Implementation of Solvers for Partial Differential Equations with PELICANS

March 15, 2010

INSTITUT DE RADIOPROTECTION ET DE SÛRETÉ NUCLÉAIRE
DIRECTION DE PRÉVENTION DES ACCIDENTS MAJEURS
SERVICE D'ETUDES ET DE MODELISATION DES INCENDIES, DU CONFINEMENT ET DU CORIUM
Centre d'Études de Cadarache - Bâtiment 702 - BP3-13115 SAINT PAUL LEZ DURANCE CEDEX
Téléphone : (33) 4.42.19.96.48 - Fax : (33) 4.42.19.91.66 - E-mail : bruno.piar@irsn.fr

Nomenclature

Notations relative to an *unknown field* $\mathbf{u}$ (in the continuous problem)

$N_{\mathrm{c}}^{\mathbf{u}}$	number of components
$X^{\mathbf{u}}$	functional space defining the required regularity conditions
$\mathcal{S}^{\mathbf{u}}$	sous espace de $X^{\mathbf{u}}$ auquel $\mathbf{u}$ appartient en tant que fonction objectif d'un problème différentiel
$\mathcal{V}^{\mathbf{u}}$	espace des fonctions tests associé à $\mathbf{u}$

Notations relatives à la triangulation du domaine géométrique

d	dimension de l'espace géométrique
$\mathcal{T}_h$	triangulation d'un ouvert $\Omega \subset \mathbb{R}^d$
$\mathcal{C}_e$	pour une triangulation $\mathcal{T}_h$ de $\Omega \subset \mathbb{R}^d$: e-ième cellule (maille de dimension d)
$\mathcal{F}_e$	pour une triangulation $\mathcal{T}_h$ de $\Omega \subset \mathbb{R}^d$: e-ième face (maille de dimension $d-1$)
K	pour une triangulation $\mathcal{T}_h$ de $\Omega \subset \mathbb{R}^d$: une maille (cellule ou face)

Notations relatives à la représentation discrète d'un champ $\mathbf{u}$

$X_h^{\mathbf{u}}$	sous espace de dimension finie de $X^{\mathbf{u}}$ associé à une triangulation $\mathcal{T}_h$
$\mathcal{S}_h^{\mathbf{u}}$	sous espace de dimension finie de $\mathcal{S}^{\mathbf{u}}$ associé à une triangulation $\mathcal{T}_h$
$\mathcal{V}_h^{\mathbf{u}}$	sous espace de dimension finie de $\mathcal{V}^{\mathbf{u}}$ associé à une triangulation $\mathcal{T}_h$
$N_{\mathrm{dof}}^{\mathbf{u}}$	nombre de degrés de liberté de la représentation discrète $\mathbf{u}_h$ de $\mathbf{u}$, dimension de $X_h^{\mathbf{u}}$
$N_{\mathrm{node}}^{\mathbf{u}}$	nombre de nœuds de la représentation discrète $\mathbf{u}_h$ de $\mathbf{u}$, nombre de fonctions de base scalaires $\mathsf{N}_i^{\mathbf{u}}$ apparaissant dans la famille génératrice de $X_h^{\mathbf{u}}$
$\mathbb{I}^{\mathbf{u}}$	ensemble des degrés de liberté de la représentation discrète $\mathbf{u}_h$ de $\mathbf{u}$ dont la valeur est *a priori* imposée
$\mathbb{U}^{\mathbf{u}}$	ensemble des degrés de liberté de la représentation discrète $\mathbf{u}_h$ de $\mathbf{u}$ qui sont des inconnues du problème discret
$N_{\mathrm{unk}}^{\mathbf{u}}$	nombre des degrés de liberté de la représentation discrète $\mathbf{u}_h$ de $\mathbf{u}$ qui appartiennent à $\mathbb{U}^{\mathbf{u}}$, dimension du vecteur des inconnues associées à $\mathbf{u}$ dans le problème discret

Notations réservées aux indixations dans la discrétisation éléments finis d'un champ $\mathbf{u}$

k $0 \leq k < N^{\mathbf{u}}_{\mathrm{dof}}$
numéro global de degré de liberté et indice des fonctions de base de $X^{\mathbf{u}}_h$

n $0 \leq n < N^{\mathbf{u}}_{\mathrm{node}}$
numéro global du nœud-géométrique $\mathbf{a}^{\mathbf{u}}_n$ et de la fonction de base scalaire associée

i $0 \leq i < N^{\mathbf{u}}_e$
pour un nœud de numéro global i donné et une maille K_e telle que support$(\mathsf{N}^{\mathbf{u}}_i) \cap K_e \neq \emptyset$, numéro local du nœud et de la fonction de base scalaire associée $\mathsf{N}^{\mathbf{u}}_i$

i_c $0 \leq i_c < N^{\mathbf{u}}_{\mathrm{c}}$
indice de composante

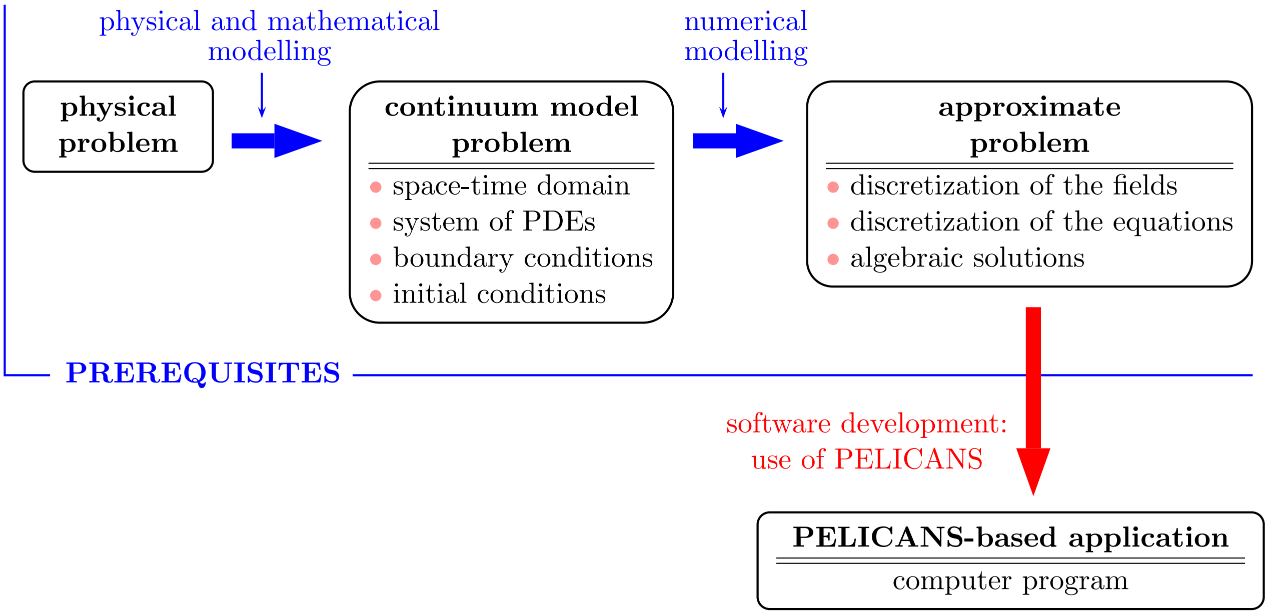
I $0 \leq I < N^{\mathbf{u}}_{\mathrm{unk}}$ indice dans le vecteur des inconnues discrètes du degré de liberté associé au nœud i et à la composante i_c de $\mathbf{u}$, degré de liberté qui appartient à $\mathbb{U}^{\mathbf{u}}$

Chapter I

Essential Concepts

I.1 An Approach to Numerical Simulation

I.1.1 Usage of PELICANS



I.1.2 Continuous Problem

Mathematical Model

a system of partial differential equations

Desired Formulation

In mathematical models, three fundamental ingredients are put forward when a subsequent use of PELICANS is foreseen:

1. the *unknown fields*, represented by **W** ;
2. the *equations* satisfied by the *unknown fields*:

$$\text{Find } \mathbf{W} \text{ such that} \left\{ \begin{array}{l} R(\mathbf{W}) = 0 \quad \text{in } Q \\ \text{boundary conditions and initial conditions} \end{array} \right. \tag{1.1}$$

3. the *parameters* of the continuous problem, *ie* the data appearing in the *equations* satisfied by the *unknown fields*.

The nomenclature and the notations are the following:

- **W** is a vector made of p scalar or vectorial functions.
- Q is a space-time domain such that the geometrical position is represented by a vector $\mathbf{x} \in \mathbb{R}^d$ where d is the number of space dimensions.
- R is a differential operator.

I.1.3 From the Continuous to the Discrete Problem

Two Distinct Notions of Discretization

The **approximation techniques** transforming the *continuous problem* into the *discrete problem* are based on **two** steps of **discretization**:

1. Build a *discrete representation* of **W**.

   Example of related method: finite element

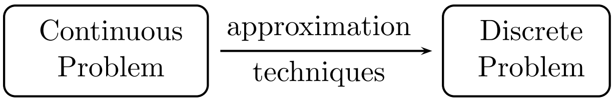
2. Build a *discrete problem*: approximate problem, deduced from the *continuous problem*, whose solution is the discrete representation of **W**.

   Examples of related method:

   - Galerkin, Galerkin Least Squares, Characteristic-Galerkin, Discontinuous Galerkin
   - finite volumes
   - finite differences

Remark : the finite element method, on the one hand, and the finite volume and finite difference methods, on the other hand, are different in nature and thus **conceptually incomparable**.

Anatomy of the Discrete Problem



Continuous Problem

approximation techniques

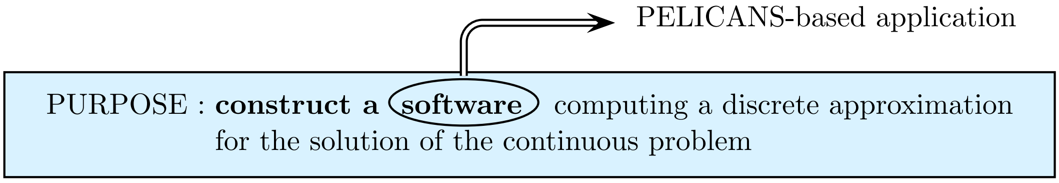
Discrete Problem

collection of **algebraic equations systems** to be solved **in sequence**

Possible Characteristics

- A sequence may correspond to:
  - a time marching procedure;
  - iterations of a nonlinear algorithm;
  - . . .
- In some numerical models of coupled problem:
  - the sequences may be encased ones in the others;
  - each step of a sequence may be another complete sequence.
  - . . .
- . . .

I.1.4 How to Find the Classes



PELICANS-based application

PURPOSE : **construct a software** computing a discrete approximation for the solution of the continuous problem

PELICANS-based application:

- object oriented software
- structure based on a decomposition into classes

“How to Find the Classes ?”

Discrete Problem

a **collection** of **algebraic equations systems** to be solved in **sequence**

What PELICANS Enforces

nothing . . .

What PELICANS Recommends

- Consider the notion of *algebraic equations systems* as

  $$\text{the conceptual pivot between} \begin{cases} \textbf{1.}\ \text{numerical modelling} \\ \textbf{2.}\ \text{choice} \left| \begin{array}{l} \text{of the classes to implement} \\ \text{of their respective responsibilities} \end{array} \right. \end{cases}$$

- For **each** *algebraic equations system*, implement two classes :
  - algebraic equations systems that may be built from elementary contributions and that may be solved
  - "discretizer" devoted to the computation of these elementary contributions and to handling the update of the discrete representation of the unknown fields

- Dedicate a specific class to the sequencing.

I.2 Discrete Representations

In order to simplify the writings, we will use specialized letters and typographies to denoted specific notions.

Hence, an *unknown field* (*i.e.* a component of $\mathbf{W}$) will be represented by the $\mathbf{u}$ sign and its number of components (if $\mathbf{u}$ is vectorial) by $N_{\text{c}}^{\mathbf{u}}$.

I.2.1 Discretization of an Unknown Field

▷ Discretizing $\mathbf{u}$ consists in choosing an approximate representation of $\mathbf{u}$ involving a finite number of real numbers, thus defining the notion of *degree of freedom* (**DOF**):

$$\mathbf{u} \in X^{\mathbf{u}} \underset{\text{approximation}}{\overset{\text{discretization}}{\rightleftarrows}} \big(u_k\big)_{0\le k<N_{\text{dof}}^{u}} \in \mathbb{R}^{N_{\text{dof}}^{u}} \tag{1.2}$$

where $X^{\mathbf{u}}$ is a functional space (infinite dimensional) formalizing the regularity constraints imposed to $\mathbf{u}$.

▷ The *degrees of freedom* are re-indexed to exhibit the *components* and, as a consequence, define **a** notion of *node*:

$$k \in [0, N_{\text{dof}}^{\mathbf{u}}[\quad \xrightarrow[\text{node and components}]{\text{indexation with}} \quad (n, i_c) \in [0, N_{\text{node}}^{\mathbf{u}}[\times[0, N_{\text{c}}^{\mathbf{u}}[\tag{1.3}$$

where n (resp. i_c) is called: node index (resp. component index).

▷

notion	index	domain
degree of freedom	k	$0 \le k < N_{\text{dof}}^{\mathbf{u}}$
node	n	$0 \le n < N_{\text{node}}^{\mathbf{u}}$
component	i_c	$0 \le i_c < N_{\text{c}}^{\mathbf{u}}$

$$N_{\text{dof}}^{\mathbf{u}} = N_{\text{node}}^{\mathbf{u}} \times N_{\text{c}}^{\mathbf{u}}$$

I.2.2 Algebraic Unknown and Discrete Representation

▷ The *continuous problem* **may** lead to *a priori* impose the value of some *degrees of freedom*.

$$\begin{aligned} &\mathbb{I}^{\mathbf{u}} \overset{\text{def}}{=} \left\{ k \in [0, N^{\mathbf{u}}_{\text{dof}}[\;\middle|\; \texttt{DOF } k \text{ has an } \texttt{imposed\_value} \right\} \quad \text{or, depending on the context} \\ &\mathbb{I}^{\mathbf{u}} \overset{\text{def}}{=} \left\{ (n, i_c) \in [0, N^{\mathbf{u}}_{\text{node}}[\times[0, N^{\mathbf{u}}_{\text{c}}[\;\middle|\; \texttt{DOF } (n, i_c) \text{ has an } \texttt{imposed\_value} \right\} \end{aligned} \tag{1.4}$$

▷ Among the $N^{\mathbf{u}}_{\text{dof}}$ degrees of freedom u_k of $\mathbf{u}$, not all are unknowns of the discrete problem: for instance, those identified by the indices of $\mathbb{I}^{\mathbf{u}}$, whose value is given *a priori*, can generally be eliminated algebraically.

Then, let us call $\mathbb{U}^{\mathbf{u}}$ the set of indices of the degrees of freedom that are unknowns of the discrete problem.

▷ These unknowns are commonly re-indexed by an increasing sequence of continuous indices starting from 0:

$$\begin{array}{ccccc} k \in [0, N^{\mathbf{u}}_{\text{dof}}[& \text{st} & k \in \mathbb{U}^{\mathbf{u}} & \xrightarrow[\text{indexation}]{\text{contiguous}} & I \in [0, N^{\mathbf{u}}_{\text{unk}}[\\ (n, i_c) \in [0, N^{\mathbf{u}}_{\text{node}}[\times[0, N^{\mathbf{u}}_{\text{c}}[& \text{st} & (n, i_c) \in \mathbb{U}^{\mathbf{u}} & & \text{with } N^{\mathbf{u}}_{\text{unk}} = \operatorname{card} \mathbb{U}^{\mathbf{u}} \end{array} \tag{1.5}$$

$$\begin{array}{|l|l|}\hline \text{unknown field} & \mathbf{u} \in X^{\mathbf{u}} \\ \hline \end{array} \longrightarrow \begin{array}{|l|l|}\hline \text{discrete representation} & (u_k)_k \in \mathbb{R}^{N^{\mathbf{u}}_{\text{dof}}} \\ \hline \text{algebraic unknown} & (u_I)_I \in \mathbb{R}^{N^{\mathbf{u}}_{\text{unk}}} \\ \hline \end{array}$$

▷ Generally (but not always) : $\mathbb{U}^{\mathbf{u}} = \begin{cases} [0, N^{\mathbf{u}}_{\text{dof}}[\setminus \mathbb{I}^{\mathbf{u}} \\ [0, N^{\mathbf{u}}_{\text{node}}[\times[0, N^{\mathbf{u}}_{\text{c}}[\setminus \mathbb{I}^{\mathbf{u}} \end{cases}$

▷ The correspondence (1.3) justifies that the same notation $\mathbb{I}^{\mathbf{u}}$ or $\mathbb{U}^{\mathbf{u}}$ is used to denote a set of indices $k \in [0, N^{\mathbf{u}}_{\text{dof}}[$ or a set of pairs $(n, i_c) \in [0, N^{u}_{\text{node}}[\times[0, N^{u}_{\text{c}}[$.

I.2.3 `PDE_DiscreteField` Class

▷ In PELICANS, there is **no** class directly associated to the following notion:

unknown field of the *continuous problem*.

▷ The **`PDE_DiscreteField`** class provides the management services for the *discrete representation* of *unknown fields*.

▷ The conceptual approximation that relates an *unknown field* and its *discrete representation* is **completely absent** from the definition of **`PDE_DiscreteField`**.

Object of type `PDE_DiscreteField`

▷ An object of type **`PDE_DiscreteField`**:

- is attached to an *unknown field*, *eg* $\mathbf{u}$;
- enables the management of:
  - $N^{\mathbf{u}}_{\text{sto}}$ **values** of the *discrete representation* of $\mathbf{u}$:

    $$(u^{(\ell)}_{n,i_c}) \qquad 0 \le n < N^{\mathbf{u}}_{\text{node}} \qquad 0 \le i_c < N^{\mathbf{u}}_{\text{c}} \qquad 0 \le \ell < N^{\mathbf{u}}_{\text{sto}}$$

  - a partial **value** of the *discrete representation* called **`imposed`**:

    $$(u^{(\texttt{imp})}_{n,i_c}) \qquad 0 \le n < N^{\mathbf{u}}_{\text{node}} \qquad 0 \le i_c < N^{\mathbf{u}}_{\text{c}} \qquad (n, i_c) \in \mathbb{I}^{\mathbf{u}}$$

▷ In the foregoing, **ff** denotes a pointer to a **PDE_DiscreteField** object attached to the *unknown field* **u**.

Some Queries of PDE_DiscreteField

expression	evaluation result
`ff->name()`	name of $\mathbf{u}$
`ff->nb_components()`	$N^{\mathbf{u}}_{\mathrm{c}}$
`ff->nb_nodes()`	$N^{\mathbf{u}}_{\mathrm{node}}$
`ff->storage_depth()`	$N^{\mathbf{u}}_{\mathrm{sto}}$
`ff->DOF_has_imposed_value( n, ic )`	**true** if $(n, i_c) \in \mathbb{I}^{\mathbf{u}}$ $\quad(n = \mathtt{n} \;\; i_c = \mathtt{ic})$
`ff->DOF_imposed_value( n, ic )`	$u^{(\mathtt{imp})}_{n,i_c}$ $\quad(n = \mathtt{n} \;\; i_c = \mathtt{ic})$
`ff->DOF_value( ll, n, ic )`	$u^{(\ell)}_{n,i_c}$ $\quad(\ell = \mathtt{ll} \;\; n = \mathtt{n} \;\; i_c = \mathtt{ic})$

Some Commands of PDE_DiscreteField

expression	side effect of the evaluation
`ff->set_DOF_value( ll, n, x, ic )`	$u^{(\ell)}_{n,i_c} \leftarrow \mathtt{x}$ $(\ell = \mathtt{ll} \;\; n = \mathtt{n} \;\; i_c = \mathtt{ic})$
`ff->set_DOF_imposed_value( n, x, ic )`	$u^{(\mathtt{imp})}_{n,i_c} \leftarrow \mathtt{x}$ $(n = \mathtt{n} \;\; i_c = \mathtt{ic})$
`ff->enforce_imposed_values_to_DOFs( ll )`	$u^{(\ell)}_{n,i_c} \leftarrow u^{(\mathtt{imp})}_{n,i_c}$ $\quad(\ell = \mathtt{ll} \;)$ $\forall (n, i_c) \in \mathbb{I}^{\mathbf{u}}$
`ff->copy_DOFs_value( s, t )`	$u^{(t)}_{n,i_c} \leftarrow u^{(s)}_{n,i_c}$ $\quad(t = \mathtt{t} \;\; s = \mathtt{s} \;)$ $\forall (n, i_c) \in [0, N^{\mathbf{u}}_{\mathrm{node}}[\times [0, N^{\mathbf{u}}_{\mathrm{c}}[$
`ff->extract_DOFs_value( ll, vv, ic )`	$v_n \leftarrow u^{(\ell)}_{n,i_c}$ $\quad \forall n \in [0, N^{\mathbf{u}}_{\mathrm{node}}[$ $(\ell = \mathtt{ll} \;\; i_c = \mathtt{ic} \;\; v = \mathtt{vv})$

Adjustment of the Set of Nodes

▷ Some simulations may necessitate modification of the set of nodes of `*ff`.

▷ It is possible to add new nodes to `*ff`.

▷ A node of `*ff` may be tagged active or not

expression	action
`ff->add_nodes( nb_supp_nodes )`	add `nb_supp_nodes` nodes
`ff->set_node_active( n )`	tag the node `n`: “active”
`ff->set_node_inactive( n )`	tag the node `n`: “not active”
`ff->node_is_active( n )`	returns `true` if the node `n` is “active”
`ff->DOF_is_free( n, ic )`	returns `true` if the node `n` is “active” and if $(n, i_c) \notin \mathbb{I}^{\mathbf{u}}$ ($n =$ `n` $i_c =$ `ic`)
`ff->DOF_is_fixed( i, ic )`	return `true` if `ff->DOF_is_free( i, ic )` returns `false`

I.2.4 `PDE_LinkDOF2Unknown` Class

The `PDE_LinkDOF2Unknown` class provides the services for the management of the correspondence between:

- the indexation of the *degrees of freedom*;
- the indexation of the *discrete unknowns*.

Object of type `PDE_LinkDOF2Unknown`

▷ An object of type `PDE_LinkDOF2Unknown` (*eg* referenced by the `ff_link` pointer)

- is associated to an object of type `PDE_DiscreteField` (*eg* referenced by the `ff` pointer), discrete representation of an unknown field $\mathbf{u}$;
- is associated to all the components or to one of the components of `*ff`;
- enables the management of the correspondence between the indexation:

$$i_c \in [0, N^{\mathbf{u}}_{\mathrm{c}}[\quad n \in [0, N^{\mathbf{u}}_{\mathrm{node}}[\quad (n, i_c) \in \mathbb{U}^{\mathbf{u}} \qquad \rightleftarrows \qquad I \in [0, N^{\mathbf{u}}_{\mathrm{unk}}[$$

  where:

  - the set $\mathbb{U}^{\mathbf{u}} \subset [0, N^{\mathbf{u}}_{\mathrm{node}}[\times [0, N^{\mathbf{u}}_{\mathrm{c}}[$
  - the relation “I function of (n, i_c)”

  are established exactly when creating the object `*ff_link`.

▷ In the foregoing, `ff` denotes a pointer to a `PDE_DiscreteField` object attached to the *unknown field* $\mathbf{u}$, and `ff_link` denotes a pointer to a `PDE_LinkDOF2Unknown` object attached to `*ff`.

Instantiation of `PDE_LinkDOF2Unknown`

▷ Creation of an object of type **`PDE_LinkDOF2Unknown`** associated to all the components of **`*ff`**:

```
ff_link = PDE_LinkDOF2Unknown::create( a_owner, ff,
                    "sequence_of_the_components", imposed_out ) ;
```

increasing I ↓	node 0	component 0	$(0,0) \in \mathbb{U}^{\mathbf{u}}$
		⇣	
	node 0	component $N_{\mathrm{c}}^{\mathbf{u}} - 1$	$(0, N_{\mathrm{c}}^{\mathbf{u}} - 1) \in \mathbb{U}^{\mathbf{u}}$
	⇣		
	node $N_{\mathrm{node}}^{\mathbf{u}} - 1$	component 0	$(N_{\mathrm{node}}^{\mathbf{u}} - 1, 0) \in \mathbb{U}^{\mathbf{u}}$
		⇣	
	node $N_{\mathrm{node}}^{\mathbf{u}} - 1$	component $N_{\mathrm{c}}^{\mathbf{u}} - 1$	$(N_{\mathrm{node}}^{\mathbf{u}} - 1, N_{\mathrm{c}}^{\mathbf{u}} - 1) \in \mathbb{U}^{\mathbf{u}}$

`imposed_out = true`

$(n, i_c) \in \mathbb{U}^{\mathbf{u}} \iff$ the node n is “active” and $(n, i_c) \notin \mathbb{I}^{\mathbf{u}}$

`imposed_out = false`

$(n, i_c) \in \mathbb{U}^{\mathbf{u}} \iff$ the node n is “active”

▷ Creation of an object of type **`PDE_LinkDOF2Unknown`** associated to all the components of **`*ff`**:

```
ff_link = PDE_LinkDOF2Unknown::create( a_owner, ff,
                    "sequence_of_the_nodes", imposed_out ) ;
```

increasing I ↓	component 0	node 0	$(0,0) \in \mathbb{U}^{\mathbf{u}}$
		⇣	
	component 0	node $N_{\mathrm{node}}^{\mathbf{u}} - 1$	$(N_{\mathrm{node}}^{\mathbf{u}} - 1, 0) \in \mathbb{U}^{\mathbf{u}}$
	⇣		
	component $N_{\mathrm{c}}^{\mathbf{u}} - 1$	node 0	$(0, N_{\mathrm{c}}^{\mathbf{u}} - 1) \in \mathbb{U}^{\mathbf{u}}$
		⇣	
	component $N_{\mathrm{c}}^{\mathbf{u}} - 1$	node $N_{\mathrm{node}}^{\mathbf{u}} - 1$	$(N_{\mathrm{node}}^{\mathbf{u}} - 1, N_{\mathrm{c}}^{\mathbf{u}} - 1) \in \mathbb{U}^{\mathbf{u}}$

`imposed_out = true`

$(n, i_c) \in \mathbb{U}^{\mathbf{u}} \iff$ the node n is “active” and $(n, i_c) \notin \mathbb{I}^{\mathbf{u}}$

`imposed_out = false`

$(n, i_c) \in \mathbb{U}^{\mathbf{u}} \iff$ the node n is “active”

▷ Creation of an object of type **`PDE_LinkDOF2Unknown`** associated to the component i_c =**`ic`** (given *a priori*) of **`*ff`**:

```
ff_link = PDE_LinkDOF2Unknown::create( a_owner, ff,
                                       ic, imposed_out ) ;
```

increasing I

node 0 component i_c $(0, i_c) \in \mathbb{U}^{\mathbf{u}}$

node $N^{\mathbf{u}}_{\text{node}} - 1$ component i_c $(N^{\mathbf{u}}_{\text{node}} - 1, i_c) \in \mathbb{U}^{\mathbf{u}}$

`imposed_out = true`

$(n, i_c) \in \mathbb{U}^{\mathbf{u}} \iff$ the node n is "active" and $(n, i_c) \notin \mathbb{I}^{\mathbf{u}}$

`imposed_out = false`

$(n, i_c) \in \mathbb{U}^{\mathbf{u}} \iff$ the node n is "active"

▷ When $N^{\mathbf{u}}_{\text{c}} = 1$, creation of an object of type **`PDE_LinkDOF2Unknown`** associated to **`*ff`**:

```
ff_link = PDE_LinkDOF2Unknown::create( a_owner, ff,
                                       imposed_out ) ;
```

increasing I

node 0 $(0, 0) \in \mathbb{U}^{\mathbf{u}}$

node $N^{\mathbf{u}}_{\text{node}} - 1$ $(N^{\mathbf{u}}_{\text{node}} - 1, 0) \in \mathbb{U}^{\mathbf{u}}$

`imposed_out = true`

$(n, 0) \in \mathbb{U}^{\mathbf{u}} \iff$ the node n is "active" and $(n, 0) \notin \mathbb{I}^{\mathbf{u}}$

`imposed_out = false`

$(n, 0) \in \mathbb{U}^{\mathbf{u}} \iff$ the node n is "active"

Some Member Functions of `PDE_LinkDOF2Unknown`

▷ Some *Queries*:

expression	evaluation result
`ff_link->field()`	**`ff`**
`ff_link->unknown_vector_size()`	$N^{\mathbf{u}}_{\text{unk}} \equiv \text{Card}\, \mathbb{U}^{\mathbf{u}}$
`ff_link->DOF_is_unknown( n, ic )`	**`true`** if $(n, i_c) \in \mathbb{U}^{\mathbf{u}}$ ($n =$**`n`** $i_c =$ **`ic`**)
`ff_link->unknown_linked_to_DOF( n, ic )`	I

▷ A *Command*:

expression	side effect of the evaluation
`ff_link->reset()`	reinitializes **`*ff_link`** (if **`*ff`** has been modified since the creation of **`*ff_link`**)

Some Commands of **PDE_DiscreteField**

expression	side effect of the evaluation

`ff->extract_unknown_DOFs_value(`
`ll, vv, ff_link )`
	$v_I \leftarrow u^{(\ell)}_{n,i_c} \quad \forall (n,i_c) \in \mathbb{U}^{\mathbf{u}}$
	($\ell =$ **`ll`** $v =$ **`vv`**)

`ff->update_DOFs_value(`
`ll, vv, ff_link )`
	$u^{(\ell)}_{n,i_c} \leftarrow v_I \quad \forall (n,i_c) \in \mathbb{U}^{\mathbf{u}}$
	($\ell =$ **`ll`** $v =$ **`vv`**)

`ff->update_free_DOFs_value(`
`ll, vv, ff_link )`
	$u^{(\ell)}_{n,i_c} \leftarrow v_I \quad \forall (n,i_c) \in \mathbb{U}^{\mathbf{u}} \setminus \mathbb{I}^{\mathbf{u}}$
	($\ell =$ **`ll`** $v =$ **`vv`**)

`ff->add_to_free_DOFs_value(`
`ll, vv, ff_link, a )`
	$u^{(\ell)}_{n,i_c}$ **`+=`** $\alpha v_I \quad \forall (n,i_c) \in \mathbb{U}^{\mathbf{u}} \setminus \mathbb{I}^{\mathbf{u}}$
	($\ell =$ **`ll`** $v =$ **`vv`** $\alpha =$ **`a`**)

I.2.5 Nothing More than Indices ...

▷ The conceptual relation between the value of a *degree of freedom* and the *unknown field* of the *continuous problem*:

$$\left\{ \begin{array}{c} \text{set of numerical values} \\ u^{(\ell)}_{n,i_c} \in \mathbb{R} \end{array} \right\} \longleftrightarrow \left\{ \begin{array}{c} \text{unknown field} \\ \mathbf{u} \in X^{\mathbf{u}} \end{array} \right\}$$

▷ The main part of the services offered by the **`PDE_DiscreteField`** class is the management of the set of numerical values $u^{(\ell)}_{n,i_c}$ *via* the three-way indexation by:

- the node index n;
- the component index i_c;
- the storage level index ℓ:

independently of any numerical significance.

▷ The main part of the services offered by the **`PDE_LinkDOF2Unknown`** class is the conversion from the indexation with the node and component indices (n, i_c) to a linear contiguous indexation with the only index I.

I.3 Meshings and Discretizations

I.3.1 Meshing of a Geometrical Domain

Subdivision into Cells

$\Omega \subset \mathbb{R}^d$ a geometrical domain.

Subdivision into polyhedra $\mathcal{C}_e \subset \mathbb{R}^d$ with non overlapping interiors: $\boxed{\overline{\Omega} = \bigcup_e \mathcal{C}_e}$ $\rightsquigarrow$ *meshing* $\mathcal{T}_h$.

Various Geometrical Entities of a Meshing

cell	**`cell`**	$\mathcal{C}_e$	one of the polyhedra $\mathcal{C}_e$
face			face of a *cell*, which belongs to at most two distinct cells
inner face	**`side`**	$\mathcal{F}_e$	*face* that belongs to exactly two distinct *cells*
boundary face	**`bound`**	$\mathcal{F}_e$	*face* that belongs to a unique *cell*, hence to the boundary of Ω
mesh	**`mesh`**	K_e	*cell* or *inner face* or *boundary face*
vertex	**`vertex`**	$\mathbf{V}_i$	vertex of a *cell*

I.3.2 GE_Color class

Markers for Clusters of Meshes

It is possible to mark clusters of geometrical entities with *colors*.

▷ In the PELICANS jargon, a *color* is an instance of the **GE_Color** class.

▷ Each *color* is associated to a character sequence (**std::string**) which univocally defines it.

▷ A *composite color* is a color made of other (non composite) colors.

▷ The *colors* can be compared.

Usage Example

```
{
   GE_Color::extend( "blue" ) ;
   GE_Color::extend( "red" ) ;
   GE_Color::extend( "green" ) ;

   stringVector nl( 2 ) ;
   nl( 0 ) = "blue" ; nl( 1 ) = "red" ;
   GE_Color::extend( "bluered", nl ) ;

   nl( 0 ) = "green" ;
   GE_Color::extend( "greenred", nl ) ;
   GE_Color const* b  = GE_Color::object( "blue" ) ;
   GE_Color const* g  = GE_Color::object( "green" ) ;
   GE_Color const* br = GE_Color::object( "bluered" ) ;
   GE_Color const* gr = GE_Color::object( "greenred" ) ;

   cout « GE_Color::exist( "blue" )  « endl ; // 1
   cout « GE_Color::exist( "modry" ) « endl ; // 0

   cout « b->is_composite()  « endl ; // 0
   cout « br->is_composite() « endl ; // 1

   cout « br->has( "blue" )  « endl ; // 1
   cout « br->has( "modry" ) « endl ; // 0

   cout « br->is_matching( b ) « endl ; // 1
   cout « br->is_matching( g ) « endl ; // 0

   cout « br->is_overlapping( b )  « endl ;  // 1
   cout « br->is_overlapping( g )  « endl ;  // 0
   cout « br->is_overlapping( gr ) « endl ;  // 1
}
```

I.3.3 Discretization of a Field on a Meshing

▷ The relationship between an *unknown field* and its *discrete representation*

$$\left\{ \begin{array}{c} \text{unknown field} \\ \mathbf{u} \in X^{\mathbf{u}} \end{array} \right\} \longleftrightarrow \left\{ \begin{array}{c} \text{set of numerical values} \\ u_{i,i_c}^{(\ell)} \in \mathbb{R} \end{array} \right\}$$

is determined by the *conceptual operation of approximation* and is a matter for the nature of the discretization.

▷ The discretizations of fields in PELICANS rely on:

- **meshing** of geometrical domains;
- **finite element interpolations**.

I.3.4 Discretization of a Collection of Fields on a Meshing

▷ On a given geometrical domain:

- the continuous problem may involve several unknown fields;
- two distinct fields may have different discretizations (*eg* in terms of finite elements).

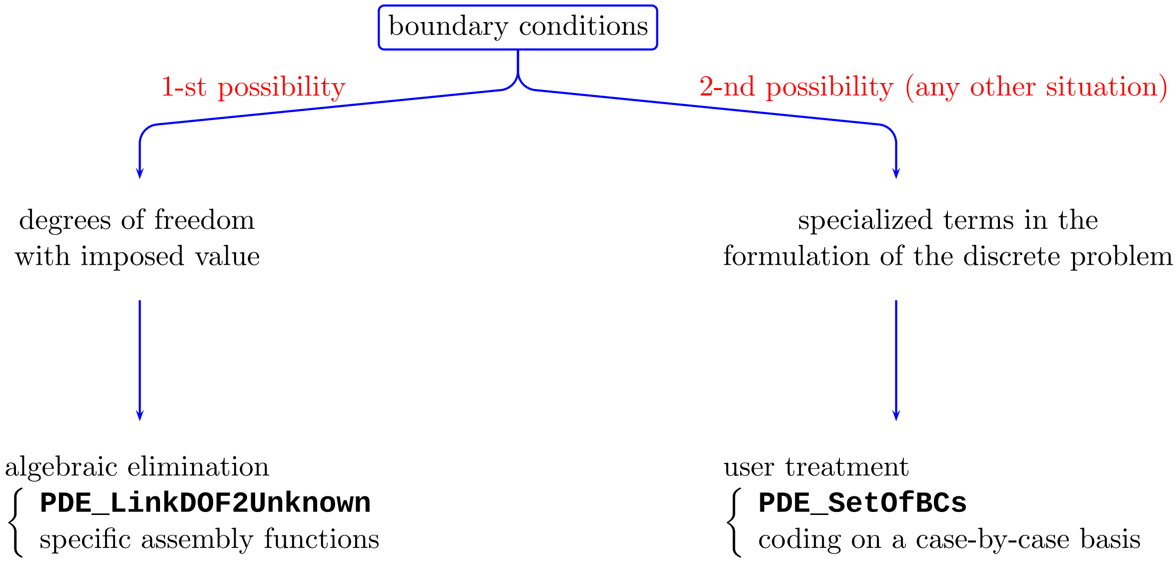
▷ On a given geometrical domain, the whole collection of discrete representations (*ie* the set of instances of **`PDE_DiscreteField`**) is managed by the **`PDE_SetOfDiscreteFields`** class.

▷ Example of services provided by an object of type **`PDE_SetOfDiscreteFields`**, references by the **`sdf`** pointer:

```
cout « "number of fields: " « sdf->nb_fields() « endl ;
if( sdf->has( "my_nice_field" ) )
{
   PDE_DiscreteField const* ff = sdf->item( "my_nice_field" ) ;
   cout « ff->name() « endl ; // "my_nice_field"
}
for( sdf->start() ; sdf->is_valid() ; sdf->go_next() )
{
   PDE_DiscreteField const* ff = sdf->item() ;
   cout « ff->name() « endl ;
}
```

I.4 Boundary Conditions

I.4.1 Taking into Account Boundary Conditions



I.4.2 `PDE_SetOfBCs` Class

▷ Objects of type **`PDE_SetOfBCs`** are sets of **`BC`**s (for **`B`**oundary **`C`**onditions).

▷ In the PELICANS jargon, a **`BC`** is a *hierarchical data structure*, with the PELICANS format, associated to:

- a *color* identifying a cluster of meshes;
- one or all components of an object of type **`PDE_DiscreteField`**.

▷ Example of **BC**:

```
MODULE xxx
   field = "my_nice_field"
   color = "turquoise"
   type = "my_user_bc"
   the_coef = 12.4
   the_string = "roll on the end!"
END MODULE xxx
```

The name and nature of the entries depend on the data of keyword **type**.

▷ Illustration of the use of **PDE_SetOfBCs**:

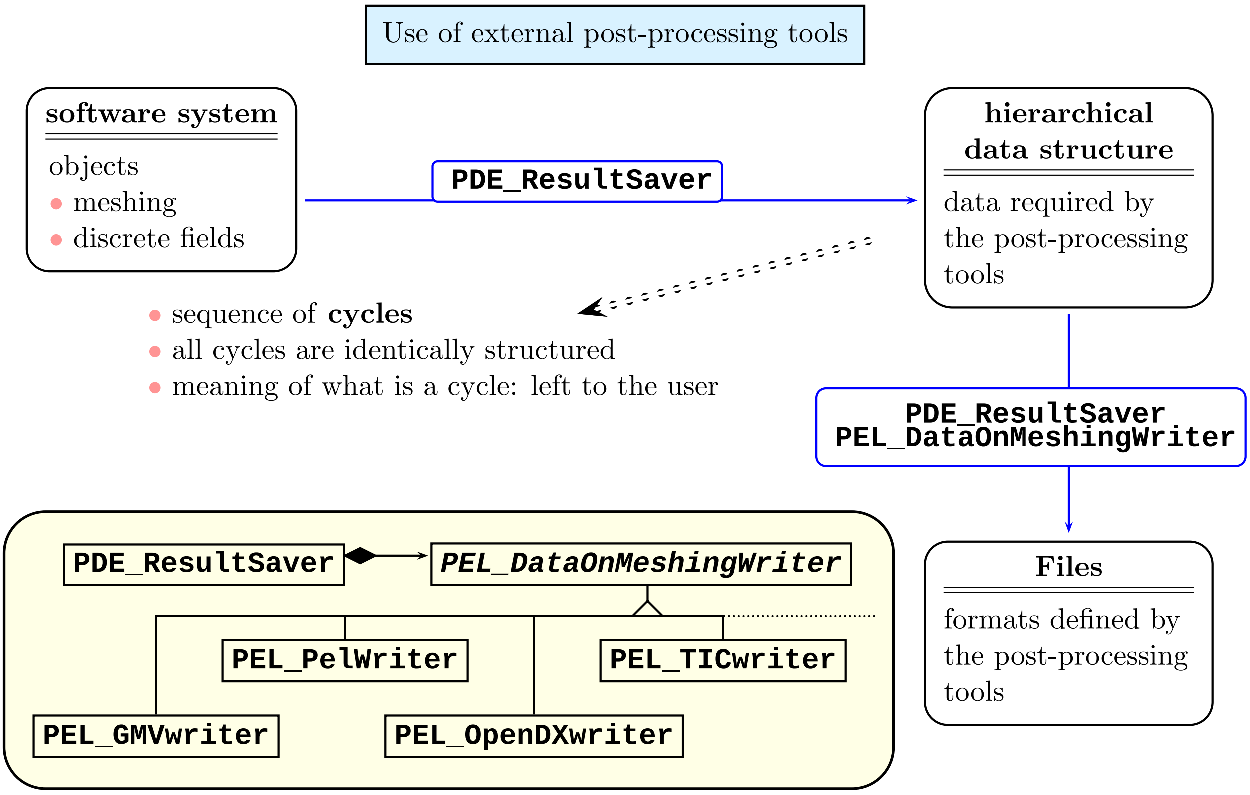
```
MODULE xxx
   field = "my_nice_field"
   color = "turquoise"
   type = "my_user_bc"
   the_coef = 12.4
   the_string = "roll on the end!"
END MODULE xxx
```

object of type **PDE_SetOfBCs**
referenced by the **bcs** pointer

```
GE_Color const* col = GE_Color::object( "turquoise" ) ;
PDE_DiscreteField const* ff = sdf->item( "my_nice_field" ) ;
if( bcs->has_BC( col, ff ) )
{
   PEL_ModuleExplorer const* ee = bcs->BC_explorer( col, ff ) ;
   string const& bc_type = ee->string_data( "type" ) ;
   if( bc_type == "my_user_bc" )
   {
      double xx = ee->double_data( "the_coef" ) ;
      cout « xx « endl ;  // 12.4
      string const& ss = ee->string_data( "the_string" ) ;
      cout « ss « endl ;  // roll on the end!
   }
}
```

I.5 Savings for Post-Processing

I.5.1 Strategy for Post-Processing


Use of external post-processing tools
software system
objects
• meshing
• discrete fields
PDE_ResultSaver
hierarchical data structure
data required by the post-processing tools
• sequence of **cycles**
• all cycles are identically structured
• meaning of what is a cycle: left to the user
PDE_ResultSaver
PEL_DataOnMeshingWriter
PDE_ResultSaver
PEL_DataOnMeshingWriter
PEL_PelWriter
PEL_TICwriter
PEL_GMVwriter
PEL_OpenDXwriter
Files
formats defined by the post-processing tools

I.5.2 Geometric Locations of the Savings

▷ The usual post-processing tools ask for the values of the fields at the **vertices** or at the **cell centers**.

