Школа-семинар

«Расширенные возможности пакета OpenFOAM»

СТРУКТУРНЫЕ ЕДИНИЦЫ OPENFOAM

М.В. Крапошин (НИЦ Курчатовский Институт)
О.И. Самоваров (Институт Системного Программирования РАН)
С.В. Стрижак (ГОУ ВПО МГТУ им. Баумана)

СОДЕРЖАНИЕ

- C++ в OPENFOAM
- УРОВНИ АБСТРАКЦИИ
- ОСНОВНЫЕ КЛАССЫ И ИХ НАЗНАЧЕНИЕ

$$pV = nRT \qquad \frac{\partial \rho}{\partial t} + \nabla \cdot \rho \boldsymbol{U} = 0$$

fvm::ddt(rho) + fvc::div(phi) = 0

$$\frac{\partial \rho \boldsymbol{U}}{\partial t} + \nabla \cdot (\rho \boldsymbol{U}\boldsymbol{U}) - \nabla \cdot \left(\mu \frac{1}{2} \left(\nabla \boldsymbol{U} + (\nabla \boldsymbol{U})^T \right) \right) = -\nabla p$$

fvm::ddt(rho, U) + fvm::div(phi,U) -
fvm::laplacian(mu,U)
=
-fvc::grad(p)

ИСПОЛЬЗОВАНИЕ ТЕХНОЛОГИИ C++ В OpenFOAM

Инкапсуляция — разработаны классы тензоров 0-го, 1-го и 2-го порядков в 3-х мерном пространстве, определены понятия сетки (пространства), времени, полей величин, схем дискретизации и решения систем линейных алгебраических уравнений. Работа с ними осуществляется как с отдельными объектами («чёрными ящиками»).

Наследование — за счет гибкого, эволюционного подхода наследования исключено дублирование кода, построена естественная иерархия объектов (пример: модель турбулентности → RAS модель → k-ε модель).

Полиморфизм — одни и те же методы используются для математических операций с полями тензоров, векторов и скаляров, матриц и массивов.

УРОВНИ АБСТРАКЦИИ OpenFOAM

Платформа OpenFOAM включает в себя следующие уровни (могут находиться в различных библиотеках):

I. Библиотека, системный уровень	1) Системный: файлы, потоки, системные команды, динамические библиотеки 2) Примитивы программирования: массивы, списки, программные указатели, хэш-списки, контейнеры
II. Библиотека, Уровень разработчика	3) Математические примитивы: тензоры, вектора, скаляры, поля тензоров, размерности физических величин 4) Физическое время 5) Пространство (сетка) 6) Средства дискретизации уравнений, матрица СЛАУ 7) Методы решения систем линейных алгебраических уравнений
III. Приложения, Уровень пользователя	8) Алгоритмы интегрирования ДУ в ЧП 9) Решатели и утилиты (приложения)

Сложность

УРОВНИ OPENFOAM СОГЛАСНО HRVOJE JASAK

I. Пространство и время: **polyMesh**, **fvMesh**, **Time**
II. Алгебраические преобразования над полями: **Field**, **DimensionedField**, **GeometricField**
III. Граничные условия: **fvPatchField** и наследуемые классы
IV. Разреженные матрицы: **lduMatrix**, **fvMatrix** и линейные решатели
V. Конечно-Объёмная дискретизация: пространства имён **fvc::** и **fvm::**

Все эти классы соответствует 2-ому уровню абстракции. Часть из них являются общими, часть предназначена только для МКО

ПРИМИТИВЫ СИСТЕМНОГО УРОВНЯ (POSIX)

- **regExp** — работа со строками, поиск выражений
- **fileStat** — статус файла (чтение и запись)
- **sigFpe** — обработка ошибки при вычислениях с плавающей точкой (SIGFPE)
- **sigInt** — обработка ошибки при прерывании с клавиатуры (SIGINT)
- **sigQuit** — обработка при выходе с клавиатуры (SIGQUIT)
- **sigSegv** — обработка ошибки при обращении к запретной области памяти (SIGSEGV)
- **cpuTime** — счет времени, затрачиваемого ЦП (CPU)
- **clockTime** — счет общего времени, затрачиваемого на выполнение
- **Mutex** — определение одноместного семафора
- И другие классы

ПРИМИТИВЫ СИСТЕМНОГО УРОВНЯ, ПАМЯТЬ

Хранение объектов в памяти

- **template <class T> Xfer<T>** - простейший класс для хранения объекта, у инкапсулируемого объекта должны быть реализованы методы **transfer()** и **copy()**
- **template <class T> autoPtr<T>** - программный указатель, уничтожает объект при вызове метода **clear()**, обнуляет хранящийся адрес при вызове метода **ptr()**
- **template <class T> tmp<T>** - программный указатель с учетом числа ссылок (инкапсулированный объект должен содержать методы **count()**, **okToDelete()**, **resetRefCount()** и операторы **++()**, и **--()**

ПРИМИТИВЫ СИСТЕМНОГО УРОВНЯ, МАССИВЫ

Хранение объектов в памяти

- **template <class T> UList<T>** - одномерный массив данных с известным размером, использованием конструктора по умолчанию для вновь создаваемых элементов, итераторами, проверкой границ и вводом/выводом
- **template <class T> List<T>** - подкласс UList<T>, с известным размером и возможностью сохранения данных при изменении размера, операциями копирования и удаления
- **template <class T> DynamicList<T>** - подкласс List<T> с поддержкой переменного размера
- **template <class T, Key, class Hash> HashTable <T, Key, Hash>** - хэш-таблица, построенная в соответствии со стандартами STL
- Другие классы

ТЕНЗОРНЫЕ ПРИМИТИВЫ

- Скаляр: **scalar** — действительное число с плавающей точкой (одинарной или двойной точности
- Вектор: **template <class C> Vector<C>** - вектор размерности N чисел с плавающей точкой и определением соответствующих операций
- Тензор: **template <class C> Tensor<C>**- тензор 3D пространства для чисел с плавающей точкой и определением операций скалярного, тензорного и прочих типов

