют источником), $\boldsymbol{\Psi}$ – вектор, содержащий искомый набор значений в центрах ячеек.

Если при аппроксимации дифференциальных операторов изменяется матрица A, то численная схема называется **неявной**. Если изменения происходят в векторе свободных членов $\boldsymbol{b}$, то схема **явная**.

Далее будут рассмотрены примеры схем дискретизации для стандартных дифференциальных операторов: $\frac{\partial}{\partial t}$, div, grad, div(grad) $\equiv \Delta$. Каждая из представленных схем дискретизации реализована в коде платформы OpenFOAM.

1.2.1 Интерполяция значений с центров ячеек на грани

Для сторон граничных ячеек, которые не смежны с другими ячейками, значения определяются в зависимости от типа граничных условий. Для граней принадлежащих внутренним ячейкам всегда есть ровно две узловых точки, через которые определяется значение на грани, этот случай показан на рис. 3. В этом случае величина Ψ_f вычисляется как взвешенная сумма значений в соседних узлах P и N, между которыми заключена грань f:

$$\Psi_f = \omega\Psi_P + (1-\omega)\Psi_N, \qquad 0 < \omega < 1 \tag{6}$$

В OpenFOAM есть метод **линейной интерполяции**, согласно этому методу вес ω выбирается на основании расстояний от центра грани f до центров ячеек:

$$\omega = \frac{\boldsymbol{n} \cdot \boldsymbol{d}_N}{\boldsymbol{n} \cdot (\boldsymbol{d}_P + \boldsymbol{d}_N)} \tag{7}$$

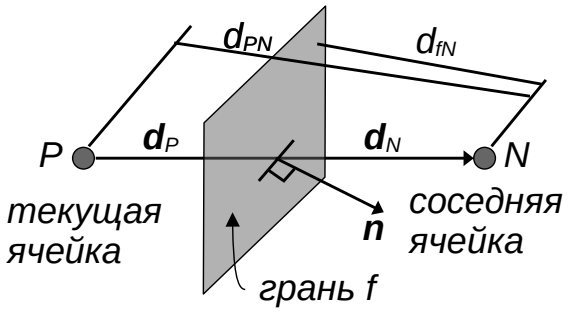


Рис. 3. Грань f расположена между ячейками с центрами в точках $\boldsymbol{P}$ и $\boldsymbol{N}$.

1.2.2 Производная по времени

При дискретизации производной по времени можно использовать **схему Эйлера**:

$$\begin{aligned} &\frac{\partial \Psi_P}{\partial t} = \frac{\Psi_P - \Psi_P^o}{\Delta t} \\ &\Psi_P \text{ – значение в узле } P \text{ на текущем временном слое,} \\ &\Psi_P^o \text{ – значение в узле } P \text{ на предыдущем временном слое,} \\ &\Delta t \text{ – величина шага по времени} \end{aligned} \tag{8}$$

В системе линейных уравнений $\mathrm{A}\boldsymbol{\Psi} = \boldsymbol{b}$ аппроксимация оператора $\frac{\partial}{\partial t}$ по схеме Эйлера изменяет как матрицу A, так и вектор свободных членов $\boldsymbol{b}$. В матрице A к *элементам на главной диагонали* добавляется величина $\frac{1}{\Delta t}$, а остальные коэффициенты остаются прежними. К каждой компоненте $\boldsymbol{b}_P$ вектора свободных членов добавляется слагаемое $\frac{\Psi_P^0}{\Delta t}$.

Для получения физически осмысленного результата, значение Δt – шага по времени нужно задавать с учетом *критерия Куранта* [2]. В данной работе уже было рассмотрено условие Куранта, выражение (**??**) реализовано в программной платформе OpenFOAM.

1.2.3 Лапласиан

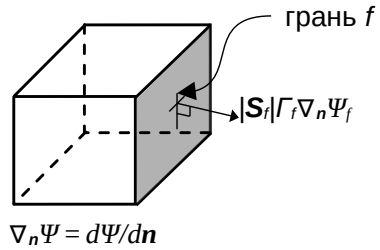


Рис. 4. Иллюстрация к дискретизации лапласиана.

В дифференциальных уравнениях оператор Лапласа часто используется для описания процесса диффузии, в тех случаях когда вещество переходит из областей с большей концентрацией в области, где она меньше. Данный процесс регулируется параметром Γ, характеризующим интенсивность диффузии. Соответствующий оператор (лапласиан) задается формулой:

$$\nabla \cdot (\Gamma \nabla \Psi) \tag{9}$$

Выражение (9) это дивергенция вектора $\Gamma\nabla\Psi$. По определению [3] дивергенция это предел отношения потока через поверхность области, когда её объем стремится к нулю. То есть выражение (9) преобразуется следующим образом:

$$\begin{gathered}\nabla \cdot (\Gamma \nabla \Psi) = \lim_{V \to 0} \frac{1}{V} \oint_{\partial V} (\Gamma \nabla \Psi \cdot d\boldsymbol{S}) = \lim_{V \to 0} \frac{1}{V} \oint_{\partial V} \Gamma \nabla_{\boldsymbol{n}} \Psi \, dS \\ \nabla_{\boldsymbol{n}} \Psi \equiv \frac{\partial \Psi}{\partial \boldsymbol{n}}, \text{ где } \boldsymbol{n} \text{ – вектор внешней нормали на поверхности } \partial V\end{gathered} \tag{10}$$

Дискретизация слагаемого $\nabla \cdot (\Gamma \nabla \Psi)$ заключается в замене последнего интеграла в формуле (10) на его численный аналог:

$$\begin{gathered}\nabla \cdot (\Gamma_P \nabla \Psi_P) \approx \frac{1}{V} \sum_f \Gamma_f \nabla_{\boldsymbol{n}} \Psi_f S_f \\ V \text{ – объем ячейки с узловой точкой } \boldsymbol{P} \\ S_f \equiv |\boldsymbol{S}_f| \text{ – площадь грани с индексом } f\end{gathered} \tag{11}$$

Для расчета по формуле (11) требуется задать схему интерполяции для величины Γ, а также метод вычисления производной по нормали для величины Ψ, также нужна информация о сетке. Значения V и S_f определяются формулами (1), (4) соответственно. Для расчета значений Γ_f можно воспользоваться линейной интерполяцией, которая была рассмотрена выше.

Аппроксимация производной по нормали на гранях ячеек Обозначение $(\nabla_{\boldsymbol{n}}\Psi)_f$ используется для численного аналога производной по нормали, вычисляемой на грани f, которая заключена между ячейками P и N:

$$\left.\frac{\partial \Psi}{\partial \boldsymbol{n}}\right|_f = (\nabla \Psi \cdot \boldsymbol{n})\big|_f \approx (\nabla_{\boldsymbol{n}} \Psi)_f \tag{12}$$

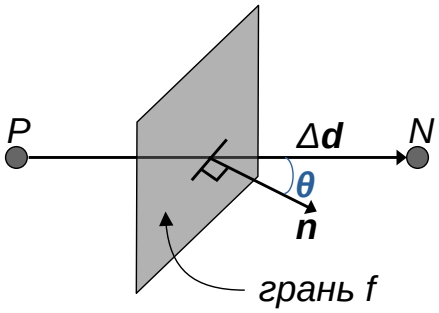


Рис. 5. Иллюстрация к вычислению нормальной производной $\nabla_{\boldsymbol{n}}\Psi$. Формула (13) применима, когда угол θ между векторами $\Delta\boldsymbol{d} = (\boldsymbol{P} - \boldsymbol{N})$ и $\boldsymbol{n}$ стремится к нулю.

Существует множество схем для расчета $(\nabla_{\boldsymbol{n}}\Psi)_f$. Простейший вариант определяется формулой:

$$(\nabla_{\boldsymbol{n}}\Psi)_f = \frac{\Psi_N - \Psi_P}{|\boldsymbol{N} - \boldsymbol{P}|} = \frac{\Psi_N - \Psi_P}{|\Delta\boldsymbol{d}|} \tag{13}$$

Формула (13) предназначена для **ортогональной сетки**. У такой сетки вектор, соединяющий соседние узлы коллинеарен вектору нормали к общей грани. Если сетка **неортогональная**, то требуется дополнительная коррекция потоков $(\nabla_{\boldsymbol{n}}\Psi)_f S_f$. Коррекция добавит слагаемые, которые будут вычисляться явно.

Таким образом, для системы уравнений $\mathrm{A}\boldsymbol{\Psi} = \boldsymbol{b}$ на ортогональной сетке дискретизация лапласиана изменяет только коэффициенты матрицы A. На неортогональной сетке изменятся как матрица A, так и вектор $\boldsymbol{b}$.

1.2.4 Конвективное (адвективное) слагаемое

Конвективное (адвективное) слагаемое отражает «перенос» некоторой величины Ψ основным течением, скорость которого задана векторным полем $\boldsymbol{u}$. Это слагаемое задается выражением вида $\nabla \cdot (\Psi\boldsymbol{u})$. В частности, переносимой величиной может быть одна из компонент скорости основного потока, например, $\nabla \cdot (u_1\boldsymbol{u})$.

Согласно определению дивергенция векторного поля $\Psi\boldsymbol{u}$ вычисляется через предельное значение потока:

$$\nabla \cdot (\Psi\boldsymbol{u}) = \lim_{V\to 0} \frac{1}{V} \oint_{\partial V} (\Psi\boldsymbol{u} \cdot d\boldsymbol{S}) \tag{14}$$

Поток по всей области V вычисляется через поверхностный интеграл. В методе контрольных объемов вместо бесконечно малой области V рассматривают ячейки расчетной сетки. Для каждой из них последний интеграл в формуле (14) аппроксимируется суммой потоков через все грани ячейки. Таким образом, дискретиза-

ция слагаемого $\nabla \cdot \boldsymbol{\Psi}$ для ячейки с центром в точке $\boldsymbol{P}$ выполняется по формуле:

$$
\begin{gathered}
[\nabla \cdot (\Psi \boldsymbol{u})]_P = \frac{1}{V_P} \sum_f (\Psi_f \boldsymbol{u}_f \cdot \boldsymbol{S}_f) \approx \frac{1}{V_P} \sum_f (\Psi_f \phi_f) \\
\phi_f \text{ – массовый поток через грань с индексом } f, \\
V_P \text{ – объем ячейки с центром в точке } \boldsymbol{P}
\end{gathered}
\tag{15}
$$

Массовые потоки ϕ_f вычисляются *явным* образом. Для этого используются данные сетки (величины V_p, $\boldsymbol{S}_f$). Значения скорости $\boldsymbol{u}_f$ на гранях ячейки должны быть известны известны. Они могут быть заданы пользователем, либо берутся данные с предыдущего временного слоя или прошлой внутренней итерации.

Схемы дискретизации конвективного слагаемое различаются методом расчета значений Ψ_f. Сложность здесь в том, что при вычислении Ψ_f нужно учесть влияние течения, так как Ψ «переносится» потоком.

Схема против потока [1]
При движении жидкости (переносящей величину Ψ) из одной ячейки в другую, жидкость «знает» о состоянии величины Ψ в предыдущих ячейках, но для следующей ячейки данных про Ψ нет. Поэтому Ψ_f вычисляется по формуле:

$$
\Psi_f = \begin{cases}
\Psi_f = \Psi_P, \quad \text{если } \phi_f > 0 \\
\Psi_f = \Psi_N, \quad \text{если } \phi_f < 0 \\
\boldsymbol{N} \text{ – центр ячейки, смежной с гранью } f
\end{cases}
\tag{16}
$$

Относительно СЛАУ $\mathrm{A}\boldsymbol{\Psi} = \boldsymbol{b}$ дискретизация конвективного слагаемого влияет только на коэффициенты матрицы A. Причем в общем случае уравнения для смежных ячеек содержат *не симметричные коэффициенты.*

1.2.5 Источниковое слагаемое

Источниковое слагаемое может не зависеть от искомой величины Ψ. В такой ситуации значения S_P источника в центрах ячеек определены явно. Поэтому изменяется лишь правая часть СЛАУ $\mathrm{A}\boldsymbol{\Psi} = \boldsymbol{b}$.