▷ Hence, the discrete representation of the fields must be **interpreted** to compute a **reconstruction** at the geometrical points requested by the post-processing tools.

$$\boxed{1}\quad \underbrace{\left\{\begin{array}{c}\text{unknown field}\\ \mathbf{u}\in X^{\mathbf{u}}\end{array}\right\} \longleftrightarrow \left\{\begin{array}{c}\text{set of numerical values}\\ u^{(\ell)}_{i,i_c}\in\mathbb{R}\end{array}\right\}}_{\begin{array}{c}\textbf{choice}\text{ of a }\textbf{reconstruction}\text{ technique}\\ \text{for the discretization of }\mathbf{u}\end{array}}$$

$\boxed{2}$ It is the value of the reconstruction, associated to the storage level (ℓ), which is computed at the vertices or at the cell centers and subsequently saved.

I.5.3 `PDE_ResultSaver` Class

▷ The services related to the computation of the reconstruction and its saving at the desired geometrical points for subsequent post-processing are provides by **`PDE_ResultSaver`** (with possibility of exceeding them partially).

▷ Example of the services provided by an object of type **PDE_ResultSaver**, referenced by the **saver** pointer:

```
// notification: beginning of a saving cycle
saver->start_cycle() ;

// saving of the meshing (geometrical grid)
saver->save_grid() ;

// saving of the reconstruction value for the chosen fields at the chosen
// points (vertices or cell centers) based on the degrees of freedom
// at 0-th storage level (cf. PDE_DiscreteField)
saver->save_fields( 0 ) ;

// saving of the real number tt with the name TIME
saver->save_variable( tt, "TIME" ) ;

// notification: end of the saving cycle
saver->terminate_cycle() ;
```

I.6 Facade for Discretization Services: PDE_DomainAndFields

The services related to the discretizations of the fields together with the building of the discrete problems are accessible *via* the **PDE_DomainAndFields** class.

Object of type PDE_DomainAndFields

- Any instance of **PDE_DomainAndFields** is associated to a meshing of a geometrical domain $\Omega \subset \mathbb{R}^d$.
- An object of type **PDE_DomainAndFields** is created by an instruction like:

```
PDE_DomainAndField* dom =
                PDE_DomainAndField::create( a_owner, exp ) ;
```

where **exp** is attached to a *hierarchical data structure* that defines:

- a meshing of Ω together with the relevant colors of various geometrical entities;
- the discretization of the unknown fields defined on Ω;
- the boundary conditions other than those describable through imposed values of degrees of freedom;
- the nature of the savings devoted to a subsequent post-processing.

Objects Delivered by PDE_DomainAndFields

The creation of an object of type **PDE_DomainAndFields**, associated to a geometrical domain Ω, triggers internally the creation of a set of fundamental objects that are subsequently placed at the disposal of the clients.

```
PDE_DomainAndFields* dom =
                PDE_DomainAdFields::create( a_owner, exp )
```

Reception of objects created in accordance with the informations of the hierarchical data structure attainable *via* **exp**:

dom->set_of_discrete_fields()
instance of **PDE_SetOfDiscreteFields**

dom->set_of_boundary_conditions()
instance of **PDE_SetOfBCs**

dom->result_saver()
instance of **PDE_ResultSaver**

Chapter II

Finite Element Discretization

II.1 Finite Element Interpolation

II.1.1 A Definition of the Finite Element Method

The starting point of the foregoing developments is a variational approximation of a boundary value problem deduced from problem (1.1) (for example one of the steps of a time semi-discretization) for which the geometrical domain is a open subset of $\mathbb{R}^d$ denoted Ω (with boundary $\Gamma = \overline{\Omega} \setminus \Omega$).

Let $\mathbf{u}\colon \Omega \subset \mathbb{R}^d \to \mathbb{R}^{N_c^{\mathbf{u}}}$ be an *unknown field* (*i.e.* a component of $\mathbf{W}$).

For the sake of writings simplicity, we will use specific letters and typography to denote particular notions. Hence we shall write:

$$\mathbf{u} \in \mathcal{S}^{\mathbf{u}} \subset X^{\mathbf{u}}$$

where:

- $X^{\mathbf{u}}$ is a (infinite dimensional) functionla space formalizing the regularity constraints imposed on $\mathbf{u}$;
- $\mathcal{S}^{\mathbf{u}}$ is a subset of $X^{\mathbf{u}}$ formalizing all or part of the essential (also called Dirichlet) boundary conditions imposed to $\mathbf{u}$. Typically:

  $$\mathcal{S}^{\mathbf{u}} = \left\{ \mathbf{v} \in X^{\mathbf{u}} \;\middle|\; \mathbf{v} = \mathbf{u}_0 \text{ sur } \Gamma_{\mathrm{D}}^{\mathbf{u}} \right\}$$

  where $\mathbf{u}_0$ and $\Gamma_{\mathrm{D}}^{\mathbf{u}} \subset \Gamma$ are given *a priori* and where the restriction of $\mathbf{v}$ to $\Gamma_{\mathrm{D}}^{\mathbf{u}}$ is to be understood in the sense of a trace theorem.

Then we define:

$$\mathcal{V}^{\mathbf{u}} = \left\{ \mathbf{v} \in X^{\mathbf{u}} \;\middle|\; \mathbf{v} = 0 \text{ sur } \Gamma_{\mathrm{D}}^{\mathbf{u}} \right\}$$

called space of test functions associated to $\mathbf{v}$. We shall use the generic notation $\mathbf{v}$ to represent any element of $\mathcal{V}^{\mathbf{u}}$.

in the continuous problem

$\mathbf{u} \in \mathcal{S}^{\mathbf{u}} \subset X^{\mathbf{u}}$

- $X^{\mathbf{u}}$: formalizes the regularity contraints imposed to $\mathbf{u}$
- $\mathcal{S}^{\mathbf{u}}$: formalizes the essential boundary conditions imposed to $\mathbf{u}$.

$$\xrightarrow[\text{interpolation}]{\text{finite element}}$$

in the discrete problem

$\mathbf{u}_h \in \mathcal{S}_h^{\mathbf{u}} \subset X_h^{\mathbf{u}}$

$\mathbf{v}_h \in \mathcal{V}_h^{\mathbf{u}}$

- $\mathbf{u}_h$ approximates $\mathbf{u}$
- $X_h^{\mathbf{u}}$ "approximates" $X^{\mathbf{u}}$
- $\mathcal{S}_h^{\mathbf{u}}$ "approximates" $\mathcal{S}^{\mathbf{u}}$

The discrete approximation of the considered variational problem is derived from finite dimensional spaces $X_h^{\mathbf{u}}$, $\mathcal{S}_h^{\mathbf{u}}$ and $\mathcal{V}_h^{\mathbf{u}}$ (respectively close to $X^{\mathbf{u}}$, $\mathcal{S}^{\mathbf{u}}$ and $\mathcal{V}^{\mathbf{u}}$ in a sense to be defined rigorously) that are built with the finite element method.

The finite element method is a method for constructing finite dimensional spaces that “approximate” a given (infinite dimensional) functional space.

II.1.2 Finite Element Spaces

▷ Starting point: a *meshing* of Ω.

▷ Approximation $X_h^{\mathbf{u}}$ of $X^{\mathbf{u}}$:

$$X_h^{\mathbf{u}} = \operatorname{span}\left\{ \boldsymbol{\varphi}_k^{\mathbf{u}} \;\middle|\; 0 \le k < N_{\mathrm{dof}}^{\mathbf{u}} \right\} \tag{2.1}$$

where the basis functions $\boldsymbol{\varphi}_k^{\mathbf{u}}$ are defined in Ω with values in $\mathbb{R}^{N_{\mathrm{c}}^{\mathbf{u}}}$.

▷ Each function $\boldsymbol{\varphi}_k^{\mathbf{u}}$ has only one of its $N_{\mathrm{c}}^{\mathbf{u}}$ components which is non-zero:

$$\varphi_{n\cdot N_{\mathrm{c}}^{\mathbf{u}}}^{\mathbf{u}}(\mathbf{x}) = \begin{bmatrix} \mathsf{N}_n^{\mathbf{u}}(\mathbf{x}) \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad \varphi_{n\cdot N_{\mathrm{c}}^{\mathbf{u}}+1}^{\mathbf{u}}(\mathbf{x}) = \begin{bmatrix} 0 \\ \mathsf{N}_n^{\mathbf{u}}(\mathbf{x}) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad \varphi_{n\cdot N_{\mathrm{c}}^{\mathbf{u}}+N_{\mathrm{c}}^{\mathbf{u}}-1}^{\mathbf{u}}(\mathbf{x}) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \mathsf{N}_n^{\mathbf{u}}(\mathbf{x}) \end{bmatrix} \tag{2.2}$$

where the $\mathsf{N}_n^{\mathbf{u}} \colon \Omega \to \mathbb{R}$ are called *scalar basis functions* ($n \in [0, N_{\mathrm{node}}^{\mathbf{u}}[$).

▷ Interpolation operator:

$$\begin{tikzcd} \mathbf{u} \in X^{\mathbf{u}} \arrow[r, "\text{finite element}", "\text{interpolation}"'] & \mathbf{u}_h \in X_h^{\mathbf{u}} \end{tikzcd}$$

$$\mathbf{u}_h(\mathbf{x}) = \sum_{0 \le k < N_{\mathrm{dof}}^{\mathbf{u}}} u_k \, \boldsymbol{\varphi}_k^{\mathbf{u}}(\mathbf{x}) \quad \forall \mathbf{x} \in \Omega$$

$$\mathbf{u}_h(\mathbf{x}) = \sum_{0 \le n < N_{\mathrm{node}}^{\mathbf{u}}} \; \sum_{0 \le i_c < N_{\mathrm{c}}^{\mathbf{u}}} u_{n,i_c} \, \mathsf{N}_n^{\mathbf{u}}(\mathbf{x}) \, \mathbf{e}_{i_c} \quad \forall \mathbf{x} \in \Omega$$

$$\mathbf{e}_{i_c} = \text{vecteur à } N_{\mathrm{c}}^{\mathbf{u}} \text{ composantes tel que } \mathbf{e}_{i_c,j} = \delta_{i_c j}$$

▷ The coefficients in these linear combinations, either noted u_k or u_{n,i_c}, are called *degrees of freedom* (**DOF**) of the discretization $\mathbf{u}_h$ of $\mathbf{u}$ (1.2).

▷ The relationship (2.2) between the vectorial basis functions $\boldsymbol{\varphi}_k^{\mathbf{u}}$ and the scalar basis functions $\mathsf{N}_n^{\mathbf{u}}$ makes concrete the notion of *node* together with the re-indexation of *degrees of freedom* in terms of *nodes* and *components* (1.3).

▷ In certain cases the unknown field $\mathbf{u}$ has prescribed value for some of its components on a part of Ω or of its boundary (for example through Dirichlet boundary conditions). This constraint might be translated by *a priori* freezing the value of some degrees of freedom of $\mathbf{u}_h$.

Generally speaking, the set of indices of the degrees of freedom having an imposed value is denoted $\mathbb{I}^{\mathbf{u}}$ (1.4) and we can write:

$$\mathbf{u}_h(\mathbf{x}) = \mathbf{u}_{\mathrm{D}h}(\mathbf{x}) + \sum_{\substack{0 \le k < N_{\mathrm{dof}}^{\mathbf{u}} \\ k \notin \mathbb{I}^{\mathbf{u}}}} u_k \, \boldsymbol{\varphi}_k^{\mathbf{u}}(\mathbf{x}) \qquad \text{avec} \qquad u_{\mathrm{D}h}(\mathbf{x}) \overset{\mathrm{def}}{=} \sum_{\substack{0 \le k < N_{\mathrm{dof}}^{\mathbf{u}} \\ k \in \mathbb{I}^{\mathbf{u}}}} u_{\mathrm{D}k} \, \boldsymbol{\varphi}_k^{\mathbf{u}}(\mathbf{x}) \tag{2.3}$$

where the reals $u_{\mathrm{D}k}$ are the imposed values of the *a priori* frozen degrees of freedom.

▷ The degrees of freedom that are chosen as unknowns of the discrete problem are re-indexed following the relationship (1.5). Les degrés de liberté qui sont choisis comme inconnues du problème discret sont réindexé conformément à la relation (1.5).

Often:

$$\mathbb{U}^{\mathbf{u}} = \begin{cases} [0, N^{\mathbf{u}}_{\mathrm{dof}}[\setminus \mathbb{I}^{\mathbf{u}} \\ [0, N^{\mathbf{u}}_{\mathrm{node}}[\times [0, N^{\mathbf{u}}_{\mathrm{c}}[\setminus \mathbb{I}^{\mathbf{u}} \end{cases}$$

but this not always the case (for instance when the physical domain is only a subset of the meshed domain, or in the case of a discretization using a hierarchical finite element basis). Thus (2.3) will be preferably written under the more general form:

$$\mathbf{u}_h(\mathbf{x}) = \mathbf{u}_{\mathrm{D}h}(\mathbf{x}) + \sum_{\substack{0 \le k < N^{\mathbf{u}}_{\mathrm{dof}} \\ k \in \mathbb{U}^{\mathbf{u}}}} u_k \, \boldsymbol{\varphi}^{\mathbf{u}}_k(\mathbf{x}) \qquad \text{with} \qquad u_{\mathrm{D}h}(\mathbf{x}) \overset{\mathrm{def}}{=} \sum_{\substack{0 \le k < N^{\mathbf{u}}_{\mathrm{dof}} \\ k \in \mathbb{I}^{\mathbf{u}}}} u_{\mathrm{D}k} \, \boldsymbol{\varphi}^{\mathbf{u}}_k(\mathbf{x}) \tag{2.4}$$

▷ Finally, the finite element method produced the following finite dimensional functional spaces:

$$\begin{aligned} X^{\mathbf{u}}_h &= \operatorname{span}\{ \, \boldsymbol{\varphi}^{\mathbf{u}}_k \mid 0 \le k < N^{\mathbf{u}}_{\mathrm{dof}} \, \} \\ \mathcal{S}^{\mathbf{u}}_h &= u_{\mathrm{D}h} + \mathcal{V}^{\mathbf{u}}_h \\ \mathcal{V}^{\mathbf{u}}_h &= \operatorname{span}\{ \, \boldsymbol{\varphi}^{\mathbf{u}}_k \mid 0 \le k < N^{\mathbf{u}}_{\mathrm{dof}} \;\; k \in \mathbb{U}^{\mathbf{u}} \, \} \;=\; \operatorname{span}\{ \, \boldsymbol{\varphi}^{\mathbf{u}}_I \mid 0 \le I < N^{\mathbf{u}}_{\mathrm{unk}} \, \} \end{aligned}$$

$$\boldsymbol{\varphi}^{\mathbf{u}}_k = \mathsf{N}^{\mathbf{u}}_n \, \mathbf{e}_{i_c} \longleftarrow N^{\mathbf{u}}_{\mathrm{c}} \left\updownarrow \begin{bmatrix} 0 \\ | \\ 0 \\ 1 \\ 0 \\ | \\ 0 \end{bmatrix} \right. \leftarrow i_c$$

basis function with $N^{\mathbf{u}}_{\mathrm{c}}$ components

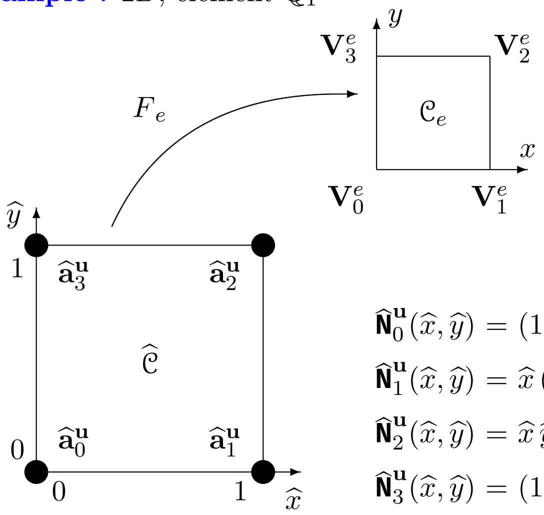
scalar basis function $0 \le n < N^{\mathbf{u}}_{\mathrm{node}}$

II.1.3 Scalar Basis Functions

Each scalar basis function $\mathsf{N}^{\mathbf{u}}_n$

- is associated to a geometrical node $\mathbf{a}^{\mathbf{u}}_n \in \overline{\Omega}$;
- is null outside the cells containing $\mathbf{a}^{\mathbf{u}}_n$;
- is defined *via* its restriction to each **cell**.

Example : 2D, element $\mathbb{Q}_1$


y
$\mathbf{V}^e_3$
$\mathbf{V}^e_2$
F_e
$\mathcal{C}_e$
x
$\mathbf{V}^e_0$
$\mathbf{V}^e_1$
$\widehat{y}$
1
$\widehat{\mathbf{a}}^{\mathbf{u}}_3$
$\widehat{\mathbf{a}}^{\mathbf{u}}_2$
$\widehat{\mathcal{C}}$
0
$\widehat{\mathbf{a}}^{\mathbf{u}}_0$
$\widehat{\mathbf{a}}^{\mathbf{u}}_1$
0
1
$\widehat{x}$

$$F_e(\widehat{x}, \widehat{y}) = \left| \begin{array}{l} \mathbf{V}^e_{0x} + \widehat{x}(\mathbf{V}^e_{1x} - \mathbf{V}^e_{0x}) \\ \mathbf{V}^e_{0y} + \widehat{y}(\mathbf{V}^e_{3y} - \mathbf{V}^e_{0y}) \end{array} \right.$$

$$\begin{aligned} \widehat{\mathsf{N}}^{\mathbf{u}}_0(\widehat{x}, \widehat{y}) &= (1 - \widehat{x})\,(1 - \widehat{y}) \\ \widehat{\mathsf{N}}^{\mathbf{u}}_1(\widehat{x}, \widehat{y}) &= \widehat{x}\,(1 - \widehat{y}) \\ \widehat{\mathsf{N}}^{\mathbf{u}}_2(\widehat{x}, \widehat{y}) &= \widehat{x}\,\widehat{y} \\ \widehat{\mathsf{N}}^{\mathbf{u}}_3(\widehat{x}, \widehat{y}) &= (1 - \widehat{x})\,\widehat{y} \end{aligned}$$

$\mathsf{N}^{\mathbf{u}}_n$ est définie par :

$$\forall \mathcal{C}_e \quad \left\{ \begin{array}{l} \text{si } \mathbf{a}^{\mathbf{u}}_n \in \mathcal{C}_e \\ \quad \mathsf{N}^{\mathbf{u}}_n|_{\mathcal{C}_e} = \widehat{\mathsf{N}}^{\mathbf{u}}_i \circ F_e^{-1} \\ \text{sinon} \\ \quad \mathsf{N}^{\mathbf{u}}_n|_{\mathcal{C}_e} = 0 \end{array} \right.$$

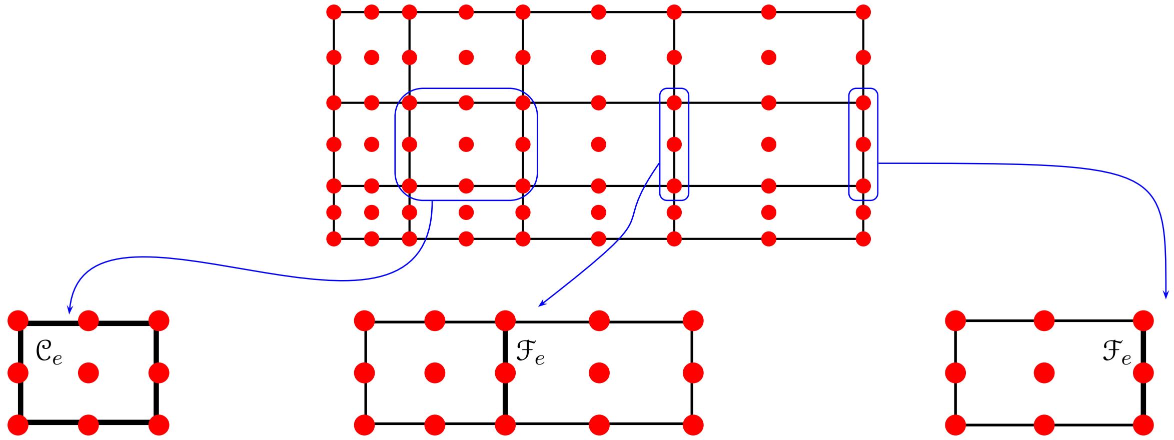
II.1.4 Numberings

▷ The nodes are then numbered locally in each mesh K_e:

$$n \in [0, N^{\mathbf{u}}_{\text{node}}[\quad \text{tq} \quad \text{support}(\mathbf{N}^{\mathbf{u}}_n) \cap K_e \neq \emptyset \quad \xrightarrow[\text{indexing}]{\text{local in } K_e} \quad i \in [0, N^{\mathbf{u}}_e[\tag{2.5}$$

The notation i is clearly incomplete since it does not refer to the considered mesh K_e or node n. Any ambiguity, however, will be resolved by the context.

Example : 2D, élément $\mathbb{Q}_2$



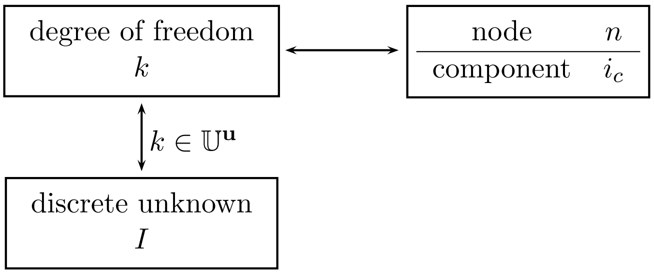
$N^{\mathbf{u}}_e = 9$
$K_e \equiv$ cell $\mathcal{C}_e$

$N^{\mathbf{u}}_e = 15$
$K_e \equiv$ face $\mathcal{F}_e$

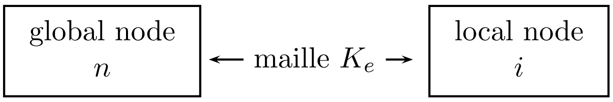
$N^{\mathbf{u}}_e = 9$
$K_e \equiv$ boundary face $\mathcal{F}_e$

▷ One of the practical difficulties induced by the finite element discretizations relates to the multiple numberings (for example, local, global node index, index of the related unknowns, *etc.*). This difficulty is managed entirely by PELICANS, so that the user is almost never confronted with it.

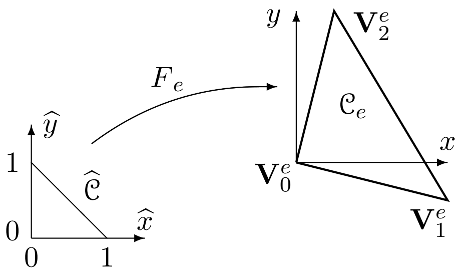
▷ In order to lighten notations as much as possible, we decided the reserve a specific index to each of the numberings defined by the relations (1.2, 2.1), (1.3), (2.5), (1.5):



$$\varphi^{\mathbf{u}}_I = \varphi^{\mathbf{u}}_k = \mathbf{N}^{\mathbf{u}}_n \mathbf{e}_{i_c}$$



$$\mathbf{N}^{\mathbf{u}}_n = \mathbf{N}^{\mathbf{u}}_i$$



$$\left\{ \begin{array}{l} \mathbf{N}^{\mathbf{u}}_i|_{\mathcal{C}_e} = \widehat{\mathbf{N}}^{\mathbf{u}}_i \circ F_e^{-1} \\ \mathcal{C}_e \quad \text{such that} \quad \mathbf{a}^{\mathbf{u}}_n \in \mathcal{C}_e \end{array} \right.$$

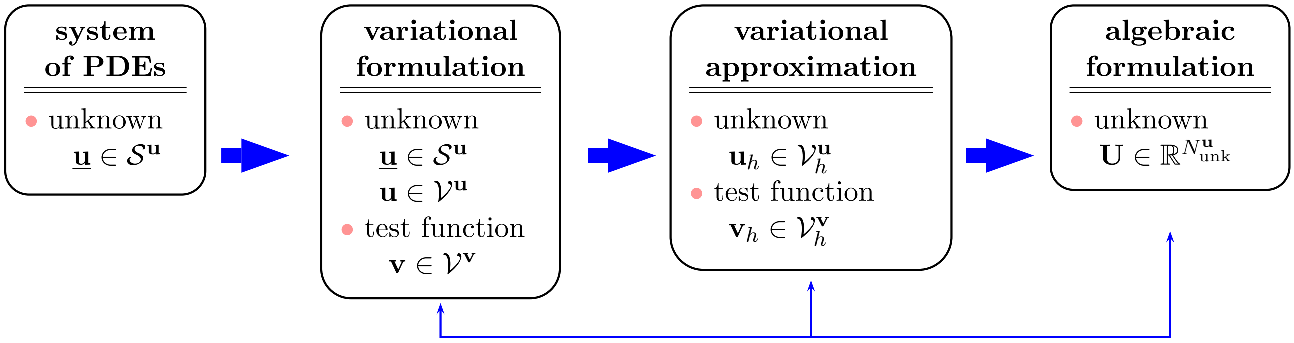
II.2 Variational Approximation

II.2.1 Algebraic Formulation

▷ A *test function* $\mathbf{v}$ is always associated to an *unknown field*, denoted here $\mathbf{u}_{\mathbf{v}}$, so that $\mathbf{u}_{\mathbf{v}} \in \mathcal{S}^{\mathbf{u}_{\mathbf{v}}} \subset X^{\mathbf{u}_{\mathbf{v}}}$ and $v \in \mathcal{V}^{\mathbf{u}_{\mathbf{v}}}$.

As soon as it does not generate confusion, the writings will be lightened by using the notation v instead of $\mathbf{u}_{\mathbf{v}}$. Thus, for example, $\mathcal{V}^{\mathbf{v}}$, $X^{\mathbf{v}}$, $N^{\mathbf{v}}_{\mathrm{c}}$, $N^{\mathbf{v}}_{\mathrm{dof}}$, $N^{\mathbf{v}}_{\mathrm{node}}$, $\mathbb{I}^{\mathbf{v}}$, $\mathbb{U}^{\mathbf{v}}$, $N^{\mathbf{v}}_{\mathrm{unk}}$ should be understood as $\mathcal{V}^{\mathbf{u}_{\mathbf{v}}}$, $X^{\mathbf{u}_{\mathbf{v}}}$, $N^{\mathbf{u}_{\mathbf{v}}}_{\mathrm{c}}$, $N^{\mathbf{u}_{\mathbf{v}}}_{\mathrm{dof}}$, $N^{\mathbf{u}_{\mathbf{v}}}_{\mathrm{node}}$, $\mathbb{U}^{\mathbf{u}_{\mathbf{v}}}$, $\mathbb{I}^{\mathbf{u}_{\mathbf{v}}}$, $N^{\mathbf{u}_{\mathbf{v}}}_{\mathrm{unk}}$.

▷ From the PDE continuous to an algebraic problem:


system of PDEs
• unknown $\underline{\mathbf{u}} \in \mathcal{S}^{\mathbf{u}}$
variational formulation
• unknown $\underline{\mathbf{u}} \in \mathcal{S}^{\mathbf{u}}$ $\mathbf{u} \in \mathcal{V}^{\mathbf{u}}$
• test function $\mathbf{v} \in \mathcal{V}^{\mathbf{v}}$
variational approximation
• unknown $\mathbf{u}_h \in \mathcal{V}^{\mathbf{u}}_h$
• test function $\mathbf{v}_h \in \mathcal{V}^{\mathbf{v}}_h$
algebraic formulation
• unknown $\mathbf{U} \in \mathbb{R}^{N^{\mathbf{u}}_{\mathrm{unk}}}$

$$\mathbf{u} \simeq \mathbf{u}_h = \sum_{0 \le I < N^{\mathbf{u}}_{\mathrm{unk}}} \mathbf{U}_I \, \boldsymbol{\varphi}^{\mathbf{u}}_I$$

▷ The algebraic formulation contains terms such that:

$$\int_{K_e} \mathcal{A}(\boldsymbol{\varphi}^{\mathbf{u}}_J, \boldsymbol{\varphi}^{\mathbf{v}}_I)\, dK_e \qquad 0 \le I < N^{\mathbf{v}}_{\mathrm{unk}} \qquad 0 \le J < N^{\mathbf{u}}_{\mathrm{unk}} \qquad \mathcal{A}(\boldsymbol{\varphi}^{\mathbf{u}}_J, \boldsymbol{\varphi}^{\mathbf{v}}_I) \in \mathbb{R} \tag{2.6}$$

$$\int_{K_e} \mathcal{F}(\boldsymbol{\varphi}^{\mathbf{v}}_I)\, dK_e \qquad 0 \le I < N^{\mathbf{v}}_{\mathrm{unk}} \qquad \mathcal{F}(\boldsymbol{\varphi}^{\mathbf{v}}_I) \in \mathbb{R} \tag{2.7}$$

where

- $\mathcal{A}$, $\mathcal{F}$ are real valued function determined by the PDE system;
- dK_e is the volume element if K_e is a *cell* and the surface element if K_e is a *face*.

II.2.2 Assembly

▷ When the index e takes all its possible values, the preceding terms generate respectively a matrix $\mathbf{A}$ and a vector $\mathbf{F}$:

elementary contribution of a mesh K_e **assemblage**

$$\begin{bmatrix} \displaystyle\int_{K_e} \mathcal{A}(\boldsymbol{\varphi}^{\mathbf{u}}_J, \boldsymbol{\varphi}^{\mathbf{v}}_I)\, dK_e \\ \displaystyle\int_{K_e} \mathcal{F}(\boldsymbol{\varphi}^{\mathbf{v}}_I)\, dK_e \end{bmatrix} \xrightarrow{e \text{ takes all its possible values}} \begin{bmatrix} \mathbf{A} \\ \mathbf{F} \end{bmatrix} = \sum_{\text{meshes } K_e} \mathsf{A} \begin{bmatrix} a^e \\ f^e \end{bmatrix}$$

▷ The elementary contributions, using local numbering, are given by:

$$\begin{aligned}
&a^e = \big[\, a^e(\, i, i_c; j, j_c\,)\,\big] \qquad f^e = \big[\, f^e(\, i, i_c)\,\big] \\
&0 \le i < N_e^{\mathbf{v}} \quad 0 \le i_c < N_{\mathrm{c}}^{\mathbf{v}} \qquad 0 \le j < N_e^{\mathbf{u}} \quad 0 \le j_c < N_{\mathrm{c}}^{\mathbf{u}} \\
&a^e(\, i, i_c; j, j_c\,) = \int_{K_e} \mathcal{A}(\mathsf{N}_j^{\mathbf{u}}\mathbf{e}_{j_c}, \mathsf{N}_i^{\mathbf{v}}\mathbf{e}_{i_c})\; dK_e \\
&f^e(\, i, i_c\,) = \int_{K_e} \mathcal{F}(\mathsf{N}_i^{\mathbf{v}}\mathbf{e}_{i_c})\; dK_e
\end{aligned}$$

▷ $\boldsymbol{\mathsf{A}}$ is an assembly operator that adds the elementary contributions at the appropriate location in the matrix $\mathbf{A}$ and in the vector $\mathbf{F}$ while taking care to eliminate, in the unknowns of the global system, the degrees of freedom with index not belonging to $\mathbb{U}^{\mathbf{u}}$.

In the foregoing, three variants $\boldsymbol{\mathsf{A}}^{\mathsf{vec}}$, $\boldsymbol{\mathsf{A}}^{\mathsf{mat}}$ and $\boldsymbol{\mathsf{A}}_{\mathsf{vec}}^{\mathsf{mat}}$ of $\boldsymbol{\mathsf{A}}$ will be defined.

▷ The elimination, at assembly time, of the *a priori* imposed degrees of freedom is the practical counterpart of the technique of change of unknown function, which is applied when deriving the weak form of a boundary value problem by substraction a lifting of the Dirichlet boundary conditions to the original unknown.

II.2.3 Numerical Integration

▷ The integrals over the meshes K_e intervening the the expressions of $a^e(i, i_c; j, j_c)$ and $f^e(i, i_c)$ are approximated numerically:

$$\int_{K_e} \square \; dK_e \;\simeq\; \sum_p \; w_p \,\square(\mathbf{x}_p)$$

where the pairs $(\mathbf{x}_p, w_p)$, with $\mathbf{x}_p \in \overline{K}_e$ and $w_p \in \mathbb{R}$, define a *quadrature rule* on the mesh K_e.

Then:

$$a^e(i, i_c; j, j_c) = \sum_p a_p^e(i, i_c; j, j_c) \qquad a_p^e(i, i_c; j, j_c) = w_p \mathcal{A}(\mathsf{N}_j^{\mathbf{u}}\mathbf{e}_{j_c}, \mathsf{N}_i^{\mathbf{v}}\mathbf{e}_{i_c})(\mathbf{x}_p) \tag{2.8}$$

$$f^e(i, i_c) = \sum_p f_p^e(i, i_c) \qquad f_p^e(i, i_c) = w_p \mathcal{F}(\mathsf{N}_i^{\mathbf{v}}\mathbf{e}_{i_c})(\mathbf{x}_p) \tag{2.9}$$

▷ The *quadrature rules* are determine theoretically to be exact when applied to some classes of polynomial functions.

▷ The choice of a *quadrature rule* is essential and left to the user.

II.2.4 Fundamental Classes

desired services	related classes
loop on a set of meshes $\{K_e\}_e$	**`PDE_LocalFE`** and derivatives
loop on integration points $\{\mathbf{x}_p\}_p \subset K_e$	**`PDE_LocalFE`** and derivatives **`GE_QRprovider`**
calculation of the elementary contributions at $\mathbf{x}_p \in K_e$: $a_p^e(i,i_c;j,j_c) = w_p \mathcal{A}(\mathbf{N}_j^{\mathbf{u}} \mathbf{e}_{j_c}, \mathbf{N}_i^{\mathbf{v}} \mathbf{e}_{i_c})(\mathbf{x}_p)$ $f_p^e(i,i_c) = w_p \mathcal{F}(\mathbf{N}_i^{\mathbf{v}} \mathbf{e}_{i_c})(\mathbf{x}_p)$	**`PDE_LocalFE`** and derivatives
storage of the elementary contributions related to K_e : $\begin{bmatrix} a^e(i,i_c;j,j_c) \\ f^e(i,i_c) \end{bmatrix}$	**`PDE_LocalEquation`**
assembly : $\begin{bmatrix} a^e(i,i_c;j,j_c) \\ f^e(i,i_c) \end{bmatrix} \rightsquigarrow \begin{bmatrix} \mathbf{A}_{IJ} \\ \mathbf{F}_I \end{bmatrix}$	**`PDE_LocalEquation`** **`PDE_LinkDOF2Unknown`** **`PDE_AssembledSystem`**

II.3 Mesh Local System

II.3.1 PDE_LocalEquation Class

The **PDE_LocalEquation** class provides the services for the management of a system $[a^e, f^e]$, related to a given mesh K_e, where:

$$\begin{array}{ll} a^e = \big[a^e(\,i, i_c; j, j_c\,) \big] & f^e = \big[f^e(\,i, i_c\,) \big] \\ 0 \le i < N_e^{\mathbf{v}} \quad 0 \le i_c < N_\mathrm{c}^{\mathbf{v}} & 0 \le j < N_e^{\mathbf{u}} \quad 0 \le j_c < N_\mathrm{c}^{\mathbf{u}} \end{array} \tag{2.10}$$

Object of type PDE_LocalEquation

An objet of type **PDE_LocalEquation**

- is created by an instruction such as:

```
PDE_LocalEquation* elm_eq = PDE_LocalEquation::create( a_owner ) ;
```

- is dimensioned by calling the *command*:

```
elm_eq->initialize( row_nodes, nb_row_cmps,
                    col_nodes, nb_col_cmps ) ;
```

where

- **row_nodes** and **col_nodes** are of type **size_t_vector** and such that
  row_nodes.size() $= N_e^{\mathbf{v}}$ **row_nodes(**i**)** $= n_i$ (global node number)
  col_nodes.size() $= N_e^{\mathbf{u}}$ **col_nodes(**j**)** $= n_j$ (local node number)
- **nb_row_cmps** $= N_\mathrm{c}^{\mathbf{v}}$ **nb_col_cmps** $= N_\mathrm{c}^{\mathbf{u}}$

Some Member Functions of PDE_LocalEquation

Let **elm_eq** be of type **PDE_LocalEquation***, attached to a local system $[a^e, f^e]$ (2.10):

▷ Some *Queries*:

expression	evaluation result
`elm_eq->nb_rows()`	$N_e^{\mathbf{v}}$
`elm_eq->nb_row_sub_indices()`	$N_\mathrm{c}^{\mathbf{v}}$
`elm_eq->row_node( i )`	global number of node $i =$ **i** (of **v**)
`elm_eq->nb_columns()`	$N_e^{\mathbf{u}}$
`elm_eq->nb_column_sub_indices()`	$N_\mathrm{c}^{\mathbf{u}}$
`elm_eq->column_node( j )`	global number of node $j =$ **j** (of **u**)
`elm_eq->matrix_item( i, j, ic, jc )`	$a^e(\,i, i_c; j, j_c\,)$ $i =$ **i** $i_c =$ **ic** $j =$ **j** $j_c =$ **jc**
`elm_eq->vector_item( i, ic )`	$f^e(\,i, i_c\,)$ $i =$ **i** $i_c =$ **ic**

▷ Some *Commands*:

expression	side effect of the evaluation
`elm_eq->add_to_matrix( x, i, j, ic, jc)`	$a^e(\,i, i_c; j, j_c\,) \mathrel{+}=$ **x** $i =$ **i** $i_c =$ **ic** $j =$ **j** $j_c =$ **jc**
`elm_eq->add_to_vector( x, i, ic )`	$f^e(\,i, i_c\,) \mathrel{+}=$ **x** $i =$ **i** $i_c =$ **ic**

II.4 Global Algebraic System

II.4.1 System of Algebraic Equations

Discrete Problem

a **set** of **systems of algebraic equations** to be solved in **sequence**

Recommendation

- For **each** *system of algebraic equations*, implement two classes:
  - “discretizers”
    - capable of calculating the elementary contributions
    - capable of managing the update for the discrete representation of the unknown fields
  - systems of algebraic equations
    - that may be assembled from elementary contributions
    - that may be solved
- Represent each kind of system of algebraic equation by a class whose implementation uses derivatives of **PDE_AssembledSystem**

II.4.2 **PDE_AssembledSystem** Class

The **PDE_AssembledSystem** class implements three assembly functions that add the elementary contributions at the appropriate location in the system of algebraic equations $[\mathbf{A};\mathbf{F}]$

```
class PDE_AssembledSystem : public PEL_Object
{
   public: //-------------------------------------

   protected: //----------------------------------
      virtual ~PDE_AssembledSystem( void ) ;
      PDE_AssembledSystem( PEL_Object* a_owner ) ;

      void assemble_in_vector( PDE_LocalEquation const* leq,
                               PDE_LinkDOF2Unknown const* r_link,
                               LA_Vector* vec ) const ;

      void assemble_in_matrix( PDE_LocalEquation const* leq,
                               PDE_LinkDOF2Unknown const* r_link,
                               PDE_LinkDOF2Unknown const* c_link,
                               LA_SparseMatrix* mat ) const ;

      void assemble_in_matrix_vector( PDE_LocalEquation const* leq,
                                      PDE_LinkDOF2Unknown const* r_link,
                                      PDE_LinkDOF2Unknown const* c_link,
                                      LA_SparseMatrix* mat,
                                      LA_Vector* vec ) const ;
} ;
```

`assemble_in_vector( leq, r_link, vec )`

`leq` ⇒	$a^e = \left[\, a^e(\, i, i_c; j, j_c\,)\,\right] \qquad f^e = \left[\, f^e(\, i, i_c)\,\right]$ $0 \le i < N_e^{\mathbf{v}} \quad 0 \le i_c < N_{\mathrm{c}}^{\mathbf{v}} \quad 0 \le j < N_e^{\mathbf{u}} \quad 0 \le j_c < N_{\mathrm{c}}^{\mathbf{u}}$
`r_link` ⇒	$0 \le i < N_e^{\mathbf{v}} \quad 0 \le i_c < N_{\mathrm{c}}^{\mathbf{v}} \quad (i, i_c) \in \mathbb{U}^{\mathbf{v}} \quad \longleftrightarrow \quad 0 \le I < N_{\mathrm{unk}}^{\mathbf{v}}$
`vec` ⇒	$\mathbf{F} = \left[\, \mathbf{F}_I \,\right] \quad 0 \le I < N_{\mathrm{unk}}^{\mathbf{v}}$

$$\begin{bmatrix} \mathbf{A} \\ \mathbf{F} \end{bmatrix} = \sum_{\text{meshes } K_e} \boldsymbol{\mathsf{A}}^{\mathsf{vec}} \begin{bmatrix} a^e \\ f^e \end{bmatrix}$$

on a given mesh K_e:

```
For each $\mathsf{N}_i^{\mathbf{v}}$ tq $(i, i_c) \in \mathbb{U}^{\mathbf{v}}$
    $\mathbf{F}_I = \mathbf{F}_I + f^e(\, i, i_c\,)$
```

`assemble_in_matrix( leq, r_link, c_link, mat )`

`leq` ⇒	$a^e = \left[\, a^e(\, i, i_c; j, j_c\,)\,\right] \qquad f^e = \left[\, f^e(\, i, i_c)\,\right]$ $0 \le i < N_e^{\mathbf{v}} \quad 0 \le i_c < N_{\mathrm{c}}^{\mathbf{v}} \quad 0 \le j < N_e^{\mathbf{u}} \quad 0 \le j_c < N_{\mathrm{c}}^{\mathbf{u}}$
`r_link` ⇒	$0 \le i < N_e^{\mathbf{v}} \quad 0 \le i_c < N_{\mathrm{c}}^{\mathbf{v}} \quad (i, i_c) \in \mathbb{U}^{\mathbf{v}} \quad \longleftrightarrow \quad 0 \le I < N_{\mathrm{unk}}^{\mathbf{v}}$
`c_link` ⇒	$0 \le j < N_e^{\mathbf{u}} \quad 0 \le j_c < N_{\mathrm{c}}^{\mathbf{u}} \quad (j, j_c) \in \mathbb{U}^{\mathbf{u}} \quad \longleftrightarrow \quad 0 \le J < N_{\mathrm{unk}}^{\mathbf{u}}$
`mat` ⇒	$\mathbf{A} = \left[\, \mathbf{A}_{IJ} \,\right] \quad 0 \le I < N_{\mathrm{unk}}^{\mathbf{v}} \quad 0 \le J < N_{\mathrm{unk}}^{\mathbf{u}}$

$$\begin{bmatrix} \mathbf{A} \\ \mathbf{F} \end{bmatrix} = \sum_{\text{meshes } K_e} \boldsymbol{\mathsf{A}}^{\mathsf{mat}} \begin{bmatrix} a^e \\ f^e \end{bmatrix}$$

on a given mesh K_e:

```
For each $\mathsf{N}_i^{\mathbf{v}}$ tq $(i, i_c) \in \mathbb{U}^{\mathbf{v}}$
    For each $\mathsf{N}_j^{\mathbf{u}}$
        if $(j, j_c) \in \mathbb{U}^{\mathbf{u}}$ then
            $\mathbf{A}_{IJ} = \mathbf{A}_{IJ} + a^e(\, i, i_c, j, j_c\,)$
```

assemble_in_matrix_vector(leq, r_link, c_link, mat, vec)

leq $\Rightarrow$	$a^e = \big[\, a^e(\,i,i_c;j,j_c\,)\big] \qquad f^e = \big[\, f^e(\,i,i_c)\big]$ $0 \le i < N_e^{\mathbf{v}} \quad 0 \le i_c < N_{\mathrm{c}}^{\mathbf{v}} \quad 0 \le j < N_e^{\mathbf{u}} \quad 0 \le j_c < N_{\mathrm{c}}^{\mathbf{u}}$
r_link $\Rightarrow$	$0 \le i < N_e^{\mathbf{v}} \quad 0 \le i_c < N_{\mathrm{c}}^{\mathbf{v}} \quad (i,i_c) \in \mathbb{U}^{\mathbf{v}} \quad \longleftrightarrow \quad 0 \le I < N_{\mathrm{unk}}^{\mathbf{v}}$
c_link $\Rightarrow$	$0 \le j < N_e^{\mathbf{u}} \quad 0 \le j_c < N_{\mathrm{c}}^{\mathbf{u}} \quad (j,j_c) \in \mathbb{U}^{\mathbf{u}} \quad \longleftrightarrow \quad 0 \le J < N_{\mathrm{unk}}^{\mathbf{u}}$
mat, **vec** $\Rightarrow$	$\mathbf{A},\mathbf{F} = \big[\,\mathbf{A}_{IJ}\big],\big[\,\mathbf{F}_I\,\big] \quad 0 \le I < N_{\mathrm{unk}}^{\mathbf{v}} \quad 0 \le J < N_{\mathrm{unk}}^{\mathbf{u}}$

$$\begin{bmatrix}\mathbf{A}\\ \mathbf{F}\end{bmatrix} = \sum_{\text{meshes } K_e} \mathsf{A}_{\mathsf{vec}}^{\mathsf{mat}} \begin{bmatrix} a^e \\ f^e \end{bmatrix}$$

on a given mesh K_e:

```
For each $\mathsf{N}_i^{\mathbf{v}}$ tq $(i,i_c) \in \mathbb{U}^{\mathbf{v}}$
    $\mathbf{F}_I = \mathbf{F}_I + f^e(\,i\,)$
    For each $\mathsf{N}_j^{\mathbf{u}}$
        if $(j,j_c) \in \mathbb{U}^{\mathbf{u}}$ then
            $\mathbf{A}_{IJ} = \mathbf{A}_{IJ} + a^e(\,i,i_c,j,j_c\,)$
        else if $(j,j_c) \in \mathbb{I}^{\mathbf{u}}$
            $\mathbf{F}_I = \mathbf{F}_I - a^e(\,i,i_c,j,j_c\,)\,u_{\mathrm{D}jj_c}$
```

II.5 Numerical Integration

II.5.1 Quadrature Rules

▷ A *quadrature rule* on a mesh K_e is a familly of pairs $(\mathbf{x}_p, w_p)_p$ that allow for approximating integrals over K_e:

$$\int_{K_e} \square \, dK_e \simeq \sum_p w_p \, \square(\mathbf{x}_p)$$

where $\mathbf{x}_p \in \overline{K}_e$ and $w_p \in \mathbb{R}$.