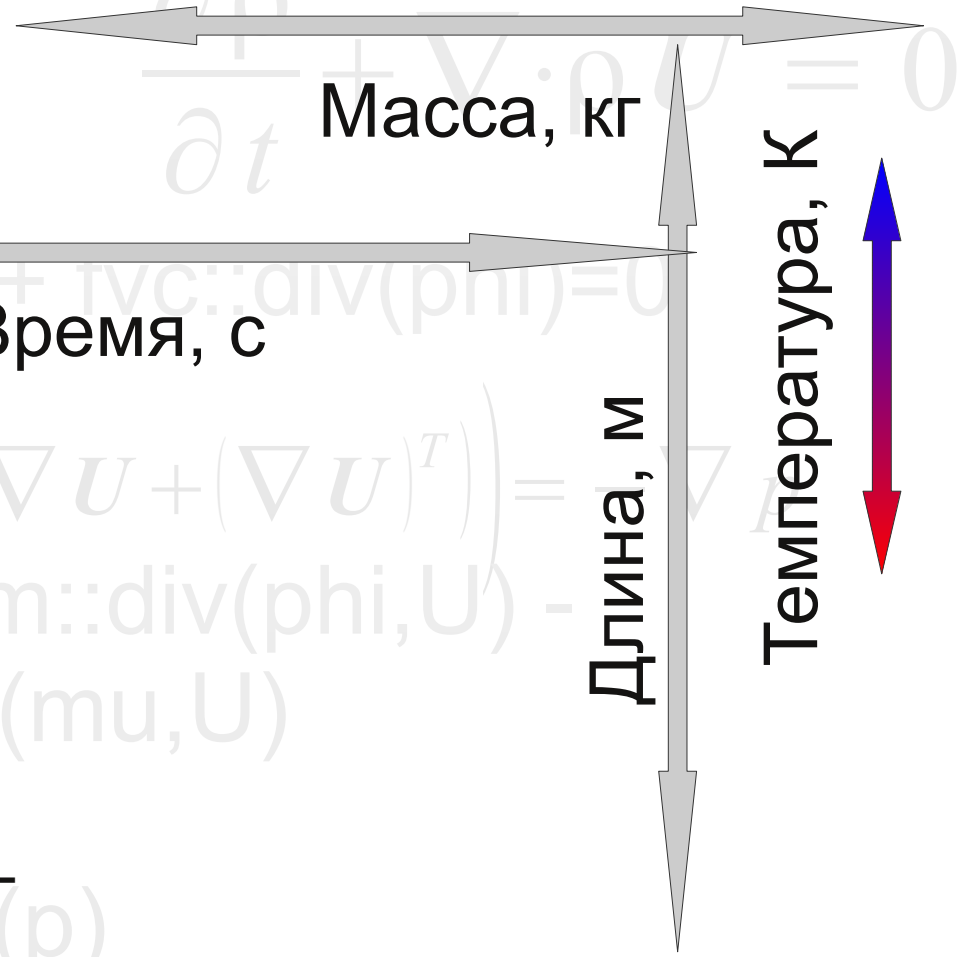
ОПЕРАЦИИ С ТЕНЗОРНЫМИ ПРИМИТИВАМИ

Для выполнения алгебраических преобразований над тензорами в OpenFOAM переопределены основные операторы:

№№	Оператор	Описание	№№	Оператор	Описание
1	=	Покомпонетное присваивание значения одного тензора другому. Ранг тензоров должен совпадать	11	T.T()	Операция транспонирования тензора
2	+, -, *,/	Покомпонетные операции сложения, вычитания и умножения и деления на скаляр	12	tr(T)	След тензора
3	*	Произведение векторов или тензоров	13	symm(T)	Симметричная составляющая тензора
4	&	Скалярное произведение	14	dev(T)	Антисимметричная составляющая тензора
5	&&	Двойное скалярное произведение	15	mag(T)	Нормал (модуль) тензора
6	^	Векторное произведение двух векторов	16	pow(T,n)	Возведение тензора в степень

РАЗМЕРНОСТЬ ФИЗИЧЕСКИХ ВЕЛИЧИН В OPENFOAM

- **dimensionSet:**
  - Масса, кг
  - Длина, м
  - Время, с
  - Температура, К
  - Количество вещества (моли), моль
  - Сила тока, А
  - Интенсивность освещения, Кд
- **dimensioned<T>:**
  - Имя string name_
  - Размерность dimensionSet dimensions_
  - Значение T value_
  - dimensionedScalar, dimensionedVector, dimensionedTensor...



ВРЕМЯ в OpenFOAM

- Управление временем модели (моделируемого процесса) — класс **Time**
- Наследует от:
  - **clock** — счетчик системного времени
  - **cpuTime** — счетчик процессорного времени
  - **TimePaths** — пути расположения основных файлов и каталогов случая (задачи OpenFOAM)
  - **objectRegistry** — регистр объектов случая OpenFOAM
  - **TimeState** — состояние времени (шаг-приращение по времени), номер шага по времени, определение момента времени для записи данных на диск
- Операции ввода/вывода в определенные моменты времени
- Операции перехода к новому шагу и критерии окончания цикла
- Поиск зарегистрированных объектов случая OpenFOAM

РЕЕСТР ОБЪЕКТОВ objectRegistry

- Разветвленное хранилище ссылок на все зарегистрированные объекты и физическое время **Time**
- Физическое время является корневым регистром. Каждый регистр может содержать под-регистры.
- Наследует от **regIOobject**, **HashTable<regIOobject*>**
- Операции работы с деревом: **parent()**, **subRegistry()**, **names()**
- Операции поиска объектов:
  - **template<class T> HashTable<T*> lookupClass();**
  - **template<class T> bool foundObject (const word& name);**
  - **template<class T> const T& lookupObject (const word& name);**
- Добавление/удаление объектов **checkIn(), checkOut()**

ДИСКРЕТИЗАЦИЯ ПРОСТРАНСТВА в OpenFOAM

- В терминах OpenFOAM пространство — это сетка (внутренняя — **internal** и пограничная — **boundary**)
- **fvMesh** (наследует **polyMesh**, **lduMesh**, **surfaceInterpolation**) — конечно-объёмная сетка, содержит все необходимые методы и данные для к.о. дискретизации
- **polyMesh** (наследует **primitiveMesh**, **objectRegistry**) — неструктурированная сетка с поддержкой контрольных объёмов произвольной формы, содержит топологию сетки и методы работы с топологией
- **primitiveMesh** — класс для установления связей между ячейками, гранями, рёбрами и точками