В случае, когда источниковое слагаемое зависит от искомой величины Ψ, для его дискретизации можно применить **линеаризацию** [1]. Например, для источника $S = S(\Psi)$ может быть допустимым представление в виде: $S = S_0 + S_\Psi \Psi$. При данном подходе значения S_0 изменяют вектор свободных членов $\boldsymbol{b}$, а S_Ψ влияют на *диагональные коэффициенты* матрицы A. Возможна ситуация, когда из-за этого на диагонали возникнут нулевые значения. Это сделает невозможным решение системы линейных уравнений. Если так случилось, то нужно выбрать другую схему дискретизации источника.

1.2.6 Дивергенция

По определению дивергенция векторного поля $\boldsymbol{\Psi}$ вычисляется по формуле:

$$\nabla \cdot \boldsymbol{\Psi} = \lim_{V \to 0} \frac{1}{V} \oint_{\partial V} (\boldsymbol{\Psi} \cdot d\boldsymbol{S}) \tag{17}$$

Поверхностный интеграл в формуле (17) аппроксимируется стандартным способом через сумму потоков по всем граням рассматриваемой ячейки. Поэтому численный аналог слагаемого $\nabla \cdot \boldsymbol{\Psi}$ для ячейки с центром в точке $\boldsymbol{P}$ определяется выражением:

$$[\nabla \cdot \boldsymbol{\Psi}]_P \approx \frac{1}{V_P} \sum_f (\boldsymbol{\Psi}_f \cdot \boldsymbol{S}_f) \tag{18}$$

Значения $\boldsymbol{\Psi}_f$ вычисляются на гранях контрольных объемов в зависимости от выбранной схемы интерполяции, величины V_P, $\boldsymbol{S}_f$ известны из расчетной сетки.

Дивергенция всегда вычисляется явным образом. Значения $\boldsymbol{\Psi}_f$ берутся либо с предыдущего временного слоя, либо с последней внутренней итерации. Таким образом, дискретизация дивергенции изменяет только вектор $\boldsymbol{b}$ в правой часть системы линейных уравнений $\mathrm{A}\boldsymbol{\Psi} = \boldsymbol{b}$.

1.2.7 Градиент

Градиент скалярной величины Ψ можно вычислить по следующей формуле [2]:

$$\nabla \Psi = \lim_{V \to 0} \frac{1}{V} \oint_{\partial V} (\Psi d\boldsymbol{S}) \tag{19}$$

При численном решении поверхностный интеграл заменяется на сумму потоков на всех гранях ячейки расчетной сетки. Поэтому дискретизация слагаемого $\nabla \Psi$ для контрольного объема с центром в точке $\boldsymbol{P}$ определяется выражением:

$$(\nabla \Psi)_P \approx \frac{1}{V_P} \sum_f (\Psi_f \boldsymbol{S}_f) \tag{20}$$

Значения Ψ_f определяются на гранях контрольных объемов с использованием схемы интерполяции, значения V_P и $\boldsymbol{S}_f$ найдены из расчетной сетки.

Градиент всегда вычисляется явным образом. Значения Ψ_f берутся либо с предыдущего временного слоя, либо с последней внутренней итерации. При аппроксимации градиента изменяется только правая часть системы линейных уравнений $\mathrm{A}\boldsymbol{\Psi} = \boldsymbol{b}$.

1.3 Общие сведения о системах уравнений, которые получаются в методе контрольных объемов

Для величины Ψ результат интегрирования и дискретизации дифференциального уравнения по одной из ячеек дает линейное уравнение следующего вида:

$$a_P \Psi_P + \sum_{nb} a_{nb} \Psi_{nb} = b_P \tag{21}$$

В выражении (21) индекс nb используется для обозначения соседних ячеек, P – для значения, вычисляемого в центре данной ячейки. Уравнение вида (21) получается *для всех* контрольных объемов расчетной сетки. Граничные условия задачи учитываются через уравнения для граничных ячеек.

Совокупность уравнений (21) образует систему:

$$\begin{cases} a_{1,1}\Psi_1 + a_{1,2}\Psi_2 + \cdots + a_{1,N}\Psi_N = b_1 \\ a_{2,1}\Psi_1 + a_{2,2}\Psi_2 + \cdots + a_{2,N}\Psi_N = b_2 \\ \cdots \\ a_{N,1}\Psi_1 + a_{N,2}\Psi_2 + \cdots + a_{N,N}\Psi_N = b_N \end{cases} \tag{22}$$

N – число ячеек в расчетной сетке

i-ое уравнение соответствует значению Ψ в центре i-ой ячейки

Матричная форма записи системы (22):

$$\mathrm{A}\boldsymbol{\Psi} = \boldsymbol{b}$$

A – результат *неявной* дискретизации

$\boldsymbol{b}$ – результат *явной* дискретизации

$\boldsymbol{\Psi}$ – *неизвестный* вектор, содержащий сеточные значения поля Ψ (23)

Для i-ой строки матрицы A ненулевые коэффициенты $a_{i,j}$ будут только для ячеек с номерами j, смежных с i-ой ячейкой. Например, уравнение для ячейки, показанной на рис. 6 будет таким:

$$a_{12,12}\Psi_{12} + a_{12,7}\Psi_7 + a_{12,11}\Psi_{11} + a_{12,17}\Psi_{17} + a_{12,13}\Psi_{13} = b_{12} \tag{24}$$

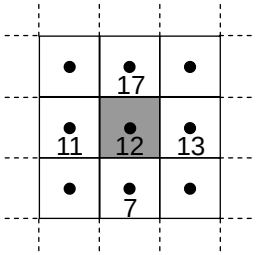


Рис. 6. Пример внутренней ячейки в двумерном случае.

В практических расчетах используют расчетные сетки с *большим количеством* контрольных объемов. При этом число ненулевых значений в матрице A существенно меньше, чем размер ячейки. Следовательно, матрица A является *разреженной*. На внутреннюю структуру матрицы A влияет как расчетная сетка, так и схемы дискретизации.

Например [2], в результате дискретизации лапласиана в матрицу добавляются симметричные коэффициенты $a_{i,j} = a_{j,i}$. Причем диагональный коэффициент $a_{i,i}$ равен сумме коэффициентов для соседних ячеек, взятой с обратным знаком. Конвективное (адвективное) слагаемое в общем случае вносит несимметричные коэффициенты. Производная по времени всегда увеличивает коэффициенты на главной диагонали матрицы и вносит вклад в правую часть СЛАУ.

Случай с несколькими неизвестными величинами
Если в рассматриваемой задаче несколько искомых величин, например, три компоненты вектора скорости (u_1, u_2, u_3) и скалярное поле давления p. То для *каждой* из них формируется собственная СЛАУ вида (23). Таким образом, в данном случае будет сформировано 4 системы уравнений.

Возможна ситуация, когда в уравнении для одной искомой величины нужно использовать другую искомую величину. Например, $\boldsymbol{u} \cdot \nabla\Psi = u_1 \frac{\partial\Psi}{\partial x^1} + u_2 \frac{\partial\Psi}{\partial x^2} + u_3 \frac{\partial\Psi}{\partial x^3}$. В этом случае в линейных уравнений для величины Ψ будут использованы значения компонент скорости (u_1, u_2, u_3), которые берутся с предыдущей внутренней итерации, либо с прошлого временного слоя.

Процедура численного решения системы линейных уравнений
В практических расчетах системы линейных уравнений решают **итерационными алгоритмами**. Это значит, что искомые величины определяются в результате многократного выполнения однотипных операций.

Для любого итерационного алгоритма задаются **начальное приближение** и **критерий остановки**. Также нужно учитывать, что последовательность итераций алгоритма дает требуемый результат только тогда, когда выполнены **условия сходимости**.

При численном решении СЛАУ ставится задача нахождения **приближенного** значения[2]. Близость найденного решения к точному можно определить с помощью **невязки**.

[2] *Точное* решение найти невозможно, как минимум, из-за ограничения разрядности чисел в компьютерах.

Что такое невязка

Пусть имеют место равенства:

$$\begin{aligned} &\mathrm{A}\boldsymbol{\Psi}^{ex} = \boldsymbol{b} \qquad \text{и} \qquad \mathrm{A}\boldsymbol{\Psi}^{i} = \boldsymbol{b}^{i}, \\ &\text{где } \boldsymbol{\Psi}^{ex} \text{ – точное решение, } \boldsymbol{\Psi}^{i} \text{ – приближенное} \end{aligned} \tag{25}$$

Тогда отклонение $\boldsymbol{\Psi}^i$ от $\boldsymbol{\Psi}^{ex}$ можно задать как вектор $\boldsymbol{r}$:

$$\boldsymbol{r} = \mathrm{A}\boldsymbol{\Psi}^{ex} - \mathrm{A}\boldsymbol{\Psi}^{i} = \boldsymbol{b} - \boldsymbol{b}^{i} \tag{26}$$

В выражении (26) значение k-ой компоненты вектора $\boldsymbol{r}$ показывает насколько правая часть k-го уравнения системы отличается от точного значения при подстановке систему $\mathrm{A}\boldsymbol{\Psi} = \boldsymbol{b}$ приближенного решения $\boldsymbol{\Psi}^i$. Компоненты вектора $\boldsymbol{r}$ могут принимать значения разных знаков.

Количественная оценка отклонения определяется как норма вектора $\boldsymbol{r}$:

$$r = ||\boldsymbol{r}|| = \left|\left|\boldsymbol{b} - \mathrm{A}\boldsymbol{\Psi}^{i}\right|\right| \tag{27}$$

Для вычисления нормы (27) могут быть использованы разные формулы. Способ расчета нормы определяет «характер близости» близости приближенного решения к точному.

Например, в платформе OpenFOAM векторная норма вычисляется по формуле:

$$\begin{aligned} &\boldsymbol{r} = (r_1, \ \ldots \ , r_N) \\ &r = ||\boldsymbol{r}|| = \sum_{k=1}^{N} |r_k| \end{aligned} \tag{28}$$

Ясно, что при сходимости приближенного решения к точному:

$$\boldsymbol{\Psi}^i \to \boldsymbol{\Psi}^{ex} : \ \begin{cases} \boldsymbol{r} = \boldsymbol{b} - \mathrm{A}\boldsymbol{\Psi}^{i} \ \to \ \boldsymbol{0} \\ r = ||\boldsymbol{r}|| \ \to \ 0 \end{cases} \tag{29}$$

В зависимости от контекста термин «**невязка**» применяется либо к *вектору* $\boldsymbol{r}$, либо к *числу* $r = ||\boldsymbol{r}||$. Таким образом, невязка является мерой различия между приближенным и точным решениями.

Критерий остановки решения СЛАУ в OpenFOAM
В программном комплексе OpenFOAM в качестве критерия остановки процедуры итерационного решения СЛАУ можно выбрать один из двух параметров:

1. r_{abs} – по значению **абсолютной** невязки
2. r_{rel} – по значению **относительной** невязки

Формула расчета **абсолютной невязки**:

$$r_{abs} = \frac{||\boldsymbol{b} - \mathrm{A}\boldsymbol{\Psi}||}{||\mathrm{A}\boldsymbol{\Psi} - \mathrm{A}\hat{\boldsymbol{\Psi}}|| + ||\boldsymbol{b} - \mathrm{A}\hat{\boldsymbol{\Psi}}||}$$

$\boldsymbol{\Psi}$ – численное решение на данной итерации

$\hat{\boldsymbol{\Psi}}$ – вектор средних значений: $\hat{\Psi}_i = \dfrac{1}{N}\sum_{j=1}^{N}\Psi_j, \quad 1 \le i \le N$

векторные нормы вычисляются по формуле (28) (30)

Формула (30) выведена таким образом, что значение r_{abs} не зависело от «масштаба» решаемой задачи (размера и типа расчетной сетки) [2].