▷ On a *mesh* K_e, the value of the pairs $(\mathbf{x}_p, w_p)_p$ of a *quadrature rule* is deduced from a *quadrature rule* $(\widehat{\mathbf{x}}_p, \widehat{w}_p)_p$ on the *reference polyhedron* $\widehat{K}_e$ and from the mapping $F_e \colon \widehat{K}_e \to K_e$.

▷ The practical details of the correspondence $\begin{bmatrix} (\widehat{\mathbf{x}}_p, \widehat{w}_p) \\ F_e \end{bmatrix} \rightsquigarrow (\mathbf{x}_p, w_p)$ are handled by PELICANS.

▷ The construction of the *discrete problem* involves integral over several kinds of *meshes* (*eg* cells, faces) that are related to several kinds of *reference polyhedra*.

▷ The choice of the *quadrature rules*, for each involved kind of mesh, is an important aspect of the numerical scheme.

II.5.2 **GE_QRprovider** and **GE_QuadratureRule** Classes

▷ A *quadrature rule* on a *reference polyhedron* is represented by a class derived from **GE_QuadratureRule**.

▷ The choice of quadrature rules, for each involved kind of mesh, is expressed by the use of an object of a concrete subclass, that represents the performed selection, through a pointer to the abstract parent class **GE_QRprovider**, which represents all the possible variants of that choice.

▷ The **GE_QRprovider** class

- describes providers of *quadrature rules* (*i.e.* of **GE_QuadratureRule** instances) for all the *reference polyhedra*;
- is an abstract class that formalizes the interface used by **PDE_LocalFE** and its descendants to bring into play an iteration on the the integration points of a given mesh.

▷ What exactly is the class **GE_QRprovider** and the relationship that she maintains with **GE_QuadratureRule** does not matter much. Selecting an object of a concrete class derived from **GE_QRprovider** comes down to selecting a **familly** of *quadrature rules* (one for each kind of *reference polyhedron*) that will be used by the descendants of **PDE_LocalFE**: such is the significant information.

▷ The classes **GE_QuadratureRule** and **GE_QRprovider** are *plug-points* of the PELICANS *framework*.

II.5.3 Objects of Classes Derived From **GE_QRprovider**

▷ The numerical schemes often require *quadrature rules* that are exact for some classes of polynomial functions (*eg* polynomials of degree lower than a given value):

$$\forall P \in \Phi \quad \int_{\widehat{K}} P - \sum_p \widehat{w}_p P(\widehat{\mathbf{x}}_p) = 0 \quad \text{where } \Phi \text{ est some space of polynomials}$$

PELICANS implements classes derived from **GE_QuadratureRule** and **GE_QRprovider** meant for this purpose.

▷ The objects of classes derived from **`GE_QRprovider`**, refered to *via* pointers to the parent class, are delivered by instructions such that:

```
GE_QRprovider const* qrp = GE_QRprovider::object( a_name ) ;
```

where **`a_name`** is the name of a concrete class derived from **`GE_QRprovider`**.

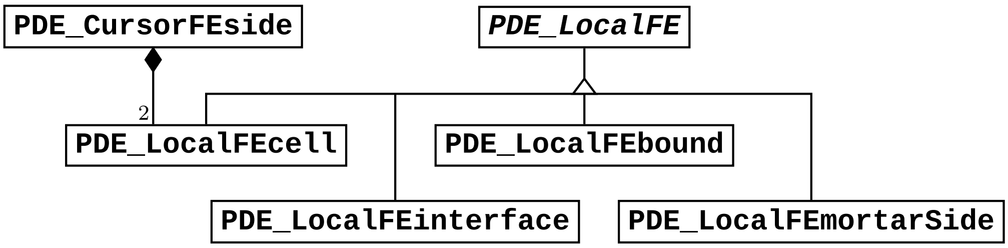
Example:

`a_name`	**`"GE_QRprovider_3"`**	**`"GE_QRprovider_5"`**	**`"GE_QRprovider_7"`**
Φ	$\mathbb{P}_3$	$\mathbb{P}_5$	$\mathbb{P}_7$

Remark: in fact, **`a_name`** is the registration name of the singleton of the concrete class derived from **`GE_QRprovider`** according to the *plug-in* mecanism of PELICANS.

II.6 **PDE_LocalFE** Class and Derivatives

The **PDE_LocalFE** class and its derivated are fundamental classes fot the **building** of *discrete problems* based on *discrete representations* of the fields with *finite elements*.



II.6.1 Iteration on Meshes

▷ Let $\Omega \subset \mathbb{R}^d$ be a meshed geometrical domain.

PDE_LocalFE

- is associated to a kind of *mesh* (*cell*, *inner face*, *boudary face*)
- defines a collection of those *meshes*
- provides the conceptual services of a *mesh iterator* to traverse through all the *meshes* of that collection

PDE_LocalFEcell

- *mesh* ≡ *cell*
- collection of *meshes* ≡ set of all the *cells* of the meshing of Ω
  (one may restrict the collection by ignoring *cells* with specified *colors*)

PDE_LocalFEbound

- *mesh* ≡ *boundary face*
- collection of *meshes* ≡ set of all the *boundary faces* of the meshing of Ω
  (one may restrict the collection by ignoring *boundary faces* with specified *colors*)

▷ Given an object of type **PDE_DomainAndFields**, referenced by the pointer **dom**, associated to a meshing of a geometrical domain Ω:

- creation of an object of type **PDE_LocalFEcell** for an iteration on the *cells* of the meshing:

```
PDE_LocalFEcell* fe = dom->create_LocalFEcell( a_owner ) ;
```

- creation of an object of type **PDE_LocalFEbound**, for an iteration on the *boundary faces* of the meshing:

```
PDE_LocalFEbound* fe = dom->create_LocalFEbound( a_owner ) ;
```

▷ Let **fe** be a pointer to a **PDE_LocalFE** object associated to a meshing of the geometrical domain $\Omega \subset \mathbb{R}^d$.

expression	result/side effect of the evaluation
fe->start()	move the *mesh iterator* to the first position
fe->is_valid()	**true** if the *mesh iterator* is effectively positioned on a *mesh* (called the *current mesh*)
fe->go_next()	move the *mesh iterator* to the next mesh
fe->polyhedron()	pointer on the object of type **GE_Mpolyhedron** representing the *current mesh*
fe->color()	pointer on the *color* of the *current mesh*
fe->mesh_id()	*identifier* of the *current mesh*
fe->go_i_th(i)	move the *mesh iterator* to the *mesh* of *identifier* **i**
fe->outward_normal()	if **fe** actually refers to an object of type **PDE_LocalFEbound**: pointer to the objet of type **GE_Vector** that represents the unit outward normal of the *current boundary face*

II.6.2 Calculation Requirements

Configuration

Let **fe** be a pointer to a **PDE_LocalFE** object associated to a meshing of the geometrical domain $\Omega \subset \mathbb{R}^d$.

An object of type **PDE_LocalFE** object associated to a meshing of the geometrical domain $\Omega \subset \mathbb{R}^d$

- will give access to the *discrete representation* of the *unknown fields* locally to the *current mesh* of the *mesh iterator*.
- **must** be configured **prior** to the iteration over the meshes, in order to be knowledgeable of:
  - the *unknown fields* to take into account;
  - the desired informations concerning their *discrete representation*.

Tools

- *Command* for the configuration of **PDE_LocalFE**:

```
void require_field_calculation( PDE_DiscreteField const* ff,
                                int order ) ;
```

- **PDE_LocalFE** defined integer constants (**int const**) to be used as second argument when calling **require_field_calculation**:

	meaning
PDE_LocalFE::N	0-th order derivative
PDE_LocalFE::dN	1-st order derivative
PDE_LocalFE::d2N	2-nd order derivative

Actors with their Prerequisites

- an object of type **PDE_DiscreteField**, referenced by the pointer **uu** and associated to the field $u_h \in \mathcal{V}_h^{\mathbf{u}} = \text{span}\left\{\mathbf{N}_i^{\mathbf{u}}\,\mathbf{e}_{i_c}\right\}$
- an object of type **PDE_LocalFE**, referenced by the pointer **fe**, which is not currently iterating over the meshes

Notification of the Requirements

- **fe->require_field_calculation(uu, PDE_LocalFE::N)**

  configures ***fe** to allow for accessing the values of the basis functions $\mathbf{N}_i^{\mathbf{u}}$ when iterating over the meshes
- **fe->require_field_calculation(uu, PDE_LocalFE::dN)**

  configures ***fe** to allow for accessing the values of the first order partial derivatives $\nabla\mathbf{N}_i^{\mathbf{u}}$ of the basis functions when iterating over the meshes
- **fe->require_field_calculation(uu, PDE_LocalFE::d2N)**

  configures ***fe** to allow for accessing the values of the second order partial derivatives $\nabla\nabla\mathbf{N}_i^{\mathbf{u}}$ of the basis functions when iterating over the meshes

Warning

The higher the required derivative order, the higher the CPU cost for the subsequent use of ***fe** (in particular the computation of the second order derivatives is highly CPU penalizing).

II.6.3 Computation of the Local Discrete System

Goal

$$\left\{\begin{array}{l}\displaystyle\int_{K_e}\mathcal{A}(\varphi_J^{\mathbf{u}},\varphi_I^{\mathbf{v}})\\ \displaystyle\int_{K_e}\mathcal{F}(\varphi_I^{\mathbf{v}})\end{array}\right\}\rightsquigarrow\left\{\begin{array}{l}\displaystyle a^e(\,i,i_c;j,j_c\,)=\int_{K_e}\mathcal{A}(\mathbf{N}_j^{\mathbf{u}}\mathbf{e}_{j_c},\mathbf{N}_i^{\mathbf{v}}\mathbf{e}_{i_c})\\ \displaystyle f^e(\,i,i_c\,)=\int_{K_e}\mathcal{F}(\mathbf{N}_i^{\mathbf{v}}\mathbf{e}_{i_c})\end{array}\right\}$$

Actors with their Prerequisites

- two objects of type **PDE_DiscreteField**, referenced by the pointers **uu**, **vv** and respectively attached to $\mathbf{u}$, $\mathbf{v}$
- an object of type **PDE_LocalFE**, referenced by the pointer **fe**
  - configured with respect to the *calculation requirements* relative to $\mathbf{u}$ and $\mathbf{v}$
  - such that the *mesh iterator* is positioned on a *mesh* K_e
- an object of type **PDE_LocalEquation**, referenced by the pointer **leq**

Process

- configuration of ***fe** so that the space of *test functions* is defined by $\mathbf{v}_h$ (*i.e.* $\mathcal{V}_h^{\mathbf{v}}$) and the space of *shape functions* is defined by $\mathbf{u}_h$ (*i.e.* $\mathcal{V}_h^{\mathbf{u}}$):

```
fe->set_row_and_col_fields( vv, uu ) ;
```

- dimensioning and initialization of `*leq` to represent $[a^e, f^e]$:

```
leq->initialize(
     fe->row_field_node_connectivity(), vv->nb_components(),
     fe->col_field_node_connectivity(), uu->nb_components() ) ;
```

- computation of the integral by numerical quadrature

II.6.4 Iteration on the Integration Points

Actors with their Prerequisites

- an object of type **PDE_LocalFE**, referenced by the pointer **fe**, associated to a meshing of the domain $\Omega \subset \mathbb{R}^d$
    - configured with respect to the *calculation requirements* of a set $\mathcal{H}$ of fields
    - such that the *mesh iterator* is positioned on a *mesh* K_e
    - configured so that the space of *test functions* is defined by $\mathbf{v}_h$ (*i.e.* $\mathcal{V}_h^{\mathbf{v}}$) and the space of *shape functions* is defined by $\mathbf{u}_h$ (*i.e.* $\mathcal{V}_h^{\mathbf{u}}$), which implies necessarily that $u_h, v_h \in \mathcal{H}$
- an object of type **GE_QRprovider**, referenced by the pointer **qrp**, enabling ***fe** to bring into play a *quadrature rule* $(\mathbf{x}_p, w_p)_p$ on K_e

Iteration

The object ***fe** allows for:

- traversing in sequence all the pairs $(\mathbf{x}_p, w_p)$;
- accessing to the values of fields of $\mathcal{H}$ and of their derivatives at the points $\mathbf{x}_p$;
- accessing to the values of the *shape functions* and *test functions* and of their derivatives at the points $\mathbf{x}_p$.

expression	result/side effect of the evaluation
`fe->start_IP_iterator( qrp )`	move the *integration point iterator* on the first point $\mathbf{x}_0$, and do all the calculations required amongst $\mathbf{N}_i^{\mathbf{v}}(\mathbf{x}_0)$ $\nabla\mathbf{N}_i^{\mathbf{v}}(\mathbf{x}_0)$ $\nabla\nabla\mathbf{N}_i^{\mathbf{v}}(\mathbf{x}_0)$ $0 \leq i < N_e^{\mathbf{v}}$ $\mathbf{N}_j^{\mathbf{u}}(\mathbf{x}_0)$ $\nabla\mathbf{N}_j^{\mathbf{u}}(\mathbf{x}_0)$ $\nabla\nabla\mathbf{N}_j^{\mathbf{u}}(\mathbf{x}_0)$ $0 \leq j < N_e^{\mathbf{u}}$ $f_h(\mathbf{x}_0)$ $\nabla f_h(\mathbf{x}_0)$ $\nabla\nabla f_h(\mathbf{x}_0)$ $f_h \in \mathcal{H}$
`fe->valid_IP()`	**true** if the *integration point iterator* is effectively positioned on a point
`fe->go_next_IP()`	move the *integration point iterator* on the next point and, if it exists (then denoted $\mathbf{x}_p$), do all the calculations required amongst $\mathbf{N}_i^{\mathbf{v}}(\mathbf{x}_p)$ $\nabla\mathbf{N}_i^{\mathbf{v}}(\mathbf{x}_p)$ $\nabla\nabla\mathbf{N}_i^{\mathbf{v}}(\mathbf{x}_p)$ $0 \leq i < N_e^{\mathbf{v}}$ $\mathbf{N}_j^{\mathbf{u}}(\mathbf{x}_p)$ $\nabla\mathbf{N}_j^{\mathbf{u}}(\mathbf{x}_p)$ $\nabla\nabla\mathbf{N}_j^{\mathbf{u}}(\mathbf{x}_p)$ $0 \leq j < N_e^{\mathbf{u}}$ $f_h(\mathbf{x}_p)$ $\nabla f_h(\mathbf{x}_p)$ $\nabla\nabla f_h(\mathbf{x}_p)$ $f_h \in \mathcal{H}$

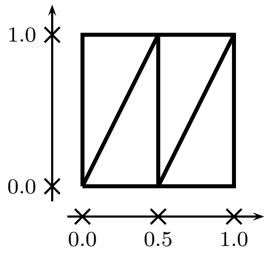
expression	evaluation result
`fe->value_at_IP( ff, ll, ic )`	$f_{h,i_c}(\mathbf{x}_p)$
`fe->gradient_at_IP( ff, ll,a ,ic )`	$\dfrac{\partial f_{h,i_c}}{\partial x_a}(\mathbf{x}_p) \qquad a = \texttt{a}$
`fe->hessian_at_IP( ff, ll,a, b, ic )`	$\dfrac{\partial^2 f_{h,i_c}}{\partial x_a \partial x_b}(\mathbf{x}_p) \quad a, b = \texttt{a}, \texttt{b}$
f_h : *discrete field* related to the *storage level* **`ll`** of **`ff`** f_{h,i_c} : component $i_c = \texttt{ic}$ of f_h	
`fe->nb_basis_functions( PDE_LocalFE::row )`	$N_e^{\mathbf{v}}$
`fe->nb_basis_functions( PDE_LocalFE::col )`	$N_e^{\mathbf{u}}$
`PDE_LocalFE` defines an enumerated type (**`enum`**) with two values ▪ **`row`** : used to identify the *test functions* defined by $\mathbf{v}$ ▪ **`col`** : used to identify the *shape functions* defined by $\mathbf{u}$	

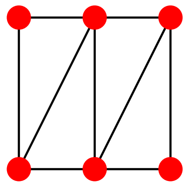
expression	evaluation result
`fe->N_at_IP( PDE_LocalFE::row, i )`	$\mathbf{N}_i^{\mathbf{v}}(\mathbf{x}_p) \qquad i = \texttt{i}$
`fe->dN_at_IP( PDE_LocalFE::row, i, a )`	$\dfrac{\partial \mathbf{N}_i^{\mathbf{v}}}{\partial x_a}(\mathbf{x}_p) \qquad i = \texttt{i} \qquad a = \texttt{a}$
`fe->d2N_at_IP( PDE_LocalFE::row, i, a, b )`	$\dfrac{\partial^2 \mathbf{N}_i^{\mathbf{v}}}{\partial x_a \partial x_b}(\mathbf{x}_p) \qquad i = \texttt{i}$ $a, b = \texttt{a}, \texttt{b}$
`fe->Ns_at_IP( PDE_LocalFE::row )`	$\mathbf{N}_i^{\mathbf{v}}(\mathbf{x}_p)$ $0 \leq i < N_e^{\mathbf{v}}$
`fe->dNs_at_IP( PDE_LocalFE::row )`	$\dfrac{\partial \mathbf{N}_i^{\mathbf{v}}}{\partial x_a}(\mathbf{x}_p)$ $0 \leq i < N_e^{\mathbf{v}} \qquad 0 \leq a < d$
`fe->d2Ns_at_IP( PDE_LocalFE::row )`	$\dfrac{\partial^2 \mathbf{N}_i^{\mathbf{v}}}{\partial x_a \partial x_b}(\mathbf{x}_p)$ $0 \leq i < N_e^{\mathbf{v}} \qquad 0 \leq a, b < d$

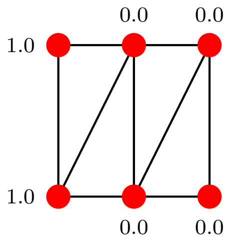
expression	evaluation result
`fe->N_at_IP( PDE_LocalFE::col, j )`	$\mathsf{N}^{\mathbf{u}}_j(\mathbf{x}_p) \qquad j = \texttt{j}$
`fe->dN_at_IP( PDE_LocalFE::col, j, a )`	$\dfrac{\partial \mathsf{N}^{\mathbf{u}}_j}{\partial x_a}(\mathbf{x}_p) \qquad j = \texttt{j} \qquad a = \texttt{a}$
`fe->d2N_at_IP( PDE_LocalFE::col, j, a, b )`	$\dfrac{\partial^2 \mathsf{N}^{\mathbf{u}}_j}{\partial x_a \partial x_b}(\mathbf{x}_p) \qquad j = \texttt{j}$ $a, b = \texttt{a}, \texttt{b}$
`fe->Ns_at_IP( PDE_LocalFE::col )`	$\mathsf{N}^{\mathbf{u}}_j(\mathbf{x}_p)$ $0 \leq j < N^{\mathbf{u}}_e$
`fe->dNs_at_IP( PDE_LocalFE::col )`	$\dfrac{\partial \mathsf{N}^{\mathbf{u}}_j}{\partial x_a}(\mathbf{x}_p)$ $0 \leq j < N^{\mathbf{u}}_e \qquad 0 \leq a < d$
`fe->d2Ns_at_IP( PDE_LocalFE::col )`	$\dfrac{\partial^2 \mathsf{N}^{\mathbf{u}}_j}{\partial x_a \partial x_b}(\mathbf{x}_p)$ $0 \leq j < N^{\mathbf{u}}_e \qquad 0 \leq a, b < d$

II.6.5 Examples

```
MODULE PDE_DomainAndFields

   verbose_level = 1
   nb_space_dimensions = 2
   type = "finite_element"

   MODULE GE_Meshing
      concrete_name = "GE_RefinedMeshing"
      mesh_polyhedron = < "GE_Segment" "GE_Triangle" >
      MODULE list_of_GE_ReferencePolyhedronRefiner
         MODULE GE_ReferencePolyhedronRefiner#1
            concrete_name = "GE_ReferenceSquareWithTriangles"
            strategy = "/"
         END MODULE GE_ReferencePolyhedronRefiner#1
      END MODULE list_of_GE_ReferencePolyhedronRefiner
      MODULE GE_Meshing
         concrete_name = "GE_BoxWithBoxes"
         vertices_coordinate_1 = regular_vector( 0.0, 1, 1.0 )
         vertices_coordinate_0 = regular_vector( 0.0, 2, 1.0 )
         mesh_polyhedron = < "GE_Segment" "GE_Rectangle" >
      END MODULE GE_Meshing
   END MODULE GE_Meshing

   MODULE interior_fields

      MODULE temperature
         name = "temp"
         nb_components = 1
         element_name = "PDE_2D_P1_3nodes"
         storage_depth = 2
         MODULE DOFs_values
            type = "uniformly_defined"
            value = < 0.0 >
         END MODULE DOFs_values
         MODULE DOFs_imposed_value
            MODULE xxx
               location = "on_bounds"
               color = "left"
               type = "uniformly_defined"
               value = < 1.0 >
            END MODULE xxx
         END MODULE DOFs_imposed_value
      END MODULE temperature

   END MODULE interior_fields

END MODULE PDE_DomainAndFields
```

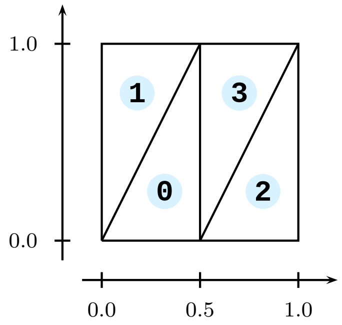






Iteration on Cells

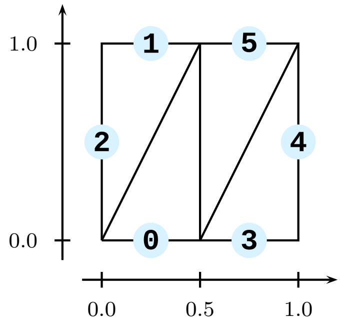
```
PDE_LocalFEcell* fe = DOM->create_LocalFEcell( 0 ) ;
for( fe->start() ; fe->is_valid() ; fe->go_next() )
{
   cout « "cell " « fe->mesh_id() « "   " ;
   GE_Mpolyhedron const* poly = fe->polyhedron() ;
   poly->print( cout, 0 ) ;
}
fe->destroy() ;
```



```
cell 0   GE_Triangle :
   ( 0.000000000e+00 , 0.000000000e+00 )
   ( 5.000000000e-01 , 0.000000000e+00 )
   ( 5.000000000e-01 , 1.000000000e+00 )
cell 1   GE_Triangle :
   ( 5.000000000e-01 , 1.000000000e+00 )
   ( 0.000000000e+00 , 1.000000000e+00 )
   ( 0.000000000e+00 , 0.000000000e+00 )
cell 2   GE_Triangle :
   ( 5.000000000e-01 , 0.000000000e+00 )
   ( 1.000000000e+00 , 0.000000000e+00 )
   ( 1.000000000e+00 , 1.000000000e+00 )
cell 3   GE_Triangle :
   ( 1.000000000e+00 , 1.000000000e+00 )
   ( 5.000000000e-01 , 1.000000000e+00 )
   ( 5.000000000e-01 , 0.000000000e+00 )
```

Iteration on Bounds

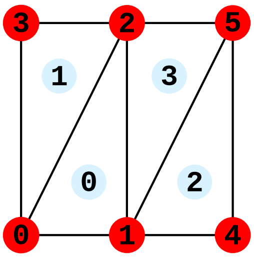
```
PDE_LocalFEbound* fe = DOM->create_LocalFEbound( 0 ) ;
for( fe->start() ; fe->is_valid() ; fe->go_next() )
{
   cout « "bound " « fe->mesh_id() « "   " ;
   GE_Mpolyhedron const* poly = fe->polyhedron() ;
   poly->print( cout, 0 ) ;
}
fe->destroy() ;
```



```
bound 0   GE_Segment :
   ( 0.000000000e+00 , 0.000000000e+00 )
   ( 5.000000000e-01 , 0.000000000e+00 )
bound 1   GE_Segment :
   ( 5.000000000e-01 , 1.000000000e+00 )
   ( 0.000000000e+00 , 1.000000000e+00 )
bound 2   GE_Segment :
   ( 0.000000000e+00 , 1.000000000e+00 )
   ( 0.000000000e+00 , 0.000000000e+00 )
bound 3   GE_Segment :
   ( 5.000000000e-01 , 0.000000000e+00 )
   ( 1.000000000e+00 , 0.000000000e+00 )
bound 4   GE_Segment :
   ( 1.000000000e+00 , 0.000000000e+00 )
   ( 1.000000000e+00 , 1.000000000e+00 )
bound 5   GE_Segment :
   ( 1.000000000e+00 , 1.000000000e+00 )
   ( 5.000000000e-01 , 1.000000000e+00 )
```

Iteration on Cells then on Nodes

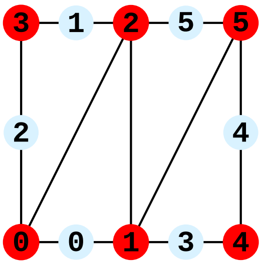
```
PDE_SetOfDiscreteFields const* sdf = DOM->set_of_discrete_fields() ;
PDE_DiscreteField const* uu = sdf->item( "temp" ) ;
PDE_LocalFEcell* fe = DOM->create_LocalFEcell( 0 ) ;
fe->require_field_calculation( uu, PDE_LocalFE::N ) ;
for( fe->start() ; fe->is_valid() ; fe->go_next() )
{
   fe->set_row_and_col_fields( uu, uu ) ;

   size_t_vector const& rc = fe->row_field_node_connectivity() ;
   cout « "cell " « fe->mesh_id() « " : noeuds " ;
    for( size_t i=0 ; i<rc.size() ; ++i )
   {
      cout « "  " « rc(i) ;
   }
   cout « endl ;
}
fe->destroy() ;
```



```
cell 0 : nodes   0  1  2
cell 1 : nodes   2  3  0
cell 2 : nodes   1  4  5
cell 3 : nodes   5  2  1
```

Iteration on Bounds then on Nodes

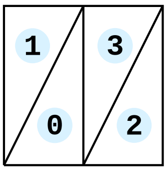
```
PDE_SetOfDiscreteFields const* sdf = DOM->set_of_discrete_fields() ;
PDE_DiscreteField const* uu = sdf->item( "temp" ) ;
PDE_LocalFEbound* fe = DOM->create_LocalFEbound( 0 ) ;
fe->require_field_calculation( uu, PDE_LocalFE::N ) ;
for( fe->start() ; fe->is_valid() ; fe->go_next() )
{
   fe->set_row_and_col_fields( uu, uu ) ;

   size_t_vector const& rc = fe->row_field_node_connectivity() ;
   cout « "bound " « fe->mesh_id() « " : noeuds " ;
   for( size_t i=0 ; i<rc.size() ; ++i )
   {
      cout « "  " « rc(i) ;
   }
   cout « endl ;
}
fe->destroy() ;
```

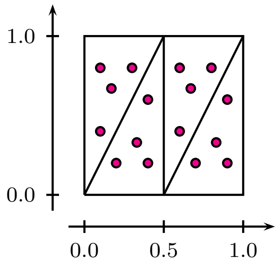


```
bound 0 : nodes   0  1  2
bound 1 : nodes   2  3  0
bound 2 : nodes   2  3  0
bound 3 : nodes   1  4  5
bound 4 : nodes   1  4  5
bound 5 : nodes   5  2  1
```

Iteration on Cells then on Integration Points

```
GE_QRprovider const* qrp = GE_QRprovider::object( "GE_QRprovider_3" ) ;
PDE_SetOfDiscreteFields const* sdf = DOM->set_of_discrete_fields() ;
PDE_DiscreteField const* uu = sdf->item( "temp" ) ;
PDE_LocalFEcell* fe = DOM->create_LocalFEcell( 0 ) ;
fe->require_field_calculation( uu, PDE_LocalFE::N ) ;
for( fe->start() ; fe->is_valid() ; fe->go_next() )
{
   fe->set_row_and_col_fields( uu, uu ) ;

   cout « "cell " « fe->mesh_id() « endl ;
   fe->start_IP_iterator( qrp ) ;
   for( ; fe->valid_IP() ; fe->go_next_IP() )
   {
      GE_Point const* pt = fe->coordinates_of_IP() ;
      cout « pt->coordinate( 0 ) « " "
           « pt->coordinate( 1 ) « endl ;
   }
}
fe->destroy() ;
```

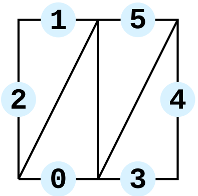


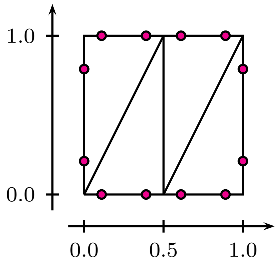


```
cell 0
0.33 0.33
0.20 0.20
0.40 0.20
0.40 0.60
cell 1
0.17 0.67
0.30 0.80
0.10 0.80
0.10 0.40
cell 2
0.83 0.33
0.70 0.20
0.90 0.20
0.90 0.60
cell 3
0.67 0.67
0.80 0.80
0.60 0.80
0.60 0.40
```

Iteration on Bounds then on Integration Points

```
GE_QRprovider const* qrp = GE_QRprovider::object( "GE_QRprovider_3" ) ;
PDE_SetOfDiscreteFields const* sdf = DOM->set_of_discrete_fields() ;
PDE_DiscreteField const* uu = sdf->item( "temp" ) ;
PDE_LocalFEbound* fe = DOM->create_LocalFEbound( 0 ) ;
fe->require_field_calculation( uu, PDE_LocalFE::N ) ;
for( fe->start() ; fe->is_valid() ; fe->go_next() )
{
   fe->set_row_and_col_fields( uu, uu ) ;

   cout « "bound " « fe->mesh_id() « endl ;
   fe->start_IP_iterator( qrp ) ;
   for( ; fe->valid_IP() ; fe->go_next_IP() )
   {
      GE_Point const* pt = fe->coordinates_of_IP() ;
      cout « pt->coordinate( 0 ) « " "
           « pt->coordinate( 1 ) « endl ;
   }
}
fe->destroy() ;
```

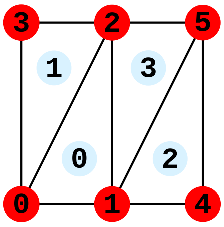


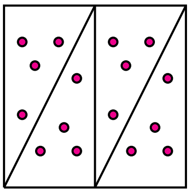


```
bound 0
0.39 0.00
0.11 0.00
bound 1
0.11 1.00
0.39 1.00
bound 2
0.00 0.21
0.00 0.79
bound 3
0.89 0.00
0.61 0.00
bound 4
1.00 0.79
1.00 0.21
bound 5
0.61 1.00
0.89 1.00
```

Iteration on Cells then on Integration Points

```
PDE_LocalFE::field_id const row = PDE_LocalFE::row ;
GE_QRprovider const* qrp = GE_QRprovider::object( "GE_QRprovider_3" ) ;
PDE_SetOfDiscreteFields const* sdf = DOM->set_of_discrete_fields() ;
PDE_DiscreteField const* uu = sdf->item( "temp" ) ;
PDE_LocalFEcell* fe = DOM->create_LocalFEcell( 0 ) ;
fe->require_field_calculation( uu, PDE_LocalFE::N ) ;
for( fe->start() ; fe->is_valid() ; fe->go_next() )
{
   fe->set_row_and_col_fields( uu, uu ) ;
   size_t nb_bfs = fe->nb_basis_functions( row ) ;
   cout « "cell " « fe->mesh_id() « endl ;
   fe->start_IP_iterator( qrp ) ;
   for( ; fe->valid_IP() ; fe->go_next_IP() )
   {
      for( size_t i=0 ; i<nb_bfs ; ++i )
         cout « fe->N_at_IP( row, i ) « " " ;
      cout « endl ;
   }
}
fe->destroy() ;
```

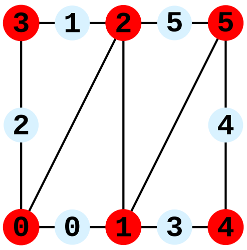


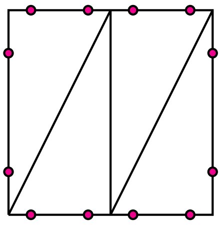


```
cell 0
0.33 0.33 0.33
0.60 0.20 0.20
0.20 0.60 0.20
0.20 0.20 0.60
cell 1
0.33 0.33 0.33
0.60 0.20 0.20
0.20 0.60 0.20
0.20 0.20 0.60
cell 2
0.33 0.33 0.33
0.60 0.20 0.20
0.20 0.60 0.20
0.20 0.20 0.60
cell 3
0.33 0.33 0.33
0.60 0.20 0.20
0.20 0.60 0.20
0.20 0.20 0.60
```

Iteration on Bounds then on Integration Points

```
PDE_LocalFE::field_id const row = PDE_LocalFE::row ;
GE_QRprovider const* qrp = GE_QRprovider::object( "GE_QRprovider_3" ) ;
PDE_SetOfDiscreteFields const* sdf = DOM->set_of_discrete_fields() ;
PDE_DiscreteField const* uu = sdf->item( "temp" ) ;
PDE_LocalFEbound* fe = DOM->create_LocalFEbound( 0 ) ;
fe->require_field_calculation( uu, PDE_LocalFE::N ) ;
for( fe->start() ; fe->is_valid() ; fe->go_next() )
{
   fe->set_row_and_col_fields( uu, uu ) ;
   size_t nb_bfs = fe->nb_basis_functions( row ) ;
   cout « "bound " « fe->mesh_id() « endl ;
   fe->start_IP_iterator( qrp ) ;
   for( ; fe->valid_IP() ; fe->go_next_IP() )
   {
      for( size_t i=0 ; i<nb_bfs ; ++i )
         cout « fe->N_at_IP( row, i ) « " " ;
      cout « endl ;
   }
}
fe->destroy() ;
```





```
bound 0
0.21 0.79 0.00
0.79 0.21 0.00
bound 1
0.21 0.79 0.00
0.79 0.21 0.00
bound 2
0.00 0.21 0.79
0.00 0.79 0.21
bound 3
0.21 0.79 0.00
0.79 0.21 0.00
bound 4
0.00 0.21 0.79
0.00 0.79 0.21
bound 5
0.21 0.79 0.00
0.79 0.21 0.00
```

Chapter III

FE Class From Library **FrameFE**

The **FE** class gathers a set of elementary functions related to different terms occuring classically in variational approximations of boundary value problems, particularly those arising from continuum thermo-mechanics (static member functions whose name start with **add**).

Each of these functions

- has a first parameter **leq** referencing an object of type **PDE_LocalEquation** which represents the system $\left\{ a^e(i, i_c; j, j_c), f^e(i, i_c) \right\}$ local to a mesh K_e;
- has a second parameter **fe** referencing an object of type **PDE_LocalFE** positioned on an given integration point $\mathbf{x}_p$ of the mesh K_e;
- is associated to a given term of kind (2.6) or (2.7);
- implements the calculation of the contribution $a^e_p(i, i_c; j, j_c)$ or $f^e_p(i, i_c)$ at point $\mathbf{x}_p$ according to the relations (2.8) or (2.9), this for all possible values of i, i_c, j, j_c ;
- adds these contributions to the local system represented by **leq**.

In the foregoing will be explicited, member function by member function, the following three features:

$\boxed{\text{c}_1}$ the term of the considered variational approximation (of kind (2.6) or (2.7));

$\boxed{\text{c}_2}$ the range of the indices i, i_c, j, j_c ;

$\boxed{\text{c}_3}$ the formulas for the computation of $a^e_p(i, i_c; j, j_c)$ or $f^e_p(i, i_c)$.

The names of these member functions have been chosen to be as evocative as possible of the related term in the variational approximation. Hence, the word **row** represents the *test functions* whereas the word **col** represents the *shape functions*. Nevertheless, evocative need not mean defining, and in the precise case of the **FE** class, only a careful reading the source code explained by the present document can ensure a judicious usage.