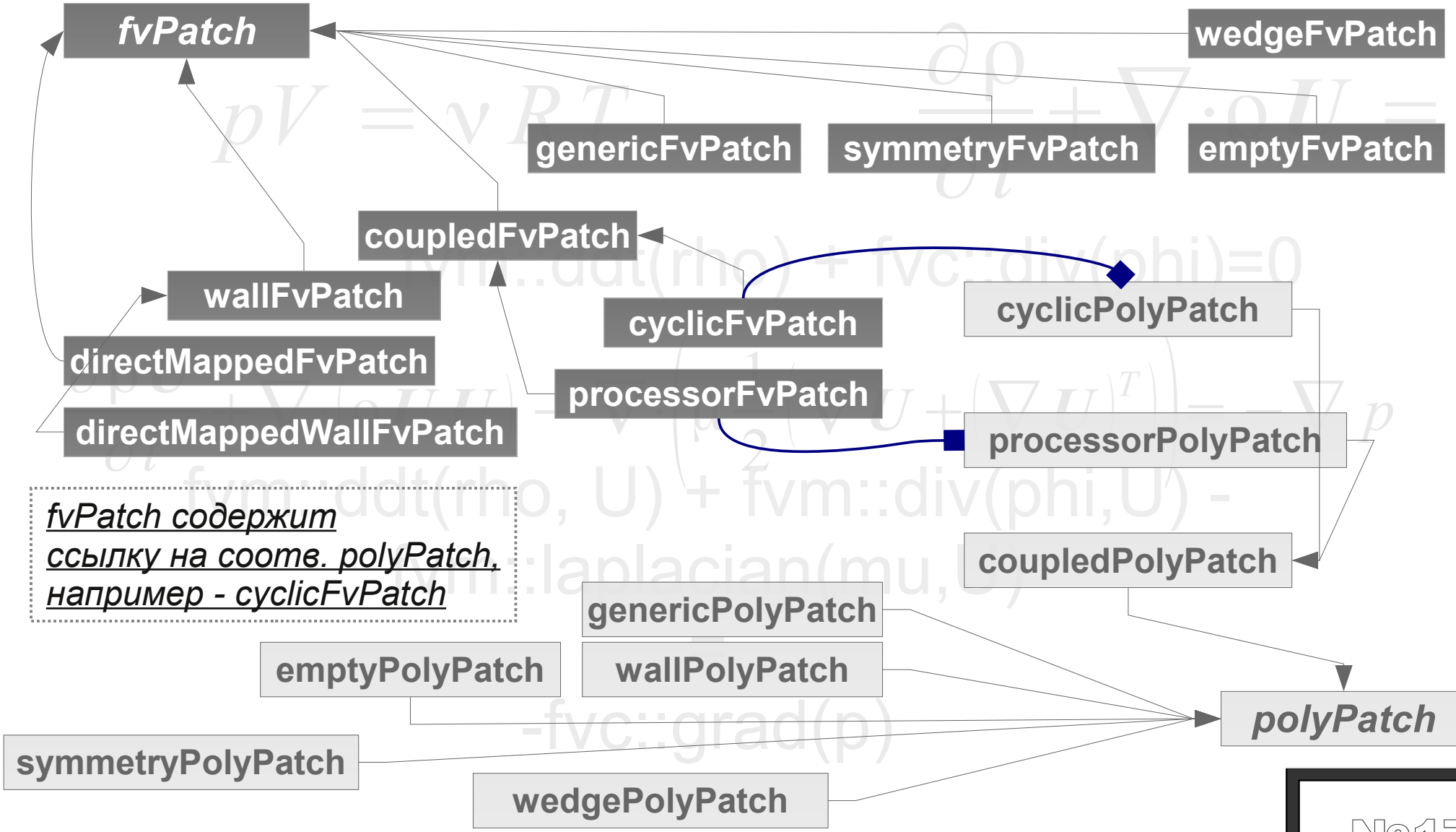
КЛАССИФИКАЦИЯ СЕТОК ПО ТИПУ РАСЧЕТНЫХ УЗЛОВ

- Дискретизация возможна в следующих типах расчетных узлов: а) центрах контрольных объёмов; б) центрах граней, ограничивающих к.о.; в) вершинах контрольных объёмов
- **template <class M> GeoMesh<M>** — базовый класс для всех типов сеток
- **volMesh** — наследует от **GeoMesh<fvMesh>** и содержит данные для выполнения конечно-объёмной дискретизации с расчетными точками в центрах ячеек (контрольных объёмов).
- **surfaceMesh** — наследует от **GeoMesh<fvMesh>** и содержит данные для выполнения конечно-объёмной дискретизации с расчетными точками в центрах граней контрольных объёмов.

ДИСКРЕТИЗАЦИЯ ФИЗИЧЕСКИХ ГРАНИЦ В OpenFOAM

- Доступ к границам: **const fvBoundaryMesh& fvMesh::boundary()** const
- **fvBoundaryMesh** наследует от **fvPatchList** (**List<fvPatch>**) - список внешних границ (патчей) расчетной области, на которых задаются граничные условия для каждого из полей задачи
- **fvPatch** содержит информацию о геометрии границы — центры граней **Cf()** и прилегающих к ним ячеек **Cn()**, их площади **magSf()**, **Sf()** и нормали **nf()**, вектора расстояний **delta()** от центров граней до центров прилегающих ячеек, ссылку на объект с топологией границы **patch()**
- **polyPatch** — наследует от **patchIdentifier**, **primitivePatch**, содержит в себе топологию границы и методы для работы с ней (центры граней, связи между гранями, связи между гранями и прилегающими ячейками, адресацию между номерами граней на данной внешней границе (локальными номерами) и глобальными (во всей расчетной области)
- **patchIdentifier** — идентификатор внешней границы (её номер в общем списке, имя и тип в виде строковой константы)
- **primitivePatch** — устанавливает адресацию между узлами и натянутыми на них гранями данной внешней границы

ТИПЫ ФИЗИЧЕСКИХ ГРАНИЦ OpenFOAM



ПРЕДСТАВЛЕНИЕ ПОЛЕЙ В OpenFOAM

- Поле как список значений в расчетных точках (например, центрах к.о.) **Field<T>** наследует от **refCount и List<T>**
- Поле физической величины определенной размерности **DimensionedField <class T, class GeoMesh>** наследует от **Field<T>** и **regIOobject**
- Геометрическое поле **GeometricField<class T, template<class> PatchField, class GeoMesh>** наследует от **DimensionedField<T, GeoMesh>** определено на всей расчетной области и вычисляется в определенных точках (к.о., гранях, узлах)
- Поле, определенное в центрах ячеек (**volFields.H**, **volFieldsFwd.H**):
  typedef GeometricField<scalar, fvPatchField,volMesh> volScalarField;
  typedef GeometricField<vector, fvPatchField,volMesh> volVectorField;
  typedef GeometricField<tensor, fvPatchField,volMesh> volTensorField;
- Поле, определенное в центрах граней (**surfaceFields.H**, **surfaceFieldsFwd.H**):
  typedef GeometricField<scalar, fvPatchField,surfaceMesh> surfaceScalarField;
  typedef GeometricField<vector, fvPatchField,surfaceMesh> surfaceVectorField