Формула расчета **относительной невязки**:

$$r_{rel} = \frac{r_{abs}}{r^1_{abs}}$$

r_{abs} – абсолютная невязка на данной итерации

r^1_{abs} – абсолютная невязка на первой итерации (31)

В любой программной реализации итерационного алгоритма в дополнение к критерию сходимости по невязке, всегда должно быть **ограничение на число итераций**.

В OpenFOAM во время конфигурации решения СЛАУ пользователь задает пороговые значения для абсолютной и относительной невязок. При необходимости он может изменить значение лимита на число итераций (по умолчанию ограничение в 1000 итераций).

Решение системы уравнений завершается в одном из двух случаев:

- Одна из невязок (30) или (31) стала меньше соответствующего порогового значения.
- Достигнут лимит по числу итераций.

2 Решение систем линейных алгебраических уравнений

2.1 Метод Гаусса-Зейделя

Основная идея метода Гаусса-Зейделя состоит в том из исходной системы уравнений «выделяется» главная диагональ. В итоге при вычислении значения Ψ_i^n (Ψ в ячейке с индексом i на n-ой итерации) используются значения Ψ из соседних ячеек, *причем часть из них была уже обновлена на текущей итерации.* Фактически, данный метод является модификацией метода простой итерации [4].

2.1.1 Итерационная формула

Вычисление компонент вектора $\mathbf{\Psi}^n$ – очередного приближения на n-ой итерации делается по формуле:

$$\Psi_i^n = \frac{1}{a_{i,j}} \left(b_i \ - \sum_{j=1}^{i-1} a_{i,j} \Psi_j^n \quad - \sum_{j=i+1}^{N} a_{i,j} \Psi_j^{n-1} \right), \tag{32}$$

где N – количество уравнений

В контексте задач механики сплошных формула (32) означает, что значение величины Ψ в ячейке с номером i вычисляется как *взвешенная* сумма по значениям соседних ячеек.

2.1.2 Условие сходимости

На каждой итерации значения Ψ_i^n отличаются от точных $\overline{\Psi}_i$ на величину ε_i^n:

$$\Psi_i^n = \overline{\Psi}_i + \varepsilon_i^n \tag{33}$$

Для любого уравнения системы выполняется равенство:

$$a_{i,i} \overline{\Psi}_i \ + \sum_{\substack{j=1 \\ j \neq i}}^{N} a_{i,j} \overline{\Psi}_{i,j} = b_i \tag{34}$$

Следовательно, для ошибок в каждом уравнении получается равенство:

$$a_{i,i} \varepsilon_i^n \ = \sum_{\substack{j=1 \\ j \neq i}}^{N} a_{i,j} \varepsilon_j^n \tag{35}$$

В формуле (35) можно сделать оценку:

$$|\varepsilon_i^n| \leq \sum_{\substack{j=1 \\ j \neq i}}^{N} \left| \frac{a_{i,j}}{a_{i,i}} \right| |\varepsilon_j^n| \leq \sum_{\substack{k=1 \\ k \neq i}}^{N} \left| \frac{a_{i,k}}{a_{i,i}} \right| \sum_{\substack{j=1 \\ j \neq i}}^{N} |\varepsilon_j^n| \tag{36}$$

Таким образом, получается *достаточное* условие сходимости [2]:

$$\begin{cases} |a_{i,\,i}| \geq \sum\limits_{j=1,\,j\neq i}^{N} |a_{i,\,j}|, \qquad 1 \leq i \leq N \\ \textbf{найдется, хотя бы одна строка } \boldsymbol{i_0}\textbf{, где неравенство строгое} \end{cases} \tag{37}$$

При выполнении условия (37) получается, что *абсолютное* значение каждого отклонения ε_i^n *не больше*, чем сумма остальных отклонений. Причем, как минимум, для одного из уравнений ошибка $\varepsilon_{i_0}^n$ уменьшится на n-ой итерации.

2.1.3 Особенности метода

- *Число операций на одной итерации пропорционально количеству ячеек.*
- *Изменения поля* Ψ *распространяются за одну итерацию на одну ячейку.*
- *Скорость сходимости зависит от способа нумерации ячеек.*
  Наилучший вариант, когда следующее уравнение записывается для ячейки смежной с данной. Тогда при расчете Ψ для следующей ячейки, используется значение Ψ в предыдущей ячейке, которое уже обновлено. То есть следующая ячейка сразу же «ощутит» изменение Ψ.

В стандартном варианте метода Гаусса-Зейделя обновленные значения поля Ψ для текущей итерации появляются «в порядке возрастания номеров ячеек». В платформе OpenFOAM реализован **симметричный алгоритм Гаусса-Зейделя** [2]. Он отличается от стандартного тем, что на четных итерациях система решается «сверху вниз» (в обычном порядке), а на нечетных «снизу вверх» (в обратном порядке). В некоторых такой прием ускоряет сходимость.

2.2 Градиентные методы

2.2.1 Основная идея

Задача решения СЛАУ $\mathrm{A}\boldsymbol{\Psi} = \boldsymbol{b}$ состоит в том, чтобы определить компоненты вектора $\boldsymbol{\Psi} = (\Psi_1, \ldots, \Psi_N)$. При выполнении *некоторых условий* её можно свести к оптимизационной задаче по минимизации функции нескольких переменных f.

Условия на матрицу A, для возможности применения градиентных методов:

$$\begin{cases} \mathrm{A} \text{ – симметричная матрица, то есть } \mathrm{A} = \mathrm{A}^T \\ \mathrm{A} \text{ – положительно определенная [4], то есть } \forall\, \boldsymbol{\Psi} :\ (\mathrm{A}\boldsymbol{\Psi}) \cdot \boldsymbol{\Psi} \geq 0 \end{cases} \tag{38}$$

Симметричная матрица с диагональным преобладанием (37), у которой все элементы на главной диагонали *положительны* удовлетворяет условиям (38) [2].

Функция $f(\boldsymbol{\Psi}) = f(\Psi_1, \ldots, \Psi_N)$ определяется специальным образом:

$$f(\boldsymbol{\Psi}) = \frac{1}{2}\boldsymbol{\Psi}^T \mathrm{A}\boldsymbol{\Psi} - \boldsymbol{b}^T\boldsymbol{\Psi} = \frac{1}{2}\boldsymbol{\Psi} \cdot (\mathrm{A}\boldsymbol{\Psi}) - \boldsymbol{b} \cdot \boldsymbol{\Psi} \tag{39}$$

В выражении (39) функция $f(\boldsymbol{\Psi})$ является *квадратичной формой* [3]. Это значит, что $f(\boldsymbol{\Psi})$ является скалярным выражением, состоящим из суммы слагаемых вида $b_{ik}\Psi_i\Psi_k$ и $b_j\Psi_j$.

В случае, когда $\mathrm{A} = \mathrm{A}^T$ градиент функции f вычисляется по формуле [5]:

$$\nabla f(\boldsymbol{\Psi}) = \mathrm{A}\boldsymbol{\Psi} - \boldsymbol{b} = -\boldsymbol{r} \tag{40}$$

Условие *положительной определенности* матрицы A используется для того, чтобы функция $f(\boldsymbol{\Psi})$ имела глобальный минимум [3]. В точке $\boldsymbol{\Psi}^0$ экстремума (минимума) функции $f(\boldsymbol{\Psi})$ выполняется условие:

$$\nabla f(\boldsymbol{\Psi})|_{\boldsymbol{\Psi}^0} = \mathrm{A}\boldsymbol{\Psi}^0 - \boldsymbol{b} = \boldsymbol{0} \tag{41}$$

Таким образом, при выполнении условий (38) задача решения СЛАУ $\mathrm{A}\boldsymbol{\Psi} = \boldsymbol{b}$ *эквивалентна* поиску минимума функции (39).

Градиентные методы поиска минимума функции $f(\boldsymbol{\Psi})$ определяют такую последовательно точек, что $f(\boldsymbol{\Psi}^{n+1}) < f(\boldsymbol{\Psi}^n)$. Ясно, что когда $n \to +\infty$, при наличии глобального минимума, последовательность значений $f(\boldsymbol{\Psi}^n)$ будет стремиться к нему. Очередное приближение к точке минимума описывается *общей формулой*:

$$\boldsymbol{\Psi}^{n+1} = \boldsymbol{\Psi}^n + \alpha^n \boldsymbol{p}^n \tag{42}$$

В рамках этого подхода алгоритмы различаются способом определения величин α^n и $\boldsymbol{p}^n$. Вектор $\boldsymbol{p}^n$ определяет направление убывания функции $f(\boldsymbol{\Psi})$ на n-ом шаге, а коэффициент α^n нужен для задания перемещения вдоль направления $\boldsymbol{p}^n$.

2.2.2 Метод наискорейшего спуска

Данный метод предназначен для нахождения экстремума функции многих переменных. При выполнении условия (38) метод можно применить для поиска решения СЛАУ $\mathrm{A}\boldsymbol{\Psi} = \boldsymbol{b}$. Функция $f = f(\Psi_1, \ldots, \Psi_N) = f(\boldsymbol{\Psi})$ определена выражением (39). При сделанных предположениях минимизация функции $f(\boldsymbol{\Psi})$ уменьшает невязку $\boldsymbol{r} = \boldsymbol{b} - \mathrm{A}\boldsymbol{\Psi}$. Причем $f(\boldsymbol{\Psi})$ имеет единственный экстремум, являющийся глобальным минимумом.

Как известно [3], градиент $\nabla f(\boldsymbol{\Psi})$ определяет направление наибольшего возрастания функции $f(\boldsymbol{\Psi})$. Следовательно, в направлении $-\nabla f$ функция будет убывать сильнее всего. Согласно определению функции $f(\boldsymbol{\Psi})$: $-\nabla f = -(\mathrm{A}\boldsymbol{\Psi} - \boldsymbol{b}) = \boldsymbol{r}$. Таким образом, $\boldsymbol{p}^n$ – направление на n-ой итерации вычисляется по формуле:

$$\boldsymbol{p}^n = -\nabla f(\boldsymbol{\Psi}^n) = \boldsymbol{r}^n \tag{43}$$

Возникает вопрос. На сколько нужно продвинуться вдоль направления $\boldsymbol{p}^n$, чтобы значение $f(\boldsymbol{\Psi}^{n+1})$ в следующей точке было как можно меньшим. Величину продвижения вдоль $\boldsymbol{p}^n$ определяет числовой коэффициент α^n.

Для случая функции двух переменных на рис. 7 показы различные способы выбора значения α^n. В первом случае функция убывает недостаточно сильно. Во втором она наоборот возрастет.

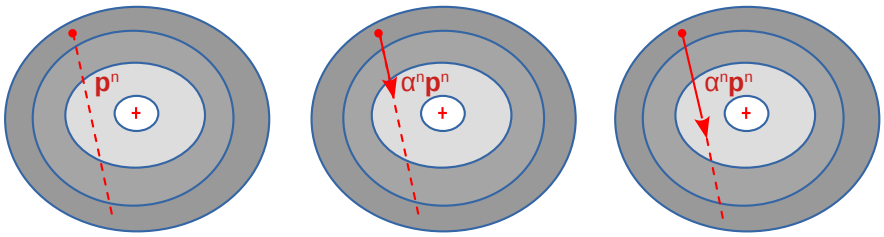


Рис. 7. Возможные продвижения вдоль направления $\boldsymbol{p}^n$ в данной точке.
На каждом контуре (они показаны синим цветом) функция $f(\Psi_1, \Psi_2)$ сохраняет некоторое постоянное значение. Точка минимума обозначена красным крестиком.