III.1 Differential Operators Applied to Basis Functions

In the foregoing, the following writing will be adopted in order to simplify the notations:

$$f_{,\beta} \overset{\text{def}}{=} \frac{\partial f}{\partial x_\beta} \quad \text{et} \quad f_{,\beta\gamma} \overset{\text{def}}{=} \frac{\partial^2 f}{\partial x_\beta \partial x_\gamma} \quad \text{where} \quad \beta, \gamma \in [0, d[$$

The following table expresses, using a mesh local indexing, the various components for the result of applying the differential operators occuring in the next sections to a basis function $\boldsymbol{\varphi}^{\mathbf{u}}_I$.

	number of components	range and restrictions	expression local to a mesh K_e
$\boldsymbol{\varphi}^{\mathbf{u}}_I$	1	component $0 \le \alpha < N^{\mathbf{u}}_{\mathrm{c}}$	$\mathsf{N}^{\mathbf{u}}_i \; \delta_{i_c\alpha}$
$\nabla\boldsymbol{\varphi}^{\mathbf{u}}_I$	2	$1-st$ component $\quad 0 \le \alpha < N^{\mathbf{u}}_{\mathrm{c}}$ $2-nd$ component $\quad 0 \le \beta < d$	$\mathsf{N}^{\mathbf{u}}_{i,\beta} \; \delta_{i_c\alpha}$
$\nabla^{\mathrm{T}}\boldsymbol{\varphi}^{\mathbf{u}}_I$	2	$1-st$ component $\quad 0 \le \alpha < d$ $2-nd$ component $\quad 0 \le \beta < N^{\mathbf{u}}_{\mathrm{c}}$	$\mathsf{N}^{\mathbf{u}}_{i,\alpha} \; \delta_{i_c\beta}$
$\nabla \cdot \boldsymbol{\varphi}^{\mathbf{u}}_I$	0	$N^{\mathbf{u}}_{\mathrm{c}} = 1$	$\mathsf{N}^{\mathbf{u}}_{i,i_c}$
$\nabla^2\boldsymbol{\varphi}^{\mathbf{u}}_I$	1	component $\;0 \le \alpha < N^{\mathbf{u}}_{\mathrm{c}}$	$\left[\sum_{\beta=0}^{d-1} \mathsf{N}^{\mathbf{u}}_{i,\beta\beta}\right] \delta_{i_c\alpha}$

III.2 Reference

III.2.1 add_row_col_S

$\boxed{\mathrm{c}_1}$ $\displaystyle\int_{K_e} \kappa \; \boldsymbol{\varphi}^{\mathbf{v}}_I \cdot \boldsymbol{\varphi}^{\mathbf{u}}_J \; dK_e \quad \text{with} \quad \mathbf{v} = \mathbf{u} \quad \text{and} \quad \kappa \colon K_e \to \mathbb{R}$

The signs **u** and **v** will be omitted in the remaining of this paragraph.

$\boxed{\mathrm{c}_2}$ $i, j \in [0, N_e[\quad i_c, j_c \in [0, N_{\mathrm{c}}[$

$\boxed{\mathrm{c}_3}$

$$\begin{aligned} a^e_p(i, i_c; j, j_c) &= w_p \; \kappa(\mathbf{x}_p) \sum_{\alpha=0}^{N_{\mathrm{c}}-1} \mathsf{N}_i(\mathbf{x}_p) \; \delta_{i_c\alpha} \; \mathsf{N}_j(\mathbf{x}_p) \; \delta_{j_c\alpha} \\ &= \begin{cases} 0 & \text{if } i_c \neq j_c \\ w_p \; \kappa(\mathbf{x}_p) \; \mathsf{N}_i(\mathbf{x}_p) \; \mathsf{N}_j(\mathbf{x}_p) & \text{if } i_c = j_c \end{cases} \end{aligned}$$

There hold the symmetry property: $a^e_p(i, i_c; j, i_c) = a^e_p(j, i_c; i, i_c)$

III.2.2 add_row_col_NS

$\boxed{\mathrm{c}_1}$ $\displaystyle\int_{K_e} \kappa \; \boldsymbol{\varphi}^{\mathbf{v}}_I \cdot \boldsymbol{\varphi}^{\mathbf{u}}_J \; dK_e \quad \text{with} \quad N^{\mathbf{v}}_{\mathrm{c}} = N^{\mathbf{u}}_{\mathrm{c}} \overset{\text{def}}{=} N_{\mathrm{c}} \quad \text{and} \quad \kappa \colon K_e \to \mathbb{R}$

$\boxed{\mathrm{c}_2}$ $i \in [0, N^{\mathbf{v}}_e[\quad j \in [0, N^{\mathbf{u}}_e[\quad i_c, j_c \in [0, N_{\mathrm{c}}[$

$$\boxed{\mathrm{c}_3}\quad \begin{aligned} a^e_p(i,i_c;j,j_c) &= w_p\;\kappa(\mathbf{x}_p)\sum_{\alpha=0}^{N_{\mathrm{c}}-1}\mathsf{N}^{\mathbf{v}}_i(\mathbf{x}_p)\;\delta_{i_c\alpha}\;\mathsf{N}^{\mathbf{u}}_j(\mathbf{x}_p)\;\delta_{j_c\alpha} \\ &= \begin{cases} 0 & \text{if } i_c\neq j_c \\ w_p\;\kappa(\mathbf{x}_p)\;\mathsf{N}^{\mathbf{v}}_i(\mathbf{x}_p)\;\mathsf{N}^{\mathbf{u}}_j(\mathbf{x}_p) & \text{if } i_c=j_c \end{cases} \end{aligned}$$

III.2.3 `add_lumped_row_col`

$$\boxed{\mathrm{c}_1}\quad \int_{K_e}\kappa\;\boldsymbol{\varphi}^{\mathbf{v}}_I\cdot\boldsymbol{\varphi}^{\mathbf{u}}_J\;dK_e \qquad \begin{array}{l}\text{in a “lumped diagonal” version,}\\ \text{with}\quad \mathbf{v}=\mathbf{u}\quad \text{and}\quad \kappa\colon K_e\to\mathbb{R}\end{array}$$

The signs **u** and **v** will be omitted in the remaining of this paragraph.

$\boxed{\mathrm{c}_2}$ $i,j\in[0,N_e[\quad i_c,j_c\in[0,N_{\mathrm{c}}[$

$$\boxed{\mathrm{c}_3}\quad a^e_p(i,i_c;j,j_c) = \begin{cases} 0 & \text{if } i\neq j \text{ or } i_c\neq j_c \\ w_p\;\kappa(\mathbf{x}_p)\;\mathsf{N}_i(\mathbf{x}_p) & \text{if } i=j \text{ and } i_c=j_c \end{cases}$$

III.2.4 `add_grad_row_grad_col_S`

$$\boxed{\mathrm{c}_1}\quad \int_{\mathcal{C}_e}\kappa\;\nabla\boldsymbol{\varphi}^{\mathbf{v}}_I\cdot\nabla\boldsymbol{\varphi}^{\mathbf{u}}_J\;d\mathcal{C}_e \quad \text{with}\quad \mathbf{v}=\mathbf{u}\quad\text{and}\quad \kappa\colon\mathcal{C}_e\to\mathbb{R}$$

The signs **u** and **v** will be omitted in the remaining of this paragraph.

$\boxed{\mathrm{c}_2}$ $i,j\in[0,N_e[\quad i_c,j_c\in[0,N_{\mathrm{c}}[$

$$\boxed{\mathrm{c}_3}\quad \begin{aligned} a^e_p(i,i_c;j,j_c) &= w_p\;\kappa(\mathbf{x}_p)\sum_{\alpha=0}^{N_{\mathrm{c}}-1}\sum_{\beta=0}^{d-1}\mathsf{N}_{i,\beta}(\mathbf{x}_p)\;\delta_{i_c\alpha}\;\mathsf{N}_{j,\beta}(\mathbf{x}_p)\;\delta_{j_c\alpha} \\ &= \begin{cases} 0 & \text{if } i_c\neq j_c \\ w_p\;\kappa(\mathbf{x}_p)\displaystyle\sum_{\beta=0}^{d-1}\mathsf{N}_{i,\beta}(\mathbf{x}_p)\;\mathsf{N}_{j,\beta}(\mathbf{x}_p) & \text{if } i_c=j_c \end{cases} \end{aligned}$$

There hold the symmetry property: $a^e_p(i,i_c;j,i_c)=a^e_p(j,i_c;i,i_c)$

III.2.5 `add_grad_row_grad_col_NS`

$$\boxed{\mathrm{c}_1}\quad \int_{\mathcal{C}_e}\kappa\;\nabla\boldsymbol{\varphi}^{\mathbf{v}}_I\cdot\nabla\boldsymbol{\varphi}^{\mathbf{u}}_J\;d\mathcal{C}_e \quad \text{with}\quad N^{\mathbf{v}}_{\mathrm{c}}=N^{\mathbf{u}}_{\mathrm{c}}\overset{\text{def}}{=}N_{\mathrm{c}}\quad\text{and}\quad \kappa\colon\mathcal{C}_e\to\mathbb{R}$$

$\boxed{\mathrm{c}_2}$ $i\in[0,N^{\mathbf{v}}_e[\quad j\in[0,N^{\mathbf{u}}_e[\quad i_c,j_c\in[0,N_{\mathrm{c}}[$

$$\boxed{\mathrm{c}_3}\quad \begin{aligned} a^e_p(i,i_c;j,j_c) &= w_p\;\kappa(\mathbf{x}_p)\sum_{\alpha=0}^{N_{\mathrm{c}}-1}\sum_{\beta=0}^{d-1}\mathsf{N}^{\mathbf{v}}_{i,\beta}(\mathbf{x}_p)\;\delta_{i_c\alpha}\;\mathsf{N}^{\mathbf{u}}_{j,\beta}(\mathbf{x}_p)\;\delta_{j_c\alpha} \\ &= \begin{cases} 0 & \text{if } i_c\neq j_c \\ w_p\;\kappa(\mathbf{x}_p)\displaystyle\sum_{\beta=0}^{d-1}\mathsf{N}^{\mathbf{v}}_{i,\beta}(\mathbf{x}_p)\;\mathsf{N}^{\mathbf{u}}_{j,\beta}(\mathbf{x}_p) & \text{if } i_c=j_c \end{cases} \end{aligned}$$

III.2.6 `add_grad_row_D_col_S`

$$\boxed{\mathrm{c}_1}\quad \int_{\mathcal{C}_e}\kappa\;\nabla\boldsymbol{\varphi}^{\mathbf{v}}_I\cdot\left(\nabla^{\mathrm{T}}\boldsymbol{\varphi}^{\mathbf{u}}_J+\nabla\boldsymbol{\varphi}^{\mathbf{u}}_J\right)\;d\mathcal{C}_e \quad \text{with}\quad v=u\;\;,\quad N^{\mathbf{v}}_{\mathrm{c}}=d\quad\text{and}\quad \kappa\colon\mathcal{C}_e\to\mathbb{R}$$

The signs **u** and **v** will be omitted in the remaining of this paragraph.

$\boxed{\mathrm{c}_2}$ $\quad i,j \in [0,N_e[\quad i_c,j_c \in [0,d[$

$$\boxed{\mathrm{c}_3} \quad \begin{aligned} a^e_p(i,i_c;j,j_c) &= w_p\,\kappa(\mathbf{x}_p)\sum_{\alpha=0}^{d-1}\sum_{\beta=0}^{d-1} \mathsf{N}_{i,\beta}(\mathbf{x}_p)\,\delta_{i_c\alpha}\left[\mathsf{N}_{j,\alpha}(\mathbf{x}_p)\,\delta_{j_c\beta}+\mathsf{N}_{j,\beta}(\mathbf{x}_p)\,\delta_{j_c\alpha}\right] \\ &= w_p\,\kappa(\mathbf{x}_p)\sum_{\beta=0}^{d-1} \mathsf{N}^{\mathbf{v}}_{i,\beta}(\mathbf{x}_p)\left[\mathsf{N}_{j,i_c}(\mathbf{x}_p)\,\delta_{j_c\beta}+\mathsf{N}_{j,\beta}(\mathbf{x}_p)\,\delta_{j_c i_c}\right] \\ &= \begin{cases} w_p\,\kappa(\mathbf{x}_p)\left[\mathsf{N}_{i,j_c}(\mathbf{x}_p)\,\mathsf{N}_{j,i_c}(\mathbf{x}_p)+\displaystyle\sum_{\beta=0}^{d-1}\mathsf{N}_{i,\beta}(\mathbf{x}_p)\,\mathsf{N}_{j,\beta}(\mathbf{x}_p)\right] & \text{uf } i_c \neq j_c \\ w_p\,\kappa(\mathbf{x}_p)\,\mathsf{N}_{i,j_c}(\mathbf{x}_p)\,\mathsf{N}_{j,i_c}(\mathbf{x}_p) & \text{if } i_c = j_c \end{cases} \end{aligned}$$

III.2.7 add_row

$\boxed{\mathrm{c}_1}$ $\quad \displaystyle\int_{K_e} a\cdot\boldsymbol{\varphi}^{\mathbf{v}}_I\,dK_e \quad \text{with} \quad a\colon K_e \to \mathbb{R}^{N^{\mathbf{v}}_{\mathrm{c}}}$

$\boxed{\mathrm{c}_2}$ $\quad i \in [0,N^{\mathbf{v}}_e[\quad i_c \in [0,N_{\mathrm{c}}[$

$$\boxed{\mathrm{c}_3} \quad \begin{aligned} f^e_p(i,i_c) &= w_p \sum_{\alpha=0}^{N_{\mathrm{c}}-1} a_\alpha(\mathbf{x}_p)\,\mathsf{N}^{\mathbf{v}}_i(\mathbf{x}_p)\,\delta_{i_c\alpha} \\ &= w_p\,a_{i_c}(\mathbf{x}_p)\,\mathsf{N}^{\mathbf{v}}_i(\mathbf{x}_p) \end{aligned}$$

III.2.8 add_grad_row

$\boxed{\mathrm{c}_1}$ $\quad \displaystyle\int_{K_e} a\cdot\nabla\boldsymbol{\varphi}^{\mathbf{v}}_I\,dK_e \quad \text{with} \quad a\colon K_e \to \mathbb{R}^d$

$\boxed{\mathrm{c}_2}$ $\quad i \in [0,N^{\mathbf{v}}_e[\quad i_c \in [0,N_{\mathrm{c}}[$

$$\boxed{\mathrm{c}_3} \quad \begin{aligned} f^e_p(i,i_c) &= w_p \sum_{\alpha=0}^{N^{\mathbf{v}}_{\mathrm{c}}-1}\left[\sum_{\beta=0}^{d-1} a_\beta(\mathbf{x}_p)\,\mathsf{N}^{\mathbf{v}}_{i,\beta}(\mathbf{x}_p)\right]\delta_{i_c\alpha} \\ &= w_p\sum_{\beta=0}^{d-1} a_\beta(\mathbf{x}_p)\,\mathsf{N}^{\mathbf{v}}_{i,\beta}(\mathbf{x}_p) \end{aligned}$$

III.2.9 add_div_row

$\boxed{\mathrm{c}_1}$ $\quad \displaystyle\int_{K_e} \kappa\;\nabla\cdot\boldsymbol{\varphi}^{\mathbf{v}}_I\,dK_e \quad \text{with} \quad \kappa\colon K_e \to \mathbb{R}$

$\boxed{\mathrm{c}_2}$ $\quad i \in [0,N^{\mathbf{v}}_e[\quad i_c \in [0,N_{\mathrm{c}}[$

$$\boxed{\mathrm{c}_3} \quad f^e_p(i,i_c) = w_p\,\kappa(\mathbf{x}_p)\,\mathsf{N}^{\mathbf{v}}_{i,i_c}(\mathbf{x}_p)$$

III.2.10 add_row_vvgrad_col

$\boxed{\mathrm{c}_1}$ $\quad \displaystyle\int_{\mathcal{C}_e} \kappa\;\boldsymbol{\varphi}^{\mathbf{v}}_I\cdot\left[(a\cdot\nabla)\boldsymbol{\varphi}^{\mathbf{u}}_J\right]d\mathcal{C}_e \quad \text{with} \quad N^{\mathbf{v}}_{\mathrm{c}} = N^{\mathbf{u}}_{\mathrm{c}} \overset{\text{def}}{=} N_{\mathrm{c}} \quad , \quad a\colon \mathcal{C}_e \to \mathbb{R}^d \quad \text{and} \quad \kappa\colon \mathcal{C}_e \to \mathbb{R}$

$\boxed{\mathrm{c}_2}$ $\quad i \in [0,N^{\mathbf{v}}_e[\quad j \in [0,N^{\mathbf{u}}_e[\quad i_c,j_c \in [0,N_{\mathrm{c}}[$

$$\begin{aligned}\boxed{\mathrm{c}_3}\quad a^e_p(i,i_c;j,j_c) &= w_p\,\kappa(\mathbf{x}_p)\sum_{\alpha=0}^{N_{\mathrm{c}}-1}\mathsf{N}^{\mathbf{v}}_i(\mathbf{x}_p)\,\delta_{i_c\alpha}\Big[\sum_{\beta=0}^{d-1}a_\beta(\mathbf{x}_p)\,\mathsf{N}^{\mathbf{u}}_{j,\beta}(\mathbf{x}_p)\Big]\,\delta_{j_c\alpha}\\ &= \begin{cases}0 & \text{if } i_c\neq j_c\\ w_p\,\kappa(\mathbf{x}_p)\,\mathsf{N}^{\mathbf{v}}_i(\mathbf{x}_p)\displaystyle\sum_{\beta=0}^{d-1}a_\beta(\mathbf{x}_p)\,\mathsf{N}^{\mathbf{u}}_{j,\beta}(\mathbf{x}_p) & \text{if } i_c=j_c\end{cases}\end{aligned}$$

III.2.11 **add_grad_row_vv_otimes_col**

$$\boxed{\mathrm{c}_1}\quad \int_{\mathcal{C}_e}\kappa\,\nabla\boldsymbol{\varphi}^{\mathbf{v}}_I\cdot\big[a\otimes\boldsymbol{\varphi}^{\mathbf{u}}_J\big]\,d\mathcal{C}_e\quad\text{with}\quad N^{\mathbf{u}}_{\mathrm{c}}=d\ ,\quad a\colon\mathcal{C}_e\to\mathbb{R}^{N^{\mathbf{v}}_{\mathrm{c}}}\quad\text{and}\quad\kappa\colon\mathcal{C}_e\to\mathbb{R}$$

$$\boxed{\mathrm{c}_2}\quad i\in[0,N^{\mathbf{v}}_e[\quad j\in[0,N^{\mathbf{u}}_e[\quad i_c\in[0,N^{\mathbf{v}}_{\mathrm{c}}[\quad j_c\in[0,d[$$

$$\begin{aligned}\boxed{\mathrm{c}_3}\quad a^e_p(i,i_c;j,j_c) &= w_p\,\kappa(\mathbf{x}_p)\sum_{\alpha=0}^{N^{\mathbf{v}}_{\mathrm{c}}-1}\sum_{\beta=0}^{d-1}\mathsf{N}^{\mathbf{v}}_{i,\beta}(\mathbf{x}_p)\,\delta_{i_c\alpha}\,a_\alpha(\mathbf{x}_p)\,\mathsf{N}^{\mathbf{u}}_j(\mathbf{x}_p)\,\delta_{j_c\beta}\\ &= w_p\,\kappa(\mathbf{x}_p)\,\mathsf{N}^{\mathbf{v}}_{i,j_c}(\mathbf{x}_p)\,a_{i_c}(\mathbf{x}_p)\,\mathsf{N}^{\mathbf{u}}_j(\mathbf{x}_p)\end{aligned}$$

III.2.12 **add_vvgrad_row_col**

$$\boxed{\mathrm{c}_1}\quad \int_{\mathcal{C}_e}\kappa\,\Big[(a\cdot\nabla)\boldsymbol{\varphi}^{\mathbf{v}}_I\Big]\cdot\boldsymbol{\varphi}^{\mathbf{u}}_J\,d\mathcal{C}_e\quad\text{with}\quad N^{\mathbf{v}}_{\mathrm{c}}=N^{\mathbf{u}}_{\mathrm{c}}\overset{\text{def}}{=}N_{\mathrm{c}}\ ,\quad a\colon\mathcal{C}_e\to\mathbb{R}^d\quad\text{and}\quad\kappa\colon\mathcal{C}_e\to\mathbb{R}$$

$$\boxed{\mathrm{c}_2}\quad i\in[0,N^{\mathbf{v}}_e[\quad j\in[0,N^{\mathbf{u}}_e[\quad i_c,j_c\in[0,N_{\mathrm{c}}[$$

$$\begin{aligned}\boxed{\mathrm{c}_3}\quad a^e_p(i,i_c;j,j_c) &= w_p\,\kappa(\mathbf{x}_p)\sum_{\alpha=0}^{N_{\mathrm{c}}-1}\Big[\sum_{\beta=0}^{d-1}a_\beta(\mathbf{x}_p)\,\mathsf{N}^{\mathbf{v}}_{i,\beta}(\mathbf{x}_p)\Big]\,\delta_{i_c\alpha}\,\mathsf{N}^{\mathbf{u}}_j(\mathbf{x}_p)\,\delta_{j_c\alpha}\\ &= \begin{cases}0 & \text{if } i_c\neq j_c\\ w_p\,\kappa(\mathbf{x}_p)\,\mathsf{N}^{\mathbf{u}}_j(\mathbf{x}_p)\displaystyle\sum_{\beta=0}^{d-1}a_\beta(\mathbf{x}_p)\,\mathsf{N}^{\mathbf{v}}_{i,\beta}(\mathbf{x}_p) & \text{if } i_c=j_c\end{cases}\end{aligned}$$

III.2.13 **add_vvgrad_row_vvgrad_col**

$$\boxed{\mathrm{c}_1}\quad \int_{\mathcal{C}_e}\kappa\,\Big[(a\cdot\nabla)\boldsymbol{\varphi}^{\mathbf{v}}_I\Big]\cdot\Big[(a\cdot\nabla)\boldsymbol{\varphi}^{\mathbf{u}}_J\Big]\,d\mathcal{C}_e\quad\begin{array}{l}\text{with}\ \ N^{\mathbf{v}}_{\mathrm{c}}=N^{\mathbf{u}}_{\mathrm{c}}\overset{\text{def}}{=}N_{\mathrm{c}}\ ,\\ a\colon\mathcal{C}_e\to\mathbb{R}^d\quad\text{and}\quad\kappa\colon\mathcal{C}_e\to\mathbb{R}\end{array}$$

$$\boxed{\mathrm{c}_2}\quad i\in[0,N^{\mathbf{v}}_e[\quad j\in[0,N^{\mathbf{u}}_e[\quad i_c,j_c\in[0,N_{\mathrm{c}}[$$

$$\begin{aligned}\boxed{\mathrm{c}_3}\quad a^e_p(i,i_c;j,j_c) &= w_p\,\kappa(\mathbf{x}_p)\sum_{\alpha=0}^{N_{\mathrm{c}}-1}\Big[\sum_{\beta=0}^{d-1}a_\beta(\mathbf{x}_p)\,\mathsf{N}^{\mathbf{v}}_{i,\beta}(\mathbf{x}_p)\Big]\,\delta_{i_c\alpha}\,\Big[\sum_{\beta=0}^{d-1}a_\beta(\mathbf{x}_p)\,\mathsf{N}^{\mathbf{u}}_{j,\beta}(\mathbf{x}_p)\Big]\,\delta_{j_c\alpha}\\ &= \begin{cases}0 & \text{if } i_c\neq j_c\\ w_p\,\kappa(\mathbf{x}_p)\Big[\displaystyle\sum_{\beta=0}^{d-1}a_\beta(\mathbf{x}_p)\,\mathsf{N}^{\mathbf{v}}_{i,\beta}(\mathbf{x}_p)\Big]\ \Big[\displaystyle\sum_{\beta=0}^{d-1}a_\beta(\mathbf{x}_p)\,\mathsf{N}^{\mathbf{u}}_{j,\beta}(\mathbf{x}_p)\Big] & \text{if } i_c=j_c\end{cases}\end{aligned}$$

III.2.14 **add_vvgrad_row_lapl_col**

$$\boxed{\mathrm{c}_1}\quad \int_{\mathcal{C}_e}\kappa\,\Big[(a\cdot\nabla)\boldsymbol{\varphi}^{\mathbf{v}}_I\Big]\cdot\nabla^2\boldsymbol{\varphi}^{\mathbf{u}}_J\,d\mathcal{C}_e\quad\text{with}\quad N^{\mathbf{v}}_{\mathrm{c}}=N^{\mathbf{u}}_{\mathrm{c}}\overset{\text{def}}{=}N_{\mathrm{c}}\ ,\quad a\colon\mathcal{C}_e\to\mathbb{R}^d\quad\text{and}\quad\kappa\colon\mathcal{C}_e\to\mathbb{R}$$

$$\boxed{\mathrm{c}_2}\quad i\in[0,N^{\mathbf{v}}_e[\quad j\in[0,N^{\mathbf{u}}_e[\quad i_c,j_c\in[0,N_{\mathrm{c}}[$$

$$\boxed{\mathrm{c}_3}\quad \begin{aligned} a^e_p(i,i_c;j,j_c) &= w_p\,\kappa(\mathbf{x}_p)\sum_{\alpha=0}^{N_{\mathrm{c}}-1}\Big[\sum_{\beta=0}^{d-1} a_\beta(\mathbf{x}_p)\,\mathsf{N}^{\mathbf{v}}_{i,\beta}(\mathbf{x}_p)\Big]\,\delta_{i_c\alpha}\Big[\sum_{\beta=0}^{d-1}\mathsf{N}^{\mathbf{u}}_{j,\beta\beta}(\mathbf{x}_p)\Big]\,\delta_{j_c\alpha} \\ &= \begin{cases} 0 & \text{if } i_c \neq j_c \\ w_p\,\kappa(\mathbf{x}_p)\Big[\displaystyle\sum_{\beta=0}^{d-1} a_\beta(\mathbf{x}_p)\,\mathsf{N}^{\mathbf{v}}_{i,\beta}(\mathbf{x}_p)\Big]\;\Big[\displaystyle\sum_{\beta=0}^{d-1}\mathsf{N}^{\mathbf{u}}_{j,\beta\beta}(\mathbf{x}_p)\Big] & \text{if } i_c = j_c \end{cases} \end{aligned}$$

III.2.15 `add_vvgrad_row_div_D_col`

$$\boxed{\mathrm{c}_1}\quad \int_{\mathcal{C}_e}\Big[(a\cdot\nabla)\boldsymbol{\varphi}^{\mathbf{v}}_I\Big]\cdot\Big[\nabla\cdot\kappa\big(\nabla^{\mathrm{T}}\boldsymbol{\varphi}^{\mathbf{u}}_J+\nabla\boldsymbol{\varphi}^{\mathbf{u}}_J\big)\Big]\,d\mathcal{C}_e \quad \text{with} \quad \begin{array}{l} v=u\ ,\quad N^{\mathbf{v}}_{\mathrm{c}}=d\ , \\ \kappa\colon \mathcal{C}_e\to\mathbb{R} \quad \text{and} \quad a\colon \mathcal{C}_e\to\mathbb{R}^d \end{array}$$

The signs **u** and **v** will be omitted in the remaining of this paragraph.

$$\boxed{\mathrm{c}_2}\quad i,j\in[0,N_e[\qquad i_c,j_c\in[0,N_{\mathrm{c}}[$$

$$\boxed{\mathrm{c}_3}\quad \begin{aligned} \Big[\nabla\cdot\kappa\big(\nabla^{\mathrm{T}}\boldsymbol{\varphi}^{\mathbf{u}}_J+\nabla\boldsymbol{\varphi}^{\mathbf{u}}_J\big)\Big]_\alpha &= \sum_{\beta=0}^{d-1}\Big[\kappa\,\mathsf{N}_{j,\alpha}\,\delta_{j_c\beta}+\kappa\,\mathsf{N}_{j,\beta}\,\delta_{j_c\alpha}\Big]_{,\beta} \\ &= \sum_{\beta=0}^{d-1}\kappa\,\mathsf{N}_{j,\alpha\beta}\,\delta_{j_c\beta}+\kappa_{,\beta}\,\mathsf{N}_{j,\alpha}\,\delta_{j_c\beta}+\kappa\,\mathsf{N}_{j,\beta\beta}\,\delta_{j_c\alpha}+\kappa_{,\beta}\,\mathsf{N}_{j,\beta}\,\delta_{j_c\alpha} \\ &= \kappa\,\mathsf{N}_{j,\alpha j_c}+\kappa_{,j_c}\,\mathsf{N}_{j,\alpha}+\delta_{j_c\alpha}\sum_{\beta=0}^{d-1}\big(\kappa\,\mathsf{N}_{j,\beta\beta}+\kappa_{,\beta}\,\mathsf{N}_{j,\beta}\big) \end{aligned}$$

$$\Big[(a\cdot\nabla)\boldsymbol{\varphi}^{\mathbf{v}}_I\Big]_\alpha=\delta_{i_c\alpha}\sum_{\beta=0}^{d-1}a_\beta\,\mathsf{N}_{i,\beta}$$

$$\begin{aligned} a^e_p(i,i_c;j,j_c) &= w_p\sum_{\alpha=0}^{N_{\mathrm{c}}-1}\Big[(a\cdot\nabla)\boldsymbol{\varphi}_I\Big]_\alpha(\mathbf{x}_p)\;\Big[\nabla\cdot\kappa\big(\nabla^{\mathrm{T}}\boldsymbol{\varphi}_J+\nabla\boldsymbol{\varphi}_J\big)\Big]_\alpha(\mathbf{x}_p) \\ &= \begin{cases} w_p\Big[\displaystyle\sum_{\beta=0}^{d-1}a_\beta(\mathbf{x}_p)\,\mathsf{N}_{i,\beta}(\mathbf{x}_p)\Big]\Big[\kappa(\mathbf{x}_p)\,\mathsf{N}_{j,i_cj_c}(\mathbf{x}_p)+\kappa_{,j_c}(\mathbf{x}_p)\,\mathsf{N}_{j,i_c}(\mathbf{x}_p)\Big] & \text{if } i_c\neq j_c \\ w_p\Big[\displaystyle\sum_{\beta=0}^{d-1}a_\beta(\mathbf{x}_p)\mathsf{N}_{i,\beta}(\mathbf{x}_p)\Big]\;\Big[\kappa(\mathbf{x}_p)\,\mathsf{N}_{j,i_cj_c}(\mathbf{x}_p)+\kappa_{,j_c}(\mathbf{x}_p)\,\mathsf{N}_{j,i_c}(\mathbf{x}_p) & \\ \qquad\qquad +\displaystyle\sum_{\beta=0}^{d-1}\kappa\,\mathsf{N}_{j,\beta\beta}(\mathbf{x}_p)+\kappa_{,\beta}(\mathbf{x}_p)\,\mathsf{N}_{j,\beta}(\mathbf{x}_p)\Big] & \text{if } i_c=j_c \end{cases} \end{aligned}$$

III.2.16 `add_vvgrad_row`

$$\boxed{\mathrm{c}_1}\quad \int_{\mathcal{C}_e} b\cdot\Big[(a\cdot\nabla)\boldsymbol{\varphi}^{\mathbf{v}}_I\Big]\,d\mathcal{C}_e \quad \text{with} \quad b\colon \mathcal{C}_e\to\mathbb{R}^{N^{\mathbf{v}}_{\mathrm{c}}} \quad \text{and} \quad a\colon \mathcal{C}_e\to\mathbb{R}^d$$

$$\boxed{\mathrm{c}_2}\quad i\in[0,N^{\mathbf{v}}_e[\qquad i_c\in[0,N_{\mathrm{c}}[$$

$$\boxed{\mathrm{c}_3}\quad \begin{aligned} f^e_p(i,i_c) &= w_p\sum_{\alpha=0}^{N^{\mathbf{v}}_{\mathrm{c}}-1}b_\alpha(\mathbf{x}_p)\Big[\sum_{\beta=0}^{d-1}a_\beta(\mathbf{x}_p)\,\mathsf{N}^{\mathbf{v}}_{i,\beta}(\mathbf{x}_p)\Big]\,\delta_{i_c\alpha} \\ &= w_p\,b_{i_c}(\mathbf{x}_p)\sum_{\beta=0}^{d-1}a_\beta(\mathbf{x}_p)\,\mathsf{N}^{\mathbf{v}}_{i,\beta}(\mathbf{x}_p) \end{aligned}$$

III.2.17 add_row_div_col

$\boxed{\mathrm{c}_1}$ $\displaystyle\int_{\mathcal{C}_e} \kappa\, \boldsymbol{\varphi}^{\mathbf{v}}_I\, \nabla\cdot\boldsymbol{\varphi}^{\mathbf{u}}_J \, d\mathcal{C}_e \quad \text{with} \quad N^{\mathbf{v}}_{\mathrm{c}} = 1 \;, \quad N^{\mathbf{u}}_{\mathrm{c}} = d \quad \text{and} \quad \kappa\colon \mathcal{C}_e \to \mathbb{R}$

$\boxed{\mathrm{c}_2}$ $i \in [0, N^{\mathbf{v}}_e[\quad j \in [0, N^{\mathbf{u}}_e[\quad i_c = 0,\, j_c \in [0, d[$

$\boxed{\mathrm{c}_3}$ $a^e_p(i, i_c; j, j_c) \;=\; w_p\, \kappa(\mathbf{x}_p)\, \mathsf{N}^{\mathbf{v}}_i(\mathbf{x}_p)\, \mathsf{N}^{\mathbf{u}}_{j,j_c}(\mathbf{x}_p)$

III.2.18 add_row_grad_col

$\boxed{\mathrm{c}_1}$ $\displaystyle\int_{\mathcal{C}_e} \kappa\, \boldsymbol{\varphi}^{\mathbf{v}}_I \cdot \nabla\boldsymbol{\varphi}^{\mathbf{u}}_J \, d\mathcal{C}_e \quad \text{with} \quad N^{\mathbf{v}}_{\mathrm{c}} = d \;, \quad N^{\mathbf{u}}_{\mathrm{c}} = 1 \quad \text{and} \quad \kappa\colon \mathcal{C}_e \to \mathbb{R}$

$\boxed{\mathrm{c}_2}$ $i \in [0, N^{\mathbf{v}}_e[\quad j \in [0, N^{\mathbf{u}}_e[\quad i_c \in [0, d[,\, j_c = 0$

$\boxed{\mathrm{c}_3}$ $a^e_p(i, i_c; j, j_c) \;=\; w_p\, \kappa(\mathbf{x}_p)\, \mathsf{N}^{\mathbf{v}}_i(\mathbf{x}_p)\, \mathsf{N}^{\mathbf{u}}_{j,i_c}(\mathbf{x}_p)$

III.2.19 add_div_row_div_col

$\boxed{\mathrm{c}_1}$ $\displaystyle\int_{\mathcal{C}_e} \kappa\, \nabla\cdot\boldsymbol{\varphi}^{\mathbf{v}}_I\; \nabla\cdot\boldsymbol{\varphi}^{\mathbf{u}}_J \, d\mathcal{C}_e \quad \text{with} \quad N^{\mathbf{v}}_{\mathrm{c}} = N^{\mathbf{u}}_{\mathrm{c}} = d \quad \text{and} \quad \kappa\colon \mathcal{C}_e \to \mathbb{R}$

$\boxed{\mathrm{c}_2}$ $i \in [0, N^{\mathbf{v}}_e[\quad j \in [0, N^{\mathbf{u}}_e[\quad i_c, j_c \in [0, d[$

$\boxed{\mathrm{c}_3}$ $a^e_p(i, i_c; j, j_c) \;=\; w_p\, \kappa(\mathbf{x}_p)\, \mathsf{N}^{\mathbf{v}}_{i,i_c}(\mathbf{x}_p)\, \mathsf{N}^{\mathbf{u}}_{j,j_c}(\mathbf{x}_p)$

III.2.20 add_row_vv_col

$\boxed{\mathrm{c}_1}$ $\displaystyle\int_{\mathcal{C}_e} \kappa\, \boldsymbol{\varphi}^{\mathbf{v}}_I\; (a\cdot\boldsymbol{\varphi}^{\mathbf{u}}_J) \, d\mathcal{C}_e \quad \text{with} \quad N^{\mathbf{v}}_{\mathrm{c}} = 1 \;, \quad a\colon \mathcal{C}_e \to \mathbb{R}^{N^{\mathbf{u}}_{\mathrm{c}}} \quad \text{and} \quad \kappa\colon \mathcal{C}_e \to \mathbb{R}$

$\boxed{\mathrm{c}_2}$ $i \in [0, N^{\mathbf{v}}_e[\quad j \in [0, N^{\mathbf{u}}_e[\quad i_c = 0 \quad j_c \in [0, N^{\mathbf{u}}_{\mathrm{c}}[$

$\boxed{\mathrm{c}_3}$
$$\begin{aligned} a^e_p(i, i_c; j, j_c) \;&=\; w_p\, \kappa(\mathbf{x}_p)\, \mathsf{N}^{\mathbf{v}}_i(\mathbf{x}_p) \sum_{\alpha=0}^{N^{\mathbf{u}}_{\mathrm{c}}-1} a_\alpha(\mathbf{x}_p)\, \mathsf{N}^{\mathbf{u}}_j(\mathbf{x}_p)\, \delta_{j_c\alpha} \\ &=\; w_p\, \kappa(\mathbf{x}_p)\, \mathsf{N}^{\mathbf{v}}_i(\mathbf{x}_p)\, a_{j_c}(\mathbf{x}_p)\, \mathsf{N}^{\mathbf{u}}_j(\mathbf{x}_p) \end{aligned}$$

III.2.21 add_lapl_row_vvgrad_col

$\boxed{\mathrm{c}_1}$ $\displaystyle\int_{\mathcal{C}_e} \kappa\, \nabla^2\boldsymbol{\varphi}^{\mathbf{v}}_I \cdot \Big[(a\cdot\nabla)\boldsymbol{\varphi}^{\mathbf{u}}_J\Big] \, d\mathcal{C}_e \quad \text{with} \quad N^{\mathbf{v}}_{\mathrm{c}} = N^{\mathbf{u}}_{\mathrm{c}} \overset{\text{def}}{=} N_{\mathrm{c}} \;, \quad \kappa\colon \mathcal{C}_e \to \mathbb{R} \quad \text{and} \quad a\colon \mathcal{C}_e \to \mathbb{R}^d$

$\boxed{\mathrm{c}_2}$ $i \in [0, N^{\mathbf{v}}_e[\quad j \in [0, N^{\mathbf{u}}_e[\quad i_c, j_c \in [0, N_{\mathrm{c}}[$

$\boxed{\mathrm{c}_3}$
$$\begin{aligned} a^e_p(i, i_c; j, j_c) \;&=\; w_p\, \kappa(\mathbf{x}_p) \sum_{\alpha=0}^{N_{\mathrm{c}}-1} \Big[\sum_{\beta=0}^{d-1} \mathsf{N}^{\mathbf{v}}_{i,\beta\beta}(\mathbf{x}_p)\Big]\, \delta_{i_c\alpha} \Big[\sum_{\beta=0}^{d-1} a_\beta(\mathbf{x}_p)\, \mathsf{N}^{\mathbf{u}}_{j,\beta}(\mathbf{x}_p)\Big]\, \delta_{j_c\alpha} \\ &=\; \begin{cases} 0 & \text{if } i_c \neq j_c \\ w_p\, \kappa(\mathbf{x}_p) \Big[\displaystyle\sum_{\beta=0}^{d-1} \mathsf{N}^{\mathbf{v}}_{i,\beta\beta}(\mathbf{x}_p)\Big] \Big[\displaystyle\sum_{\beta=0}^{d-1} a_\beta(\mathbf{x}_p)\, \mathsf{N}^{\mathbf{u}}_{j,\beta}(\mathbf{x}_p)\Big] & \text{if } i_c = j_c \end{cases} \end{aligned}$$

III.2.22 add_row_lapl_col

$\boxed{\mathrm{c}_1}$ $\displaystyle\int_{\mathcal{C}_e} \kappa\, \boldsymbol{\varphi}^{\mathbf{v}}_I \cdot \nabla^2\boldsymbol{\varphi}^{\mathbf{u}}_J \, d\mathcal{C}_e \quad \text{with} \quad N^{\mathbf{v}}_{\mathrm{c}} = N^{\mathbf{u}}_{\mathrm{c}} \overset{\text{def}}{=} N_{\mathrm{c}} \quad \text{and} \quad \kappa\colon \mathcal{C}_e \to \mathbb{R}$

$\boxed{\mathrm{c}_2}$ $i \in [0, N^{\mathbf{v}}_e[\quad j \in [0, N^{\mathbf{u}}_e[\quad i_c, j_c \in [0, N_{\mathrm{c}}[$

$$\boxed{\mathrm{c}_3} \quad a_p^e(i,i_c;j,j_c) \;=\; w_p\;\kappa(\mathbf{x}_p) \sum_{\alpha=0}^{N_\mathrm{c}-1} \boldsymbol{\mathsf{N}}_i^{\mathbf{v}}(\mathbf{x}_p)\;\delta_{i_c\alpha} \left[\sum_{\beta=0}^{d-1} \boldsymbol{\mathsf{N}}^{\mathbf{u}}_{j,\beta\beta}(\mathbf{x}_p)\right] \delta_{j_c\alpha}$$

$$= \begin{cases} 0 & \text{if } i_c \neq j_c \\ w_p\;\kappa(\mathbf{x}_p)\;\boldsymbol{\mathsf{N}}_i^{\mathbf{v}}(\mathbf{x}_p) \displaystyle\sum_{\beta=0}^{d-1} \boldsymbol{\mathsf{N}}^{\mathbf{u}}_{j,\beta\beta}(\mathbf{x}_p) & \text{if } i_c = j_c \end{cases}$$