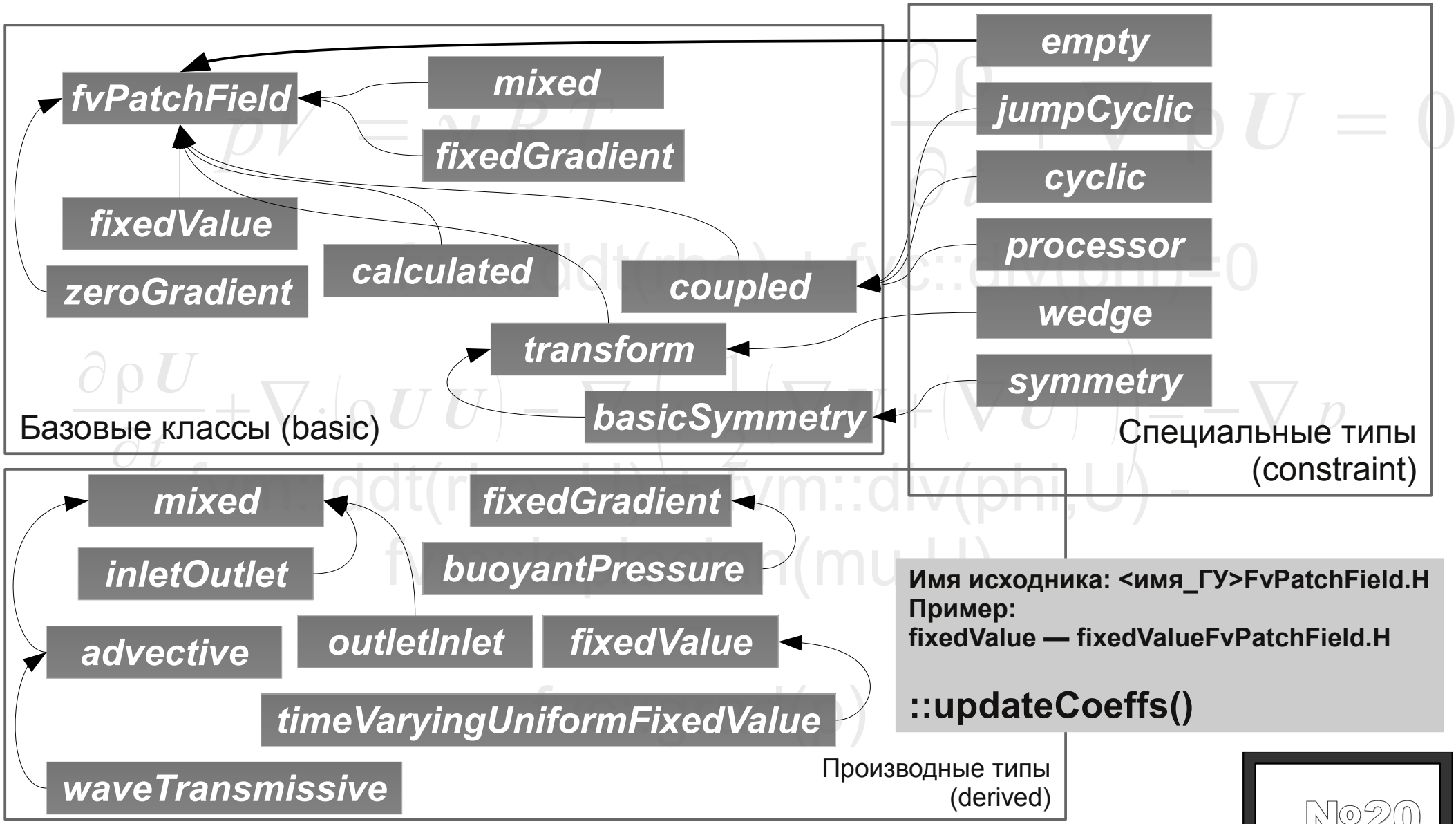
ДИСКРЕТИЗАЦИЯ ПОЛЕЙ НА ВНЕШНЕЙ ГРАНИЦЕ РАСЧЕТНОЙ ОБЛАСТИ

- Также, как и расчетная область (сетка), поля физических величин представлены значениями внутри расчетной области и значениями на границе. Последние группируются по именованному множеству граней и выступают как численная реализация граничного условия определенного типа.

- Поле, определенное на некоторой границе (**fvPatch**) определяется либо в шаблоне **fvPatchField<T>** (при дискретизации в центрах к.о., соответствует **vol***Field**) либо **fvsPatchField<T>** (при дискретизации в центрах граней, соответствует **surface***Field**). Параметр шаблона **T** - это тип поля (вектор, скаляр, тензор и прочее). Три звездочки это не известное **слово**, а: Scalar, Vector, Tensor и т.д.

- Класс **fv(s)PatchField** обязательно содержит: а) ссылку на сетку данной границы **patch()**, б) регистр объектов данного уровня **db()**, в) внутренее поле **internalField()**, г) производную по нормали **snGrad()**, д) диагональные коэффициенты матрицы **valueInternalCoeffs()** и **gradientInternalCoeffs()**, е) коэффициенты правой части уравнений **valueBoundaryCoeffs()** и **gradientBoundaryCoeffs()**, ж) операторы присваивания и простейшие арифметические операторы, з) маркер, определяющий является ли это ГУ 1-го рода — **fixesValue()**.