Функция многих переменных $f(\boldsymbol{\Psi}) = f(\Psi_1, \ldots, \Psi_N)$ на направлении $\boldsymbol{p}^n$ может быть представлена как сложная функция от одной переменной α:

$$\hat{f}(\alpha) = f(\boldsymbol{\Psi}^n + \alpha\boldsymbol{p}^n) = f(\Psi_1^n + \alpha p_1^n, \ldots, \Psi_N^n + \alpha p_N^n) \tag{44}$$

Оптимальное значение для α^n определяется из условия локального экстремума функции $\hat{f}(\alpha)$, то есть когда $\frac{d\hat{f}}{d\alpha}\big|_{\alpha^n} = 0$.

Перед получением выражения для α^n удобно записать функцию $\hat{f}(\alpha)$ в развернутом виде:

$$\hat{f}(\alpha) = f(\boldsymbol{\Psi}^n + \alpha\boldsymbol{p}^n) = \frac{1}{2}(\boldsymbol{\Psi}^n + \alpha\boldsymbol{p}^n)\cdot(\mathrm{A}(\boldsymbol{\Psi}^n + \alpha\boldsymbol{p}^n)) - \boldsymbol{b}\cdot(\boldsymbol{\Psi}^n + \alpha\boldsymbol{p}^n) =$$
$$= \frac{1}{2}\boldsymbol{\Psi}^n\cdot(\mathrm{A}\boldsymbol{\Psi}^n) + \frac{\alpha}{2}\boldsymbol{\Psi}^n\cdot(\mathrm{A}\boldsymbol{p}^n) + \frac{\alpha}{2}\boldsymbol{p}^n\cdot(\mathrm{A}\boldsymbol{\Psi}^n) + \frac{\alpha^2}{2}\boldsymbol{p}^n\cdot(\mathrm{A}\boldsymbol{p}^n) - \boldsymbol{b}\cdot\boldsymbol{\Psi}^n - \alpha\boldsymbol{b}\cdot\boldsymbol{p}^n$$

Матрица симметричная $\mathrm{A} = \mathrm{A}^T$, поэтому $\boldsymbol{p}^n\cdot(\mathrm{A}\boldsymbol{\Psi}^n) = \boldsymbol{\Psi}^n\cdot(\mathrm{A}\boldsymbol{p}^n)$

$$\hat{f}(\alpha) = \frac{1}{2}\boldsymbol{\Psi}^n\cdot(\mathrm{A}\boldsymbol{\Psi}^n) - \boldsymbol{b}\cdot\boldsymbol{\Psi}^n + \alpha\boldsymbol{p}^n\cdot(\mathrm{A}\boldsymbol{\Psi}^n) + \frac{\alpha^2}{2}\boldsymbol{p}^n\cdot(\mathrm{A}\boldsymbol{p}^n) - \alpha\boldsymbol{b}\cdot\boldsymbol{p}^n$$

На данном этапе: $\hat{f}(\alpha) = f(\boldsymbol{\Psi}) + \alpha\boldsymbol{p}^n\cdot(\mathrm{A}\boldsymbol{\Psi}^n) + \frac{\alpha^2}{2}\boldsymbol{p}^n\cdot(\mathrm{A}\boldsymbol{p}^n) - \alpha\boldsymbol{b}\cdot\boldsymbol{p}^n$

Дальнейшее преобразование: $\boldsymbol{p}^n\cdot(\mathrm{A}\boldsymbol{\Psi}^n) - \boldsymbol{b}\cdot\boldsymbol{p}^n = \boldsymbol{p}^n\cdot(\mathrm{A}\boldsymbol{\Psi}^n) - \boldsymbol{b}) = -\boldsymbol{p}^n\cdot\boldsymbol{r}^n$

В итоге: $\hat{f}(\alpha) = f(\boldsymbol{\Psi}) + \frac{\alpha^2}{2}\boldsymbol{p}^n\cdot(\mathrm{A}\boldsymbol{p}^n) - \alpha\boldsymbol{p}^n\cdot\boldsymbol{r}^n$

(45)

Нужное значение α^n определяется из условия $\frac{d\hat{f}}{d\alpha}\big|_{\alpha^n} = 0$:

$$\frac{d\hat{f}}{d\alpha} = \frac{d}{d\alpha}\left(f(\boldsymbol{\Psi}) + \frac{\alpha^2}{2}\boldsymbol{p}^n\cdot(\mathrm{A}\boldsymbol{p}^n) - \alpha\boldsymbol{p}^n\cdot\boldsymbol{r}^n\right) = \cancelto{0}{f(\boldsymbol{\Psi})} + \alpha\boldsymbol{p}^n\cdot(\mathrm{A}\boldsymbol{p}^n) - \boldsymbol{p}^n\cdot\boldsymbol{r}^n = 0$$

С учетом формулы (43): $\boldsymbol{p}^n = \boldsymbol{r}^n$

$$\frac{d\hat{f}}{d\alpha} = \alpha\boldsymbol{r}^n\cdot(\mathrm{A}\boldsymbol{r}^n) - \boldsymbol{r}^n\cdot\boldsymbol{r}^n = 0$$

(46)

С учетом последнего равенства в выражении (46) формула для расчета α^n:

$$\alpha^n = \frac{\boldsymbol{r}^n\cdot\boldsymbol{r}^n}{\boldsymbol{r}^n\cdot(\mathrm{A}\boldsymbol{r}^n)} \tag{47}$$

Алгоритм метода наискорейшего спуска

1. Вычислить невязку: $\boldsymbol{r}^n = \boldsymbol{b} - \mathrm{A}\boldsymbol{\Psi}^n$.
2. Определить направление спуска: $\boldsymbol{p}^n = \boldsymbol{r}^n$.
3. Вычислить коэффициент $\alpha^n = \frac{\boldsymbol{r}^n\cdot\boldsymbol{r}^n}{\boldsymbol{r}^n\cdot(\mathrm{A}\boldsymbol{r}^n)}$.
4. Найти новое приближение: $\boldsymbol{\Psi}^{n+1} = \boldsymbol{\Psi}^n + \alpha^n\boldsymbol{p}^n$.
5. Проверить критерий сходимости по невязке для $\boldsymbol{r}^{n+1}$.
   Если сходимость не достигнута, то вернутся к шагу 1.

2.2.3 Метод сопряженных градиентов

Можно показать [5], что в методе наискорейшего спуска каждое новое направление *ортогонально* предыдущему. Поэтому путь к минимуму функции (решению СЛАУ) состоит из «зиг-загов». Получается, что на некоторых итерациях продвижения вдоль одного и того же направления повторяются. В методе сопряженных градиентов процедура определения пути модифицирована так, благодаря чему продвижение вдоль одного направления происходит только один раз. Поэтому итоговое число итераций (шагов) меньше.

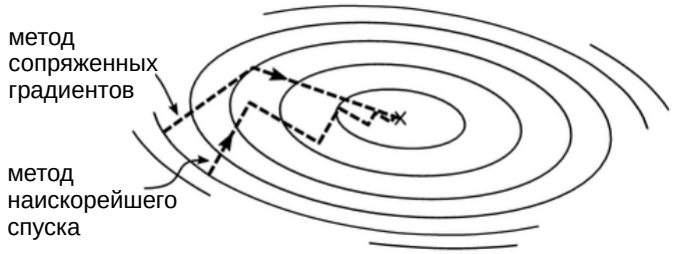


Рис. 8. Схема, показывающая различие между методом наискорейшего спуска и методом сопряженных градиентов.

Для любого приближенного решения $\boldsymbol{\Psi}^n$ можно определить $\boldsymbol{\varepsilon}^n$ – **ошибку**, на которую текущее приближение $\boldsymbol{\Psi}^n$ отличается от точного значения $\boldsymbol{\Psi}^{ex}$.

$$\boldsymbol{\varepsilon}^n = \boldsymbol{\Psi}^n - \boldsymbol{\Psi}^{ex} \tag{48}$$

Геометрически вектор ошибки $\boldsymbol{\varepsilon}^n(\boldsymbol{\Psi}^n)$ соединяет точку $\boldsymbol{\Psi}^n$ (N-мерного пространства) с точкой $\boldsymbol{\Psi}^{ex}$ – точным решением, какой бы произвольной точка $\boldsymbol{\Psi}^n$ не была.

В дальнейшем будет использовано следующее соотношение:

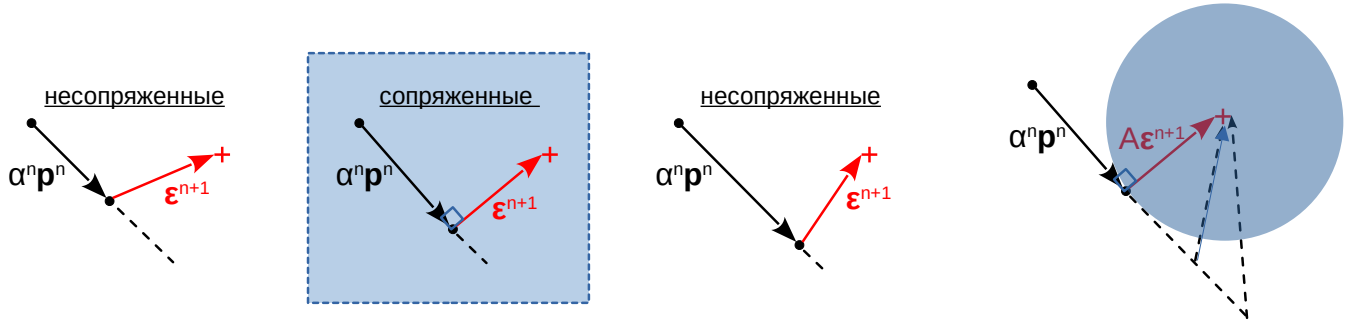
$$\begin{aligned} &\boldsymbol{\Psi}^{n+1} = \boldsymbol{\Psi}^n + \alpha^n \boldsymbol{p}^n \\ &\boldsymbol{\Psi}^{n+1} - \boldsymbol{\Psi}^{ex} = \boldsymbol{\Psi}^n - \boldsymbol{\Psi}^{ex} + \alpha^n \boldsymbol{p}^n \quad \Rightarrow \\ &\Rightarrow \quad \boldsymbol{\varepsilon}^{n+1} = \boldsymbol{\varepsilon}^n + \alpha^n \boldsymbol{p}^n \end{aligned} \tag{49}$$

Ключевая особенность метода
Метод сопряженных градиентов отличается специальным способом расчета коэффициента α^n. Это делается так, чтобы приближение $\boldsymbol{\Psi}^{n+1} = \boldsymbol{\Psi}^n + \alpha^n \boldsymbol{p}^n$ стало наиболее точным из всех возможных, которые могут быть при движении вдоль выбранного направления $\boldsymbol{p}^n$.

Вектор $\boldsymbol{\varepsilon}^{n+1}$ указывает на точное решение. Согласно формуле (49) он может быть вычислен с использованием вектора $\alpha^n \boldsymbol{p}^n$. Значение α^n подбирается так, чтобы векторы $\alpha^n \boldsymbol{p}^n$ и $\boldsymbol{\varepsilon}^{n+1}$ стали **сопряженными**.

Условие **сопряженности** (по матрице A) для векторов $\boldsymbol{a}$ и $\boldsymbol{b}$ [2]:

$$\boldsymbol{a} \cdot (\mathrm{A}\boldsymbol{b}) = 0 \tag{50}$$



(a) Результаты различных вариантов выбора значения α^n.
По центру показаны сопряженные (по матрице A) векторы $\boldsymbol{p}^n$ и $\boldsymbol{\varepsilon}^{n+1}$.

(b) Вектор, сопряженный с вектором $\boldsymbol{p}^n$ наиболее близок к точному решению.

Рис. 9. Иллюстрация к выбору значения α^n в методе сопряженных градиентов.