III.2.23 `add_lapl_row_lapl_col`

$$\boxed{\mathrm{c}_1} \quad \int_{\mathcal{C}_e} \kappa\;\nabla^2\boldsymbol{\varphi}_I^{\mathbf{v}} \cdot \nabla^2\boldsymbol{\varphi}_J^{\mathbf{u}}\;d\mathcal{C}_e \quad \text{with} \quad N_\mathrm{c}^{\mathbf{v}} = N_\mathrm{c}^{\mathbf{u}} \overset{\text{def}}{=} N_\mathrm{c} \quad \text{and} \quad \kappa\colon \mathcal{C}_e \to \mathbb{R}$$

$$\boxed{\mathrm{c}_2} \quad i \in [0, N_e^{\mathbf{v}}[\quad j \in [0, N_e^{\mathbf{u}}[\quad i_c, j_c \in [0, N_\mathrm{c}[$$

$$\boxed{\mathrm{c}_3} \quad a_p^e(i,i_c;j,j_c) \;=\; w_p\;\kappa(\mathbf{x}_p) \sum_{\alpha=0}^{N_\mathrm{c}-1} \left[\sum_{\beta=0}^{d-1} \boldsymbol{\mathsf{N}}^{\mathbf{v}}_{i,\beta\beta}(\mathbf{x}_p)\right] \delta_{i_c\alpha} \left[\sum_{\beta=0}^{d-1} \boldsymbol{\mathsf{N}}^{\mathbf{u}}_{j,\beta\beta}(\mathbf{x}_p)\right] \delta_{j_c\alpha}$$

$$= \begin{cases} 0 & \text{if } i_c \neq j_c \\ w_p\;\kappa(\mathbf{x}_p) \left[\displaystyle\sum_{\beta=0}^{d-1} \boldsymbol{\mathsf{N}}^{\mathbf{v}}_{i,\beta\beta}(\mathbf{x}_p)\right] \left[\displaystyle\sum_{\beta=0}^{d-1} \boldsymbol{\mathsf{N}}^{\mathbf{u}}_{j,\beta\beta}(\mathbf{x}_p)\right] & \text{if } i_c = j_c \end{cases}$$

III.2.24 `add_lapl_row_col`

$$\boxed{\mathrm{c}_1} \quad \int_{\mathcal{C}_e} \kappa\;\nabla^2\boldsymbol{\varphi}_I^{\mathbf{v}} \cdot \boldsymbol{\varphi}_J^{\mathbf{u}}\;d\mathcal{C}_e \quad \text{with} \quad N_\mathrm{c}^{\mathbf{v}} = N_\mathrm{c}^{\mathbf{u}} \overset{\text{def}}{=} N_\mathrm{c} \quad \text{and} \quad \kappa\colon \mathcal{C}_e \to \mathbb{R}$$

$$\boxed{\mathrm{c}_2} \quad i \in [0, N_e^{\mathbf{v}}[\quad j \in [0, N_e^{\mathbf{u}}[\quad i_c, j_c \in [0, N_\mathrm{c}[$$

$$\boxed{\mathrm{c}_3} \quad a_p^e(i,i_c;j,j_c) \;=\; w_p\;\kappa(\mathbf{x}_p) \sum_{\alpha=0}^{N_\mathrm{c}-1} \left[\sum_{\beta=0}^{d-1} \boldsymbol{\mathsf{N}}^{\mathbf{v}}_{i,\beta\beta}(\mathbf{x}_p)\right] \delta_{i_c\alpha}\;\boldsymbol{\mathsf{N}}_j^{\mathbf{u}}(\mathbf{x}_p)\;\delta_{j_c\alpha}$$

$$= \begin{cases} 0 & \text{if } i_c \neq j_c \\ w_p\;\kappa(\mathbf{x}_p)\;\boldsymbol{\mathsf{N}}_j^{\mathbf{u}}(\mathbf{x}_p) \displaystyle\sum_{\beta=0}^{d-1} \boldsymbol{\mathsf{N}}^{\mathbf{v}}_{i,\beta\beta}(\mathbf{x}_p) & \text{if } i_c = j_c \end{cases}$$

III.2.25 `add_lapl_row`

$$\boxed{\mathrm{c}_1} \quad \int_{K_e} a \cdot \nabla^2\boldsymbol{\varphi}_I^{\mathbf{v}}\;dK_e \quad \text{with} \quad a\colon K_e \to \mathbb{R}^{N_\mathrm{c}^{\mathbf{v}}}$$

$$\boxed{\mathrm{c}_2} \quad i \in [0, N_e^{\mathbf{v}}[\quad i_c \in [0, N_\mathrm{c}[$$

$$\boxed{\mathrm{c}_3} \quad f_p^e(i,i_c) \;=\; w_p \sum_{\alpha=0}^{N_\mathrm{c}^{\mathbf{v}}-1} a_\alpha(\mathbf{x}_p) \left[\sum_{\beta=0}^{d-1} \boldsymbol{\mathsf{N}}^{\mathbf{v}}_{i,\beta\beta}(\mathbf{x}_p)\right] \delta_{i_c\alpha}$$

$$= w_p\;a_{i_c}(\mathbf{x}_p) \sum_{\beta=0}^{d-1} \boldsymbol{\mathsf{N}}^{\mathbf{v}}_{i,\beta\beta}(\mathbf{x}_p)$$

Chapter IV

Multilevel Adaptive Local Refinement

IV.1 Selection of a Refinement Method

IV.1.1 Wishes

- no modification of the discrete problem due to the meshing adaptation
- cells divided into cells of the same type, uniformly applying the same subdivision pattern
- implicit handling of the possible non-conformity of the adapted meshing
- no particular treatment due to particular finite elements (*eg* $\mathbb{P}_1, \mathbb{P}_2, \mathbb{Q}_1, \mathbb{Q}_2$)
- no particular treatment due to the number of dimensions

IV.1.2 Choice

CHARMS : Conforming, Hierarchical Adaptive Refinement MethodS [1] [2] [3]

IV.2 Multilevel Finite Element Interpolation

IV.2.1 Triangulation

subdivision of $\Omega \subset \mathbb{R}^d$ into polyhedra $\mathcal{C}_e \subset \mathbb{R}^d$ with non overlapping interiors

$$\overline{\Omega} = \bigcup_e \mathcal{C}_e \qquad \mathcal{T} = \{\mathcal{C}_e\} \qquad \mathcal{C}_e \text{ called } \textit{cell}$$

IV.2.2 Lagrange Finite Elements

- $\mathcal{T}_j$ a *conforming* triangulation of Ω : any face of any polyhedron $\mathcal{C}_e$ is either a face of another polyhedron or a subset of the boundary Γ
- X_j a Lagrange finite element space that is $\mathrm{H}^1(\Omega)$ *conforming*

$$X_j = \mathrm{span}\{\, \varphi_k^{[j]} \,;\, k = 1, \ldots, N_{\mathrm{dof}}^{[j]} \,\} \;\subset\; \mathrm{H}^1(\Omega)$$

- every basis function $\varphi_k^{[j]}$
  - is associated to a geometrical node $\mathbf{a}_k^{[j]}$: $\varphi_k^{[j]}(\mathbf{a}_\ell^{[j]}) = \delta_{k\ell}$
  - vanishes outside the cells containing $\mathbf{a}_k^{[j]}$: $\mathrm{supp}[\varphi_k^{[j]}] = \{\text{cells containing } \mathbf{a}_k^{[j]}\}$
  - is defined by its restriction to these cells

IV.2.3 Finite Element Spaces

Ω a polyhedral domain

- $\mathcal{T}_j = \bigcup\limits_e \mathcal{C}_e^{[j]}$ a **conforming** triangulation
- A Lagrange finite element $(\mathcal{C}_e^{[j]}, P_e^{[j]}, \Sigma_e^{[j]})$ is defined on each cell from a reference finite element $(\widehat{\mathcal{C}}, \widehat{P}, \widehat{\Sigma})$ and a mapping $F_e^{[j]} \colon \widehat{\mathcal{C}} \to \mathcal{C}_e^{[j]}$:
  - $\widehat{\Sigma} = \{\widehat{\mathbf{a}}_k\}_k$ a set of N distinct points of $\widehat{\mathcal{C}}$
  - $\Sigma_e^{[j]} = \{\mathbf{a}_{ek}^{[j]}\}_k$ with $\mathbf{a}_{ek}^{[j]} = F_e^{[j]}(\widehat{\mathbf{a}}_k)$
  - $\widehat{P} = \{\widehat{\varphi}_k \colon \widehat{\mathcal{C}} \to \mathbb{R}\}_k$ a set of N polynomial functions such that $\widehat{\varphi}_k(\widehat{\mathbf{a}}_\ell) = \delta_{k\ell}$
  - $P_e^{[j]} = \{\varphi_{ek}^{[j]} \colon \mathcal{C}_e^{[j]} \to \mathbb{R} \mid \varphi_{ek}^{[j]} \circ F_e^{[j]} \in \widehat{P}\}$ with the property: $\varphi_{ek}^{[j]}(\mathbf{a}_{e\ell}^{[j]}) = \delta_{k\ell}$
- $X_j = \{v \in \mathrm{C}^0(\overline{\Omega}) \mid \forall \mathcal{C}_e^{[j]} \in \mathcal{T}_j \quad v|_{\mathcal{C}_e^{[j]}} \in P_e^{[j]}\}$
- $\Sigma^{[j]} = \bigcup\limits_e \Sigma_e^{[j]} = \{\mathbf{a}_k^{[j]} \ ; \ k = 1, \ldots, N_{\mathrm{dof}}^{[j]}\}$
- $X_j = \mathrm{span}\{\varphi_k^{[j]} \ ; \ k = 1, \ldots, N_{\mathrm{dof}}^{[j]}\}$

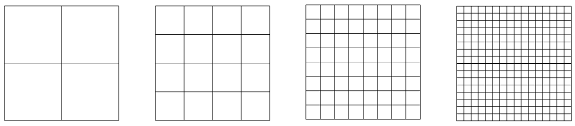
  $\varphi_k^{[j]}(\mathbf{a}_\ell^{[j]}) = \delta_{k\ell}$

  $\varphi_k^{[j]}$ defined by its restriction $\varphi_{ek}^{[j]}$ in each cell $\mathcal{C}_e^{[j]}$ which contains the node $\mathbf{a}_\ell^{[j]}$

IV.2.4 Conceptual Hierarchy of Nested Conforming FE Spaces

$$\begin{array}{lcccll}
 & \text{triangulation} & \text{finite element space} & & & \text{basis function set} \\
\text{level } 0 & \mathcal{T}_0 & X_0 & = & \text{span } \mathcal{B}_0 & \mathcal{B}_0 = \{\varphi_k^{[0]}; k = 1, \ldots, N_{\mathrm{dof}}^{[0]}\} \\
 & & \cap & & & \\
\text{level } 1 & \mathcal{T}_1 & X_1 & = & \text{span } \mathcal{B}_1 & \mathcal{B}_1 = \{\varphi_k^{[1]}; k = 1, \ldots, N_{\mathrm{dof}}^{[1]}\} \\
 & & \cap & & & \\
\vdots & \vdots & \vdots & & \vdots & \\
 & & \cap & & & \\
\text{level } J & \mathcal{T}_J & X_J & = & \text{span } \mathcal{B}_J & \mathcal{B}_J = \{\varphi_k^{[J]}; k = 1, \ldots, N_{\mathrm{dof}}^{[J]}\}
\end{array}$$

- $\mathcal{T}_j \to \mathcal{T}_{j+1}$:
  uniform refinement
- X_j :
  conforming Lagrange



▷ $\mathcal{T}_j$ is the same for all the unknown fields whereas there may be a different X_j for each of them.

IV.2.5 Multilevel Finite Element Spaces

General Framework

- **Select** some functions of $\mathcal{B}_j$, thus defining a subset $\widehat{\mathcal{B}}_j \subset \mathcal{B}_j$ by a process that guaranties the linear independence of $\cup_{j=0}^{J} \widehat{\mathcal{B}}_j$.

Elements of $\mathcal{B}_j$ that belong to $\widehat{\mathcal{B}}_j$ are qualified: *active*.

- Define the **multilevel** *basis function set*: $\mathcal{B} = \bigcup_{j=0}^{J} \widehat{\mathcal{B}}_j$
- Define the **multilevel** *finite element space*: $\mathcal{V}_h = \operatorname{span} \mathcal{B} \subset \mathcal{V}$

Multilevel Finite Element Expansion

- $u_h \in \mathcal{V}_h$ is expanded as:

$$u_h(\mathbf{x}) = \sum_{\varphi \in \mathcal{B}} u_\varphi \varphi(\mathbf{x}) \qquad \text{or} \qquad u_h(\mathbf{x}) = \sum_{j=0}^{J} \Big\{ \sum_{k \,|\, \varphi_k^{[j]} \in \widehat{\mathcal{B}}_j} u_k^{[j]} \, \varphi_k^{[j]}(\mathbf{x}) \Big\}$$

- $u_k^{[j]} \in \mathbb{R}$ but $u_k^{[j]} \neq u_h(\mathbf{a}_k^{[j]})$

IV.3 Adaptation

IV.3.1 Principle

Starting form a (possibly multilevel) basis function set $\mathcal{B}^\star$ (*eg* $\mathcal{B}_0$), an adapted basis function set $\mathcal{B}$, with better approximation properties, is built through a succession of refinement/unrefinement steps.

$$\left.\begin{array}{ll} \text{Refinement:} & \textbf{remove} \text{ coarse functions} \\ & \textbf{replace} \text{ them by finer functions} \\ & \\ \text{Unrefinement:} & \textbf{remove} \text{ fine functions} \\ & \textbf{replace} \text{ them by coarser functions} \end{array}\right\} \begin{array}{c} \textbf{activation/deactivation} \\ \text{of basis functions} \end{array}$$

basic principle of the adaptation procedure:

refine/unrefine **primarily** basis functions instead of cells

IV.3.2 Basis Function Refinement/Unrefinement

Refinement Equations

$$X_j \subset X_{j+1} \quad \Rightarrow \quad \varphi_k^{[j]}(\mathbf{x}) = \sum_{\ell} \beta_{k\ell}^{[j+1]} \varphi_\ell^{[j+1]}(\mathbf{x}) \qquad \forall \mathbf{x} \in \Omega$$

Child-Parent Relationship

- refinement set of $\varphi_k^{[j]}$: $\mathcal{R}(\varphi_k^{[j]}) = \Big\{ \varphi_\ell^{[j+1]} \;\Big|\; \beta_{k\ell}^{[j+1]} \neq 0 \Big\}$
- if $\varphi_\ell^{[j+1]} \in \mathcal{R}(\varphi_k^{[j]})$: $\varphi_k^{[j]}$ is a *parent* of $\varphi_\ell^{[j+1]}$
  $\varphi_\ell^{[j+1]}$ is a *child* of $\varphi_k^{[j]}$

One-Level-Difference Refinement Rule

- $\varphi_k^{[j]}$ may be refined only if all its parents on level $[j-1]$ have been refined.
- $\varphi_k^{[j]}$ may be unrefined only if:
  - it was previously refined
  - none of its children on level $[j+1]$ are refined

IV.3.3 (Quasi-)Hierarchical Refinement/Unrefinement

Let $\mathcal{B}^\star$ be a (multilevel) basis function set.

Hierarchical Refinement/Unrefinement

- refinement of $\varphi_k^{[j]} \in \mathcal{B}^\star$

  activation of all children of $\varphi_\ell^{[j+1]}$ of $\varphi_k^{[j]}$ except the one such that $\mathbf{a}_\ell^{[j+1]} \equiv \mathbf{a}_k^{[j]}$
- unrefinement of a previously refined $\varphi_k^{[j]} \in \mathcal{B}^\star$

  deactivation of those children of $\varphi_k^{[j]}$ which have no other refined parent

Quasi-Hierarchical Refinement/Unrefinement

- refinement of $\varphi_k^{[j]} \in \mathcal{B}^\star$
  - deactivation of $\varphi_k^{[j]}$
  - activation of all children $\varphi_\ell^{[j+1]}$ of $\varphi_k^{[j]}$
- unrefinement of a previously refined $\varphi_k^{[j]} \notin \mathcal{B}^\star$
  - activation of $\varphi_k^{[j]}$
  - deactivation of those children of $\varphi_k^{[j]}$ which have no other active parent
- **preferred** compared to hierarchical: it generates **sparser** assembled matrices

IV.3.4 Reference Element Refinement

- cornerstone of the method : the *refinement equations*

$$\forall j \in [0, J] \qquad \forall k \in [1, N_{\text{dof}}^{[j]}] \qquad \varphi_k^{[j]} = \sum_\ell \beta_{k\ell}^{[j+1]} \, \varphi_\ell^{[j+1]}$$

- every $\varphi_k^{[j]} \in \mathcal{B}_j$ is defined by its restriction $\varphi_{ek}^{[j]}$ to the cells $\mathcal{C}_e^{[j]}$ of $\mathcal{T}_j$
- the basis functions restrictions $\varphi_{ek}^{[j]}$ are completely determined by:
  - the reference element basis functions $\widehat{\varphi}_{\hat{k}}$ defined over the reference cell $\widehat{\mathcal{C}}$
  - the mapping $F_e^{[j]} : \widehat{\mathcal{C}} \to \mathcal{C}_e^{[j]}$

The *refinement equations* are **all** deduced from the two-level *reference element* refinement equations:

$$\forall \hat{k} \in [1, \widehat{N}_{\text{dof}}] \qquad \widehat{\varphi}_{\hat{k}}^{[0]} = \sum_{\hat{\ell}} \widehat{\beta}_{\hat{k}\hat{\ell}} \, \widehat{\varphi}_{\hat{\ell}}^{[1]}$$

IV.4 Multilevel Variational Approximation

IV.4.1 Model Problem

given $u_h^\star \in \mathcal{V}_h^\star = \operatorname{span} \mathcal{B}^\star$ find $u_h \in \mathcal{V}_h = \operatorname{span} \mathcal{B}$ such that:

$$\forall v \in \mathcal{V}_h \qquad \int_\Omega \frac{u_h - u_h^\star}{\Delta t} \cdot v + \int_\Omega \nabla u_h : \nabla v = \int_\Omega f \cdot v$$

where:

- $\mathcal{B}^\star$ and $\mathcal{B}$ are multilevel basis function sets;
- $\mathcal{B}$ has been obtained from $\mathcal{B}^\star$ by some steps of refinement/unrefinement.

IV.4.2 Active Degrees Of Freedom

- active DOF of a discrete field: pair (j,k) s.t. $\varphi_k^{[j]}$ occurs in its multilevel expansion.
- $\mathcal{A}_{\text{dof}} = \left\{ (j,k) \,\middle|\, (j,k) \text{ is an active DOF of at least one discrete field} \right\}$

Observations

$\boxed{1}$ variational approximation is posed on $\mathcal{V}_h = \operatorname{span} \mathcal{B}$ (usual formulation)

$\boxed{2}$ multiple discrete fields involved, either as unknowns or data

$\boxed{3}$ active DOFs: do not necessarily correspond to active basis functions of $\mathcal{B}$

IV.4.3 Active Cells

▷ A cell $\mathcal{C}_e^{[j]}$ of $\mathcal{T}_j$ is called *active* iff:

- $\exists (k,j) \in \mathcal{A}_{\text{dof}}$ such that $\mathcal{C}_e^{[j]} \bigcap \operatorname{supp}[\varphi_k^{[j]}] \neq \emptyset$
- $\forall (k,j') \in \mathcal{A}_{\text{dof}} \quad j' > j \quad \Rightarrow \quad \mathcal{C}_e^{[j]} \bigcap \operatorname{supp}[\varphi_k^{[j']}] = \emptyset$

▷ The *multilevel triangulation* $\mathcal{T}$ is defined from the set of all active DOFs $\mathcal{A}_{\text{dof}}$:

- $\widehat{\mathcal{T}}_j$ = set of active cells of $\mathcal{T}_j$
- $\mathcal{T} = \bigcup_{j=0}^{J} \widehat{\mathcal{T}}_j$

- multilevel triangulations are not conforming, but
- multilevel finite element spaces are conforming **by construction**

$$\operatorname{span} \mathcal{B} \subset X_J$$

▷ Active cells are

- elementary integration domains
- part of the piecewise definition of basis functions

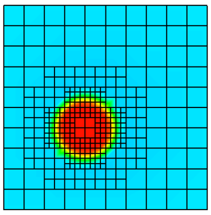
▷ Integrals on an active cell $\mathcal{C}$ involving any basis function associated to an active DOF, *eg*:

$$\int_{\mathcal{C}} \varphi \cdot \varphi^\star \qquad \forall \varphi \in \mathcal{B} \qquad \forall \varphi^\star \in \mathcal{B}^\star$$

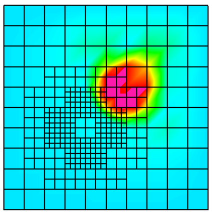
can be computed **exactly**.

no need of *transfer operators* for fields defined on two distinct refined cells resulting from successive refinements of the same initial meshing

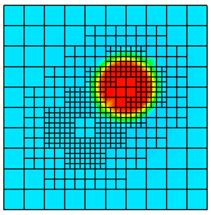
▷ Example (manufactured solution)



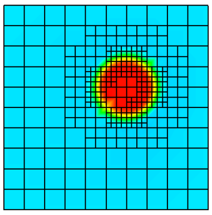
$u_h^\star$



computation of u_h
first adaptation step



computation of u_h
final adaptation step



$u_h^\star \leftarrow u_h$

IV.4.4 Refinement/Unrefinement Criterion

Let $\mathcal{B}^\star$ be a multilevel basis function set, and $\mathcal{T}^\star$ an associated multilevel triangulation.

Adaptation of $\mathcal{B}^\star$: refinement/unrefinement of basis functions determined by
a *per-basis-function-indicator*.

$\boxed{1}$ A *per-cell-indicator* $\eta_{\mathcal{C}}$ is computed for all $\mathcal{C} \in \mathcal{T}^\star$.

$\boxed{2}$ For a basis function φ that is either

- *active* and *unrefined* basis functions (possible candidate for refinement), or
- *refined* basis functions (possible candidate for unrefinement)

the indicator η_φ is computed as follows:

$$\eta_\varphi = \frac{1}{|\mathrm{supp}[\varphi]|} \sum_{\mathcal{C} \in \mathrm{supp}[\varphi]} \xi(\mathcal{C}) \qquad \text{with} \quad \begin{cases} \text{if } \mathcal{C} \in \mathcal{T}^\star & \xi(\mathcal{C}) = |\mathcal{C}|\, \eta_{\mathcal{C}} \\ \text{else} & \xi(\mathcal{C}) = \displaystyle\sum_{\mathcal{C}_e \text{ child of } \mathcal{C}} \xi(\mathcal{C}_e) \end{cases}$$

Bibliography

[1] Petr Krysl, Abhishek Trivedi, and Baozhi Zhu. Object-oriented hierarchical mesh refinement with CHARMS. *International Journal for Numerical Methods in Engineering*, 60:1401–1424, 2004.

[2] Lance Endres and Petr Krysl. Octasection-based refinement of finite element approximations on tetrahedral meshes that guarantees shape quality. *International Journal for Numerical Methods in Engineering*, 59(1):69–82, 2004.

[3] P. Krysl, E. Grinspun, and P. Schröder. Natural hierarchical refinement for finite element methods. *International Journal for Numerical Methods in Engineering*, 56(8):1109–1124, 2003.

Chapter V

Hierarchical Data Structures for the Definition of Discretizations

V.1 `MODULE PDE_DomainAndFields`

V.1.1 Example

```
MODULE PDE_DomainAndFields
   verbose_level = 1
   nb_space_dimensions = 3
   type = "finite_element"
   MODULE GE_Meshing
      ...
   END MODULE GE_Meshing
   MODULE interior_fields
      ...
   END MODULE interior_fields
   MODULE boundary_fields
      ...
   END MODULE boundary_fields
   MODULE PDE_ResultSaver
      ...
   END MODULE PDE_ResultSaver
END MODULE PDE_DomainAndFields
```

V.1.2 Entries

`type` : type of discrete representation chosen for the unknown fields

- data type: **`PEL_Data::String`**
- only possible value: **`"finite_element"`**
- mandatory

`nb_space_dimensions` : geometrical dimension d of the computational domain $\Omega \subset \mathbb{R}^d$

- data type: **`PEL_Data::Int`**
- only possible values: **`1`**, **`2`**, **`3`**
- mandatory

`verbose_level` : level of verbosity

- data type: **`PEL_Data::Int`**
- value **`0`** : silent output. Domain and fields building process will be briefly described.
- value **`1`** : Domain and fields building process will generate processing traces on the standard output

- mandatory

verbose_level : for multidomain computations, name of the current **PDE_DomainAndFields** instance

- data type: **PEL_Data::String**
- optional

V.1.3 Submodules

MODULE GE_Meshing (§V.13) : geometrical definition of the *meshing*

- mandatory

MODULE macro_colors (§V.8) : definition of composite colors from pre-existent non composite colors

- optional

MODULE interior_fields (§V.2) : definition of discrete representations of fields defined in the interior of the geometrical domain

- mandatory

MODULE boundary_fields (§V.3) : definition of discrete representations of fields defined in the interior of the geometrical domain

- optional

MODULE boundary_conditions (§V.9) : definition of the boundary conditions that are not taken into account through **imposed** values of the **DOF**s

- optional
- mutually exclusive with **MODULE macro_boundary_conditions**

MODULE macro_boundary_conditions (§V.10) : definition of the boundary conditions that are not taken into account through **imposed** values of the **DOF**s

- optional
- mutually exclusive with **MODULE boundary_conditions**

MODULE PDE_ResultSaver (§V.11) : definition of the savings for subsequent post-processing

- optional

V.2 **MODULE interior_fields**

V.2.1 Example

The following Hierarchical Data Structure sets three discrete representations of unknown fields, called respectively **"velocity"**, **"pressure"** and **"temperature"**, defined on the interior of the geometrical domain attached to the current **PDE_DomainAndFields** instance:

```
MODULE interior_fields
   MODULE xxx
      name = "velocity"
      ...
   END MODULE xxx
   MODULE yyy
      name = "pressure"
      ...
   END MODULE yyy
   MODULE zzz
      name = "temperature"
      ...
   END MODULE zzz
END MODULE interior_fields
```

V.2.2 Submodules

The module **`interior_fields`** contains a generic list of submodules, each of them setting a discrete representation of an unknown field. (§V.4).

The name of these modules is not significant.

V.3 MODULE boundary_fields

V.3.1 Example

The following Hierarchical Data Structure sets two discrete representations of unknown fields, called respectively **`"lambda"`**, and **`"uu_0_bd"`**, defined on the boundary of the geometrical domain attached to the current **`PDE_DomainAndFields`** instance:

```
MODULE boundary_fields
   MODULE xxx
      name = "lambda"
      ...
   END MODULE xxx
   MODULE yyy
      name = "uu_0_bd"
      ...
   END MODULE yyy
END MODULE boundary_fields
```

V.3.2 Submodules

The module **`boundary_fields`** contains a generic list of submodules, each of them setting a discrete representation of an unknown field. (§V.4).

The name of these modules is not significant.

V.4 Generic Submodule of MODULE interior_fields

V.4.1 Example

The following Hierarchical Data Structure defines a one component field, named **`"temp"`**, on a 2D meshing made of cells having **`GE_ReferenceSquare`** as reference polyhedron, such that:

- the discrete representation is based on bi-quadratic finite elements;

- the **DOF**s values are initialized to $(\sin x + 3)$ where x denotes the coordinate **0** of the associated nodal geometrical point;
- 4 values of the whole discrete representation will be handled;
- an additional **imposed** value, uniformly set at **3.0**, will be handled for all **DOF**s located on bounds of color **"entry"**.

```
MODULE interior_fields
   MODULE t_Q2
      name = "temp"
      nb_components = 1
      element_name  = "PDE_2D_Q2_9nodes"
      storage_depth = 4
      MODULE DOFs_values
         type  = "uniformly_defined"
         value = vector( sin(component($DV_X,0)) + 3.0 )
      END MODULE DOFs_values
      MODULE DOFs_imposed_value
         MODULE entry_values
            location = "on_bounds"
            color    = "entry"
            type     = "uniformly_defined"
            value    = < 3.0 >
         END MODULE entry_values
      END MODULE DOFs_imposed_value
   END MODULE t_Q2
END MODULE interior_fields
```

V.4.2 Entries

name : name of the field

- data type: **PEL_Data::String**
- mandatory

nb_components : number of components of the field

- data type: **PEL_Data::Int**
- mandatory

storage_depth :

- data type: **PEL_Data::Int**
- mandatory

element_name :

- data type: **PEL_Data::String**
- optional
- mutually exclusive with **element_names**, one of both being requested

element_names :

- data type: **PEL_Data::StringVector**
- optional
- mutually exclusive with **element_name**, one of both being requested

V.4.3 Submodules

MODULE DOFs_values (§V.6) : initialization of the values of the **DOFs**

- mandatory

MODULE DOFs_imposed_value (§V.7) : setting of **imposed** value for some **DOFs**

- optional

V.5 Generic Submodule of **MODULE boundary_fields**

V.5.1 Examples

The following Hierarchical Data Structure defines a one component field, named **"bdL1"**, on the boundary of a 2D meshing, such that:

- the discrete representation is based on linear finite elements;
- the **DOF**s values are initialized to $\sqrt{xy}$ where x and y denote respectively the coordinate **0** and **1** of the associated nodal geometrical point;
- 2 values of the whole discrete representation will be handled.

```
MODULE boundary_fields
   MODULE bd_L1
      name = "bdl1"
      nb_components = 1
      element_name = "PDE_1D_P1_2nodes"
      storage_depth = 2
      MODULE DOFs_values
         type = "uniformly_defined"
         value = vector( sqrt( component($DV_X,0) * component($DV_X,1) ) )
      END MODULE DOFs_values
   END MODULE bd_L1
END MODULE boundary_fields
```

V.5.2 Entries and Submodules

Entries and submodules are the same as for **interior_fields** (§V.4) except for the **DOFs_imposed_value** module that must not appear (**imposed** values cannot be defined from the Hierarchical Data Structure).

V.6 **MODULE DOFs_values**

V.6.1 Examples

The following Hierarchical Data Structure leads to the initialization of all the **DOF**s of the current one component field to the value **0**.

```
MODULE DOFs_value
   type  = "uniformly_defined"
   value = < 0. >
END MODULE DOFs_value
```

The following Hierarchical Data Structure is related to a two components field and relies on the nodal geometric position (whose coordinates **0** and **1** will be respectively denoted x and y) : the **DOF**s of component 0 will be initialized to the value $\sqrt{2-xy}$ whereas the **DOF**s of component 1 will be initialized to the value $0.32 + x/2$.

```
$DS_x = component( $DV_X, 0 )
$DS_y = component( $DV_X, 1 )
MODULE DOFs_value
   type  = "uniformly_defined"
   value = vector( sqrt(2.-$DS_x*$DS_y),0.5*$DS_x+0.32 )
END MODULE DOFs_value
```

The following Hierarchical Data Structure leads to a zero initialization of all values of the **DOF**s except those associated to a nodal geometric position lying in a cell of color **"blue"**, in which case it is initialized to $2x$ where x denotes the coordinate **0** of that nodal geometric position.

```
MODULE DOFs_value
   type  = "mesh_defined"
   MODULE value
      default = <0.>
      blue = vector( 2.*component($DV_X,0) )
   END MODULE value
END MODULE DOFs_value
```

The following Hierarchical Data Structure leads to a piecewise initialization of the **DOF**s based on their associated nodal geometric position (whose coordinate **0** will be denoted x).

- If this latter is located inside a cell having at least one vertex of color **"red"**, an interpolation of values at the cell vertices will be used. For vertices of color **"red"**, this value will be 2, and for other vertices this value will $2V_x$ (where V_x is the coordinate **0** of the vertex).
- Otherwise, the initilization value will be $2x$.

```
MODULE DOFs_value
   type = "vertex_defined"
   MODULE value
      default = vector( 2.*component($DV_X,0) )
      red = <2.>
   END MODULE value
END MODULE DOFs_value
```

V.7 MODULE DOFs_imposed_value

V.8 MODULE macro_colors

V.8.1 Example

The following Hierarchical Data Structure leads to the definition of three new colors called respectively **"tops"**, **"bottoms"** and **"lefts"**.

```
MODULE macro_colors
   tops    = < "top_left" "top" "top_right" >
   bottoms = < "bottom_left" "bottom" "bottom_right" >
   lefts   = < "left" >
END MODULE macro_colors
```

V.8.2 Entries

The module **macro_colors** contains a generic list of entries, each of them with the following characteristics:

- keyword: any allowed sequence of characters that will further represent the name of the defined composite color
- data type: **PEL_Data::StringVector**
- description: name of the leaf colors (themselves must not be composite colors) that will define the composite color

V.9 MODULE boundary_conditions

The module **boundary_conditions** contains the definition of a sequence of **BC**s (§I.4.1), each of them being defined in one specialized module.

V.9.1 Example

The following Hierarchical Data Structure leads to the definition of two **BC**s (§I.4.1). The first one, named **"toto"**, is relative to the field **"temperature"** on bounds of color **"right"**. The second one, named **"titi"**, is relative to the second component of the field **"velocity"** on bounds of color **"Interface"**.

The content and the meaning of the modules **xxx** and **yyy** is left to the user's PELICANS-based application.

```
MODULE boundary_conditions
   MODULE xxx
      type = "toto"
      field = "temperature"
      color = "right"
      prefactor = 10.
      field_out_value = 573.
   END MODULE xxx
   MODULE yyy
      type = "titi"
      field = "velocity"
      component = 1
      color = "Interface"
      penalization_coefficient = 1.0E8
      param_normal_velocity = "vn"
   END MODULE yyy
END MODULE boundary_conditions
```

V.9.2 Submodules

The module **boundary_conditions** contains a generic list of submodules, each of them setting a **BC**. (§V.9.3). The name of these modules is not significant.

V.9.3 Entries of a Generic Submodule

type : name of the **BC**

- data type: **PEL_Data::String**
- mandatory

field : name of the field concerned by this **BC**

- data type: **PEL_Data::String**
- mandatory

component : field component concerned by this **BC** (in the absence of this entry, all field components are concerned)

- data type: **PEL_Data::Int**
- optional

color : color name of the bounds concerned by this **BC** (in the absence of this entry, all bounds are concerned)

- data type: **PEL_Data::String**

- optional

additional specific entries associated to the above **type**.

V.10 MODULE macro_boundary_conditions

The module **macro_boundary_conditions** contains the definition of a sequence of **BC**s (§I.4.1), each of them being defined by two enclosed modules (instead of a single module in the case of module **boundary_conditions**). The outer module, which sets up a so called **macro_BC** in the jargon of PELICANS, specifies:

- the first part of the **BC** identifier (*i.e* the **macro_BC** identifier),
- the color of the bound,

while the inner module specifies:

- the last part of the **BC** indentifier,
- the field concerned with the **BC** (an possibly its component),
- additional data specific to the **BC**.

The modules **boundary_conditions** and **macro_boundary_conditions** are mutually exclusive. The latter compared to the former allows a more thorough control over the relevance of the boundary conditions that are enforced to multiple fields on a given part of the domain boundary.

V.10.1 Example

The following Hierarchical Data Structure leads to the definition of two **BC**s (§I.4.1) on bounds of color **"right"**. The first one, identified by the combination **"wall"**–**"adiabatic"**, is relative to the field **"temperature"**. The second one, identified by the combination **"wall"**–**"friction"**, is relative to the field **"velocity"**.

The content and the meaning of the modules **bc#1** and **bc#2** is left to the user's PELICANS-based application.

```
MODULE macro_boundary_conditions
   MODULE mbc#1
      type = "wall"
      color = "right"
      MODULE bc#1
         type = "adiabatic"
         field = "temperature"
      END MODULE bc#1
      MODULE bc#2
         type = "friction"
         field = "velocity"
         friction_coefficient = 12.
      END MODULE bc#2
   END MODULE mbc#1
END MODULE macro_boundary_conditions
```

V.10.2 Submodules

The module **macro_boundary_conditions** contains:

a generic list of submodules, qualified outer (§V.10.3),

each of them containing another generic list of subsubmodules, qualified inner (§V.10.4).

The name of these modules is not significant.

V.10.3 Entries of a Generic Outer Submodule

type : identifier of the **macro_BC** and first part of the **BC** identifier

- data type: **PEL_Data::String**
- mandatory

color : color name of the bounds concerned by this **BC** (in the absence of this entry, all bounds are concerned)

- data type: **PEL_Data::String**
- optional

V.10.4 Entries of a Generic Inner Subsubmodule

type : last part of the **BC** identifier

- data type: **PEL_Data::String**
- mandatory

field : name of the field concerned by this **BC**

- data type: **PEL_Data::String**
- mandatory

component : field component concerned by this **BC** (in the absence of this entry, all field components are concerned)

- data type: **PEL_Data::Int**
- optional

additional specific entries associated to the above **type**.

V.11 MODULE PDE_ResultSaver

PDE_ResultSaver is aimed to organize the savings of the grids, the fields and additional variables. Actually output is made using some specific functionalities of the concrete sub-classes of **PEL_DataOnMeshingWriter** that **PDE_ResultSaver** calls. Those functionalities allow to write the saved entities in a suitable output format so that they can be used with an associated external postprocessor. Functionalities have to be activated using the **writers** entry.
Saving is organized in successive cycles. Each cycle will contain the values of the entities that have been declared to be saved. Saving of grid, fields and additional variables must be switch on in a PELICANS based application by calling directly when necessary some **PDE_ResultSaver}** functionalities as **start_cycle**, **save_grid**, **save_fields**, **save_variable** and **save_integration_domain**. One may note that any additional variable must be flagged by a character string, its name. Those of the fields to be saved when calling the **save_fields** functionality must be selected in the Hierarchical Data Structure to be detailed in this subsection.

V.11.1 Examples

The following Hierarchical Data Structure leads to the saving for subsequent post-processing of the three fields **"velocity"**, **"pressure"** and **"temperature**, respectively under the names **"VV"**, **"PP"** and **"TEMP"**.

```
MODULE PDE_ResultSaver
   writers = < "PEL_TICwriter" "PEL_GMVwriter" "PEL_OpenDXwriter" >
   writing_mode = "binary"
   files_basename = "save_new_result"
   MODULE xxx
      where_to_save = "at_vertices"
      entry_name = "VV"
      field = "velocity"
   END MODULE xxx
   MODULE yyy
      where_to_save = "at_cell_centers"
      entry_name = "PP"
      field = "pressure"
   END MODULE yyy
   MODULE zzz
      where_to_save = "at_vertices"
      entry_name = "TEMP"
      field = "temperature"
   END MODULE zzz
END MODULE PDE_ResultSaver
```

V.11.2 Entries

writers : list of concrete classes derived from **PEL_DataOnMeshingWriter** that will be responsible for producing output files with their specific format

- data type: **PEL_Data::StringVector**
- possible item: **"PEL_PelWriter"**, for outputs in the format of PELICANS Hierarchical Data Structures (generated files with the **.pel** or **.pel.bin** extensions)
- possible item: **"PEL_TICwriter"**, for subsequent post-processing with TIC (generated files with the **.gene** extension)
- possible item: **"PEL_GMVwriter"**, for subsequent post-processing with GMV (generated files with the **.gmv.*** extension)
- possible item: **"PEL_OpenDXwriter"**, for subsequent post-processing with OpenDX (generated files with the **.dx** extension)
- possible item: **"EXT_SiloWriter"**, for outputs in the silo format (generated files with the **.silo** extension)
- possible item: **"PEL_1Dwriter"**, for 1D calculations
- possible item: **"PEL_VTKwriter"**, for subsequent post-processing with paraview (generated files with the **.vtu** extension)
- mandatory

files_basename : base names used by **PEL_DataOnMeshingWriter** objects to build their output file names

- data type: **PEL_Data::String**
- mandatory

writing_mode : specification of the output file format, either text or ascii

- data type: **PEL_Data::String**
- 1-st possible value: **"text"**
- 2-nd possible value: **"binary"**
- mandatory

append_mode :

- data type: **PEL_Data::Bool**
- optional

V.11.3 Submodules

The module **PDE_ResultSaver** contains a generic list of submodules, each of them setting saving requests for a given field. (§V.11.4).

The name of these modules is not significant.

V.11.4 Entries of a Generic Submodule

field : name of the field concerned by this saving request

- data type: **PEL_Data::String**
- mandatory

entry_name : name used by the **PEL_DataOnMeshingWriter** objects to save the current field in their outputs

- data type: **PEL_Data::String**
- mandatory

where_to_save : definition of the geometrical points where the values of the current field have to be computed and saved

- data type: **PEL_Data::String**
- 1-st possible value: **"at_cell_centers"**
- 2-nd possible value: **"at_vertices"**
- mandatory

V.12 MODULE GE_Colorist

The class **GE_Colorist** is meant to define colors or change afterwards the current color of vertices, faces of cells.

V.12.1 Example

The following Hierarchical Data Structure

```
MODULE GE_Colorist
   MODULE vertices
      red = in_box( $DV_X, < 0.9 -0.1 >, < 1.1 0.6 > )
   END MODULE vertices
   MODULE cells
      red = in_box( $DV_X, < 0.0 0.5 >, < 0.5 1.0 > ) ||
            in_box( $DV_X, < 0.5 0.0 >, < 1.0 0.5 > )
      black = in_box( $DV_X, < 0.0 0.0 >, < 0.5 0.5 > ) ||
              in_box( $DV_X, < 0.5 0.5 >, < 1.0 1.0 > )
   END MODULE cells
   MODULE faces
      yellow = in_box( $DV_X, < -0.1 0.2 >, < 0.1 0.5 > )
   END MODULE faces
END MODULE GE_Colorist
```

leads to an afterwards assignment of

- the color of name **"red"** to the vertices located in $[0.9, 1] \times [-0.1, 0.6]$;
- the color of name **"red"** to the cells whose center is located in $[0, 0.5] \times [0.5, 1] \cup [0.5, 1] \times [0, 0.5]$;
- the color of name **"black"** to the cells whose center is located in $[0., 0.5] \times [0.0, 0.5] \cup [0.5, 1] \times [0.5, 1]$;

- the color of name **"yellow"** to the faces whose center is located in $[-0.1, 0.1] \times [0.2, 0.5]$.

These assignments of colors overrides the previously attached colors (that where set for example by **GE_Meshing** objects).