ИЕРАРХИЯ КЛАССОВ ГРАНИЧНЫХ УСЛОВИЙ



Имя исходника: <имя_ГУ>FvPatchField.H
Пример:
fixedValue — fixedValueFvPatchField.H

::updateCoeffs()

ПРЕДСТАВЛЕНИЕ РАЗРЕЖЕННЫХ МАТРИЦ. ИСХОДНАЯ МАТРИЦА

$Ax = b$ Или, разворачивая запись, получаем

$$A_{ii}\psi_i + \sum_j A_{ij}\psi_j = S_i$$

A_{ii} Диагональные элементы матрицы *A*

ψ_j Искомое поле, определенное в расчетных точках

$\sum_j A_{ij}\psi_j$ Сумма недиагональных элементов в строке *i*

S_i Правая часть уравнения (источник)

Имеем матрицу *N*x*N*, где *N* — число расчетных точек (контрольных объёмов)

ПРЕДСТАВЛЕНИЕ РАЗРЕЖЕННЫХ МАТРИЦ В ПАМЯТИ

1	2	3
6	5	4
7	8	9

L-D-U — матрица по сути описывает соединение контрольных объёмов с её соседями. L — соседи с номером меньшим номера к.о., D — диагональный элемент (данный к.о.) и U — верхний треугольник, элементы с большим номером

$$A_{ii}\psi_i + \sum_j A_{ij}\psi_j = S_i$$

lduMatrix

i	1	2	3	4	5	6	7	8	9
1	D	U				U			
2	L	D	U		U				
3		L	D	U					
4			L	D	U				U
5		L		L	D	U		U	
6	L				L	D	U		
7						L	D	U	
8					L		L	D	U
9				L				L	D

Отдельно хранятся диагональные элементы, верхний и нижний треугольники (хранятся только не нулевые значения). Получаем, что при $S_i = 0$, сумма недиагональных по строке должна иметь знак, обратный к диагональному.

Должно выполняться условие диагонального преобладания

ПРЕДСТАВЛЕНИЕ РАЗРЕЖЕННЫХ МАТРИЦ В ПАМЯТИ. АДРЕСАЦИЯ

COO (Coordinate storage) — все непустые значения хранятся в одном массиве (values). Помимо него используются ещё два вектора — номера непустых колонок (cols) и номера непустых рядов (rows)

OpenFOAM COO — отдельно хранятся диагональные элементы (D), верхние (U) — номера ячеек с индексом U > D, и нижние (L) с индексом L < D

i	1	2	3	4	5	6	7	8	9
1	D	U				U			
2	L	D	U		U				
3		L	D	U					
4			L	D	U				U
5		L		L	D	U		U	
6	L				L	D	U		
7						L	D	U	
8					L		L	D	U
9				L				L	D

D = 1, 2, 3, 4, 5, 6, 7, 8, 9

U = 2, 6, 3, 5, 4, 5, 9, 6, 8, 7, 8, 9

L = 2, 3, 3, 2, 4, 1, 5, 6, 5, 7, 4, 8

R = 1, 1, 2, 2, 3, 4, 4, 5, 5, 6, 7, 8

C = 1, 1, 2, 2, 3, 4, 4, 5, 5, 6, 7, 8

В представлении матрицы OpenFOAM каждой ячейке соответствуют: а) диагональный элемент, б) список элементов верхнего треугольника (по строкам), в) список элементов нижнего треугольника (по рядам)

СВЯЗЬ АДРЕСАЦИИ МАТРИЦЫ С РАСЧЕТНОЙ СЕТКОЙ

1	2	3
6	5	4
7	8	9

Структура представления расчетной сетки OpenFOAM соответствует представлению данных матрицы. Расчетная сетка представляет собой:

1) Список координат вершин, образующих контрольные объёмы
2) Список граней, образующих контрольные объёмы, каждый элемент такого списка содержит массив номеров узлов из списка вершин пункта 1, сортированный таким образом, что нормаль грани «смотрит» во-вне контрольного объёма
3) Список граней, принадлежащих контрольным объёмам (соединяющим один объём с другим, чей номер выше данного); нормаль такой грани направлена наружу упомянутого объёма (**N**eighbours), Число элементов — кол-во внутренних.
4) Список граней, принадлежащих соседним контрольным объёмам (с номером, меньшим чем данный) — **O**wners. Число элементов - общее количество внутренних граней (внутренние + внешние)

i	1	2	3	4	5	6	7	8	9
1	D	O				O			
2	N	D	O		O				
3		N	D	O					
4			N	D	O				O
5		N		N	D	O		O	
6	N				N	D	O		
7						N	D	O	
8					N		N	D	O
9				N				N	D

Owners — верхний треугольник, Neighbours - нижний

ПРЕДСТАВЛЕНИЕ РАЗРЕЖЕННЫХ МАТРИЦ В ПАМЯТИ

- Матрица содержится в классе lduMatrix:
  - Коэффициенты матрицы lowerPtr_, diagPtr_, upperPtr_
  - Адресация осуществляется через объект типа lduAddressing - ::lduAddr()
  - Доступ к матрице — lower(), diag(), upper()
  - Доступ к операторам H() и A() $A_P \psi_P + H_P = S_P$
  - Операторы сложения и вычитания -=, +=
  - Ссылку на расчетную область mesh()
  - Подклассы: а) solver; б) solverPerfomance; в) smoother; г) preconditioner

РЕШЕНИЕ СИСТЕМЫ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

Класс lduMatrix::solver:

- Имя искомой величины
- Ссылка на параметры поиска
- Критерий выхода из итераций
- Процедура решения — BICCG, ICCG, PCG, PbiCG
- Для решения используется процедура Matrix::solve()

СТАТИСТИКА СОЛВЕРА

IduMatrix::solverPerfomance — информация о процессе решения системы линейных алгебраических уравнений (private):

- Имя решателя solverName_
- Имя поля fieldName_
- Начальная невязка initialResidual_
- Конечная невязка finalResidual_
- Число итераций noIterations_
- Сходимость решения converged_
- Сингулярность решения singular_

СГЛАЖИВАНИЕ РЕШЕНИЯ

Сглаживание — процедура подавления высокочастотных составляющих решения (релаксация) при использовании многосеточных методов. Целью сглаживателя (smoother) ставится сглаживание решения для улучшения приближения на грубой сетке

Базовый абстрактный класс сглаживателя определен в классе lduMatrix::smoother, его реализации в классах:

- DICSmoother — сглаживание методом неполного разложения Холецкого
- DICGaussSeidelSmoother — сглаживание методом неполного разложения Холецкого и Гаусса-Зейделя
- DILUSmoother — сглаживание методом LU-разложения
- DILUGaussSeidelSmoother — сглаживание методом LU-разложения и Гаусса-Зейделя
- GaussSeidelSmoother — сглаживание методов Гаусса-Зейделя

ПРЕДОБУСЛАВЛИВАТЕЛИ

Базовый класс для всех предобуславливателей — lduMatrix::preconditioner, их реализации содержатся в классах:

- noPreconditioner — без предобуславливания
- diagonalPreconditioner — диагональный предобуславливатель
- GAMGPreconditioner — многосеточный предобуславливатель
- DILUPreconditioner — предобусловливание методом LU-разложения
- DICPreconditioner — предобусловливание методом неполного разложения Холецкого
- FDICPreconditioner — предобусловливание методом быстрого неполного разложения Холецкого