Определение коэффициента α^n

Оказывается, что условие сопряженности для векторов $\boldsymbol{p}^n$ и $\boldsymbol{\varepsilon}^{n+1}$ является оптимальным. В этом можно убедиться, если рассмотреть рис. 9.

Условие на выбор коэффициента α^n следует из формулы (49) и определения сопряженности (50):

$$\begin{aligned} &\boldsymbol{p}^n \cdot (\mathrm{A}\boldsymbol{\varepsilon}^{n+1}) = 0 \quad \text{(условие сопряженности)} \\ &\boldsymbol{p}^n \cdot (\mathrm{A}(\boldsymbol{\varepsilon}^n + \alpha^n \boldsymbol{p}^n)) = 0 \quad \Rightarrow \\ &\Rightarrow \quad \alpha^n = -\frac{\boldsymbol{p}^n \cdot (\mathrm{A}\boldsymbol{\varepsilon}^n)}{\boldsymbol{p}^n \cdot (\mathrm{A}\boldsymbol{p}^n)} \end{aligned} \tag{51}$$

Формула (51) не может быть использована в практических расчетах, так как вектор $\boldsymbol{\varepsilon}^n$ неизвестен. Эту проблему можно преодолеть, если немного преобразовать выражение (48):

$$\mathrm{A}\boldsymbol{\varepsilon}^n = \mathrm{A}(\boldsymbol{\Psi}^n - \boldsymbol{\Psi}^{ex}) = \mathrm{A}\boldsymbol{\Psi}^n - \boldsymbol{b} = -\boldsymbol{r}^n \tag{52}$$

Таким образом, коэффициент α^n вычисляется по формуле:

$$\alpha^n = \frac{\boldsymbol{p}^n \cdot \boldsymbol{r}^n}{\boldsymbol{p}^n \cdot (\mathrm{A}\boldsymbol{p}^n)} \tag{53}$$

Определение следующего направления $\boldsymbol{p}^{n+1}$

Вектор $\boldsymbol{\varepsilon}^{n+1}$ определяет направление к точному решению, и он сопряжен с $\boldsymbol{p}^n$. Поэтому естественно для следующего направления $\boldsymbol{p}^{n+1}$ потребовать *сонаправленность* с вектором $\boldsymbol{\varepsilon}^{n+1}$. Следовательно одно из условий на $\boldsymbol{p}^{n+1}$ состоит в том, что векторы $\boldsymbol{p}^n$ и $\boldsymbol{p}^{n+1}$ *должны быть сопряженными.*

Определение сопряженности (50) и формула (52) для $(n+1)$-ой итерации дают следующий результат:

$$\boldsymbol{p}^n \cdot \boldsymbol{r}^{n+1} = 0 \tag{54}$$

Формула (54) означает, что при выполнении всех условий, перечисленных выше, следующая невязка $\boldsymbol{r}^{n+1}$ ортогональна предыдущему направлению $\boldsymbol{p}^n$.

С учетом требования сопряженности $\boldsymbol{p}^n$ с $\boldsymbol{\varepsilon}^{n+1}$ и ортогональности между $\boldsymbol{p}^n$ и $\boldsymbol{r}^{n+1}$ получается следующая формула для направления $\boldsymbol{p}^{n+1}$:

$$\boldsymbol{p}^{n+1} = \boldsymbol{r}^{n+1} + \beta^{n+1}\boldsymbol{p}^n \tag{55}$$

Формулу (55) хорошо разъясняет рис. 10.

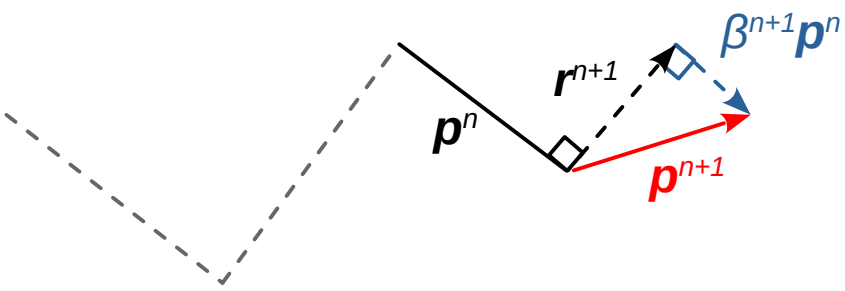


Рис. 10. Схема расчета следующего направления $\boldsymbol{p}^{n+1}$.

Для определения коэффициента β^{n+1} нужно воспользоваться условием сопряженности векторов $\boldsymbol{p}^n$ и $\boldsymbol{p}^{n+1}$:

$$\begin{aligned}
&\boldsymbol{p}^n \cdot (\mathrm{A}\boldsymbol{p}^{n+1}) = 0, \qquad \boldsymbol{p}^{n+1} = \boldsymbol{r}^{n+1} + \beta^{n+1}\boldsymbol{p}^n \\
&\boldsymbol{p}^n \cdot (\mathrm{A}(\boldsymbol{r}^{n+1} + \beta^{n+1}\boldsymbol{p}^n)) = \boldsymbol{p}^n \cdot (\mathrm{A}\boldsymbol{r}^{n+1}) + \beta^{n+1}\boldsymbol{p}^n(\mathrm{A}\boldsymbol{p}^n) = 0 \quad \Rightarrow \\
&\Rightarrow \quad \beta^{n+1} = -\frac{\boldsymbol{p}^n \cdot (\mathrm{A}\boldsymbol{r}^{n+1})}{\boldsymbol{p}^n(\mathrm{A}\boldsymbol{p}^n)}
\end{aligned} \tag{56}$$

Выбор начального направления $\boldsymbol{p}^1$ и рекомендованные формулы

В качестве исходного направления рекомендуется выбирать направление наискорейшего спуска, которое определяется через отрицательный градиент.

$$\boldsymbol{p}^1 = -\nabla f(\boldsymbol{\Psi}) = \boldsymbol{r}^1 = \boldsymbol{b} - \mathrm{A}\boldsymbol{\Psi}^1 \tag{57}$$

Можно показать [5], что выражения для расчета коэффициентов α^n и β^{n+1} могут быть упрощены:

$$\begin{aligned}
\alpha^n &= \frac{\boldsymbol{p}^n \cdot \boldsymbol{r}^n}{\boldsymbol{p}^n \cdot (\mathrm{A}\boldsymbol{p}^n)} \quad \rightarrow \quad \alpha^n = \frac{\boldsymbol{r}^n \cdot \boldsymbol{r}^n}{\boldsymbol{p}^n \cdot (\mathrm{A}\boldsymbol{p}^n)} \\
\beta^{n+1} &= -\frac{\boldsymbol{p}^n \cdot (\mathrm{A}\boldsymbol{r}^{n+1})}{\boldsymbol{p}^n(\mathrm{A}\boldsymbol{p}^n)} \quad \rightarrow \quad \beta^{n+1} = \frac{\boldsymbol{r}^{n+1} \cdot \boldsymbol{r}^{n+1}}{\boldsymbol{r}^n \cdot \boldsymbol{r}^n}
\end{aligned} \tag{58}$$

Формулы в (58) считаются стандартными для метода сопряженных градиентов.

Алгоритм метода сопряженных градиентов

1. Найти начальное направление: $\boldsymbol{p}^1 = \boldsymbol{r}^1 = \boldsymbol{b} - \mathrm{A}\boldsymbol{\Psi}^1$.
2. Вычислить перемещение $\alpha^n = \frac{\boldsymbol{r}^n \cdot \boldsymbol{r}^n}{\boldsymbol{p}^n \cdot (\mathrm{A}\boldsymbol{p}^n)}$.
3. Обновить приближенное решение: $\boldsymbol{\Psi}^{n+1} = \boldsymbol{\Psi}^n + \alpha^n \boldsymbol{p}^n$.
4. Вычислить невязку $\boldsymbol{r}^{n+1} = \boldsymbol{b} - \mathrm{A}\boldsymbol{\Psi}^{n+1}$.
5. Определить следующее направление:

$$\beta^{n+1} = \frac{\boldsymbol{r}^{n+1} \cdot \boldsymbol{r}^{n+1}}{\boldsymbol{r}^n \cdot \boldsymbol{r}^n}, \qquad \boldsymbol{p}^{n+1} = \boldsymbol{r}^n + 1 + \beta^{n+1}\boldsymbol{p}^n$$

6. Вернуться к шагу 2, если норма невязки $||\boldsymbol{r}^{n+1}||$ недостаточно мала.

Реализация метода сопряженных градиентов в OpenFOAM
Для ещё большего ускорения сходимости в OpenFOAM метод сопряженных градиентов комбинируется с процедурой **предобуславливания**:

$$\mathrm{M}^{-1}\mathrm{A}\boldsymbol{\Psi} = \mathrm{M}^{-1}\boldsymbol{b} \tag{59}$$

Смысл этой операции в том, чтобы получить эквивалентную (с тем же решением) систему, матрица $\mathrm{M}^{-1}\mathrm{A}$ которой будет более близка к единичной.

Описанный здесь метод сопряженных градиентов можно применять, когда матрица системы уравнений симметричная. В OpenFOAM для систем с *несимметричными* матрицами реализовано несколько процедур предобуславливания, а также **метод би-сопряженных градиентов** [2].

2.3 Многосеточный метод

2.3.1 Основная идея

Пусть с помощью некоторого итерационного алгоритма найдено $\boldsymbol{\Psi}^c$ – приближенное решение СЛАУ $\mathrm{A}\boldsymbol{\Psi} = \boldsymbol{b}$. На данном этапе $\boldsymbol{\Psi}^c$ отличается от точного решения $\boldsymbol{\Psi}^{ex}$ на величину $\Delta\boldsymbol{\Psi} = \boldsymbol{\Psi}^{ex} - \boldsymbol{\Psi}^c$. Можно составить уравнение для поправки $\Delta\boldsymbol{\Psi}$:

$$\begin{aligned} &\mathrm{A}\Delta\boldsymbol{\Psi} = \mathrm{A}(\boldsymbol{\Psi}^{ex} - \boldsymbol{\Psi}^c) = \mathrm{A}\boldsymbol{\Psi}^{ex} - \mathrm{A}\boldsymbol{\Psi}^c = \boldsymbol{b} - \mathrm{A}\boldsymbol{\Psi}^c = \boldsymbol{r} \\ &\mathrm{A}\Delta\boldsymbol{\Psi} = \boldsymbol{r} \end{aligned} \tag{60}$$

Тогда $\boldsymbol{\Psi}^{ex}$ – точное решение СЛАУ $\mathrm{A}\boldsymbol{\Psi} = \boldsymbol{b}$ можно получить по формуле:

$$\begin{aligned} &\boldsymbol{\Psi}^{ex} = \boldsymbol{\Psi}^c + \Delta\boldsymbol{\Psi}^{ex}, \\ &\text{где } \Delta\boldsymbol{\Psi}^{ex} \text{ – точное решение системы (60)} \end{aligned} \tag{61}$$

Но решение системы $\mathrm{A}\Delta\boldsymbol{\Psi} = \boldsymbol{r}$ по сложности эквивалентно нахождению точного решения исходной задачи $\mathrm{A}\boldsymbol{\Psi} = \boldsymbol{b}$. Если расчетная сетка содержит много ячеек, то итерационное решение любой из СЛАУ: $\mathrm{A}\boldsymbol{\Psi} = \boldsymbol{b}$, $\mathrm{A}\Delta\boldsymbol{\Psi} = \boldsymbol{r}$ потребует много итераций (будет долгим).