V.12.2 Submodules

MODULE vertices (§V.12.3) : afterwards assignment of colors to some vertices

- optional

MODULE cells (§V.12.3) : afterwards assignment of colors to some cells

- optional

MODULE faces (§V.12.3) : afterwards assignment of colors to some faces

- optional

V.12.3 Entries of MODULEs vertices, cells and faces

The modules **vertices**, **cells** and **faces** contain a generic list of entries, each of them with the following characteristics:

- keyword: name of the color to be assigned
- data: an expression of type **PEL_Data::Bool** depending of the variable **$DV_X**

For each of these entries, and for all vertices (resp. cells) (resp. faces):

- the value of variable **$DV_X** of the vertex (resp. the center of the cell) (resp. the center of the face);
- the expression is evaluated;
- if the evaluation result is **true**, the color of name the keyword is assigned to the vertex (resp. cell) (resp. face).

V.13 MODULE GE_Meshing

Meshing functionalities are provided by objects of the concrete classes derived of **GE_Meshing**.

V.13.1 Entries

concrete_name : name of the concrete class derived from **GE_Meshing**

- data type: **PEL_Data::String**
- only possible values:
- mandatory

additional specific entries associated to the above **concrete_name**.

V.13.2 Submodules

MODULE GE_Colorist (§V.12) : afterward definition or re-definition of colors

- optional

other MODULEs associated to the above **concrete_name**.

V.14 MODULE GE_Meshing concrete_name="GE_BoxWithBoxes"

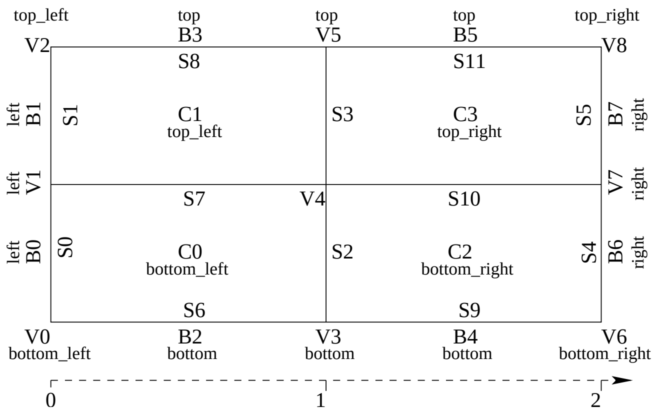
The **GE_BoxWithBoxes** class is devoted to the triangulation of geometrical domains Ω having initially the shape of a box with cells themselves having the shape of boxes. What is meant here by "box" is *an orthogonal parallelotope whose bounds are parallel to the coordinate axes.* Thus a 1D box is a segment, a 2D box is a rectangle and a 3D box is a cuboid.

V.14.1 Example

The following Hierarchical Data Structure

```
MODULE GE_Meshing
   concrete_name = "GE_BoxWithBoxes"
   vertices_coordinate_0 = regular_vector( 0.0, 2, 2.0 )
   vertices_coordinate_1 = regular_vector( 0.0, 2, 1.0 )
   mesh_polyhedron = < "GE_Segment" "GE_Rectangle" >
END MODULE GE_Meshing
```

leads to the following 2D box meshing, where Vi, Si, Bi, Ci stand respectively for the i-th vertex, side, bound, cell. Joined character strings are the associated colors.



V.14.2 Entries

vertices_coordinate_0 : list of the coordinate **0** of the vertices

- data type: **PEL_Data::DoubleVector**
- mandatory

vertices_coordinate_1 : list of the coordinate **1** of the vertices

- data type: **PEL_Data::DoubleVector**
- mandatory in 2D or 3D, forbidden in 1D

vertices_coordinate_2 : list of the coordinate **2** of the vertices

- data type: **PEL_Data::DoubleVector**
- mandatory in 3D, forbidden in 1D or 2D

mesh_polyhedron : names of the **GE_Mpolyhedron** derived-classes, for the faces then for the cells

- data type: **PEL_Data::StringVector**
- only possible values in 1D: **< "GE_Mpoint" "GE_Segment" >**

  - 1-st possible values in 2D: **`< "GE_Segment" "GE_Rectangle" >`** ,
    devoted to meshings that will not be subsequently deformed.
  - 2-nd possible values in 2D: **`< "GE_Segment" "GE_Quadrilateral" >`** ,
    devoted to meshings that may be deformed.
  - 3-rd possible values in 2D: **`< "GE_Segment" "GE_Trapezoid" >`** , devoted to meshings that may be deformed, yet maintaining cells representable by **`GE_Trapezoid`** instances.
  - 1-st possible values in 3D: **`< "GE_Rectangle" "GE_Cuboid" >`** ,
    devoted to meshing that will not be subsequently deformed.
  - 2-nd possible values in 3D: **`< "GE_Quadrilateral" "GE_Hexahedron" >`**,
    devoted to meshing that may be subsequently deformed.
  - mandatory

V.14.3 Coloring

- **`GE_BoxWithBoxes`** assigns colors to all bounds and to some vertices and cells (but not to sides).

- The boundary Γ of Ω can be partionned into subboundaries, each of them being assigned an identifier.

1D $\Gamma = \Gamma_{\text{left}} \cup \Gamma_{\text{right}}$
2D $\Gamma = \Gamma_{\text{left}} \cup \Gamma_{\text{right}} \cup \Gamma_{\text{bottom}} \cup \Gamma_{\text{top}}$
3D $\Gamma = \Gamma_{\text{left}} \cup \Gamma_{\text{right}} \cup \Gamma_{\text{bottom}} \cup \Gamma_{\text{top}} \cup \Gamma_{\text{front}} \cup \Gamma_{\text{behind}}$

subboundary	identifier	
Γ_{left}	**`left`**	surface perpendicular to the axis of coordinates **`0`**
Γ_{right}	**`right`**	surface perpendicular to the axis of coordinates **`0`**
the coordinates **`0`** are increasing from Γ_{left} to Γ_{right}		
Γ_{bottom}	**`bottom`**	surface perpendicular to the axis of coordinates **`1`**
Γ_{top}	**`top`**	surface perpendicular to the axis of coordinates **`1`**
the coordinates **`1`** are increasing from Γ_{bottom} to Γ_{top}		
Γ_{front}	**`front`**	surface perpendicular to the axis of coordinates **`2`**
Γ_{behind}	**`behind`**	surface perpendicular to the axis of coordinates **`2`**
the coordinates **`2`** are increasing from Γ_{behind} to Γ_{front}		

- The color name of a *bound* is the identifier of the subboundary of Ω to which it belongs (either **`"left"`**, **`"right"`**, **`"bottom"`**, **`"top"`**, **`"front"`**, **`"behind"`**).

- Color names of *vertices* and *cells* are concatenations of subboundaries identifiers, separated by underscores, as follows : the identifiers **`"front"`** or **`"behind"`** occur first (if present), followed by **`"bottom"`** or **`"top"`** (if present) followed by **`"left"`** or **`"right"`** (if present).

  This leads to the following color names:

"front"	**"front_bottom"**	**"front_bottom_left"**
		"front_bottom_right"
	"front_top"	**"front_top_left"**
		"front_top_right"
	"front_left"	
	"front_right"	
"behind"	**"behind_bottom"**	**"behind_bottom_left"**
		"behind_bottom_right"
	"behind_top"	**"behind_top_left"**
		"behind_top_right"
	"behind_left"	
	"behind_right"	
"bottom"	**"bottom_left"**	
	"bottom_right"	
"top"	**"top_left"**	
	"top_right"	
"left"		
"right"		

- A *vertex* belonging to a bound is assigned a color whose name is the concatenation of the color names of all the bounds to which it belongs.

- A *cell* whose vertices are colored is assigned a color whose name is the concatenation of the color names of its vertices.

V.15 MODULE GE_Meshing concrete_name="GE_CurveWithSegments"

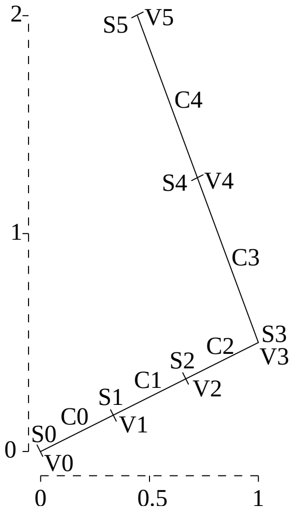
The **GE_CurveWithSegments** class is devoted to the subdivision into of a domain Ω whose shape is a broken line (in a geometric space of any number of dimensions). *Cells* are segments.

V.15.1 Example

The following Hierarchical Data Structure

```
MODULE GE_Meshing
   concrete_name = "GE_CurveWithSegments"
   closed_curve = false
   bends = array( < 0. 0.>, <1.0 0.5>, <0.5 2.0> )
   subdivisions = regular_vector( 0.0, 3, 1.0 )  «  < 0.0 0.75 1.0 >
   mesh_polyhedron = < "GE_Mpoint" "GE_Segment" >
END MODULE GE_Meshing
```

leads to the following meshing of a broken line, where Vi, Si and Ci stand respectively for the i-th vertex, side and cell.



V.15.2 Entries

closed_curve : Is the broken line closed?

- data type: **PEL_Data::Bool**
- mandatory

bends : list of the coordinates of the geometrical points where the line is broken

- data type: **PEL_Data::DoubleArray2D**
- mandatory

subdivisions : definition of the subdivision of each rectilinear part of Ω

- data type: **PEL_Data::DoubleVector**
- values: each item is the relative coordinate α of the mesh vertex V in the current rectilinear part $[AB]$:

  $$V = \alpha A + (1-\alpha)B$$

  As the original vertices of the broken line will become mesh vertices, for k line segments the value will have the following form:

  $$\texttt{<}\ \ \texttt{0.} = \alpha_0^0\quad \alpha_1^0\quad ..\quad \alpha_n^0 = \texttt{1.}\quad \dots\quad \texttt{0.} = \alpha_0^k\quad \alpha_1^k\quad ..\quad \alpha_m^k = \texttt{1.}\ \ \texttt{>}$$

  where the $(\alpha_i^0)_i \dots (\alpha_j^k)_j$ are growing series of relative coordinates.
- mandatory

mesh_polyhedron : names of the **GE_Mpolyhedron** derived-classes, for the faces then for the cells

- data type: **PEL_Data::StringVector**
- only possible values: **< "GE_Mpoint" "GE_Segment" >**
- mandatory

V.15.3 Coloring

GE_CurveWithSegments assigns no color to geometric entities.

V.16 MODULE GE_Meshing concrete_name="GE_ExtrudedMeshing"

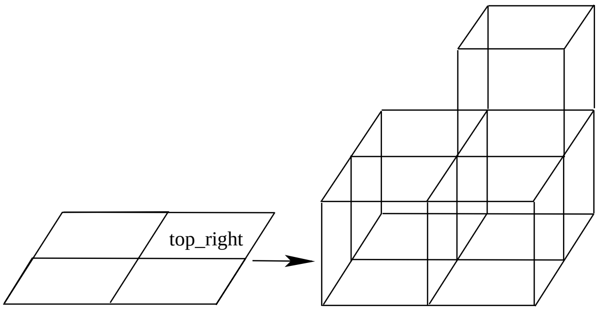
The class **GE_ExtrudedMeshing** builds a 3D meshing from a pre-existent 2D meshing by extruding it along the vertical direction. The extrusion is processed cell by cell by adding successive layers of vertices upon the current cell.

V.16.1 Example

The following Hierarchical Data Structure

```
MODULE GE_Meshing
   concrete_name = "GE_ExtrudedMeshing"
   MODULE GE_Meshing
      concrete_name = "GE_BoxWithBoxes"
      vertices_coordinate_1 = regular_vector( 0.0, 2, 1.0 )
      vertices_coordinate_0 = regular_vector( 0.0, 2, 2.0 )
      mesh_polyhedron = < "GE_Segment" "GE_Rectangle" >
   END MODULE GE_Meshing
   mesh_polyhedron = < "GE_Rectangle" "GE_Cuboid" >
   MODULE vertices_coordinate_2
      default = regular_vector( 0.0, 1, 1.0 )
      top_right = regular_vector( 0.0, 2, 2.0 )
   END MODULE vertices_coordinate_2
END MODULE GE_Meshing
```

leads to the following 3D meshing:



V.16.2 Entries

mesh_polyhedron : names of the **GE_Mpolyhedron** derived-classes, for the faces then for the cells

- mandatory

V.16.3 Submodules

MODULE GE_Meshing (§V.13) : any Hierarchical Data Structure that leads to a 2D meshing

- mandatory

MODULE vertices_coordinate_2 (§V.16.4) :

- mandatory

V.16.4 Entries of MODULE vertices_coordinate_2

The module **vertices_coordinate_2** contains a generic list of entries, each of them defining the list of the third coodinate of the vertices created by the extrusion of groups of 2D cells (identified by their color).

- keyword: the color name of cells of the pre-existent 2D meshing, or **"default"**

- data type: **PEL_Data::DoubleVector**
- mandatory

The pre-existent vertices of the 2D meshing will be swept according the list of third coordinates of their 2D cell color, if any, or, if none, according to the list of third coordinates associated to the **"default"** keyword.

V.16.5 Coloring

V.17 MODULE GE_Meshing concrete_name="GE_EMC2Meshing"

The class **GE_EMC2Meshing** builds a meshing from ouput files of the EMC2 mesh program.

V.17.1 Example

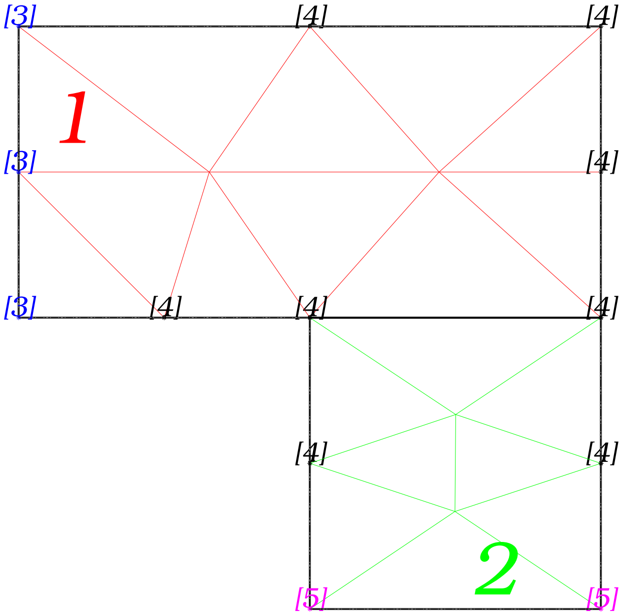
The following Hierarchical Data Structure

```
MODULE GE_Meshing
   concrete_name = "GE_EMC2Meshing"
   filename = join( this_file_dir(), "emc2.ftq" )
   format = "ftq"
   mesh_polyhedron = < "GE_Segment" "GE_Triangle" >
END MODULE GE_Meshing
```

where **"emc2.ftq"** is the following EMC2 output file (except for comments that have been manually added afterwards)

```
17 19 19 0   // Number of vertices, cells, triangles, quadrangles.
3 1 2 14 1   // List of cells :
3 3 4 2 1    // for each cell, number of vertices,
3 4 5 14 1   //                id number of each vertex and color.
3 5 16 14 1
3 15 17 7 2
3 7 17 5 2
3 6 8 15 2
3 13 11 16 1
3 17 9 5 2
3 16 11 12 1
3 9 16 5 1
3 16 12 14 1
3 6 15 7 2
3 9 13 16 1
3 17 10 9 2
3 15 10 17 2
3 4 14 2 1
3 8 10 15 2
3 14 12 1 1
  0.000000E+00   1.000000E+00       3 // List of vertices.
  0.000000E+00   7.500000E-01       3 // For each vertex,
  0.000000E+00   5.000000E-01       3 //    coordinates and color.
  2.500000E-01   5.000000E-01       4
  5.000000E-01   5.000000E-01       4
  5.000000E-01   0.000000E+00       5
  5.000000E-01   2.500000E-01       4
  1.000000E+00   0.000000E+00       5
  1.000000E+00   5.000000E-01       4
  1.000000E+00   2.500000E-01       4
  1.000000E+00   1.000000E+00       4
  5.000000E-01   1.000000E+00       4
  1.000000E+00   7.500000E-01       4
  3.276249E-01   7.499999E-01       0
  7.492648E-01   1.673947E-01       0
  7.216671E-01   7.500000E-01       0
  7.502111E-01   3.337847E-01       0
```

leads to the meshing depicted below with its EMC2 references for vertices and cells. These references are used by PELICANS for assigning colors to vertices, faces and cells.



V.17.2 Entries

filename : name of the EMC2 output file

- data type: **PEL_Data::String**
- only possible values:
- mandatory

format : format of the EMC2 output file

- data type: **PEL_Data::String**
- 1-st possible value: **"amdba"**
- 2-nd possible value: **"ftq"**
- mandatory

mesh_polyhedron : names of the **GE_Mpolyhedron** derived-classes, for the faces then for the cells

- mandatory

Coloring

- EMC2 allows to mark the vertices and the cells by an integer (see the node reference and domain reference functionalities).
- **GE_EMC2Meshing** uses those references to associate colors to mesh entities by adding the character '**r**' before the reference (a.e. an EMC2 node that is referenced by the integer **"2"** will be interpreted as a mesh vertex with color name **"r2"**).
- The faces are colored by the **GE_EMC2Meshing** component as a combination of the colors of its two vertices. For example, a face with two vertices of the same color **"r1"** will be assigned the color of name **"r1"** whereas a face with two vertices of color **"r2"** and **"r1"** will be assigned the color of name **"r1r2"**.
- The vertices and cells that are referenced with the null integer will be assigned the color of name **"r0"**.

V.18 MODULE GE_Meshing concrete_name="GE_MefistoMeshing"

The class **GE_MefistoMeshing** builds a meshing from output files of the Méfisto software (in **xyznpef** format).

V.18.1 Example

The following Hierarchical Data Structure

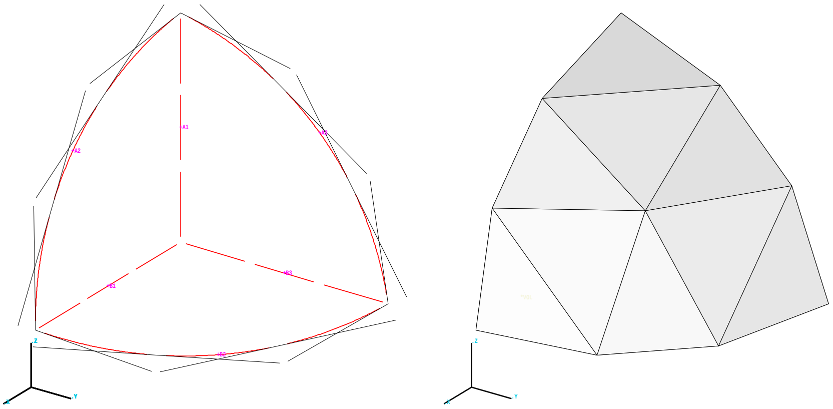
```
MODULE GE_Meshing
   concrete_name = "GE_MefistoMeshing"
   filename = join( this_file_dir(), "xyznpef.TETRAHEDRE" )
   mesh_polyhedron = < "GE_Triangle" "GE_Tetrahedron" >
END MODULE GE_Meshing
```

where **"xyznpef.TETRAHEDRE"** is the following Méfisto output file (displayed here in two columns with shorter blank characters)

```
TETRAHEDRE                    {NOM DE L'OBJET}
       3   {DIMENSION 2 ou 3 DE L'ESPACE DE L'OBJET}
      20   {NBNOEUDS NOMBRE DE NOEUDS DE L'OBJET}
       0   {NBTG     NOMBRE DE TANGENTES DE L'OBJET}
  0.0000000E+00  0.0000000E+00  0.0000000E+00
  0.0000000E+00  0.3333333E+00  0.0000000E+00
  0.0000000E+00  0.6666667E+00  0.0000000E+00
  0.0000000E+00  0.1000000E+01  0.0000000E+00
  0.3333333E+00  0.0000000E+00  0.0000000E+00
  0.0000000E+00  0.0000000E+00  0.3333333E+00
  0.3924646E+00  0.3924645E+00 -0.5913124E-01
 -0.5913124E-01  0.3924646E+00  0.3924646E+00
 -0.8869686E-01  0.7612126E+00  0.4161812E+00
  0.4161812E+00  0.7612125E+00 -0.8869686E-01
  0.0000000E+00  0.0000000E+00  0.6666667E+00
 -0.8869686E-01  0.4161812E+00  0.7612125E+00
  0.0000000E+00  0.0000000E+00  0.1000000E+01
  0.6666667E+00  0.0000000E+00  0.0000000E+00
  0.3924646E+00 -0.5913124E-01  0.3924645E+00
  0.7612126E+00  0.4161812E+00 -0.8869686E-01
  0.3924646E+00  0.3924646E+00  0.3924645E+00
  0.4161812E+00 -0.8869686E-01  0.7612125E+00
  0.1000000E+01  0.0000000E+00  0.0000000E+00
  0.7612126E+00 -0.8869686E-01  0.4161812E+00
       1   {NBOBIN NOMBRE DE MATERIAUX}
       1   {est le NUMERO dans VOLUME     de }
VOL
       4   {NBOBCL NOMBRE D'OBJETS AUX LIMITES}
       4   {NOMBRE DE SURFACE   }
       1   {est le NUMERO dans SURFACE    de }
BASE
       3   {est le NUMERO dans SURFACE    de }
GAUCHE
       4   {est le NUMERO dans SURFACE    de }
DERRIERE
       2   {est le NUMERO dans SURFACE    de }
FACE
       0   {NOMBRE DE LIGNE     }
       0   {NOMBRE DE POINT     }
       1   {NBTYEL NOMBRE DE TYPES D'EF de l'OBJET}
      27  {NBELEM NOMBRE D'EF DE CE TYPE D'EF}
       4  {NBNOE  NOMBRE DE NOEUDS DE CE TYPE D'EF}
       4  {NBSOE  NOMBRE DE SOMMETS DE CE TYPE D'EF}
       6  {NBARET NOMBRE D'ARETES DE CE TYPE D'EF}
       4  {NBFACE NOMBRE DE FACES DE CE TYPE D'EF}
       1  {NBVOL  NOMBRE DE VOLUME DE CE TYPE D'EF}
```

```
  7  15  17   8   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   1
  8  10   9  17   0   0   0   0   0   0
  0   0   0   0   0   0   0   2   1
  6  15   8  11   0   0   0   0   0   0
  0   0   0   0   0   4   3   0   1
 15  11  18  17   0   0   0   0   0   0
  0   0   0   0   3   0   0   0   1
 11  17  12  18   0   0   0   0   0   0
  0   0   0   0   0   0   0   2   1
  8  11  17  12   0   0   0   0   0   0
  0   0   0   0   0   0   4   0   1
  8  17   9  12   0   0   0   0   0   0
  0   0   0   0   0   4   0   2   1
 15   8  11  17   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   1
  7   8  17  10   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   1
  1   5   2   6   0   0   0   0   0   0
  0   0   0   0   1   4   3   0   1
 11  18  12  13   0   0   0   0   0   0
  0   0   0   0   0   4   3   2   1
  3   8  10   9   0   0   0   0   0   0
  0   0   0   0   0   0   4   0   1
  7  15  16  17   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   1
 14  19  16  20   0   0   0   0   0   0
  0   0   0   0   1   0   3   2   1
  5   2   6   7   0   0   0   0   0   0
  0   0   0   0   0   0   1   0   1
  7   3   8  10   0   0   0   0   0   0
  0   0   0   0   0   0   1   0   1
 14  15  20  16   0   0   0   0   0   0
  0   0   0   0   3   0   0   0   1
 15  16  17  20   0   0   0   0   0   0
  0   0   0   0   0   0   0   2   1
  5   6  15   7   0   0   0   0   0   0
  0   0   0   0   3   0   0   0   1
  5  14   7  15   0   0   0   0   0   0
  0   0   0   0   1   0   3   0   1
 14   7  15  16   0   0   0   0   0   0
  0   0   0   0   0   0   1   0   1
  2   6   7   8   0   0   0   0   0   0
  0   0   0   0   0   0   4   0   1
  3  10   4   9   0   0   0   0   0   0
  0   0   0   0   1   4   0   2   1
 15  20  17  18   0   0   0   0   0   0
  0   0   0   0   0   0   3   2   1
  7  16  10  17   0   0   0   0   0   0
  0   0   0   0   1   0   0   2   1
  2   7   3   8   0   0   0   0   0   0
  0   0   0   0   1   4   0   0   1
  6   7   8  15   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   1
 27 {NOMBRE TOTAL D'EF de l'OBJET TETRAHEDRE }
```

leads to meshing depiced below (right figure) of the half of a sphere quarter (left figure).



V.18.2 Entries

filename : name of the Méfisto-Maillages output file in **xyznpef** format

- data type: **PEL_Data::String**
- only possible values:
- mandatory

mesh_polyhedron : names of the **GE_Mpolyhedron** derived-classes, for the faces then for the cells

- mandatory

V.18.3 Coloring

When saving the meshing in **xyznpef** format (option number **91** in the main menu of Méfisto-Maillages) one has first to give a name to the to-be-saved object and then to exclude tangent saving by selecting **0**. Be careful, Méfisto-Maillage can mesh volumes with pentahedra whereas they are not now defined in PELICANS. One can give references (names) to vertices, faces (lines in 2D and facets in 3D) and cells by saving at the same time than the main object the concerned points, lines, surfaces and volumes.

Vertices – A mesh vertex lying on a saved physical point takes its reference (name). Others vertices that belong to a saved line in 2D or to a saved surface in 3D take their reference. Otherwise by default any vertex takes the reference of the surfaces in 2D and the volumes in 3D it belongs to (Méfisto-Maillages says materials).

Faces – A face lying on a saved physical line in 2D and surface in 3D takes its reference. Otherwise by default any face takes the reference of the lines in 2D and of the surfaces in 2D and the volumes in 3D it belongs to (Méfisto-Maillages says materials).

Cells – A cell takes the reference of the surfaces in 2D and the volumes in 3D (Méfisto-Maillages says materials) it belongs to.

If a vertex or a face has to take the reference of several saved elements (a.e. two lines), its reference will be compound with all reference names listed alphabetically and separated by an underscore. As an example if a vertex of the 3D space lies on surfaces **EAST** and **NORTH**, it will be associated to reference **EAST_NORTH**, if a sides lies on volumes (materials) **TOP** and **BOTTOM** it will takes the reference **BOTTOM_TOP**.

V.19 MODULE GE_Meshing concrete_name="GE_GmshMeshing"

The class **GE_GmshMeshing** builds a meshing from output files of the Gmsh software.

V.19.1 Example

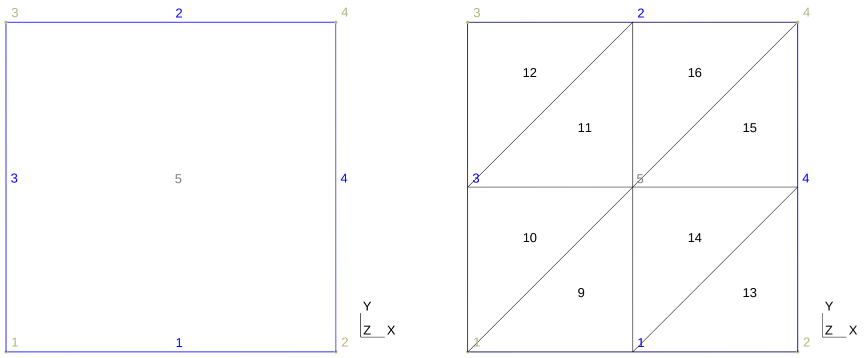
The following Hierarchical Data Structure

```
MODULE GE_Meshing
   concrete_name = "GE_GmshMeshing"
   filename = join( this_file_dir(), "untitled.msh" )
   mesh_polyhedron = <  "GE_Segment" "GE_Triangle" >
   format = "2.0"
END MODULE GE_Meshing
```

where **"untitled.msh"** is the following Gmsh output file (displayed here in two columns with shorter blank characters)

```
                | $Elements
                | 20
                | 1 15 2 0 1 1
                | 2 15 2 0 2 2
$MeshFormat     | 3 15 2 0 3 3
2 0 8           | 4 15 2 0 4 4
$EndMeshFormat  | 5 1 3 0 1 0 1 5
$Nodes          | 6 1 3 0 1 0 5 2
9               | 7 1 3 0 2 0 3 6
1 0 0 0         | 8 1 3 0 2 0 6 4
2 1 0 0         | 9 1 3 0 3 0 1 7
3 0 1 0         | 10 1 3 0 3 0 7 3
4 1 1 0         | 11 1 3 0 4 0 2 8
5 0.5 0 0       | 12 1 3 0 4 0 8 4
6 0.5 1 0       | 13 2 3 0 5 0 1 5 9
7 0 0.5 0       | 14 2 3 0 5 0 1 9 7
8 1 0.5 0       | 15 2 3 0 5 0 7 9 6
9 0.5 0.5 0     | 16 2 3 0 5 0 7 6 3
$EndNodes       | 17 2 3 0 5 0 5 2 8
                | 18 2 3 0 5 0 5 8 9
                | 19 2 3 0 5 0 9 8 4
                | 20 2 3 0 5 0 9 4 6
                | $EndElements
```

leads to meshing depicted below (right figure) of the unit square (left figure).



V.19.2 Usage Notes

- No all the functionnalities of Gmsh can be exploited within PELICANS. For example, Gmsh can mesh volumes with prisms, pyramids and second order elements (*i.e.* specific polyhedra which include nodes that do not correspond to geometrical vertices of these polyhedra), whereas such cells are not handled as such in PELICANS.

- The version 2.0 of the '**.msh**' file format is Gmsh's new native file format. It is highly recommended to use it.

V.19.3 Entries

filename : name of the Gmsh output file in the '**.msh**' format

- data type: **PEL_Data::String**
- mandatory

mesh_polyhedron : names of the **GE_Mpolyhedron** derived-classes, for the faces then for the cells

- mandatory

format : version of the '**.msh**' file format

- data type: **PEL_Data::String**
- 1-st possible value: **"1.0"**
- 2-nd possible value: **"2.0"**
- mandatory

V.19.4 Coloring

Gmsh marks the vertices, the faces (lines in 2D and facets in 3D) and the cells by an integer (see the Chapter 9: File formats of the Gmsh reference manual). **GE_GmshMeshing** uses those references to build colors by adding the character '**r**' before the reference (a.e. an Gmsh cell that is referenced by the integer **"2"** will be interpreted as a cell with color name **"r2"**).

Vertices – A vertex takes the number of the geometrical lines in 2D and the geometrical surfaces in 3D it it belongs to. If a vertex has to take the reference of several geometrical lines in 2D or several geometrical surfaces in 3D, its reference will be compound with all reference listed alphabetically. As an example if a vertex of the 3D space lies on the three geometrical surfaces numbered by Gmsh **1**, **12** and **3**, it will be associated to reference **r1r3r12**.

Faces – A face lying on a geometrical line in 2D and surface in 3D takes its reference.

Cells – A cell takes the reference of the surface in 2D and the volume in 3D it belongs to.

V.20 MODULE GE_Meshing concrete_name="GE_RefinedMeshing"

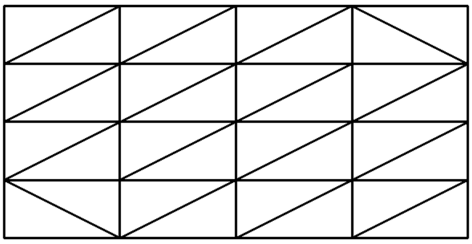
The class **GE_RefinedMeshing** builds a meshing from a pre-existent one by refining its cells according to a pattern defined for the reference polyhedra.

V.20.1 Examples

▷ The following Hierarchical Data Structure:

```
MODULE GE_Meshing
   concrete_name = "GE_RefinedMeshing"
   mesh_polyhedron = < "GE_Segment" "GE_Triangle" >
   MODULE GE_Meshing
      concrete_name = "GE_BoxWithBoxes"
      vertices_coordinate_0 = regular_vector( 0.0, 4, 2.0 )
      vertices_coordinate_1 = regular_vector( 0.0, 4, 1.0 )
      mesh_polyhedron = < "GE_Segment" "GE_Rectangle" >
   END MODULE GE_Meshing
   MODULE list_of_GE_ReferencePolyhedronRefiner
      MODULE GE_ReferencePolyhedronRefiner#1
         concrete_name = "GE_ReferenceSquareWithTriangles"
         strategy = "/"
      END MODULE GE_ReferencePolyhedronRefiner#1
      MODULE GE_ReferencePolyhedronRefiner#2
         concrete_name = "GE_ReferenceSquareWithTriangles"
         color = "bottom_left"
         strategy = "\"
      END MODULE GE_ReferencePolyhedronRefiner#2
      MODULE GE_ReferencePolyhedronRefiner#3
         concrete_name = "GE_ReferenceSquareWithTriangles"
         color = "top_right"
         strategy = "\"
      END MODULE GE_ReferencePolyhedronRefiner#3
   END MODULE list_of_GE_ReferencePolyhedronRefiner
END MODULE GE_Meshing
```

leads to the following meshing:



▷ The following Hierarchical Data Structure leads to the creation of a meshing with tetrahedral cells by subdividing an original meshing made of cuboids.

```
MODULE GE_Meshing
   concrete_name = "GE_RefinedMeshing"
   mesh_polyhedron = < "GE_Triangle" "GE_Tetrahedron" >
   MODULE GE_Meshing
      concrete_name = "GE_BoxWithBoxes"
      vertices_coordinate_0 = regular_vector( 0.0, 2, 1.0 )
      vertices_coordinate_1 = regular_vector( 0.0, 2, 1.0 )
      vertices_coordinate_2 = regular_vector( 0.0, 2, 1.0 )
      mesh_polyhedron = < "GE_Rectangle" "GE_Cuboid" >
   END MODULE GE_Meshing
   MODULE list_of_GE_ReferencePolyhedronRefiner
      MODULE GE_ReferencePolyhedronRefiner#1
         concrete_name = "GE_ReferenceCubeWithTetrahedra"
      END MODULE GE_ReferencePolyhedronRefiner#1
   END MODULE list_of_GE_ReferencePolyhedronRefiner
END MODULE GE_Meshing
```

V.20.2 Methodology

Let $\mathcal{T}_0 = \bigcup_e \mathcal{C}_e^{[0]}$ be an initial meshing.

Each cell $\mathcal{C}_e^{[0]}$ is the image of a *reference polyhedron* $\widehat{\mathcal{C}}_e$ *via* the mapping $F_e^{[0]}$:

$$\mathcal{C}_e^{[0]} = F_e^{[0]}\big(\widehat{\mathcal{C}}_e\big)$$

There are very few kinds of possible reference polyhedra, each being represented by a class derived from **GE_ReferencePolyhedron**. For example, if $\mathcal{T}_0$ is a two dimensional meshing made of four sided cells, all reference cells will be the unit square $[0,1]^2$ represented by the class **GE_ReferenceSquare**.

A *reference polyhedron* may be partitionned into (few) non overlapping smaller polyhedra:

$$\widehat{\mathcal{C}}_e = \bigcup_i \mathcal{R}_e(\widehat{\mathcal{C}}_e, i) \quad \text{where for all } i: \quad \mathcal{R}_e(\widehat{\mathcal{C}}_e, i) \quad \text{is a polyhedron such that} \quad \mathcal{R}_e(\widehat{\mathcal{C}}_e, i) \subset \widehat{\mathcal{C}}_e \quad (5.1)$$

Such a partition defines a *refinement* of the *reference polyhedron* and is represented by a class derived from **GE_ReferencePolyhedronRefiner**.

Given a *refinement* $\mathcal{R}_e$ for all the reference polyhedra $\widehat{\mathcal{C}}_e$ occuring in $\mathcal{T}_0$, the refinement meshing $\mathcal{T}_{\text{refined}}$ built by **GE_RefinedMeshing** reads:

$$\mathcal{T}_{\text{refined}} = \bigcup_e \bigcup_i F_e^{[0]}\big(\mathcal{R}_e(\widehat{\mathcal{C}}_e, i)\big)$$

V.20.3 Entries

mesh_polyhedron : names of the **GE_Mpolyhedron** derived-classes, for the faces then for the cells

- data type: **PEL_Data::StringVector**
- values: vector of two names of classes derived from **GE_Mpolyhedron**, the first one representing the faces, the second one representing the cells
  The named polyhedra should be compatible with the chosen refinement for the reference polyhedra of the meshing defined in the submodule **GE_Meshing**.
- mandatory

V.20.4 Submodules

MODULE GE_Meshing (§V.13) : any Hierarchical Data Structure defining the meshing $\mathcal{T}_0$ to be refined

- mandatory

MODULE list_of_GE_ReferencePolyhedronRefiner : a generic list of submodules (§V.20.5, §V.20.6, §V.20.7, §V.20.8, §V.20.9), each defining

  - ◆ an object of a class derived from **GE_ReferencePolyhedronRefiner** representing a *refinement* $\mathcal{R}$
  - ◆ possibly a cell color

- mandatory

This leads to a set of refiners $\mathcal{R}_{\mathbf{c}_i}$ associated to the colors $\mathbf{c}_i$ and to a set of refiners $\mathcal{R}_{\text{default}_j}$ that where defined independently of any color.

The *refinement* $\mathcal{R}_e$ that is applied to a cell $\mathcal{C}_e$ of the initial meshing $\mathcal{T}_0$ is determined as follows:

– if there exists i such that the color of $\mathcal{C}_e$ is $\mathbf{c}_i$ and such that the reference polyhedron of $\mathcal{C}_e$ is handled by $\mathcal{R}_{\mathbf{c}_i}$, then $\mathcal{R}_e \equiv \mathcal{R}_{\mathbf{c}_i}$;

– else $\mathcal{R}_e \equiv \mathcal{R}_{\text{default}_j}$ where j is such that $\mathcal{R}_{\text{default}_j}$ handles the reference polyhedron of $\mathcal{C}_e$.

A fatal error is raised if the content of **MODULE list_of_GE_ReferencePolyhedronRefiner** leads to a failure of the above algorithm.

V.20.5 MODULE GE_ReferencePolyhedronRefiner concrete_name="GE_ReferenceCubeWithCubes"

GE_ReferenceCubeWithCubes defines a *refinement* of the reference square $[0,1]^3$ by partitioning it into n^3 identical subcubes.

Entries

nb_subintervals_per_edge : number of splitting of the original cube in each directions

- data type: **PEL_Data::Int**
- values: n greater or equal 0
  leads to refining the original square into n^3 identical subcubes (the value **0** means no refinement at all)
- mandatory

Example

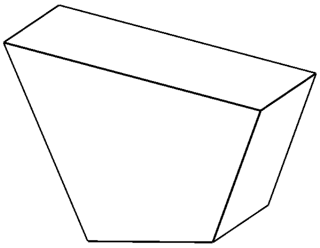
Consider the following Hierarchical Data Structure, where the file with a GMSH format **onecell.msh** describes a meshing made of a single hexahedron.

```
MODULE GE_Meshing
   concrete_name = "GE_RefinedMeshing"
   mesh_polyhedron = < "GE_Quadrilateral"
                       "GE_Hexahedron" >
   MODULE GE_Meshing
      concrete_name = "GE_GmshMeshing"
      mesh_polyhedron = < "GE_Quadrilateral"
                          "GE_Hexahedron" >
      filename = join( this_file_dir(),
                       "onecell.msh" )
      format = "2.0"
   END MODULE GE_Meshing
   MODULE list_of_GE_ReferencePolyhedronRefiner
      MODULE GE_ReferencePolyhedronRefiner#1
         concrete_name = "GE_ReferenceCubeWithCubes"
         nb_subintervals_per_edge = $IS_n
      END MODULE GE_ReferencePolyhedronRefiner#1
   END MODULE list_of_GE_ReferencePolyhedronRefiner
END MODULE GE_Meshing
```

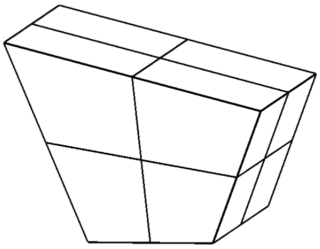
file **onecell.msh**

```
$MeshFormat
2 0 8
$EndMeshFormat
$Nodes
8
1  0    0   0
2  1    0   0
3  1.5  1   0
4 -0.5  1.5 0
5  0    0   1
6  1    0   1
7  1.5  1   1
8 -0.5  1.5 1
$EndNodes
$Elements
1
1 5 0 1 2 3 4 5 6 7 8
$EndElements
```

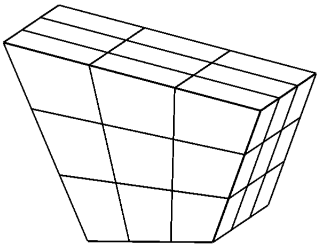
This leads, for three values of **$IS_n**, to the following meshings:



$IS_n = 1



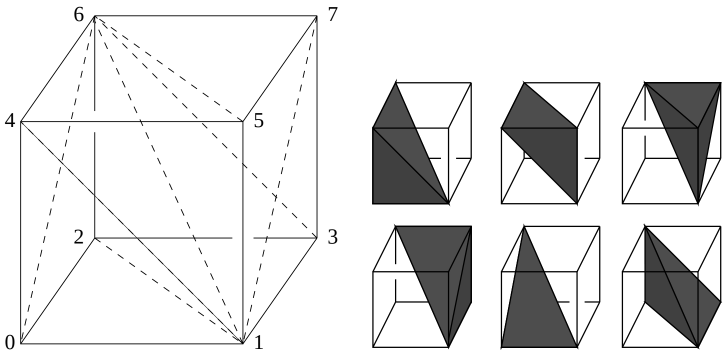
$IS_n = 2



$IS_n = 3

V.20.6 MODULE GE_ReferencePolyhedronRefiner concrete_name="GE_ReferenceCubeWithTetrahedra"

GE_ReferenceCubeWithTetrahedra defines a *refinement* of the reference cube $[0,1]^3$ by partitioning it into 6 tetrahedra.