РЕШЕНИЕ СЛАУ, ПОЛУЧЕННЫХ ПОСЛЕ ДИСКРЕТИЗАЦИИ УРАВНЕНИЙ МЕТОДОМ КОНЕЧНЫХ ОБЪЁМОВ

fvMatrix — реализация **lduMatrix**, предназначенная для решения СЛАУ, полученных МКО, наследует от **refCount** и **lduMatrix**, осуществляет выбор решателя (**fvMatrix::fvSolver**). Уравнения могут решаться только для одной переменной (скаляр), решение уравнений для тензоров и векторов производится последовательно:

- ✗ solve() - решение системы СЛАУ, полученных после дискретизации уравнений на сетке
- ✗ psi() - искомое поле (скаляр)
- ✗ source() - источник
- ✗ другие функции связанные lduMatrix

Класс fvMatrix обладает свойствами самотождественности, аддитивности и коммутативности через переопределенные операторы:

$$LA = L_1 A + L_2 A = L_2 A + L_1 A = \left(L_1 + L_2\right) A$$

MULES

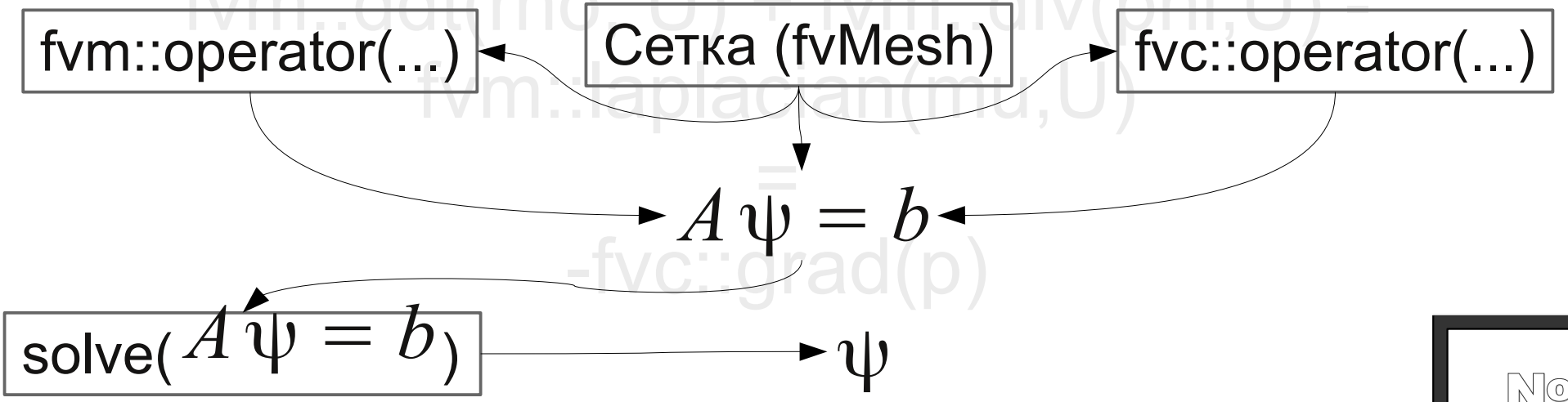
Для решения конвективных задач можно воспользоваться методом MULES — Multidimensional Universal Limiter with Explicit Solution — Многомерный Ограниченный Явный Метод Решения. Метод реализован в виде набора функций пространства имен MULES:

- MULES::explicitSolve(rho, psi, …) - явное решение с учетом плотности
- MULES::explicitSolve(psi, ...) - явное решение без учета плотности
- MULES::implicitSolve(rho, gamma, ...) - неявное решение с учетом плотности
- MULES::implcitSolve(gamma, …) - неявное решение без учета плотности
- MULES::limiter(...) - ограничение переменной

ОБЩИЙ ПОРЯДОК ИНТЕГРИРОВАНИЯ УРАВНЕНИЙ

Решение задачи проводится на три этапа. Результат выполнения каждого из трех — **fvMatrix**. В случае неявной дискретизации (**fvm::**) - матрица коэффициентов в левой части, в случае явной (**fvc::**) — вектор правой части (поле, определенное в расчетных точках)

1) Неявная дискретизация слагаемых уравнений (**fvm::**)
2) Явная дискретизация слагаемых уравнений (**fvc::**)
3) Дискретизация по времени (**fvm::ddt,fvm::d2dt2** и **fvc::dtdt, fvc::d2tdt2**)
4) Обновление граничных условий (возможно на шагах 1 и 2)
5) Решение системы уравнений



ПРОСТРАНСТВА ИМЕН FVC и FVM

- fvc:: — Finite Volume Calculus
- fvm:: — Finite Volume Method -

Общая схема вызовов:

Шаблоны функций операторов численного дифференцирования (fvm:: и fvc::) - fvm::div(...), fvc::ddt(...)

Статическая функция создания конкретной реализации численного оператора определенного класса fvc::, fvm:: operatorScheme::New(...) (divScheme::New(...), ddtScheme::New(...))

Реализация конкретного численного оператора дифференцирования определенного класса (градиент, дивергенция, диффузия, производная по времени)

Линия с кружочком означает вызов, а не наследование!!!

СХЕМА ВЫЗОВОВ (ОТ ОПЕРАТОРА ДО РЕАЛИЗАЦИИ)

Оператор fv::operator(psi)

Селектор класса оператора (fv::convectionScheme, fv::ddtScheme, fv::gradScheme, fv::laplacianScheme, fv::ddtScheme)

Селектор реализации оператора operatorScheme::New

Селектор явного/неявного дифференцирования (operatorScheme.fvcOperator(...), operatorScheme.fvmOperator(...))

fvm::div(phi,U)

fv::convectionScheme

fv::convectionScheme ::New(....) - static

fv::convectionScheme ::fvmDiv(phi,U)

ДИСКРЕТИЗАЦИЯ ДИВЕРГЕНЦИИ

$$\int_V \nabla\cdot(\rho \boldsymbol{U}\,\psi)\,dV = \int_S d\,\boldsymbol{S}\cdot(\rho \boldsymbol{U}\,\psi) = \sum_f S_f\cdot(\rho \boldsymbol{U})_f\,\psi_f$$

$$\int_V \nabla\cdot\psi\,dV = \int_S d\,\boldsymbol{S}\cdot(\psi) = \sum_f S_f\cdot(\psi)_f$$