Процесс решения системы $\mathrm{A}\Delta\boldsymbol{\Psi} = \boldsymbol{r}$ можно существенно ускорить, выбрав хорошее начальное приближение $\Delta\boldsymbol{\Psi}_0$. Для этого можно перейти к более грубой сетке, на которой решение соответствующей СЛАУ $\mathrm{A}'\Delta\boldsymbol{\Psi}' = \boldsymbol{r}'$ потребует меньше итераций. Найденное решение $\Delta\boldsymbol{\Psi}'^{,\,ex}$ можно интерполировать на исходную сетку и использовать как начальное приближение для системы (60). В то же время, для СЛАУ $\mathrm{A}'\Delta\boldsymbol{\Psi}' = \boldsymbol{r}'$ можно провести аналогичные рассуждения и перейти к ещё более грубой сетке с системой $\mathrm{A}''\Delta\boldsymbol{\Psi}'' = \boldsymbol{r}''$ и так далее. В этом и есть сущность данного метода.

Для *ускорения* решения исходной СЛАУ (60) формируется набор «более простых» систем линейных уравнений. Каждая следующая система содержит меньше уравнений. Начальное приближение для k-ой системы получается, как результат интерполяции численного решения $(k+1)$-ой системы. Следовательно, решение k-ой системы требует меньше итераций. Итоговое решение системы $\mathrm{A}\boldsymbol{\Psi} = \boldsymbol{b}$ определяется по формуле (61).

При переходах между разными системами уравнений (сетками) нужно выполнять следующие действия:

- **Агломерация**: формирование матриц для систем уравнений на более грубых сетках.
- **Сужение (рестрикция)**: вычисление правых частей для новых систем уравнений.
- **Продолжение (инъекция)**: использование решения $(k+1)$-ой в качестве начального приближения для k-ой системы.

Нужно отметить, что выбранный итерационный алгоритм решения СЛАУ должен обладать свойством **сглаживания невязки** [6]. Смысл этого свойства в том, что начиная с определенной итерации невязка становится однородной. То есть все компоненты становится примерно одинаковыми. Пример можно видеть на рис. 11.

Невязка в правой части $(k+1)$-ой системы уравнений вычисляется на основании невязки в k-ой системе. Если невязка в k-ой системе содержала компоненты, изменяющиеся скачкообразно (не была однородной), то $(k+1)$-я система не будет корректно аппроксимировать k-ю систему.

Сглаживающее свойство есть у метода Гаусса-Зейделя и метода сопряженных градиентов [2].

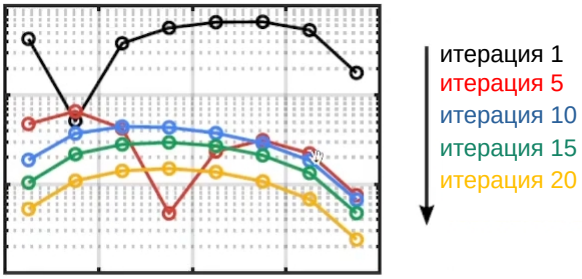


Рис. 11. Пример графика значений компонент невязки в соседних точках расчетной сетки на разных итерациях. Синий, зеленый и желтый графики показывают, что невязка стала однородной

2.3.2 Реализация в OpenFOAM – алгоритм GAMG

Для упрощения записей при описании алгоритма GAMG (Geometric Algebraic Multi-Grid) будет рассмотрен всего один переход:

$$\mathrm{A}\boldsymbol{\Psi} = \boldsymbol{b} \quad \rightarrow \quad \mathrm{A}'\boldsymbol{\Psi}' = \boldsymbol{b}', \qquad (\text{сетки } G,\, G') \tag{62}$$

Снова считается, что выбран один из подходящих итерационных алгоритмов решения СЛАУ. С помощью этого алгоритма найденное приближенное решение $\boldsymbol{\Psi}^c$ и вычислена невязка $\boldsymbol{r} = \boldsymbol{b} - \mathrm{A}\boldsymbol{\Psi}^c$. На грубой сетке G' одна ячейка может состоять из нескольких соседних ячеек в G.

Этап сужения ($\boldsymbol{r} \rightarrow \boldsymbol{r}'$)

В методе контрольных объемов системы линейных уравнений являются аналогами дифференциальных уравнений. В свою очередь дифференциальные уравнения отражают законы сохранения. Точное решение системы линейных уравнений удовлетворяет дискретному закону сохранения. Ненулевое значение компоненты невязки в i-ой ячейки можно интерпретировать, как избыток (энергии, силы и так далее в зависимости от уравнения) в данной ячейке. При переходе к более грубой сетке G' «избытки» в соседних ячейках сетки G, которые попали в одну ячейку в G' можно просуммировать.
Пример формирования вектора $\boldsymbol{r}'$ показан на рис. 12

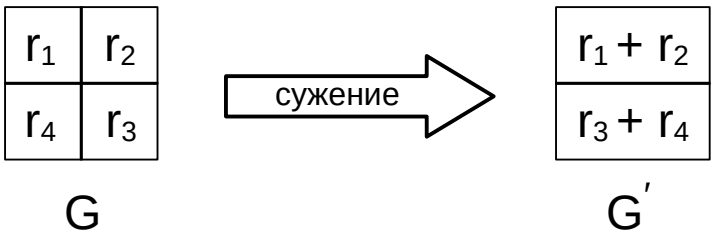


Рис. 12. Схема построения невязки $\boldsymbol{r}'$ по значениям $\boldsymbol{r}$

Формирование $\boldsymbol{r}'$ можно представить в матричном виде:

$$\boldsymbol{r}' = \mathrm{R}\boldsymbol{r} \tag{63}$$

В формуле (63) в общем случае матрица R *не квадратная*. Строк в R столько, сколько ячеек в G'. Столбцов в R столько, сколько ячеек в G. Элементы $R_{i,j}$ могут принимать одно из двух значений 0 или 1.

Для примера, показанного на рис. 12 вектор $\boldsymbol{r}'$ вычисляется следующим образом:

$$\begin{bmatrix} r'_1 \\ r'_2 \end{bmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix} \tag{64}$$

Этап продолжения («инъекция» решения с сетки G' на сетку G)

Предполагается, что значение величины $\Delta\Psi$ однородно в пределах одной ячейки на грубой сетке G'. Это же значение присваивается всем ячейкам мелкой сетки G, которые находятся внутри данной ячейки грубой сетки, так называемый кусочно-постоянный профиль.

Данную операцию можно выразить через матричное умножение:

$$\begin{aligned} &\Delta\boldsymbol{\Psi}^{G} = \mathrm{P}\Delta\boldsymbol{\Psi}^{G'}, \\ &\Delta\boldsymbol{\Psi}^{G'} \text{ – результат решения СЛАУ } \mathrm{A}'\Delta\boldsymbol{\Psi}' = \boldsymbol{r}' \text{ на сетке } G' \\ &\Delta\boldsymbol{\Psi}^{G} \text{ – результат интерполяции } \Delta\boldsymbol{\Psi}^{G'} \text{ на сетку } G \end{aligned} \tag{65}$$

В формуле (65) матрица P *не обязательно квадратная*. Число строк в P столько, сколько ячеек в G. Столбцов столько, сколько ячеек в G'. Элементы $P_{i,j}$ могут принимать значения либо 0, либо 1.

Кусочно-постоянная интерполяция проиллюстрирована на рис. 13.

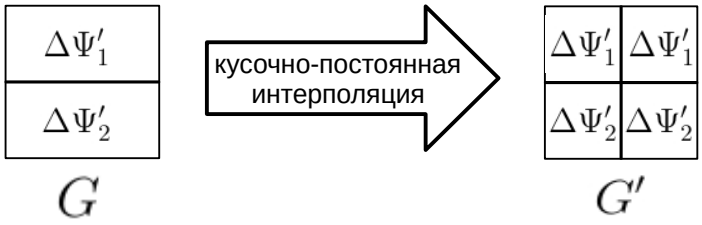


Рис. 13. Пример интерполяции (инъекции) значений с сетки G' на сетку G

Для примера, изображенного на рис. 13 вектор $\Delta\boldsymbol{\Psi}^{G}$ вычисляется следующим образом:

$$\begin{bmatrix} \Delta\Psi_1 \\ \Delta\Psi_2 \\ \Delta\Psi_3 \\ \Delta\Psi_4 \end{bmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \begin{bmatrix} \Delta\Psi'_1 \\ \Delta\Psi'_2 \end{bmatrix} \tag{66}$$

Этап агломерации ($\mathrm{A} \rightarrow \mathrm{A}'$)
В системе уравнений $\mathrm{A}\Delta\Psi = \boldsymbol{r}$ на поправку $\Delta\Psi$ используется матрица, которая получается в результате дискретизации дифференциального уравнения на Ψ. Поэтому при переходе от сетки G к более грубой сетке G' новая матрица A' *может* быть найдена после дискретизации на сетке G'. Но такая процедура достаточно трудоемкая.

В алгоритме GAMG вместо истинной матрицы A' вычисляется её *аппроксимация* алгебраическими путем (отсюда и название алгоритма).

При итерационном решении «исходной» СЛАУ $\mathrm{A}\boldsymbol{\Psi} = \boldsymbol{b}$, на очередной итерации:

$$\mathrm{A}\boldsymbol{\Psi}^c = \boldsymbol{b} \tag{67}$$

Для текущего приближенного решения $\boldsymbol{\Psi}^c$ можно найти невязку $\boldsymbol{r}^c$:

$$\boldsymbol{r}^c = \boldsymbol{b} - \mathrm{A}\boldsymbol{\Psi}^c \tag{68}$$

Цель в том, чтобы невязка $\boldsymbol{r}^c$ стремилась к нулю, следовательно на более грубой сетке будет тот же результат, поэтому можно записать:

$$\begin{aligned} \mathrm{R}\boldsymbol{r}^c &\approx \mathbf{0} \\ \mathrm{R}(\boldsymbol{b} - \mathrm{A}\boldsymbol{\Psi}^c) &\approx \mathbf{0} \end{aligned} \tag{69}$$

Текущее приближение $\boldsymbol{\Psi}^c$ может быть определено через предыдущее $\boldsymbol{\Psi}^p$ с использование коррекции $\Delta\boldsymbol{\Psi}$:

$$\begin{aligned} \boldsymbol{\Psi}^c &= \boldsymbol{\Psi}^p + \Delta\boldsymbol{\Psi} \\ \boldsymbol{\Psi}^c &= \boldsymbol{\Psi}^p + \mathrm{P}\boldsymbol{\Psi}' \end{aligned} \tag{70}$$

Подстановка выражения (70) в (69) приводит к следующей цепочке равенств:

$$\begin{aligned} \mathrm{R}\boldsymbol{r}^c = \mathrm{R}(\boldsymbol{b} - \mathrm{A}\boldsymbol{\Psi}^c) = \mathrm{R}(\boldsymbol{b} - \mathrm{A}(\boldsymbol{\Psi}^p + \mathrm{P}\boldsymbol{\Psi}')) &\approx \mathbf{0} \\ \mathrm{RAP}\Delta\boldsymbol{\Psi}' &\approx \mathrm{R}(\boldsymbol{b} - \mathrm{A}\boldsymbol{\Psi}) \\ \underbrace{\mathrm{RAP}}_{\mathrm{A}'}\Delta\boldsymbol{\Psi}' &\approx \boldsymbol{r}' \end{aligned} \tag{71}$$

Таким образом, при переходе от сетки G к сетке G' соответствующая матрица A' вычисляется по формуле:

$$\mathrm{A}' = \mathrm{RAP} \tag{72}$$

В данном случае матрицы R и P связаны свойством:

$$\mathrm{P} = \mathrm{R}^T \tag{73}$$

Алгоритм метода GAMG:

1. Выбрать подходящий итерационный метод решения СЛАУ. Выполнить несколько итераций, чтобы получить приближенное решение $\boldsymbol{\Psi}^c$.
   Вычислить невязку $\boldsymbol{r}^c = \boldsymbol{b} - \mathrm{A}\boldsymbol{\Psi}^c = \mathrm{A}\Delta\boldsymbol{\Psi}^c$.