The above meshing can be produced with the following Hierarchical Data Structure:

```
MODULE GE_Meshing
   concrete_name = "GE_RefinedMeshing"
   mesh_polyhedron = < "GE_Triangle" "GE_Tetrahedron" >
   MODULE GE_Meshing
      concrete_name = "GE_BoxWithBoxes"
      vertices_coordinate_0 = regular_vector( 0.0, 1, 1.0 )
      vertices_coordinate_1 = regular_vector( 0.0, 1, 1.0 )
      vertices_coordinate_2 = regular_vector( 0.0, 1, 1.0 )
      mesh_polyhedron = < "GE_Rectangle" "GE_Cuboid" >
   END MODULE GE_Meshing
   MODULE list_of_GE_ReferencePolyhedronRefiner
      MODULE GE_ReferencePolyhedronRefiner#1
         concrete_name = "GE_ReferenceCubeWithTetrahedra"
      END MODULE GE_ReferencePolyhedronRefiner#1
   END MODULE list_of_GE_ReferencePolyhedronRefiner
END MODULE GE_Meshing
```

V.20.7 MODULE GE_ReferencePolyhedronRefiner concrete_name="GE_ReferenceSquareWithSquares"

GE_ReferenceSquareWithSquares defines a *refinement* of the reference square $[0,1]^2$ by partitioning it into n^2 identical subsquares.

Entries

nb_subintervals_per_edge : number of splitting of the original square in each directions

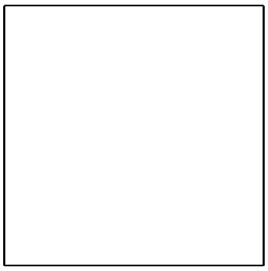
- data type: **PEL_Data::Int**
- values: n strictly greater than 0
  leads to refining the original square into n^2 identical subsquares (the value **0** means no refinement at all)
- mandatory

Example

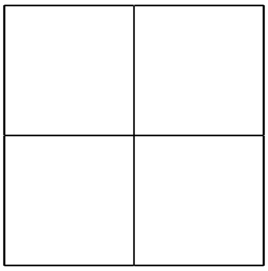
The following Hierarchical Data Structure:

```
MODULE GE_Meshing
   concrete_name = "GE_RefinedMeshing"
   mesh_polyhedron = < "GE_Segment" "GE_Rectangle" >
   MODULE GE_Meshing
      concrete_name = "GE_BoxWithBoxes"
      vertices_coordinate_0 = regular_vector( 0.0, 1, 1.0 )
      vertices_coordinate_1 = regular_vector( 0.0, 1, 1.0 )
      mesh_polyhedron = < "GE_Segment" "GE_Rectangle" >
   END MODULE GE_Meshing
   MODULE list_of_GE_ReferencePolyhedronRefiner
      MODULE GE_ReferencePolyhedronRefiner#1
         concrete_name = "GE_ReferenceSquareWithSquares"
         nb_subintervals_per_edge = $IS_n
      END MODULE GE_ReferencePolyhedronRefiner#1
   END MODULE list_of_GE_ReferencePolyhedronRefiner
END MODULE GE_Meshing
```

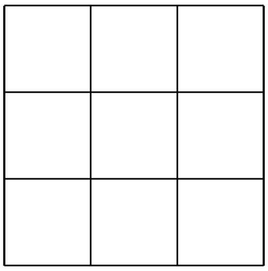
leads, for three values of **$IS_n**, to the following meshings:



$IS_n = 1



$IS_n = 2



$IS_n = 3

V.20.8 MODULE GE_ReferencePolyhedronRefiner concrete_name="GE_ReferenceSquareWithTriangles"

GE_ReferenceSquareWithTriangles defines different forms of *refinement* of the reference square by partitioning it into triangles.

Entries

strategy : form of the refinement for the reference square

- data type: **PEL_Data::String**
- 1-st possible value: **"/"**

  The original square is split in 2 triangles by joining the lower-left and the top-right corners.
- 2-nd possible value: **"\"**

  The original square is split in 2 triangles by joining the top-left and the bottom-right corners.
- 3-rd possible value: **"x"** or **"X"**

  The orginal square is split in 4 triangles by joining the top-left and the bottom-right corners and by joining the lower-left and the top-right corners.
- 4-th possible value: **"26_acute_triangles"**

  The orginal square is split in 26 triangles in which all the three angles are strictly lower than $\pi/2$.
- mandatory

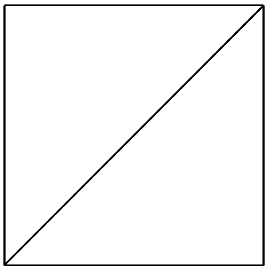
Example

The following Hierarchical Data Structure:

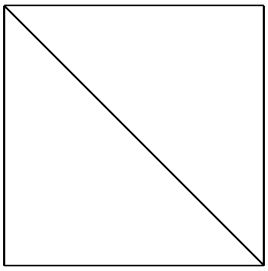
```
MODULE GE_Meshing
   concrete_name = "GE_RefinedMeshing"
   mesh_polyhedron = < "GE_Segment" "GE_Triangle" >
   MODULE GE_Meshing
      concrete_name = "GE_BoxWithBoxes"
      vertices_coordinate_0 = regular_vector( 0.0, 1, 1.0 )
      vertices_coordinate_1 = regular_vector( 0.0, 1, 1.0 )
      mesh_polyhedron = < "GE_Segment" "GE_Rectangle" >
   END MODULE GE_Meshing
   MODULE list_of_GE_ReferencePolyhedronRefiner
      MODULE GE_ReferencePolyhedronRefiner#1
         concrete_name = "GE_ReferenceSquareWithTriangles"
         strategy = $SS_str
      END MODULE GE_ReferencePolyhedronRefiner#1
   END MODULE list_of_GE_ReferencePolyhedronRefiner
END MODULE GE_Meshing
```

leads, for the 4 possible values of **`$SS_str`**, to the following meshings:

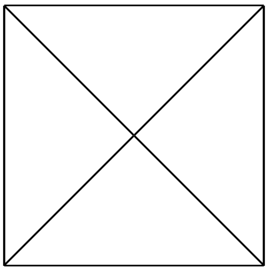
`$SS_str = "/"`

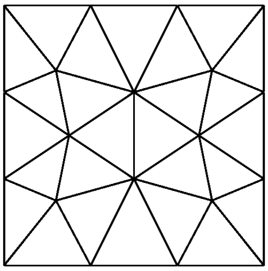


`$SS_str = "\"`



`$SS_str = "X"`





`$SS_str = "26_acute_triangles"`

V.20.9 MODULE GE_ReferencePolyhedronRefiner concrete_name="GE_ReferenceTriangleWithTriangles"

`GE_ReferenceTriangleWithTriangles` defines a *refinement* of the reference triangle $\left\{ \, x \geq 0 \; ; \; y \geq 0 \; ; \; x + y \leq 1 \, \right\}$ by partitioning it into n^2 identical subtriangles.

<u>Entries</u>

`nb_subintervals_per_edge` : number of splitting of the original triangle in each directions

- data type: **`PEL_Data::Int`**
- values: n strictly greater than 0
  leads to refining the original square into n^2 identical subtriangles (the value **`0`** means no refinement at all)
- mandatory

Example

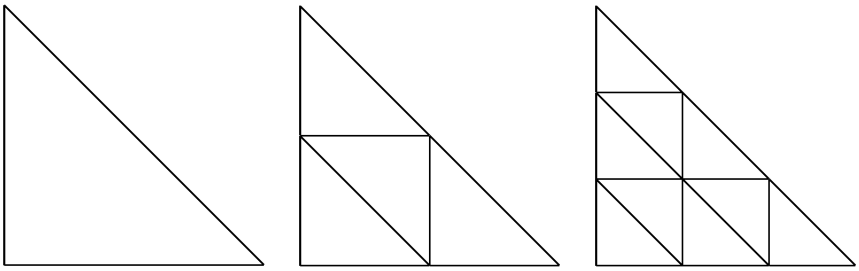
Consider the following Hierarchical Data Structure, where the file with a GMSH format **`utri.msh`** describes a meshing made of the only unit triangular cell.

```
MODULE GE_Meshing
  concrete_name = "GE_RefinedMeshing"
  mesh_polyhedron = < "GE_Segment" "GE_Triangle" >
  MODULE GE_Meshing
    concrete_name = "GE_GmshMeshing"
    mesh_polyhedron = < "GE_Segment" "GE_Triangle" >
    filename = join( this_file_dir(), "utri.msh" )
    format = "2.0"
  END MODULE GE_Meshing
  MODULE list_of_GE_ReferencePolyhedronRefiner
    MODULE GE_ReferencePolyhedronRefiner#1
      concrete_name = "GE_ReferenceTriangleWithTriangles"
      nb_subintervals_per_edge = $IS_n
    END MODULE GE_ReferencePolyhedronRefiner#1
  END MODULE list_of_GE_ReferencePolyhedronRefiner
END MODULE GE_Meshing
```

file **`utri.msh`**

```
$MeshFormat
2 0 8
$EndMeshFormat
$Nodes
3
1 0 0 0
2 1 0 0
3 0 1 0
$EndNodes
$Elements
1
1 2 3 0 5 0 1 2 3
$EndElements
```

This leads, for three values of **`$IS_n`**, to the following meshings:



`$IS_n = 1` **`$IS_n = 2`** **`$IS_n = 3`**

V.20.10 Coloring

▷ **`GE_RefinedMeshing`** assigns colors to the cells, faces and vertices of $\mathcal{T}_{\text{refined}}$ on basis of the colors of $\mathcal{T}_0$.

▷ The color of a cell in $\mathcal{T}_{\text{refined}}$ is that of the cell of $\mathcal{T}_0$ into which it is included.

▷ The color of a face in $\mathcal{T}_{\text{refined}}$ is:

- the color of the face of $\mathcal{T}_0$ into which it is included if any;
- if none, the color of the cell of $\mathcal{T}_0$ into which it is included.

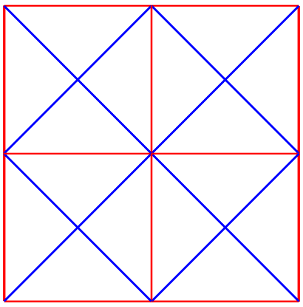
▷ The color of a vectex in $\mathcal{T}_{\text{refined}}$ is:

- its color in $\mathcal{T}_0$ if it already existed as a vertex in $\mathcal{T}_0$;
- if not, the color of the face of $\mathcal{T}_0$ into which it is included if any;
- if none, the color of the cell of $\mathcal{T}_0$ into which it is included.

▷ As an example, consider the following Hierarchical Data Structure:

```
MODULE GE_Meshing
   concrete_name = "GE_RefinedMeshing"
   mesh_polyhedron = < "GE_Segment" "GE_Triangle" >
   MODULE list_of_GE_ReferencePolyhedronRefiner
      MODULE GE_ReferencePolyhedronRefiner#1
         concrete_name = "GE_ReferenceSquareWithTriangles"
         strategy = "X"
      END MODULE GE_ReferencePolyhedronRefiner#1
   END MODULE list_of_GE_ReferencePolyhedronRefiner
   MODULE GE_Meshing
      concrete_name = "GE_BoxWithBoxes"
      vertices_coordinate_0 = regular_vector( 0.0, 2, 1.0 )
      vertices_coordinate_1 = regular_vector( 0.0, 2, 1.0 )
      mesh_polyhedron = < "GE_Segment" "GE_Rectangle" >
      MODULE GE_Colorist
         MODULE faces
            red = true
         END MODULE faces
         MODULE cells
            blue = true
         END MODULE cells
      END MODULE GE_Colorist
   END MODULE GE_Meshing
END MODULE GE_Meshing
```

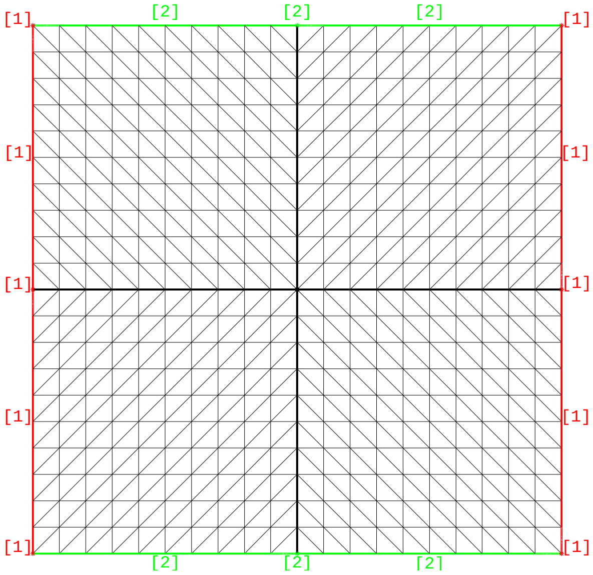
The initial meshing $\mathcal{T}_0$ has 4 cells, all with the color of name **"blue"**, and 12 faces, all with the color of name **"red"**. The refined meshing is displayed on the right, using blue (resp. red) lines for the faces with the color of name **"blue"** (resp. **"red"**).



V.21 Example

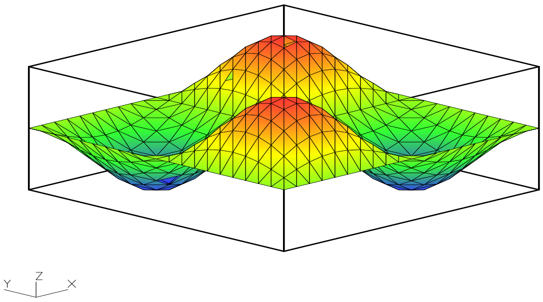
The purpose of this example is to build on the unit square discrete representations of a scalar field (denoted f) and of a vector field (denoted $\mathbf{u}$) using piecewise linear finite elements.

The meshing is made of 800 triangles.

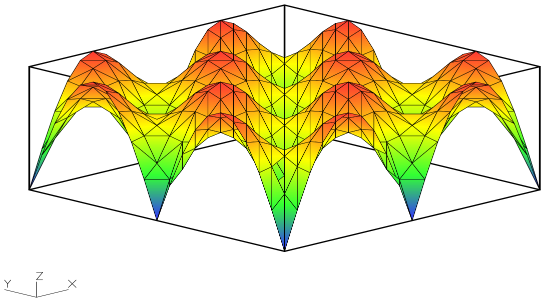


The initialization value of the **DOF**s will be deduced from :

$$\begin{aligned} f(x,y) &= \sin(2\pi x) * \sin(2\pi y) \\ \mathbf{u}(x,y) &= -2\pi \begin{bmatrix} \cos(2\pi x)\ \sin(2\pi y) \\ \sin(2\pi x)\ \cos(2\pi y) \end{bmatrix} \end{aligned}$$



f (minimum: -1 maximum: 1)



magnitude of $\mathbf{u}$ (minimum: 0, maximum: 2π)

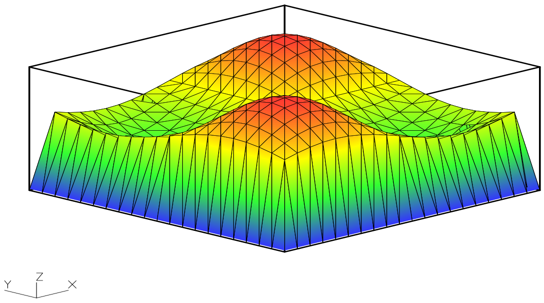
The Hierarchical Data Structure devoted to the definition of theses discretizations is given below.

```
MODULE PDE_DomainAndFields
   verbose_level = 1
   nb_space_dimensions = 2
   type = "finite_element"
   MODULE GE_Meshing
      concrete_name = "GE_EMC2Meshing"
      mesh_polyhedron = < "GE_Segment" "GE_Triangle" >
      filename= join( this_file_dir(), "example1.ftq" )
      format="ftq"
   END MODULE GE_Meshing
   MODULE interior_fields
      $DS_p = 2.*pi()
      $DS_x = component( $DV_X, 0 )
      $DS_y = component( $DV_X, 1 )
      MODULE vv
         name = "vectorial_field"
         nb_components = 2
         element_name = "PDE_2D_P1_3nodes"
         storage_depth = 1
         MODULE DOFs_values
            type = "uniformly_defined"
            value = vector( - $DS_p*cos($DS_p*$DS_x)*sin($DS_p*$DS_y) ,
                            - $DS_p*sin($DS_p*$DS_x)*cos($DS_p*$DS_y) )
         END MODULE DOFs_values
      END MODULE vv
      MODULE ff
         name = "scalar_field"
         nb_components = 1
         element_name = "PDE_2D_P1_3nodes"
         storage_depth = 1
         MODULE DOFs_values
            type = "uniformly_defined"
            value = vector( sin($DS_p*$DS_x)*sin($DS_p*$DS_y) )
         END MODULE DOFs_values
      END MODULE ff
   END MODULE interior_fields
   MODULE PDE_ResultSaver
      writers = < "PEL_GMVwriter" >
      writing_mode = "text"
      files_basename = "save"
      MODULE vv
         where_to_save = "at_vertices"
         entry_name = "V"
         field = "vectorial_field"
      END MODULE vv
      MODULE ff
         where_to_save = "at_vertices"
         entry_name = "F"
         field = "scalar_field"
      END MODULE ff
   END MODULE PDE_ResultSaver
END MODULE PDE_DomainAndFields
```

As an extension one can enforce the vectorial field values at the computational domain boundaries and perturb the scalar field values at the boundary by imposing them to -2.

```
MODULE DOFs_imposed_value
   MODULE vertical
      location = "on_bounds"
      type = "uniformly_defined"
      color = "r1"
      value = vector( -$DS_p*sin($DS_p*$DS_y), 0. )
   END MODULE vertical
   MODULE horizontal
      location = "on_bounds"
      type = "uniformly_defined"
      color = "r2"
      value = vector( 0., -$DS_p*sin($DS_p*$DS_x) )
   END MODULE horizontal
END MODULE DOFs_imposed_value
```

```
MODULE DOFs_imposed_value
   MODULE everywhere
      location = "on_bounds"
      type = "uniformly_defined"
      value = < -2.0 >
   END MODULE everywhere
END MODULE DOFs_imposed_value
```



f with **imposed** values on the boundary

Chapter VI

A Galerkin Solver for the Unsteady Diffusion Problem

VI.1 Position du Problème

▷ Considérons le transport diffusif d'une grandeur physique scalaire régi par le système d'équations suivant :

$$
\begin{aligned}
\alpha \frac{\partial \underline{u}}{\partial t} &= \nabla \cdot (\kappa \nabla \underline{u}) + \pi && \text{dans} && [0,T] \times \Omega && \text{(6.1a)} \\
\underline{u} &= \underline{u}_0 && \text{sur} && [0,T] \times \Gamma_{\mathrm{D}} && \text{(6.1b)} \\
-\kappa \nabla \underline{u} \cdot \mathbf{n} &= q_0 && \text{sur} && [0,T] \times \Gamma_{\mathrm{N}} && \text{(6.1c)} \\
-\kappa \nabla \underline{u} \cdot \mathbf{n} &= h(\underline{u} - u_\infty) && \text{sur} && [0,T] \times \Gamma_{\mathrm{R}} && \text{(6.1d)} \\
\underline{u} &= \underline{u}_{\text{initial}} && \text{dans} && \{0\} \times \Omega && \text{(6.1e)}
\end{aligned}
$$

où

- $[0,T]$ représente l'intervalle temporel d'étude ;
- Ω est un ouvert connexe borné de $\mathbb{R}^d$ ($d = 1$, 2 ou 3) dont la frontière Γ, supposée suffisamment régulière, est subdivisée en $\Gamma = \Gamma_{\mathrm{D}} \cup \Gamma_{\mathrm{N}} \cup \Gamma_{\mathrm{R}}$ de façon que Γ_{D}, Γ_{N} et Γ_{R} soient deux à deux d'intersection vide ;
- la grandeur transportée (par exemple la température ou la concentration d'une espèce), définie dans $[0,T] \times \Omega$ à valeurs dans $\mathbb{R}$ est notée $\underline{u}$ et prend à chaque instant la valeur $\underline{u}_0$ sur Γ_{D} ;
- le préfacteur du terme instationnaire α, le coefficient de diffusion κ et la production interne π dépendent éventuellement de l'espace ;
- en tous points de Γ_{N} (resp. Γ_{R}) et à chaque instant, la densité de flux diffusif dans la direction de la normale unitaire sortante $\mathbf{n}$ prend la valeur q_0 (resp. $h(\underline{u} - u_\infty)$).

▷ Le problème que nous nous proposons d'étudier est le suivant :

> Etant *a priori* données les grandeurs α, κ, π, $\underline{u}_0$, q_0, h, u_∞, $\underline{u}_{\text{initial}}$,
> trouver $\underline{u}$ solution du système d'équations (6.1) (6.2)

ou, reformulé en discriminant les trois ingrédients traditionnels des modèles mathématiques en vue d'une utilisation ultérieure de PELICANS (§I.1.2) :

- Champ inconnu :

$$\mathbf{W} \overset{\text{def}}{=} \underline{u} : [0,T] \times \Omega \to \mathbb{R}$$

- Equations vérifiées par le champ inconnu :

$$
\begin{array}{ll}
R(\mathbf{W}) \overset{\text{def}}{=} \alpha \dfrac{\partial \underline{u}}{\partial t} - \nabla \cdot (\kappa \nabla \underline{u}) - \pi = 0 & \text{dans } [0,T] \times \Omega \\
\underline{u} = \underline{u}_0 & \text{sur } [0,T] \times \Gamma_{\mathrm{D}} \\
-\kappa \nabla \underline{u} \cdot \mathbf{n} = q_0 & \text{sur } [0,T] \times \Gamma_{\mathrm{N}} \\
-\kappa \nabla \underline{u} \cdot \mathbf{n} = h(\underline{u} - u_\infty) & \text{sur } [0,T] \times \Gamma_{\mathrm{R}} \\
\underline{u} = \underline{u}_{\text{initial}} & \text{dans } \{0\} \times \Omega
\end{array}
$$

- Paramètres du modèle : $\alpha, \kappa, \pi, \underline{u}_0, q_0, h, u_\infty, \underline{u}_{\text{initial}}$

VI.2 Discrétisation en Temps

▷ L'élaboration d'une technique permettant d'obtenir une approximation discrète de la solution du problème (6.2) requiert d'une part la formalisation d'une représentation discrète de l'inconnue $\underline{u}$ et d'autre part l'établissement d'un problème approché dont cette dernière est solution.

▷ Le parti pris est ici

1. de dissocier la dépendance spatiale de la dépendance temporelle,
2. de considérer une suite de fonctions inconnues représentant chacune une approximation à un instant donné (ne dépendant donc plus que de l'espace),
3. d'approcher les dérivées temporelles par différences finies.

▷ On considère donc une partition $0 = t_0 < t_1 < \cdots < t_N = T$ de l'intervalle de temps $[0,T]$ telle que le $n^{\text{ième}}$ sous-intervalle $[t_n, t_{n+1}]$ soit de mesure Δt.

Nous allons établir une suite de N problèmes aux valeurs limites (où le temps n'apparaît plus) dont les solutions, notées $\underline{u}^n$ pour $1 \leq n \leq N$, sont des approximations de la solution exacte $\underline{u}(t_n, \cdot)$ aux instants t_n. Cette suite de problèmes constitue la semi-discrétisation en temps du problème continu (6.2).

VI.2.1 Système Semi-Discrétisé en Temps

Par une discrétisation en différences finies de type Euler implicite d'ordre 1, la résolution du problème (6.2) se réduit à la résolution d'une suite de problèmes aux valeurs limites définie comme suit :

- $t = 0$: $\underline{u}^0 = \underline{u}_{\text{initial}}$
- $t = t_{n+1}$: $\underline{u}^{n+1}$ solution de :

$$
\begin{array}{rll}
\alpha \dfrac{\underline{u}^{n+1} - \underline{u}^n}{\Delta t} = \nabla \cdot (\kappa \nabla \underline{u}^{n+1}) + \pi & \text{dans} & \Omega \\
\underline{u}^{n+1} = \underline{u}_0 & \text{sur} & \Gamma_{\mathrm{D}} \\
-\kappa \nabla \underline{u}^{n+1} \cdot \mathbf{n} = q_0 & \text{sur} & \Gamma_{\mathrm{N}} \\
-\kappa \nabla \underline{u}^{n+1} \cdot \mathbf{n} = h(\underline{u}^{n+1} - u_\infty) & \text{sur} & \Gamma_{\mathrm{R}}
\end{array}
\qquad (6.3)
$$

VI.3 Formulation Variationnelle

▷ On s'intéresse ici au problème (6.3) que l'on réécrit sous la forme :

Trouver u tel que :

$$\left|\begin{array}{lll} \dfrac{\alpha}{\Delta t}\, u - \nabla\cdot(\kappa\nabla u) = \dfrac{\alpha}{\Delta t}\, u^n + \pi & \text{dans} & \Omega \\ u = \underline{u}_0 & \text{sur} & \Gamma_{\mathrm{D}} \\ -\kappa\nabla u\cdot\mathbf{n} = q_0 & \text{sur} & \Gamma_{\mathrm{N}} \\ -\kappa\nabla u\cdot\mathbf{n} = h(u-u_\infty) & \text{sur} & \Gamma_{\mathrm{R}} \end{array}\right. \tag{6.4}$$

▷ La formulation variationnelle du problème (6.4) s'écrit :

Trouver $u \in \mathcal{S}^u$ tel que $\forall v \in \mathcal{V}^u$:

$$\left|\begin{array}{l} \displaystyle\int_\Omega \frac{\alpha}{\Delta t}\, u\, v + \int_\Omega \kappa\,\nabla u\cdot\nabla v + \int_{\Gamma_{\mathrm{R}}} h\, u\, v = \\ \qquad\qquad \displaystyle\int_\Omega \frac{\alpha}{\Delta t} u^n v + \int_\Omega \pi v + \int_{\Gamma_{\mathrm{R}}} h u_\infty v + \int_{\Gamma_{\mathrm{N}}} -q_0 v \end{array}\right. \tag{6.5}$$

où les espaces fonctionnels $\mathcal{S}^u$ et $\mathcal{V}^u$ sont définis comme suit :

$$\mathcal{S}^u = \left\{ v \in \mathrm{H}^1(\Omega) \;\middle|\; v = \underline{u}_0 \text{ sur } \Gamma_{\mathrm{D}} \right\}$$
$$\mathcal{V}^u = \left\{ v \in \mathrm{H}^1(\Omega) \;\middle|\; v = 0 \text{ sur } \Gamma_{\mathrm{D}} \right\}$$

▷ Supposons que $\underline{u}_0 \in \mathrm{H}^{1/2}(\Gamma_{\mathrm{D}})$. Alors il existe $u_{\mathrm{D}} \in \mathrm{H}^1(\Omega)$ tel que $u_{\mathrm{D}}|_{\Gamma_{\mathrm{D}}} = u_0$ (au sens d'un théorème de trace), ce qui va nous permettre d'introduire une nouvelle inconnue u^{n+1} au problème (6.3) définie comme suit :

$$u^{n+1} = \underline{u}^{n+1} - u_{\mathrm{D}} \tag{6.6}$$

▷ Définissons, pour $u, v \in \mathrm{H}^1(\Omega)$:

$$m(u,v) = \int_\Omega \frac{\alpha}{\Delta t}\, u\, v \tag{6.7a}$$

$$k(u,v) = \int_\Omega \kappa\,\nabla u\cdot\nabla v \tag{6.7b}$$

$$m_{\mathrm{R}}(u,v) = \int_{\Gamma_{\mathrm{R}}} h\, u\, v \tag{6.7c}$$

$$a(u,v) = m(u,v) + k(u,v) + m_{\mathrm{R}}(u,v) \tag{6.7d}$$

$$\ell(v) = \int_\Omega (\frac{\alpha}{\Delta t}\, \underline{u}^n + \pi)\, v \tag{6.7e}$$

$$\ell_{\mathrm{N}}(v) = \int_{\Gamma_{\mathrm{N}}} -q_0 v \tag{6.7f}$$

$$\ell_{\mathrm{R}}(v) = \int_{\Gamma_{\mathrm{R}}} h u_\infty v \tag{6.7g}$$

$$f(v) = \ell(v) + \ell_{\mathrm{N}}(v) + \ell_{\mathrm{R}}(v) - a(u_{\mathrm{D}}, v) \tag{6.7h}$$

▷ La nouvelle inconnue u^{n+1} est solution de :

$$\text{Trouver } u \in \mathcal{V}^u \text{ tel que } \forall v \in \mathcal{V}^u : \quad a(u,v) = f(v) \tag{6.8}$$

VI.4 Approximation Variationnelle

Une approximation de Galerkin du problème variationnel (6.8) s'obtient simplement en remplaçant l'espace $\mathcal{V}^u$ par un espace de dimension finie $\mathcal{V}_h^u$ (proche de $\mathcal{V}^u$, en un sens à définir rigoureusement).

$$\text{Trouver } u_h \in \mathcal{V}_h^u \text{ tel que } \forall v_h \in \mathcal{V}_h^u : \quad a(u_h, v_h) = f(v_h) \tag{6.9}$$

L'espace $\mathcal{V}_h^u$ que nous considérerons par la suite est construit par la méthode des éléments finis.

VI.4.1 Espaces Eléments Finis

▷ Soit $\mathcal{T}_h$ une triangulation régulière de Ω, composée de *cellules* (polyèdres de $\mathbb{R}^d$ d'intérieurs disjoints, dont la réunion recouvre $\overline{\Omega}$), de *faces* (faces des cellules) qui appartiennent soit à deux cellules distinctes, soit à la frontière Γ de Ω (auquel cas on les qualifiera de *faces frontières*). On appellera *maille* de $\mathcal{T}_h$ une cellule ou une face. De façon générique, les faces seront notées $\mathcal{F}_e$, les cellules $\mathcal{C}_e$ et les mailles K_e (l'indice e prenant dans chaque cas des valeurs appropriées).

▷ Introduisons X_h^u une approximation éléments finis de $\mathrm{H}^1(\Omega)$ construite à partir de $\mathcal{T}_h$, telle que :

$$X_h^u = \mathrm{span}\left\{ \varphi_k^u \;\middle|\; 0 \leq k < N_{\mathrm{dof}}^u \right\} \tag{6.10}$$

où les fonctions de base φ_k^u sont définies dans Ω à valeurs dans $\mathbb{R}$. Chacune d'elles peut être associée à un nœud géométrique et a son support réduit à la réunion des mailles contenant ce nœud.

▷ Toute fonction $v \in \mathrm{H}^1(\Omega)$ peut alors être approximée par son interpolation éléments finis $v_h \in X_h^u$ définie par :

$$\forall x \in \Omega \qquad v_h(x) = \sum_{0 \leq k < N_{\mathrm{dof}}^u} u_k \varphi_k^u(x)$$

où les scalaires $u_k \in \mathbb{R}$ sont appelés *degrés de liberté*. La finesse de l'approximation est déterminée par majoration d'une mesure de l'écart $(u - u_h)$.

▷ L'espace $\mathcal{V}_h^u$ qui nous intéresse ici peut être construit à partir de X_h^u en prenant en compte les valeurs imposées sur Γ_{D}.

Appelons $\mathbb{I}^u$ l'ensemble des indices k tels que $\varphi_k^u|_{\Gamma_{\mathrm{D}}}$ ne soit pas identiquement nulle. Considérons $u \in \mathcal{V}^u$ et son interpolation éléments finis $u_h \in X_h^u$ qui s'écrit :

$$u_h(x) = \sum_{0 \leq k < N_{\mathrm{dof}}^u} u_k \varphi_k^u(x) \qquad \forall x \in \Omega$$

Afin d'annuler u_h sur Γ_{D}, nous imposons $u_k = 0$ pour tout $k \in \mathbb{I}^u$, de sorte que :

$$u_h(x) = \sum_{\substack{0 \leq k < N_{\mathrm{dof}}^u \\ k \notin \mathbb{I}^u}} u_k \varphi_k^u(x) \qquad \forall x \in \Omega$$

Finalement :

$$\mathcal{V}_h^u = \mathrm{span}\left\{ \varphi_k^u \;\middle|\; 0 \leq k < N_{\mathrm{dof}}^u \;\; k \notin \mathbb{I}^u \right\} \tag{6.11}$$

▷ Dans la pratique, le relèvement des conditions aux limites u_{D} apparaît dans l'approximation variationnelle sous forme approchée $u_{\mathrm{D}h}$ au travers d'un développement suivant la famille $\left\{\varphi_k^u \middle| k \in \mathbb{I}^u\right\}$:

$$u_{\mathrm{D}h} = \sum_{k \in \mathbb{I}^u} u_{\mathrm{D}k} \varphi_k^u$$

les scalaires $u_{\mathrm{D}k} \in \mathbb{R}$ étant déterminés à partir de la donnée $\underline{u}_0$.

VI.4.2 Formulation Matricielle du Problème Discret

▷ Le problème discret (6.8) peut être formulé sous forme matricielle dans un premier temps en choisissant pour v chacun des vecteurs de la base donnée par la relation (6.11), puis, dans un second temps, en développant u dans cette même base. Les coefficients intervenant dans ces développements seront les composantes du vecteur inconnu du système algébrique. Cette démarche requiert une réindexation contiguë des fonctions de base de $\mathcal{V}_h^u$:

$$k \in [0, N^u_{\text{dof}}[\backslash \mathbb{I}^u \quad \xrightarrow[\text{contiguë}]{\text{indexation}} \quad I \in [0, N^u_{\text{unk}}[\qquad \text{avec} \quad N^u_{\text{unk}} = N^u_{\text{dof}} - \operatorname{card} \mathbb{I}^u \tag{6.12}$$

Ainsi, on a :

$$\mathcal{V}_h^u = \operatorname{span}\left\{ \; \varphi_I^u \; \middle| \; 0 \leq I < N^u_{\text{unk}} \; \right\} \tag{6.13}$$

▷ Etant donnés le problème discret (6.8) et la base de $\mathcal{V}_h^u$ exhibée par la relation (6.13), l'algorithme de marche en temps conduit, au pas de temps n, à rechercher les vecteur $\mathbf{U}^{n+1}$ solution de :

$$\text{Trouver } \mathbf{U} \in \mathbb{R}^{N^u_{\text{unk}}} \qquad \text{tel que} \qquad \mathbf{AU} = \mathbf{F} \tag{6.14}$$

avec :

$$\mathbf{A}_{IJ} = a(\varphi_J^u, \varphi_I^u) = m(\varphi_J^u, \varphi_I^u) + k(\varphi_J^u, \varphi_I^u) + m_{\text{R}}(\varphi_J^u, \varphi_I^u) \qquad 0 \leq I, J < N^u_{\text{unk}} \tag{6.15}$$

$$\mathbf{F}_I = f(\varphi_I^u) = \ell(\varphi_I^u) + \ell_{\text{N}}(\varphi_I^u) + \ell_{\text{R}}(\varphi_I^u) - a(u_{\text{D}h}, \varphi_I^u) \qquad 0 \leq I < N^u_{\text{unk}} \tag{6.16}$$

$$\tag{6.17}$$

Sous réserve que l'on sache résoudre (6.14) à chaque pas de temps, on aura obtenu une approximation discrète de la solution du problème continu (6.2) sous la forme :

$$\underline{u}(n\Delta t, x) \approx \sum_{0 \leq I < N^u_{\text{unk}}} \mathbf{U}_I \varphi_I^u(x) + u_{\text{D}h} \qquad \forall n \in [0, N] \quad \forall x \in \Omega \tag{6.18}$$

VI.4.3 Assemblages Elémentaires

▷ Le champ $\underline{u}$ a une unique composante. Cependant, afin de rester au plus près du formalisme habituel des applications PELICANS (§I.2.1), nous allons différencier l'indexation des *degrés de liberté* et celle des *nœuds*.

Ainsi, chaque fonction φ_k^u de la base de X_k^u (6.10) définit une fonction de base scalaire $\boldsymbol{\mathsf{N}}_i^u$ (avec ici $i \equiv k$ puis $\underline{u}$ a une unique composante) qui a la propriété fondamentale d'être associée à un nœud géométrique $\mathbf{a}_i^u \in \overline{\Omega}$, d'être nulle en dehors des mailles contenant $\mathbf{a}_i^u$, et d'être définie à travers sa restriction à chacune d'elles. En conséquence, les fonctions de base $\boldsymbol{\mathsf{N}}_i^u$ peuvent être agréablement numérotées localement à chaque maille K_e :

$$\forall i \in [0, N^u_{\text{node}}[\quad \text{tq} \quad \operatorname{support}(\,\boldsymbol{\mathsf{N}}_i^u\,) \cap K_e \neq \emptyset \quad \xrightarrow[\text{locale à } K_e]{\text{indexation}} \quad i \in [0, N_e^u[\tag{6.19}$$

La notation i est clairement incomplète, car elle ne contient aucune référence à la maille K_e considéré. Toute ambiguité sera cependant levée par le contexte.

▷ Le système algébrique (6.14)(6.15) est construit par ajouts successifs des contributions de chaque maille :

$$\begin{bmatrix} \mathbf{A} \\ \mathbf{F} \end{bmatrix} = \sum_{e \,:\, \text{cellules}} \boldsymbol{\mathsf{A}}^{\mathsf{mat}}_{\mathsf{vec}} \begin{bmatrix} m^e + k^e \\ \ell^e \end{bmatrix} + \sum_{\substack{e \,:\, \text{faces} \\ \text{de } \Gamma_{\text{R}}}} \boldsymbol{\mathsf{A}}^{\mathsf{mat}}_{\mathsf{vec}} \begin{bmatrix} m^e_{\text{R}} \\ \ell^e_{\text{R}} \end{bmatrix} + \sum_{\substack{e \,:\, \text{faces} \\ \text{de } \Gamma_{\text{N}}}} \boldsymbol{\mathsf{A}}^{\mathsf{mat}}_{\mathsf{vec}} \begin{bmatrix} 0 \\ \ell^e_{\text{N}} \end{bmatrix} \tag{6.20}$$

où :

- les contributions élémentaires sont données par[1] :

$$
\begin{aligned}
& m^e = \big[\, m^e(\,i,j\,) \,\big] \qquad k^e = \big[\, k^e(\,i,j\,) \,\big] \qquad m^e_{\mathrm{R}} = \big[\, m^e_{\mathrm{R}}(\,i,j\,) \,\big] \qquad 0 \le i,j < N^u_e \\
& m^e(\,i,j\,) = \int_{\mathcal{C}_e} \frac{\alpha}{\Delta t}\, \varphi^u_J\, \varphi^u_I = \int_{\mathcal{C}_e} \frac{\alpha}{\Delta t}\, \mathsf{N}^u_j\, \mathsf{N}^u_i \\
& k^e(\,i,j\,) = \int_{\mathcal{C}_e} \kappa\, \nabla\varphi^u_J \cdot \nabla\varphi^u_I = \int_{\mathcal{C}_e} \kappa\, \nabla\mathsf{N}^u_j \cdot \nabla\mathsf{N}^u_i \\
& m^e_{\mathrm{R}}(\,i,j\,) = \int_{\mathcal{F}_e} h\, \varphi^u_J\, \varphi^u_I = \int_{\mathcal{F}_e} h\, \mathsf{N}^u_j\, \mathsf{N}^u_i \qquad \mathcal{F}_e \subset \Gamma_{\mathrm{R}} \\
& \ell^e = \big[\, \ell^e(\,i\,) \,\big] \qquad \ell^e_{\mathrm{N}} = \big[\, \ell^e_{\mathrm{N}}(\,i\,) \,\big] \qquad \ell^e_{\mathrm{R}} = \big[\, \ell^e_{\mathrm{R}}(\,i\,) \,\big] \qquad 0 \le i < N^u_e \\
& \ell^e(\,i\,) = \int_{\mathcal{C}_e} \big(\frac{\alpha}{\Delta t}\, \underline{u}^n + \pi\big)\, \varphi^u_I = \int_{\mathcal{C}_e} \big(\frac{\alpha}{\Delta t}\, \underline{u}^n + \pi\big)\, \mathsf{N}^u_i \\
& \ell^e_{\mathrm{N}}(\,i\,) = \int_{\mathcal{F}_e} -q_0\, \varphi^u_I = \int_{\mathcal{F}_e} -q_0\, \mathsf{N}^u_i \qquad \mathcal{F}_e \subset \Gamma_{\mathrm{N}} \\
& \ell^e_{\mathrm{R}}(\,i\,) = \int_{\mathcal{F}_e} h\, u_\infty\, \varphi^u_I = \int_{\mathcal{F}_e} h\, u_\infty\, \mathsf{N}^u_i \qquad \mathcal{F}_e \subset \Gamma_{\mathrm{R}}
\end{aligned}
\tag{6.21}
$$

- $\mathsf{A}^{\mathsf{mat}}_{\mathsf{vec}}$ est l'opérateur d'assemblage qui ajoute les contributions élémentaires à l'endroit approprié dans la matrice **A** et le vecteur **F**, en prenant soin d'éliminer, dans les inconnues du système global, les degrés de liberté d'indice appartenant à $\mathbb{I}^u$ et de reporter les contributions associées au second membre (§II.2.2). Cette élimination des degrés de liberté imposés *a priori*, au moment de l'assemblage, est la méthode utilisée dans la pratique pour soustraire à l'inconnue $\underline{u}$ le relèvement u_{D} des conditions aux limites sur Γ_{D} (équations (6.6) et (6.7h)).