Дискретизация конвективного слагаемого осуществляется с помощью группы функций **div(...)** пространства имён **fvc::** и **fvm::**, файлы **fvmDiv.H, fvcDiv.H**

```
template<class Type> tmp<fvMatrix<Type> >
fvm::div (const surfaceScalarField& flux, GeometricField<Type, fvPatchField, volMesh>&
vf,   const word& name);

template<class Type> tmp <GeometricField<
typename innerProduct<vector, Type>::type, fvPatchField, volMesh> >
div (const GeometricField<Type, fvPatchField, volMesh>& vf, const word& name)
```

Классы-реализации операторов:
fv::convectionScheme (fvm::, fvc::), fv::divScheme (fvc)

ДИСКРЕТИЗАЦИЯ ГРАДИЕНТА

$$\int_V \nabla(\psi)\, dV = \int_S (d\boldsymbol{S})\, \psi = \sum_f \boldsymbol{S}_f \psi_f$$

Дискретизация градиента (метод наименьших квадратов, теорема Гаусса и т. д.) осуществляется с помощью группы функций **grad(...)** пространства имён **fvc::,** для дискретизации градиента определенным методом используются **fvc::gGrad(...), fvc::lsGrad(...) и т. д.** Файл **fvcGrad.H**

```
template<class Type> tmp <GeometricField<
      typename outerProduct<vector,Type>::type, fvPatchField, volMesh> >
fvc::grad (
   const GeometricField<Type, fvPatchField, volMesh>& vf,
   const word& name)
```

Операторы реализуются в классе **gradScheme**

ДИСКРЕТИЗАЦИЯ ДИФФУЗИОННОГО СЛАГАЕМОГО

$$\int_V \nabla\cdot(\Gamma_\psi \nabla\psi)\,dV = \int_S d\boldsymbol{S}\cdot(\Gamma_\psi \nabla\psi) = \sum_f \boldsymbol{S}_f\cdot(\Gamma_\psi \nabla\psi)_f$$

Дискретизация диффузионного слагаемого (лапласиан в OpenFOAM) осуществляется с помощью группы функций **laplacian(...)** пространства имён **fvc::** и **fvm::,** файлы **fvmLaplacian.H** и **fvcLaplacian.H**

template<class Type, class GType> tmp<fvMatrix<Type> >
fvm::laplacian (const GeometricField<GType, fvsPatchField, surfaceMesh>& gamma,
GeometricField<Type, fvPatchField, volMesh>& vf, const word& name)

template<class Type, class Gtype> tmp<GeometricField<Type, fvPatchField, volMesh> >
fvc::laplacian (const GeometricField<GType, fvsPatchField, surfaceMesh>& gamma,
const GeometricField<Type, fvPatchField, volMesh>& vf, const word& name);

Операторы реализуются в классе **laplacianScheme**

ПЕРВАЯ ПРОИЗВОДНАЯ ПО ВРЕМЕНИ

$$\frac{\partial}{\partial t}\int_V \rho\psi\, dV$$

Дискретизация первой производной осуществляется с помощью группы функций **ddt(...)** пространства имён **fvc::** и **fvm::,** используются схемы первого порядка (Эйлера) и второго порядка (обратного дифференцирования), файлы **fvmDdt.H**, **fvcDdt.H**

template<class Type> tmp<fvMatrix<Type> >
fvm::ddt (GeometricField<Type, fvPatchField, volMesh>& vf)

template<class Type> tmp<GeometricField<Type, fvPatchField, volMesh> >
fvc::ddt (const dimensioned<Type> dt, const fvMesh& mesh)

Операторы реализуются в классе **ddtScheme**

ВТОРАЯ ПРОИЗВОДНАЯ ПО ВРЕМЕНИ

$$\frac{\partial}{\partial t}\int_V \rho \frac{\partial \psi}{\partial t} dV$$

Дискретизация первой производной осуществляется с помощью группы функций **d2dt2(...)** пространства имён **fvc::** и **fvm::,** используются схема первого порядка (Эйлера), файлы **fvmD2dt2.H**, **fvcD2dt2.H**

template<class Type> tmp<fvMatrix<Type> > fvm::d2dt2 (GeometricField<Type, fvPatchField, volMesh>& vf)

template<class Type> tmp<GeometricField<Type, fvPatchField, volMesh> > fvc::d2dt2 (const GeometricField<Type, fvPatchField, volMesh>& vf)

Операторы реализуются в классе **d2dt2Scheme**

ПРОИЗВОДНАЯ ПО НОРМАЛИ

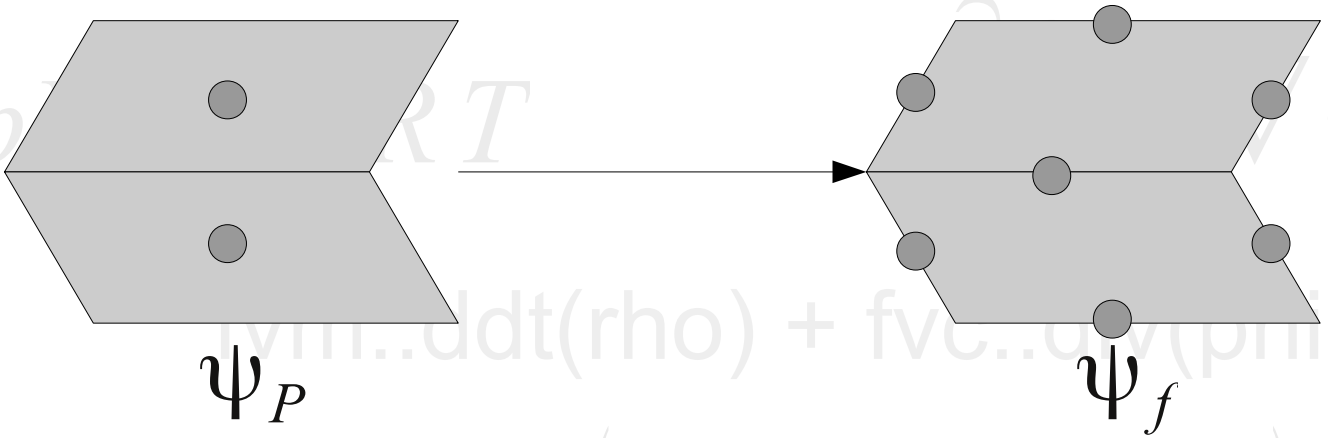
$$\frac{\partial \psi}{\partial \boldsymbol{n}}$$