2. Выполнить N этапов *агломерации* и *сужения*:

$$\mathrm{A}', \ldots, \mathrm{A}'^{(N)}, \qquad \boldsymbol{r}', \ldots, \boldsymbol{r}'^{(N)}$$

   В результате сформированы СЛАУ вида:

$$\mathrm{A}'^{(k)}\Delta\boldsymbol{\Psi}'^{(k)} = \boldsymbol{r}'^{(k)}, \qquad 1 \le k \le N$$

3. Решить последнюю систему уравнений $\mathrm{A}'^{(N)}\Delta\boldsymbol{\Psi}'^{(N)} = \boldsymbol{r}'^{(N)}$ *методом исключения Гаусса* [4] (определяется точное решение).
   Для остальных систем применить выбранный *итерационный* метод.
   Начальные приближения $\Delta\boldsymbol{\Psi}_0^k$ определять интерполяцией решений $\Delta\boldsymbol{\Psi}_{ex}^{k+1}$.
   Так продолжать до $k = 0$ *включительно*, то есть до СЛАУ на исходной сетке $\mathrm{A}\Delta\boldsymbol{\Psi} = \boldsymbol{r}^c$ (в конце она решается итерационно до получения $\Delta\boldsymbol{\Psi}^{final}$).

4. Найти окончательное решение для СЛАУ $\mathrm{A}\boldsymbol{\Psi} = \boldsymbol{b}$ по формуле:

$$\boldsymbol{\Psi}^{final} = \boldsymbol{\Psi}^c + \Delta\boldsymbol{\Psi}^{final}$$

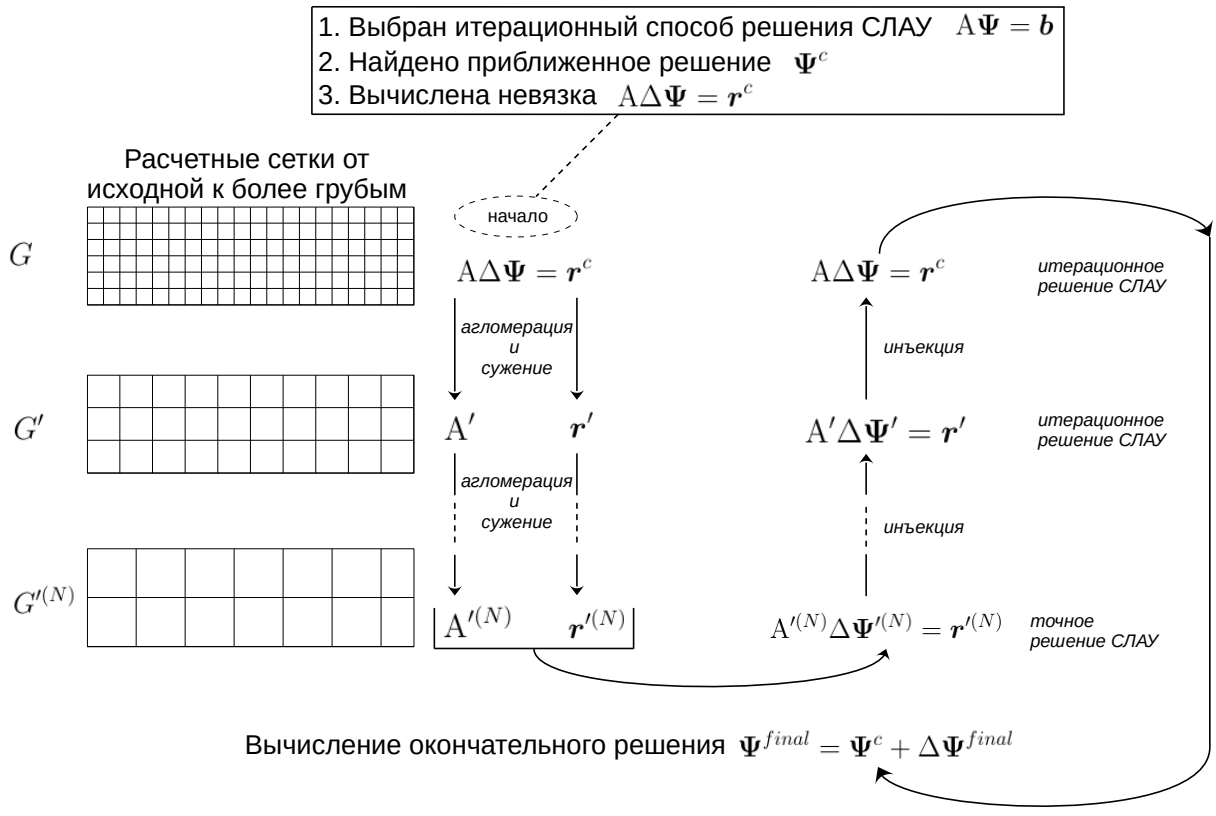


Рис. 14. Схема алгоритма GAMG

3 Численное решение уравнений гидродинамики

Система уравнений гидродинамики вязкой несжимаемой жидкости состоит из уравнения движения (74) и уравнения неразрывности (75). В данном случае используется векторная форма записи этих уравнений. В уравнении движения (74) не учитывается действие объемных сил (гравитация, сила Лоренца и прочие).

Уравнение движения (переноса импульса):

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) = -\frac{\nabla p}{\rho} + \nabla \cdot (\nu \nabla \boldsymbol{u}) \tag{74}$$

Уравнение неразрывности:

$$\nabla \cdot \boldsymbol{u} = 0 \tag{75}$$

Уравнение движения (74) содержит *нелинейное* слагаемое $\nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u})$. Большинство алгоритмов итерационные, поэтому выражение $\nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u})$ *линеаризуется* (подробнее эта проблема рассмотрена в [1], [7]).

Процедура линеаризации представляется формулой:

$$[\nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u})]\Big|_P = \frac{1}{V_P} \sum_f (\boldsymbol{S}_f \cdot \mathring{\boldsymbol{u}}_f) \boldsymbol{u}_f = a_P \boldsymbol{u}_P + \sum_N a_N \boldsymbol{u}_N \tag{76}$$

Выражение (76) записано для некоторой ячейки с центром в точке $\boldsymbol{P}$, объем которой равен V_P. Индекс f пробегает по всем граням этой ячейки, а индекс N применяется для ячеек смежных с данной. Вектор $\boldsymbol{S}_f$ направлен по внешней нормали к грани с индексом f, а его модуль равен площади этой грани. Согласно формуле (76) линеаризация слагаемого $\nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u})$ достигается за счет использования значений скорости $\mathring{\boldsymbol{u}}$, которые берутся либо с предыдущего временного слоя, либо с прошлой итерации. В любом случае потоки $\boldsymbol{S}_f \cdot \mathring{\boldsymbol{u}}_f$ *известны*. Величины $\boldsymbol{u}_f$ вычисляются обычным образом через значения в центрах ячеек согласно схеме дискретизации, что выражено в последнем равенстве формулы (76).

Нужно заметить, что коэффициенты a_P, a_N зависят от скорости, так как они вычисляются через в том числе с помощью известного на данный момент поля скорости $\mathring{\boldsymbol{u}}$. В процессе решения коффициенты a_P, a_N могут изменяться (на каждой итерации алгоритма). Таким способом преодолевается проблема нелинейности уравнений гидродинамики.

Система уравнений (74) , (75) имеет ещё одну особенность. В системе отсутсвует уравнение на давление p. Причем даже, если давление задано, скорость должна удовлетворять не только уравнению движения(74), но также и уравнению неразрывности (75).

При численном решении часто используется **раздельный подход**. Это значит, что связь между скоростью и давлением устанавливается последовательно, с помощью того или иного итерационного алгоритма. В частности, наиболее распространены алгоритмы SIMPLE [1] и его модификации, а также алгоритм PISO [8]. Оба алгорита реализованы в платформе OpenFOAM [9].

3.1 Основные формулы

Частичная дискретизация уравнения движения (74) записывается в виде [8], [10]:

$$\mathrm{M}[\boldsymbol{u}] = -\nabla p, \tag{77}$$

В выражении (77) символ M это **дискретный оператор**. Данный оператор действует на исходное уравнение движения (74), в результате для *каждой* компоненты скорости (u_1, u_2, u_3) формируется матрица, содержащая коэффициенты соответствующей системы линейных уравнений Результат частичной дискретизации, представленной в формуле (77), можно показать более наглядно, например, для компоненты скорости u_1:

$$\underbrace{\begin{pmatrix} m_{11} & m_{12} & \dots & m_{1N} \\ m_{21} & m_{22} & \dots & m_{2N} \\ \dots & \dots & \dots & \dots \\ m_{N1} & m_{N2} & \dots & m_{NN} \end{pmatrix}}_{\mathrm{M}_1} \begin{pmatrix} u_1^1 \\ u_1^2 \\ \dots \\ u_1^N \end{pmatrix} = \begin{pmatrix} -\left(\frac{\partial p}{\partial x^1}\right)^1 - \frac{\hat{u}_1^1}{\Delta t} \\ -\left(\frac{\partial p}{\partial x^1}\right)^2 - \frac{\hat{u}_1^2}{\Delta t} \\ \dots \\ -\left(\frac{\partial p}{\partial x^1}\right)^N - \frac{\hat{u}_1^N}{\Delta t} \end{pmatrix} \tag{78}$$

здесь верхний индекс означает номер ячейки

значения $\hat{\boldsymbol{u}}$ известны из предыдущего временного слоя

Формально на этапе частичной дискретизации (77) *градиент давления не подвергается дискретизации.*

В дальнейшем оператор M преобразуется в другую форму [8], [10]:

$$\mathrm{M}[\boldsymbol{u}] = \mathrm{A}[\boldsymbol{u}] - \boldsymbol{H}(\boldsymbol{u}) \tag{79}$$

В выражении (79) символом A обозначен **дискретный оператор**,
$\boldsymbol{H}(\boldsymbol{u})$ это **дискретная векторная функция**.

Оператор A выделяет диагональные компоненты из матриц M_i $(i = 1, 2, 3)$, в результате чего появляются соответствующие диагональные матрицы $\mathrm{A}_1, \mathrm{A}_2, \mathrm{A}_3$.

Функция $\boldsymbol{H}(\boldsymbol{u})$ состоит из двух частей. Первая часть содержит результат произведения недиагональных коэффициентов (они также зависят от скорости) матриц на соответствующие значения копонент скорости, причем используются известные на данный момент значения скорости. Во вторую часть входит «источниковая» часть (предыдущий временной слой при дискретизации производной по времени, объемные силы и так далее).

При отсутствии объемных сил вектор $\boldsymbol{H}(\boldsymbol{u})$ вычисляется по формуле [7]:

$$\boldsymbol{H}(\boldsymbol{u}) = -\sum_N a_N \mathring{\boldsymbol{u}}_N + \frac{\hat{\boldsymbol{u}}}{\Delta t}$$

значения $\mathring{\boldsymbol{u}}$ известны (с предыдущего временного слоя или прошлой итерации)

значения $\hat{\boldsymbol{u}}$ берутся с предыдущего временного слоя

индекс N означает, пробегает только по соседним ячейкам

(80)

Преобразование (79) можно продемонстрировать более наглядно следующим выражением, которое записано для компоненты скорости u_1:

$$\underbrace{\begin{pmatrix} m_{11} & 0 & \dots & 0 \\ 0 & m_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & m_{NN} \end{pmatrix}}_{\mathrm{A}_1} \begin{pmatrix} u_1^1 \\ u_1^2 \\ \dots \\ u_1^N \end{pmatrix} - \underbrace{\begin{pmatrix} -\sum_{k=1,\,k\neq 1}^{N} m_{1k}\mathring{u}_1^k - \frac{\hat{u}_1^1}{\Delta t} \\ -\sum_{k=1,\,k\neq 2}^{N} m_{2k}\mathring{u}_1^k - \frac{\hat{u}_1^2}{\Delta t} \\ \dots \\ -\sum_{k=1,\,k\neq N}^{N} m_{Nk}\mathring{u}_1^k - \frac{\hat{u}_1^N}{\Delta t} \end{pmatrix}}_{\boldsymbol{H_1(\mathring{u})_1}} = \begin{pmatrix} -\left(\frac{\partial p}{\partial x^1}\right)^1 \\ -\left(\frac{\partial p}{\partial x^1}\right)^2 \\ \dots \\ -\left(\frac{\partial p}{\partial x^1}\right)^N \end{pmatrix}$$

значения $\mathring{\boldsymbol{u}}_1^k$ известны с предыдущего временного слоя или из прошлой итерации

значения $\dfrac{\hat{u}_1^k}{\Delta t}$ взяты с предыдущего временного слоя

величины u_1^k это искомый набор значений первой компоненты

(81)

С помощью преобразования (79) частично-дискретную форму уравнения движения (74) можно записать таким способом:

$$\begin{aligned} &\text{компонентная форма: } a_P \boldsymbol{u}_P = \left(\boldsymbol{H(u)} - \nabla p\right)\Big|_P \\ &\text{операторная форма: } \mathrm{A}[\boldsymbol{u}] = \boldsymbol{H(u)} - \nabla p \\ &\text{индекс } P \text{ пробегает по всем ячейкам расчетной сетки} \end{aligned} \tag{82}$$

Из-за нелинейности уравнения движения в формуле (82) коэффициенты a_P частично (из-за конвективного слагаемого) зависят от скорости. У функции $\boldsymbol{H(u)}$ зависимость ещё сложнее. Во-первых, прямое вычисление $\boldsymbol{H(u)}$ для разных значей скорости даёт разные результаты, а также недиагональные коэффициенты (см.(78) и (81)), используемые в $\boldsymbol{H(u)}$, тоже зависят от скрости.

Все «приёмы», проделанные выше, нацелены на то, чтобы в итоге получить недостающее уравнение для давления. Сейчас получается, что оператор A – **неявный** (результат его действия это наборы коэффициентов в линейных уравнениях), а функция $\boldsymbol{H(u)}$ – **явная**, то есть она вычисляется по известным значениям.

Первый шаг при получении дополнительного уравнения на давление это состоит в том, чтобы выразить скорость из формулы (82) (некоторые индексы исключены для упрощения записи):

$$\boldsymbol{u}_P = \frac{\boldsymbol{H}(\boldsymbol{u})}{a_P} - \frac{1}{a_P}\nabla p \tag{83}$$

Дискретная форма уравнения неразрывности (75):

$$[\nabla \cdot \boldsymbol{u}]\Big|_P = \frac{1}{V_P}\sum_f (\boldsymbol{S}_f \cdot \boldsymbol{u}_f) = 0$$

V_P – объем ячейки индексом P (пробегает по всем ячейкам)

f – индекс, пробегающий по всем граням данной ячейки

$\boldsymbol{S}_f$ – вектор, направленный по нормали к грани, $|\boldsymbol{S}_f|$ равен площади грани

(84)

Величины $\boldsymbol{u}_f$ определяются на гранях ячеек через интерполяцию по значениям в центрах граней ячеек. С учетом формулы (83) для $\boldsymbol{u}_f$ получается следующее выражение:

$$\boldsymbol{u}_f = \left(\frac{\boldsymbol{H}(\boldsymbol{u})}{a_P}\right)_f - \left(\frac{1}{a_P}\nabla p\right)_f \tag{85}$$

Наконец, подстановка скорости, выраженной в виде (83), в уравнение неразрывности (75) даёт недостающее **уравнение на давление**:

$$\nabla \cdot \left(\frac{1}{a_P}\nabla p\right) = \nabla \cdot \left(\frac{\boldsymbol{H}(\boldsymbol{u})}{a_P}\right) \tag{86}$$

Численная аппроксимация слагаемого ∇p выполняется таким образом:

$$\nabla p = \sum_f p_f \boldsymbol{S}_f \tag{87}$$

Дискретная форма системы уравнений гидродинамики вязкой несжимаемой жидкости записывается так [7]:

$$a_P \boldsymbol{u}_P = \boldsymbol{H}(\boldsymbol{u}) - \sum_f p_f \boldsymbol{S}_f \tag{88}$$

$$\sum_f \boldsymbol{S}_f \cdot \left[\left(\frac{1}{a_P}\right)_f (\nabla p)_f\right] = \sum_f \boldsymbol{S}_f \cdot \left(\frac{\boldsymbol{H}(\boldsymbol{u})}{a_P}\right)_f \tag{89}$$

Формула (88) соответствует уравнению движения (74). А формула (89) выведена с учетом уравнения неразрывности 75.

Во многих задачах кроме уравнений гидродинамики решаются также и дополнительные уравнения. Часто дополнительные уравнения, например на скалярную величину Ψ, содержат слагаемые вида $\boldsymbol{u} \cdot \nabla\Psi$. Как только удается получить

значения скорости и давления для данного временного слоя, для дальнейшего решения дополнительных уравнений нужно выполнить **коррекцию массовых потоков** [7], [10]. Формула корреции массового потока ϕ_f для одной из граней:

$$\phi_f = \boldsymbol{S}_f \cdot \left[\left(\frac{1}{a_P} \right)_f (\nabla p)_f - \left(\frac{\boldsymbol{H}(\boldsymbol{u})}{a_P} \right)_f \right] \tag{90}$$

Уточнение массовых потоков изменяет коэффициенты a_P, a_N, которые используются при вычислении функции $\boldsymbol{H}(\boldsymbol{u})$.

3.2 Алгоритм SIMPLE

Алгоритм SIMPLE (Semi-Implicit Method for Pressure Linked Equations) [1] используется для решения **стационарных задач**. Чтобы предотвратить расходимость для скорости и давления применяется **нижняя релаксация** [1], [7].

Для скорости релаксации подвергеются уравнение движения (коэффициенты матриц для каждой компоненты скорости) [1], [7]:

$$\begin{gathered} \frac{1}{\alpha_u} a_p \boldsymbol{u}_P = -\sum_N a_N \boldsymbol{u}_N - \nabla p + \frac{1-\alpha_u}{\alpha_u} a_p \mathring{\boldsymbol{u}}_p, \qquad 0 < \alpha_u \leq 1 \\ \text{значения } \mathring{\boldsymbol{u}} \text{ известны с предыдущей итерации} \end{gathered} \tag{91}$$

В случае с давлением релаксация используется для самих значений [1], [7]:

$$\begin{gathered} p = \mathring{p} + \alpha_p (p - \mathring{p}), \qquad 0 < \alpha_p \leq 1 \\ p \text{ – новое значение} \\ \mathring{p} \text{ – значение с предыдущей итерации} \end{gathered} \tag{92}$$

Последовательность действий в алгоритме SIMPLE [1], [7], [11]:

1. Определить начальные значения скорости $\mathring{\boldsymbol{u}}^n$ и давления p^n.
   Верхний индекс n обозначает номер итерации.
2. Сформировать уравнение движения (88), используя $\mathring{\boldsymbol{u}}^n$, p^n и применяя нижнюю релаксацию (91).
   Решить полученное уравнение движения (88).
   В результате рассчитана скорость $\boldsymbol{u}^n$.
3. Сформировать уравнение (89) на давление.
   Решить полученное уравнение (89).
   В результате рассчитано давление p^{n+1}.
4. Выполнить коррекцию потоков согласно формуле (90).
5. Сделать поправку значений скорости по формуле (83).
   В этих вычислениях используется давление, которое подверглось нижней релаксации согласно формуле (92).
   В результате получено поле скорости $\boldsymbol{u}^{n+1}$.

6. Если сходимость не достигуна, то $\mathring{\boldsymbol{u}}^n = \boldsymbol{u}^{n+1}$, $p^n = p^{n+1}$ и возврат к шагу 1. При достижении сходимости завершить выполнение алгоритма.

Дополнительные уравнения (например, при расчете турбулентного течения) решаются стандартным способом (согласно методу контрольных объемов), в этом случае численные уравнения будут содержать массовые потоки. Поэтому дополнительные уравнения решаются после этапа 4 алгоритма SIMPLE.

Сходимость достигается, когда для каждой ячейки *суммарный* массовый поток стал достаточно близок к нулевому значению.

3.3 Алгоритм PISO

Алгоритм PISO (Pressure Implicit with Splitting of Operators) [8] применяется для **нестационарных задач**. С одной стороны в таких задачах ошибки могут возникать как из-за нелинейности уравнений (они линеаризуются и это порождает ошибки), так и благодаря измению течения со временем.

Различие между методами SIMPLE и PISO заключается в том, что в этих методах назначены разные приоритеты ошибкам. В SIMPLE уточнение нелинейности уделяется «больше внимания», тогда как метод PISO в первую очередь нацелен на снижение ошибок, связанных с имзенением течения со временем.

Последовательность действий в алгоритме PISO [2], [7]:

1. Определить начальные значения скорости $\mathring{\boldsymbol{u}}^n$ и давления p^n.
   Верхний индекс n обозначает номер временного слоя.
2. Решить уравнение двиежния (88).
   В результате получена скорость $\boldsymbol{u}^*$.
3. Решить уравнение на давление (86), используя найденную скорость $\boldsymbol{u}^*$.
   В результате получено давление p^*.
4. Уточнить поле скорости, согласно формуле (83).
   Получена скорость $\boldsymbol{u}^*$.
5. Обновить значение функции $\boldsymbol{H(u)}$ по формуле (80).
6. Если сходимость недостигнута, то обновить значения: $\mathring{\boldsymbol{u}}^n = \boldsymbol{u}^*, p^n = p^*$.
   Затем перейти к шагу 3.
   Когда сходимость достигунта перейти к следующему шагу.
7. Сделать коррекцию потоков для всех граней по формуле (90).
8. Решить дополнительные уравнения задачи, если такие есть.

Основной цикл алгоритма PISO включает в себя шаги 3, 4, 5 и обычны выполняются 1-2 цикла повторений этих шагов. **Сходимость** считается достигнутой, когда во всех ячейках *суммарные* массовые потоки стали достаточно малыми, чтобы их можно было считать практически нулевыми.

Литература

[1] Патанкар С. Численные методы решения задач теплообмена и динамики жидкости. — 1984.

[2] Greenshields C. Weller H. Notes on Computational Fluid Dynamics: General Principles. — 2022.

[3] Ильин В.А. Позняк Э.Г. Основы математического анализа: В 2-х ч. Часть 1. — 2005.

[4] Амосов А.А. Дубинский Ю.А. Копченова Н.В. Вычислительные методы для инженеров. — 1994.

[5] R.J. Shewchuk. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. — 1994. — Режим доступа: https://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf.

[6] A. Brandt. Multi-Level Adaptive Solutions to Boundary-Value Problems. — 1977.

[7] Hrvoje J. Error Analysis and Estimation for the Finite Volume Method With Applications to Fluid Flows. — 1996.

[8] Issa R.I. Локальная структура турбулентности в несжимаемой вязкой жидкости при очень больших числах Рейнольдса // Journal of Computational Physics. — 1986. — Режим доступа: https://www.scirp.org/reference/referencespapers?referenceid=2615422.

[9] Документация для платформы OpenFOAM. — Режим доступа: https://www.openfoam.com/documentation/guides/latest/api/.

[10] Pressure-velocity algorithms in OpenFOAM. — Режим доступа: https://doc.openfoam.com/2306/tools/processing/solvers/pressure-velocity/.

[11] SIMPLE algorithm in OpenFOAM. — Режим доступа: https://doc.openfoam.com/2306/tools/processing/solvers/pressure-velocity/simple/.