Дискретизация производной по нормали осуществляется с помощью группы функций **snGrad(...)** пространства имён **fvc::,** множество расчетных значений определено на поверхности (на гранях расчетной сетки), файлы **fvcSnGrad.H**

```
template<class Type> tmp<GeometricField<Type, fvsPatchField, surfaceMesh> >
fvc::snGrad (const GeometricField<Type, fvPatchField, volMesh>& vf, const word&
name)
{
    return fv::snGradScheme<Type>::New
    (vf.mesh(), vf.mesh().snGradScheme(name))().snGrad(vf);
}
```

Операторы реализуются в классе **snGradScheme**

ИНТЕРПОЛЯЦИЯ



- Интерполяция с узлов ячеек на центры ячеек: **pointVolInterpolation** (веса обратно пропорциональны расстоянию)
- Интерполяция с центров ячеек на узлы ячеек **volPointInterpolation** (веса обратно пропорциональны расстоянию)
- Интерполяция поля (объёмного или поверхностного) в произвольной точке пространства, классы **interpolationCell**, **interpolationCellPoint**, **interpolationCellPointFace**
- Интерполяция с объёмного поля на поверхностное поле — surfaceInterpolationScheme. Ограниченные схемы интерполяции — папка **limitedSchemes**, неограниченные — **schemes**, многомерные схемы интерполяции — папка **multivariateSchemes**

ПАРАЛЛЕЛЬНОЕ ВЫПОЛНЕНИЕ: УПРАВЛЕНИЕ ПОТОКАМИ

- Класс Pstream (почти все члены статические)
- bool parRun()
- nProcs()
- master()
- masterNo()
- myProcNo()
- Классы IPstream (ввод данных, оператор >>), OPstream (вывод данных, оператор <<)

ПОДДЕРЖКА ДВИЖУЩИХСЯ СЕТОК

- dynamicFvMesh наследует от fvMesh
- fvMotionSolver — движение сетки (наследует от motionSolver) без изменения топологии
- motionSmoother — контроль за качеством сетки при её деформации
- topoChangerFvMesh — движение сетки с изменением топологии (остальные классы наследуют от этого, например mixerFvMesh)

ДИНАМИЧЕСКИЕ БИБЛИОТЕКИ: МОДЕЛИ ТУРБУЛЕНТНОСТИ

$$R^{Eff} = (\mu + \mu^t)\left[\nabla \boldsymbol{U} + (\nabla \boldsymbol{U})^T\right]$$

(mu + mut) * (fvc::grad(U) + fvc::grad(U).T())

1) Все RAS-модели опираются на одно и тоже предположение Буссинеска
2) Расчет эффективного тензора Рейнольдса зависит только от параметров конкретной модели
3) Эти особенности выносятся в отдельную библиотеку, например, **libincompressibleRASModels**
4) Множественность вариантов одной и той же функции осуществляется за счет полиморфизма — механизма виртуальных функций
4) Для включения в динамическую библиотеку информации о возможных вариантах виртуальной функции используются макросы:

- **defineTypeNameAndDebug(kEpsilon,0)**
  addToRunTimeSelectionTable(RASModel,kEpsilon,dictionary)
-
- **defineTemplateTypeNameAndDebugWithName(<name>, <debug level>)**

<debug level> - уровень отладки. 0 — без отладки, 1 — минимальный вывод
2 — вывод всех сообщений

ОБЗОР БИБЛИОТЕК (ПОДПАПКИ src), 1

- **ODE** — методы интегрирования обыкновенных дифференциальных уравнений
- **OSspecific** — системные вызовы и функции
- **OpenFOAM** — ядро программы, её основные классы
- **Pstream** — работа с потоками исполнения
- **autoMesh** — алгоритмы автоматического создания сеток
- **conversion** — алгоритмы конвертации форматов сеток
- **decompositionMethods** — методы декомпозиции задачи по пространству
- **dummyThirdParty** — интерфейсы к сторонним методам декомпозиции по пространству
- **dynamicFvMesh** — конечно-объёмная сетка с поддержкой её деформации
- **dynamicMesh** — методы деформации и изменения сетки
- **edgeMesh** — сетка, состоящая из отрезков (рёбер)
- **engine** — библиотека для решения задач, связанных с горением в двигателе
- **errorEstimation** — оценка погрешностей, ошибок аппроксимации, невязок
- **finiteVolume** — реализация метода конечных объёмов в OpenFOAM

ОБЗОР БИБЛИОТЕК (ПОДПАПКИ src), 2

- **fvAgglomerationMethods** — методы укрупнения сеток при использовании многосеточных алгоритмов решения систем линейных алгебраических уравнений
- **fvMotionSolver** — методы деформации сетки без изменения топологии по заданным перемещениям внешних границ расчетной области
- **genericPatchFields** — описание нетипизированного (абстрактного) граничного условия
- **lagrangian** — решение задач в переменных Лагранжа
- **meshTools** — утилиты для работы с сеткой
- **postProcessing** — анализ расчетных результатов
- **randomProcesses** — анализ случайных процессов (например, БПФ)
- **sampling** — анализ выборки из общих расчетных данных (полей)
- **surfMesh** — поверхностная сетка
- **thermophysicalModels** — термодинамические и теплофизические модели
- **topoChangerFvMesh** — изменение топологии конечно-объёмной сетки
- **transportModels** — модели транспорта (Ньютоновская и не-Ньютоновские)
- **triSurface** — триангулированная поверхность
- **turbulenceModels** — сжимаемые и несжимаемые RAS, LES и DES модели турбулентности

ПРИЛОЖЕНИЕ OpenFOAM. ОСНОВНЫЕ ЭТАПЫ

1)Инициализация структуры каталогов рабочей задачи

2)Инициализация потоков выполнения (MPI)

3)Инициализация физического времени

4)Инициализация расчетной сетки

5)Инициализация искомых полей

6)Инициализация цикла интегрирования (время)

7)Интегрирование уравнений и запись результатов

8)Завершение работы

СПАСБО ЗА ВНИМАНИЕ!

$$pV = \nu RT$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \boldsymbol{U} = 0$$

fvm::ddt(rho) + fvc::div(phi)=0

$$\frac{\partial \rho \boldsymbol{U}}{\partial t} + \nabla \cdot (\rho \boldsymbol{U}\boldsymbol{U}) - \nabla \cdot \left(\mu \frac{1}{2} \left(\nabla \boldsymbol{U} + (\nabla \boldsymbol{U})^T \right) \right) = -\nabla p$$

fvm::ddt(rho, U) + fvm::div(phi,U) -
fvm::laplacian(mu,U)
=
-fvc::grad(p)