VI.5 Implémentation Informatique

VI.5.1 Une Itération en Temps

`MI_Diffusion`

Données :

u_h	(u)	(e)	α	κ	π

conditions aux limites sur Γ_{N} : q_0

conditions aux limites sur Γ_{R} : h, u_∞

`MI_Diffusion::do_one_inner_iteration`

- Calculer les termes élémentaires :

[1] l'indice e désigne alternativement une cellule ou une face frontière.

boucles sur les cellules	
$m^e(\, i, j \,) = \displaystyle\int_{\mathcal{C}_e} \frac{\alpha}{\Delta t}\, \varphi^u_J \, \varphi^u_I$	`FE::add_row_col_S`
$k^e(\, i, j \,) = \displaystyle\int_{\mathcal{C}_e} \kappa \, \nabla\varphi^u_J \cdot \nabla\varphi^u_I$	`FE::add_grad_row_grad_col_S`
$\ell^e(\, i \,) = \displaystyle\int_{\mathcal{C}_e} (\frac{\alpha}{\Delta t}\, u_h^{(\mathrm{e})} + \pi) \, \varphi^u_I$	`FE::add_row`
boucles sur les faces de Γ_{R}	
$m^e_{\mathrm{R}}(\, i, j \,) = \displaystyle\int_{\mathcal{F}_e} h \, \varphi^u_J \, \varphi^u_I$	`FE::add_row_col_S`
$\ell^e_{\mathrm{R}}(\, i \,) = \displaystyle\int_{\mathcal{F}_e} h \, u_\infty \, \varphi^u_I$	`FE::add_row`
boucles sur les faces de Γ_{N}	
$\ell^e_{\mathrm{N}}(\, i \,) = \displaystyle\int_{\mathcal{F}_e} -q_0 \, \varphi^u_I$	`FE::add_row`

- Assemblage de $\mathbf{A}$ et $\mathbf{F}$ (6.20) :
  utilisation de la classe `PDE_BlockAssembledSystem`.
- Calcul de $\mathbf{U}$ solution de $\mathbf{AU} = \mathbf{F}$ (6.14) :
  utilisation de la classe `PDE_BlockAssembledSystem`.
- Mise à jour :

$$u_h^{(\mathrm{u})} = \sum_{0 \leq I < N^u_{\mathrm{unk}}} \mathbf{U}_I \varphi^u_I(x) + u_{\mathrm{D}h}$$

VI.5.2 Une Marche en Temps

Sur la base d'un objet `Diff` de type `MI_Diffusion`, les itérations suivantes sont réalisées :

```
Du pas de temps 1 au pas de temps $N$ :
    Diff.do_one_inner_iteration()
Fin de la marche en temps
```

Chapter VII

Galerkin Solvers for the Incompressible Navier-Stokes Problem

VII.1 Position du Problème

L'écoulement isotherme incompressible d'un fluide Newtonien est régi par le système d'équations de Navier-Stokes, que nous écrirons en variables primitives sous la forme :

$$\rho\Big[\frac{\partial\underline{\mathbf{u}}}{\partial t}+(\underline{\mathbf{u}}\cdot\nabla)\underline{\mathbf{u}}\Big]=-\nabla\underline{p}+\nabla\cdot\tau(\underline{\mathbf{u}})+\rho\mathbf{g}\qquad\text{dans}\qquad[0,T]\times\Omega\qquad(7.1a)$$

$$\nabla\cdot\underline{\mathbf{u}}=0\qquad\text{dans}\qquad[0,T]\times\Omega\qquad(7.1b)$$

$$\underline{\mathbf{u}}=\underline{\mathbf{u}}_0\qquad\text{sur}\qquad[0,T]\times\Gamma_{\mathrm{D}}\qquad(7.1c)$$

$$-\underline{p}\mathbf{n}+\tau(\underline{\mathbf{u}})\cdot\mathbf{n}=\boldsymbol{\tau}_{\mathrm{N}}\qquad\text{sur}\qquad[0,T]\times\Gamma_{\mathrm{N}}\qquad(7.1d)$$

$$\underline{\mathbf{u}}=\underline{\mathbf{u}}_{\text{initial}}\qquad\text{dans}\qquad\{0\}\times\Omega\qquad(7.1e)$$

où

- $[0,T]$ représente l'intervalle temporel d'étude ;
- Ω est un ouvert connexe borné de $\mathbb{R}^d$ ($d=2$ ou 3) dont la frontière Γ, supposée suffisamment régulière, est subdivisée en $\Gamma=\Gamma_{\mathrm{D}}\cup\Gamma_{\mathrm{N}}$ avec $\Gamma_{\mathrm{D}}\neq\emptyset$ et $\Gamma_{\mathrm{D}}\cap\Gamma_{\mathrm{N}}=\emptyset$;
- ρ est la masse volumique (kg.m^{-3}) ;
- la vitesse du fluide, définie dans $[0,T]\times\Omega$ à valeurs dans $\mathbb{R}^d$ est notée $\underline{\mathbf{u}}$ et prend à chaque instant la valeur $\underline{\mathbf{u}}_0$ sur Γ_{D} ;
- la pression thermodynamique, définie dans $[0,T]\times\Omega$ à valeurs dans $\mathbb{R}$ est notée $\underline{p}$;
- les efforts à distance exercés sur tout domaine matériel inclus dans Ω sont représentés par la densité massique $\mathbf{g}$;
- les efforts de contact exercés à la frontière de tout domaine matériel inclus dans Ω sont représentés par le tenseur des contraintes $\boldsymbol{\sigma}$, du second ordre, symétrique, dont la partie sphérique admet la pression thermodynamique (au signe près) comme valeur propre et dont le déviateur dépend linéairement du gradient de vitesse symétrisé :

  $$\boldsymbol{\sigma}=-\underline{p}\mathrm{Id}+\tau(\underline{\mathbf{u}})\quad\text{avec}\quad\tau(\underline{\mathbf{u}})=\eta(\nabla\underline{\mathbf{u}}+\nabla\underline{\mathbf{u}}^{\mathrm{T}})\qquad(7.2)$$

  où η est la viscosité dynamique (Pa.s) ;
- en tous points de Γ_{N} et à chaque instant, le vecteur contrainte pour la direction définie par la normale unitaire sortante $\mathbf{n}$ prend la valeur $\boldsymbol{\tau}_{\mathrm{N}}$.

Le problème que nous nous proposons d'étudier est le suivant :

> Etant *a priori* données les grandeurs ρ, $\mathbf{g}$, η, $\underline{\mathbf{u}}_0$, $\boldsymbol{\tau}_{\mathrm{N}}$, $\underline{\mathbf{u}}_{\mathrm{initial}}$
> trouver $(\underline{\mathbf{u}}, \underline{p})$ solution du système d'équations (8.15) (7.3)

Remarque :

Il est possible de généraliser les conditions aux limites du système (8.15) : supposons la frontière de Ω subdivisée en plusieurs parties sans recouvrement et donnons nous pour l'une quelconque de ces parties un système de coordonnées. Les développements qui vont suivre présupposent uniquement que pour chaque direction e_i de ce système de coordonnées, soit la composante $\underline{\mathbf{u}} \cdot e_i$ de la vitesse est imposée, soit la composante $(\boldsymbol{\sigma} \cdot \mathbf{n}) \cdot e_i$ de la contrainte est imposée. Pour alléger les notations, nous nous limiterons aux conditions (7.1c) et (7.1d) tout en gardant en mémoire le cadre plus général dans lequel elles s'inscrivent.

VII.2 Discrétisation en Temps

L'élaboration d'une technique permettant d'obtenir une approximation discrète de la solution[1] du problème (7.3) requiert d'une part la formalisation d'une représentation discrète de l'inconnue $(\underline{\mathbf{u}}, \underline{p})$ et d'autre part l'établissement d'un problème approché dont cette dernière est solution.

Le parti pris est ici

1. de dissocier la dépendance spatiale de la dépendance temporelle,
2. de considérer une suite de fonctions inconnues représentant chacune une approximation à un instant donné (ne dépendant donc plus que de l'espace),
3. d'approcher les dérivées temporelles par différences finies.

On considère donc une partition $0 = t_0 < t_1 < \cdots < t_N = T$ de l'intervalle de temps $[0, T]$ telle que le $n^{\text{ième}}$ sous-intervalle $[t_n, t_{n+1}]$ soit de mesure Δt.

Nous allons établir une suite de N problèmes aux valeurs limites (où le temps n'apparaît plus) dont les solutions, notées $[\underline{\mathbf{u}}^n, \underline{p}^n]$ pour $1 \leq n \leq N$, sont des approximations de la solution exacte $[\underline{\mathbf{u}}(t_n, \cdot), \underline{p}(t_n, \cdot)]$ aux instants t_n. Cette suite de problèmes constitue la semi-discrétisation en temps du problème continu (7.3).

VII.2.1 Approximation par Différentiation Rétrograde de la Dérivée Temporelle

Soit $\phi \colon [0, T] \to \mathbb{R}$ une fonction arbitraire suffisamment régulière.

Notons $\phi^n = \phi(t_n)$ pour $0 \leq n \leq N$.

Etant donnés $\phi^0, \ldots, \phi^n$, on obtient une approximation à l'ordre q de la valeur de ϕ à l'instant t_{n+1} par l'extrapolation de Richardson :

$$\phi(t_{n+1}) = \phi^{\star,n+1} + O(\Delta t^q) \quad \text{avec} \quad \phi^{\star,n+1} \overset{\text{def}}{=} \sum_{j=0}^{q-1} \gamma_j \phi^{n-j} \tag{7.4}$$

De plus, une approximation à l'ordre q de la dérivée de ϕ à l'instant t_{n+1} est donnée par la formule de différentiation rétrograde :

$$\frac{\partial \phi}{\partial t}(t_{n+1}) = \frac{D\phi^{n+1}}{\Delta t} + O(\Delta t^q) \quad \text{avec} \quad D\phi^{n+1} \overset{\text{def}}{=} \beta_q \phi^{n+1} - \sum_{j=0}^{q-1} \beta_j \phi^{n-j} \tag{7.5}$$

[1] il ne faut pas voir dans l'expression “la solution” une référence quelconque à un éventuel théorème d'existence et d'unicité qui dépasse le cadre du présent travail.

Dans la pratique, nous utiliserons des approximations à l'ordre 1 et 2 où les coefficient β_j et γ_j sont donnés comme suit :

$$q = 1 \qquad D\phi^{n+1} = \phi^{n+1} - \phi^n \qquad \phi^{\star,n+1} = \phi^n \tag{7.6}$$

$$q = 2 \qquad D\phi^{n+1} = \frac{3}{2}\phi^{n+1} - 2\phi^n + \frac{1}{2}\phi^{n-1} \qquad \phi^{\star,n+1} = 2\phi^n - \phi^{n-1} \tag{7.7}$$

VII.2.2 Système Semi-Discrétisé en Temps

En utilisant les approximations (7.4) et (7.5), la résolution du problème (7.3) se réduit à la résolution d'une suite de problèmes aux valeurs limites définie comme suit :

- $t = 0$: $\underline{\mathbf{u}}^0 = \underline{\mathbf{u}}_{\text{initial}}$
- $t = t_{n+1}$: $(\underline{\mathbf{u}}^{n+1}, \underline{p}^{n+1})$ solution de :

$$\rho\Big[\frac{D\underline{\mathbf{u}}^{n+1}}{\Delta t} + (\underline{\mathbf{u}}^{\star,n+1}\cdot\nabla)\underline{\mathbf{u}}^{n+1}\Big] = -\nabla\underline{p}^{n+1} + \nabla\cdot\tau(\underline{\mathbf{u}}^{n+1}) + \rho\mathbf{g} \quad \text{dans} \quad \Omega \tag{7.8a}$$

$$\nabla\cdot\underline{\mathbf{u}}^{n+1} = 0 \quad \text{dans} \quad \Omega \tag{7.8b}$$

$$\underline{\mathbf{u}}^{n+1} = \underline{\mathbf{u}_0}^{n+1} \quad \text{sur} \quad \Gamma_{\mathrm{D}} \tag{7.8c}$$

$$-\underline{p}^{n+1}\mathbf{n} + \tau(\underline{\mathbf{u}}^{n+1})\cdot\mathbf{n} = \boldsymbol{\tau}_{\mathrm{N}}{}^{n+1} \quad \text{sur} \quad \Gamma_{\mathrm{N}} \tag{7.8d}$$

VII.3 Formulation Variationnelle

▷ On s'intéresse ici au problème (7.8) que l'on réécrit sous la forme :

Trouver $(\mathbf{u}, p)$ tel que :

$$\begin{aligned} \frac{\beta_q}{\Delta t}\rho\mathbf{u} + (\rho\underline{\mathbf{u}}^{\star,n+1}\cdot\nabla)\mathbf{u} - \nabla\cdot\tau(\mathbf{u}) + \nabla p &= \rho\mathbf{g} + \sum_{j=0}^{q-1}\frac{\beta_j}{\Delta t}\rho\mathbf{u}^{n-j} && \text{dans} \quad \Omega \\ \nabla\cdot\mathbf{u} &= 0 && \text{dans} \quad \Omega \\ \mathbf{u} &= \underline{\mathbf{u}_0}^{n+1} && \text{sur} \quad \Gamma_{\mathrm{D}} \\ -p\mathbf{n} + \tau(\mathbf{u})\cdot\mathbf{n} &= \boldsymbol{\tau}_{\mathrm{N}}{}^{n+1} && \text{sur} \quad \Gamma_{\mathrm{N}} \end{aligned} \tag{7.9}$$

▷ La formulation variationnelle du problème (8.5) s'écrit :

Trouver $(\mathbf{u}, p) \in \mathcal{S}^{\mathbf{u}} \times \mathcal{V}^p$ tel que $\forall(\mathbf{v}, q) \in \mathcal{V}^{\mathbf{u}} \times \mathcal{V}^p$:

$$\left|\begin{array}{l} \displaystyle\frac{\beta_q}{\Delta t}\int_\Omega \rho\mathbf{u}\cdot\mathbf{v} + \int_\Omega (\rho\underline{\mathbf{u}}^{\star,n+1}\cdot\nabla)\mathbf{u}\ \cdot\mathbf{v} + \int_\Omega \tau(\mathbf{u})\colon\nabla\mathbf{v} - \int_\Omega p\nabla\cdot\mathbf{v} = \\ \displaystyle\qquad\qquad \int_\Omega \rho\mathbf{g}\cdot\mathbf{v} + \int_{\Gamma_{\mathrm{N}}} \boldsymbol{\tau}_{\mathrm{N}}{}^{n+1}\cdot\mathbf{v} + \int_\Omega \sum_{j=0}^{q-1}\frac{\beta_j}{\Delta t}\rho\mathbf{u}^{n-j}\cdot\mathbf{v} \\ \displaystyle\int_\Omega q\nabla\cdot\mathbf{u} = 0 \end{array}\right. \tag{7.10}$$

où les espaces fonctionnels $\mathcal{S}^{\mathbf{u}}$, $\mathcal{V}^{\mathbf{u}}$ et $\mathcal{V}^p$ sont définis comme suit :

$$\mathcal{S}^{\mathbf{u}} = \left\{\mathbf{v} \in \mathrm{H}^1(\Omega)^d \;\middle|\; \mathbf{v} = \underline{\mathbf{u}}_0 \text{ sur } \Gamma_{\mathrm{D}}\right\}$$

$$\mathcal{V}^{\mathbf{u}} = \left\{\mathbf{v} \in \mathrm{H}^1(\Omega)^d \;\middle|\; \mathbf{v} = 0 \text{ sur } \Gamma_{\mathrm{D}}\right\}$$

$$\mathcal{V}^p = \mathrm{L}^2(\Omega) \quad \text{si} \quad \Gamma_{\mathrm{N}} \neq \emptyset$$

$$\mathcal{V}^p = \left\{q \in \mathrm{L}^2(\Omega) \;\middle|\; \int_\Omega q = 0\right\} \quad \text{si} \quad \Gamma_{\mathrm{N}} = \emptyset$$

▷ Supposons que $\underline{\mathbf{u}}_0 \in \mathrm{H}^{1/2}(\Gamma_\mathrm{D})^d$. Alors il existe $\mathbf{u}_\mathrm{D} \in \mathrm{H}^1(\Omega)^d$ tel que $\mathbf{u}_\mathrm{D}|_{\Gamma_\mathrm{D}} = \underline{\mathbf{u}}_0$ (au sens d'un théorème de trace), ce qui va nous permettre d'introduire une nouvelle inconnue $(\mathbf{u}^{n+1}, p^{n+1})$ au problème (7.8) définie comme suit :

$$\mathbf{u}^{n+1} = \underline{\mathbf{u}}^{n+1} - {\mathbf{u}_\mathrm{D}}^{n+1} \qquad p^{n+1} = \underline{p}^{n+1} \tag{7.11}$$

▷ Définissons, pour $u, v, w \in \mathrm{H}^1(\Omega)^d$ et $q \in \mathrm{L}^2(\Omega)$:

$$k(u,v) = \int_\Omega \tau(u) \colon \nabla v \tag{7.12a}$$

$$b(v,q) = -\int_\Omega q \nabla \cdot v \tag{7.12b}$$

$$c(u;v,w) = \int_\Omega \rho (u \cdot \nabla) v \cdot w \tag{7.12c}$$

$$m(u,v) = \int_\Omega \rho u \cdot v \tag{7.12d}$$

$$\ell(v) = \int_\Omega \rho \mathbf{g} \cdot v + \int_{\Gamma_\mathrm{N}} {\boldsymbol{\tau}_\mathrm{N}}^{n+1} \cdot v \tag{7.12e}$$

$$f(v) = \ell(v) - \frac{1}{\Delta t} m(\beta_q {\mathbf{u}_\mathrm{D}}^{n+1} + \sum_{j=0}^{q-1} \beta_j \underline{\mathbf{u}}^{n-j}, v) - c(\underline{\mathbf{u}}^{\star,n+1}; {\mathbf{u}_\mathrm{D}}^{n+1}, v) - k({\mathbf{u}_\mathrm{D}}^{n+1}, v) \tag{7.12f}$$

$$g(q) = -b({\mathbf{u}_\mathrm{D}}^{n+1}, p) \tag{7.12g}$$

▷ La nouvelle inconnue $(\mathbf{u}^{n+1}, p^{n+1})$ est solution de :

Trouver $(\mathbf{u}, p) \in \mathcal{V}^\mathbf{u} \times \mathcal{V}^p$ tel que $\forall (\mathbf{v}, q) \in \mathcal{V}^\mathbf{u} \times \mathcal{V}^p$:

$$\left| \begin{array}{l} \dfrac{\beta_q}{\Delta t} m(\mathbf{u}, \mathbf{v}) + c(\underline{\mathbf{u}}^{\star,n+1}; \mathbf{u}, \mathbf{v}) + k(\mathbf{u}, \mathbf{v}) + b(\mathbf{v}, p) = f(\mathbf{v}) \\ b(\mathbf{u}, q) = g(q) \end{array} \right. \tag{7.13}$$

VII.4 Méthode de Correction de Pression

▷ Les techniques de résolution numérique du système (7.8) qui seront présentées par la suite utilisent toutes comme point de départ une approximation de Galerkin du problème variationnel (8.6) et se différencient par le traitement ultérieur du problème matriciel qui en découle. Ainsi les méthodes de projection seront-elles exposées dans un contexte algébrique, ce qui présente un double avantage.

- D'un point de vue pratique, l'implémentation informatique naturelle qui en résulte est bien encapsulée (construction du système discret d'une part, résolution approchée de ce système d'autre part).
- D'un point de vue numérique, il est préférable de construire des termes d'augmentation liés à la contrainte $\nabla \cdot \underline{\mathbf{u}}^{n+1} = 0$ à partir de la formulation algébrique, puisque cette contrainte n'est généralement pas satisfaite, au sens donné dans le problème continu, par la solution discrète. En effet, souvent, seule la divergence faible de l'interpolation éléments finis $\underline{\mathbf{u}}_h$ de la vitesse est nulle :

$$\nabla \cdot \underline{\mathbf{u}}_h \neq 0 \quad \text{mais} \quad \int_\Omega q_h \, \nabla \cdot \underline{\mathbf{u}}_h = 0 \quad \text{pour toute fonction test } q_h \text{ de pression}$$

Cependant, une présentation exclusivement algébrique comporte un aspect arbitraire qui cache certaines idées fondatrices issues des stratégies à pas fractionnaires découplés appliquées à la résolution de (7.8). C'est pourquoi nous allons ici succinctement exposer l'une d'elles : la méthode de correction de pression.

- ▷ Dans sa version classique, la méthode de correction de pression décompose chaque pas de temps en sous-pas comme décrit ci-dessous.
  - ▪ Dans une première étape, le gradient de pression est fixé à sa valeur de début de pas de temps dans l'équation de bilan de quantité de mouvement, dont la résolution conduit alors à une estimation $\widetilde{\underline{\mathbf{u}}}$ de la vitesse au temps t_{n+1} qui ne vérifie pas la contrainte d'incompressibilité :

$$\rho\Big[\frac{\beta_q\widetilde{\underline{\mathbf{u}}}-\sum_{j=0}^{q-1}\beta_j\underline{\mathbf{u}}^{n-j}}{\Delta t}+(\underline{\mathbf{u}}^{\star,n+1}\cdot\nabla)\widetilde{\underline{\mathbf{u}}}\Big]=-\nabla\underline{p}^n+\nabla\cdot\tau(\widetilde{\underline{\mathbf{u}}})+\rho\mathbf{g} \qquad \text{dans } \Omega \qquad (7.14a)$$

$$\widetilde{\underline{\mathbf{u}}}={\underline{\mathbf{u}}_0}^{n+1} \qquad \text{sur } \Gamma_{\mathrm{D}} \qquad (7.14b)$$

$$-\underline{p}^n\mathbf{n}+\tau(\widetilde{\underline{\mathbf{u}}})\cdot\mathbf{n}={\boldsymbol{\tau}_{\mathrm{N}}}^{n+1} \qquad \text{sur } \Gamma_{\mathrm{N}} \qquad (7.14c)$$

  - ▪ On introduit ensuite la variété affine :

$$\mathrm{H}=\big\{v\in\mathrm{L}^2(\Omega)^d\ \big|\ \nabla\cdot v=0\ \ \text{dans}\ \ \Omega\quad\text{et}\quad v\cdot\mathbf{n}={\underline{\mathbf{u}}_0}^{n+1}\cdot\mathbf{n}\ \text{sur}\ \Gamma_{\mathrm{D}}\big\}$$

puis on effectue une projection $\mathrm{L}^2(\Omega)$-orthogonale de $\widetilde{\underline{\mathbf{u}}}$ sur H, sous la forme :

$$\rho\beta_q\frac{\underline{\mathbf{u}}^{n+1}-\widetilde{\underline{\mathbf{u}}}}{\Delta t}+\nabla\phi=0 \qquad \text{dans}\ \ \Omega \qquad (7.15a)$$

$$\nabla\cdot\underline{\mathbf{u}}^{n+1}=0 \qquad \text{dans}\ \ \Omega \qquad (7.15b)$$

$$\underline{\mathbf{u}}^{n+1}={\underline{\mathbf{u}}_0}^{n+1} \qquad \text{sur}\ \ \Gamma_{\mathrm{D}} \qquad (7.15c)$$

$$\nabla\phi\cdot\mathbf{n}=0 \qquad \text{sur}\ \ \Gamma_{\mathrm{D}} \qquad (7.15d)$$

$$\phi=0 \qquad \text{sur}\ \ \Gamma_{\mathrm{N}} \qquad (7.15e)$$

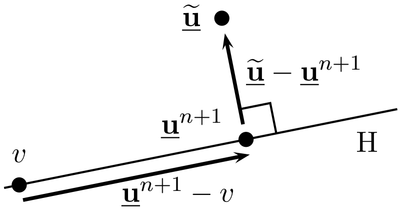
enfin, une identification formelle entre d'une part le bilan de quantité de mouvement au temps t_{n+1} (7.8a) et d'autre part la somme des relations (7.14a) et (7.15a) suggère l'expression suivante :

$$\underline{p}^{n+1}=\underline{p}^n+\phi$$

Ainsi le champ ϕ est-il qualifié d'incrément de pression.

- ▷ La condition aux limites (7.15d) est une conséquence directe des relations (7.15a) et (7.14b), (7.15c).
- ▷ La condition aux limites (7.15e) est essentielle pour pouvoir interpréter (7.15a),(7.15b) comme une étape de projection : en effet, la condition de L^2-orthogonalité s'écrit :

$$\int_\Omega(\widetilde{\underline{\mathbf{u}}}-\underline{\mathbf{u}}^{n+1})\cdot(\underline{\mathbf{u}}^{n+1}-v)=0\quad\forall v\in\mathrm{H}$$


$\widetilde{\underline{\mathbf{u}}}$
$\widetilde{\underline{\mathbf{u}}}-\underline{\mathbf{u}}^{n+1}$
$\underline{\mathbf{u}}^{n+1}$
H
v
$\underline{\mathbf{u}}^{n+1}-v$

soit :

$$0=\int_\Omega\nabla\phi\cdot(\underline{\mathbf{u}}^{n+1}-v)=\oint_\Gamma\phi\,(\underline{\mathbf{u}}^{n+1}-v)\cdot\mathbf{n}-\int_\Omega\phi\,\nabla\cdot(\underline{\mathbf{u}}^{n+1}-v) \qquad (7.16)$$

$$=\int_{\Gamma_{\mathrm{N}}}\phi(\underline{\mathbf{u}}^{n+1}-v)\cdot\mathbf{n} \qquad (7.17)$$

ce qui est vérifié si : $\phi=0$ sur Γ_{N}.

- ▷ Les conditions aux limites (7.15d) et (7.15e) sont qualifiées d'artificielles car les relations sur la pression qui en découlent :

$$\nabla\underline{p}^{n+1}=\nabla\underline{p}^n=\cdots=\nabla\underline{p}^0 \qquad \text{sur} \qquad \Gamma_{\mathrm{D}} \qquad (7.18)$$

$$\underline{p}^{n+1}=\underline{p}^n=\cdots=\underline{p}^0 \qquad \text{sur} \qquad \Gamma_{\mathrm{N}} \qquad (7.19)$$

n'ont *a priori* aucune raison d'être satisfaites par la solution physique. En particulier, si $\Gamma_\mathrm{N} \neq \emptyset$, la condition (7.15e) peut conduire à une dégradation significative de l'approximation de la pression.

▷ Pour des raisons d'efficacité, le problème de Darcy (7.15) est reformulé en prenant la divergence de (7.15a), ce qui conduit à un problème de Poisson pour ϕ :

$$\nabla^2 \phi = \frac{\rho \beta_q}{\Delta t} \nabla \cdot \underline{\tilde{\mathbf{u}}} \qquad \text{dans} \quad \Omega \tag{7.20a}$$

$$\nabla \phi \cdot \mathbf{n} = 0 \qquad \text{sur} \quad \Gamma_\mathrm{D} \tag{7.20b}$$

$$\phi = 0 \qquad \text{sur} \quad \Gamma_\mathrm{N} \tag{7.20c}$$

VII.5 Approximation Variationnelle

Une approximation de Galerkin du problème variationnel (8.6) s'obtient simplement en remplaçant les espaces $\mathcal{V}^\mathbf{u}$ et $\mathcal{V}^p$ par des espaces de dimension finie $\mathcal{V}_h^\mathbf{u}$ et $\mathcal{V}_h^p$ (proches de $\mathcal{V}^\mathbf{u}$ et $\mathcal{V}^p$, en un sens à définir rigoureusement).

Trouver $(\mathbf{u}_h, p_h) \in \mathcal{V}_h^\mathbf{u} \times \mathcal{V}_h^p$ tel que $\forall (\mathbf{v}_h, q_h) \in \mathcal{V}_h^\mathbf{u} \times \mathcal{V}_h^p$:

$$\left| \begin{array}{l} \dfrac{\beta_q}{\Delta t} m(\mathbf{u}_h, \mathbf{v}_h) + c(\underline{\mathbf{u}}_h^{\star,n+1}; \mathbf{u}_h, \mathbf{v}_h) + k(\mathbf{u}_h, \mathbf{v}_h) + b(\mathbf{v}_h, p_h) = f(\mathbf{v}_h) \\ b(\mathbf{u}_h, q_h) = g(q_h) \end{array} \right. \tag{7.21}$$

Les espaces $\mathcal{V}_h^\mathbf{u}$ et $\mathcal{V}_h^p$ que nous considérerons par la suite sont construits par la méthode des éléments finis.

VII.5.1 Espaces Eléments Finis

▷ Soit $\mathcal{T}_h$ une triangulation régulière de Ω.

Introduisons $\{X_h, M_h\}$ une approximation éléments finis mixte de $\{\mathrm{H}^1(\Omega)^d, \mathrm{L}^2(\Omega)\}$ construite à partir de $\mathcal{T}_h$, telle que :

$$X_h = \mathrm{span}\{\ \boldsymbol{\varphi}_k^\mathbf{u} \ \big|\ 0 \leq k < N_\mathrm{dof}^\mathbf{u}\ \} \tag{7.22}$$

$$M_h = \mathrm{span}\{\ \varphi_k^p \ \big|\ 0 \leq k < N_\mathrm{dof}^p\ \} \tag{7.23}$$

où les fonctions de base $\boldsymbol{\varphi}_k^\mathbf{u}$ (resp. φ_k^p) sont définies dans Ω à valeur dans $\mathbb{R}^d$ (resp. $\mathbb{R}$). Chacune d'elles peut être associée à un nœud géométrique et a son support réduit à la réunion des mailles contenant ce nœud.

▷ Tout couple de fonctions $(v, q) \in \mathrm{H}^1(\Omega)^d \times \mathrm{L}^2(\Omega)$ peut alors être approximé par son interpolation éléments finis $(v_h, q_h) \in X_h \times M_h$ définie par :

$$\forall x \in \Omega \qquad v_h(x) = \sum_{0 \leq k < N_\mathrm{dof}^\mathbf{u}} v_k \boldsymbol{\varphi}_k^\mathbf{u}(x) \qquad q_h(x) = \sum_{0 \leq k < N_\mathrm{dof}^p} q_k \varphi_k^p(x)$$

où les scalaires $v_k \in \mathbb{R}$ et $q_k \in \mathbb{R}$ sont appelés degrés de liberté. La finesse de l'approximation est déterminée par majoration d'une mesure des écarts $(v - v_h)$ et $(q - q_h)$.

▷ Les espaces $\mathcal{V}_h^\mathbf{u}$ et $\mathcal{V}_h^p$ qui nous intéressent ici peuvent être construits à partir de X_h et M_h en prenant en compte les valeurs imposées sur Γ_D.

Appelons $\mathbb{I}^\mathbf{u}$ l'ensemble des indices k tels que $\boldsymbol{\varphi}_k^\mathbf{u}|_{\Gamma_\mathrm{D}}$ ne soit pas identiquement nulle. Considérons $v \in \mathcal{V}^\mathbf{u}$ et son interpolation éléments finis $v_h \in X_h$ qui s'écrit :

$$v_h(x) = \sum_{0 \leq k < N_\mathrm{dof}^\mathbf{u}} v_k \boldsymbol{\varphi}_k^\mathbf{u}(x) \qquad \forall x \in \Omega$$

Afin d'annuler v_h sur Γ_{D}, nous imposons $v_k = 0$ pour tout $k \in \mathbb{I}^{\mathbf{u}}$, de sorte que :

$$v_h(x) = \sum_{\substack{0 \le k < N^{\mathbf{u}}_{\mathrm{dof}} \\ k \notin \mathbb{I}^{\mathbf{u}}}} v_k \boldsymbol{\varphi}^{\mathbf{u}}_k(x) \qquad \forall x \in \Omega$$

Finalement :

$$\begin{aligned} \mathcal{V}^{\mathbf{u}}_h &= \mathrm{span}\{\ \boldsymbol{\varphi}^{\mathbf{u}}_k \ \big|\ 0 \le k < N^{\mathbf{u}}_{\mathrm{dof}} \ \ k \notin \mathbb{I}^{\mathbf{u}}\ \} \\ \mathcal{V}^{p}_h &= \mathrm{span}\{\ \varphi^{p}_k \ \big|\ 0 \le k < N^{p}_{\mathrm{dof}}\ \} \end{aligned} \tag{7.24}$$

▷ Dans la pratique, le relèvement des conditions aux limites $\mathbf{u}_{\mathrm{D}}$ apparaît dans l'approximation variationnelle sous forme approchée $\mathbf{u}_{\mathrm{D}h}$ au travers d'un développement suivant la famille $\left\{\boldsymbol{\varphi}^{\mathbf{u}}_k \middle| k \in \mathbb{I}^{\mathbf{u}}\right\}$:

$$\mathbf{u}_{\mathrm{D}h} = \sum_{k \in \mathbb{I}^{\mathbf{u}}} u_{\mathrm{D}k} \boldsymbol{\varphi}^{\mathbf{u}}_k$$

les scalaires $u_{\mathrm{D}k} \in \mathbb{R}$ étant déterminés à partir de la donnée $\underline{\mathbf{u}}_0$.

VII.5.2 Formulation Matricielle du Problème Discret

▷ Le problème discret (8.6) peut être formulé sous forme matricielle dans un premier temps en choisissant pour $\mathbf{v}$ et q chacun des vecteurs des bases données par la relation (7.24), puis, dans un second temps, en développant $\mathbf{u}$ et p dans ces mêmes bases. Les coefficients intervenant dans ces développements seront les composantes des vecteurs inconnus du système algébrique. Cette démarche requiert une réindexation contiguë des fonctions de base de $\mathcal{V}^{\mathbf{u}}_h$:

$$k \in [0, N^{\mathbf{u}}_{\mathrm{dof}}[\backslash \mathbb{I}^{\mathbf{u}} \ \xrightarrow[\text{contiguë}]{\text{indexation}}\ I \in [0, N^{\mathbf{u}}_{\mathrm{unk}}[\qquad \text{avec} \quad N^{\mathbf{u}}_{\mathrm{unk}} = N^{\mathbf{u}}_{\mathrm{dof}} - \mathrm{card}\,\mathbb{I}^{\mathbf{u}} \tag{7.25}$$

Bien que cela ne soit pas strictement nécessaire, et par soucis de cohérence, nous adopterons le même type de notation pour les fonctions de base de $\mathcal{V}^{p}_h$. Ainsi, on a :

$$\begin{aligned} \mathcal{V}^{\mathbf{u}}_h &= \mathrm{span}\{\ \boldsymbol{\varphi}^{\mathbf{u}}_I \ \big|\ 0 \le I < N^{\mathbf{u}}_{\mathrm{unk}}\ \} \\ \mathcal{V}^{p}_h &= \mathrm{span}\{\ \varphi^{p}_I \ \big|\ 0 \le I < N^{p}_{\mathrm{unk}}\ \} \end{aligned} \tag{7.26}$$

▷ Etant donnés le problème discret (8.6) et les bases de $\mathcal{V}^{\mathbf{u}}_h$ et de $\mathcal{V}^{p}_h$ exhibées par les relations (7.26), l'algorithme de marche en temps conduit, au pas de temps n, à rechercher les vecteurs $\mathbf{U}^{n+1}, \mathbf{P}^{n+1}$ solutions de :

$$\text{Trouver } (\mathbf{U}, \mathbf{P}) \in \mathbb{R}^{N^{\mathbf{u}}_{\mathrm{unk}}} \times \mathbb{R}^{N^{p}_{\mathrm{unk}}} \quad \text{tel que} \quad \left\{ \begin{aligned} \frac{\beta_q}{\Delta t}\mathbf{MU} + \mathbf{AU} + \mathbf{B}^{\mathrm{T}}\mathbf{P} &= \mathbf{F} \\ \mathbf{BU} &= \mathbf{G} \end{aligned} \right. \tag{7.27}$$

avec :

$$\begin{aligned} \mathbf{M}_{IJ} &= m(\boldsymbol{\varphi}^{\mathbf{u}}_J, \boldsymbol{\varphi}^{\mathbf{u}}_I) && 0 \le I, J < N^{\mathbf{u}}_{\mathrm{unk}} \\ \mathbf{A}_{IJ} &= c(\underline{\mathbf{u}}^{\star,n+1}_h; \boldsymbol{\varphi}^{\mathbf{u}}_J, \boldsymbol{\varphi}^{\mathbf{u}}_I) + k(\boldsymbol{\varphi}^{\mathbf{u}}_J, \boldsymbol{\varphi}^{\mathbf{u}}_I) && 0 \le I, J < N^{\mathbf{u}}_{\mathrm{unk}} \\ \mathbf{B}_{IJ} &= b(\boldsymbol{\varphi}^{\mathbf{u}}_J, \varphi^{p}_I) && 0 \le I < N^{p}_{\mathrm{unk}} \quad 0 \le J < N^{\mathbf{u}}_{\mathrm{unk}} \\ \mathbf{F}_I &= f(\boldsymbol{\varphi}^{\mathbf{u}}_I) && 0 \le I < N^{\mathbf{u}}_{\mathrm{unk}} \\ \mathbf{G}_I &= g(\varphi^{p}_I) && 0 \le I < N^{p}_{\mathrm{unk}} \end{aligned}$$

Sous réserve que l'on sache résoudre (8.26) à chaque pas de temps, on aura obtenu une approximation discrète de la solution du problème continu (7.3) sous la forme :

$$\begin{cases} \underline{\mathbf{u}}(n\Delta t, x) \approx \displaystyle\sum_{0 \leq I < N^{\mathbf{u}}_{\text{unk}}} \mathbf{U}^n_I \boldsymbol{\varphi}^{\mathbf{u}}_I(x) + \mathbf{u}_{\mathrm{D}h} \\ \underline{p}(n\Delta t, x) \approx \displaystyle\sum_{0 \leq I < N^{p}_{\text{unk}}} \mathbf{P}^n_I \varphi^{p}_I(x) \end{cases} \qquad \forall n \in [0, N] \quad \forall x \in \Omega \tag{7.28}$$

▷ Le système linéaire (8.26) peut être réécrit sous la forme :

$$\underbrace{\begin{pmatrix} \frac{\beta_q}{\Delta t}\mathbf{M} + \mathbf{A} & \mathbf{B}^{\mathrm{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix}}_{\mathcal{M}} \begin{pmatrix} \mathbf{U} \\ \mathbf{P} \end{pmatrix} = \begin{pmatrix} \mathbf{F} \\ \mathbf{G} \end{pmatrix} \tag{7.29}$$

La matrice $\mathcal{M}$ est creuse, potentiellement de grande taille et non symétrique. Par ailleurs, elle possède un noyau donné par :

$$\mathrm{N}()M = \left\{ \begin{pmatrix} \mathbf{0} \\ \mathbf{P} \end{pmatrix} \quad \text{avec} \quad \mathbf{P} \in \mathrm{N}()^{\mathrm{T}}\mathbf{B} \right\}$$

On peut alors montrer l'alternative suivante :

- si $\mathbf{G} \notin \mathrm{R}(\mathbf{B})$, le système (7.29) n'a pas de solution ;
- si $\mathbf{G} \in \mathrm{R}(\mathbf{B})$, le système (7.29) a une infinité de solutions qui sont toutes de la forme $\begin{pmatrix} \widehat{\mathbf{U}} \\ \widehat{\mathbf{P}} + \mathbf{P}_2 \end{pmatrix}$ où $\widehat{\mathbf{U}}$ et $\widehat{\mathbf{P}}$ sont communs à toutes les solutions et où $\mathbf{P}_2 \in \mathrm{N}()^{\mathrm{T}}\mathbf{B}$ varie d'une solution à l'autre.

Ces spécificités du système (8.26) en rendent l'inversion numérique délicate. L'objet des paragraphes suivants est d'explorer différentes techniques ségrégées conçues pour accomplir cette tâche.

VII.6 Méthode du Lagrangien Augmenté

Le système (8.26) est résolu par un processus itératif :

$$\mathbf{U}^{n+1} = \lim_k \mathbf{U}^{(k+1)}$$
$$\mathbf{P}^{n+1} = \lim_k \mathbf{P}^{(k+1)}$$

avec la formule de récurrence suivante :

$$\begin{aligned} &\frac{\beta_q}{\Delta t}\mathbf{M}\mathbf{U}^{(k+1)} + \mathbf{A}\mathbf{U}^{(k+1)} + \mathbf{B}^{\mathrm{T}}\mathbf{P}^{(k)} + r\mathbf{B}^{\mathrm{T}}\mathbf{M}_{\mathbf{p}}^{-1}\big(\mathbf{B}\mathbf{U}^{(k+1)} - \mathbf{G}\big) = \mathbf{F} \\ &\mathbf{P}^{(k+1)} = \mathbf{P}^{(k)} + \rho\mathbf{M}_{\mathbf{p}}^{-1}\big(\mathbf{B}\mathbf{U}^{(k+1)} - \mathbf{G}\big) \end{aligned} \tag{7.30}$$

où r et ρ sont des réels positifs appelés respectivement paramètre d'augmentation et paramètre de descente. La matrice $\mathbf{M}_{\mathbf{p}}$ est la matrice masse de pression, de coefficients :

$$\mathbf{M}_{\mathbf{p}IJ} = \int_\Omega \varphi^p_I \varphi^p_J \qquad 0 \leq I, J < N^p_{\text{unk}}$$

Afin d'éviter une inversion coûteuse, la matrice masse de pression est en pratique remplacée par sa version condensée $\mathbf{M}_{\mathbf{p}\ell}$ définie par :

$$\mathbf{M}_{\mathbf{p}\ell IJ} = \Big(\sum_{0 \leq K < N^p_{\text{unk}}} \mathbf{M}_{\mathbf{p}IK} \Big)\, \delta_{IJ} \quad \text{(symbole de Kronecker)} \qquad 0 \leq I, J < N^p_{\text{unk}} \tag{7.31}$$

L'itération (8.27) pourrait être réinterprétée comme une technique de recherche de point selle d'un Lagrangien si $\mathbf{A}$ était symétrique (d'où le nom de la méthode). Un critère suffisant de convergence est $0 < \rho < 2r$. Dans la pratique, on utilise $\rho = r$.

VII.7 Méthode de Projection Augmentée

▷ Introduisons les inconnues auxiliaires $\widetilde{\mathbf{U}}$, $\mathbf{\Phi}$.

$(\mathbf{U}, \mathbf{P})$ est solution de (8.26) si et seulement si :

$$\begin{aligned}
&\frac{\beta_q}{\Delta t}\mathbf{M}\widetilde{\mathbf{U}} + \mathbf{A}\mathbf{U} + \mathbf{B}^{\mathrm{T}}\mathbf{P}^n + r\mathbf{B}^{\mathrm{T}}\mathbf{M}_{\mathbf{p}}^{-1}\big(\mathbf{B}\widetilde{\mathbf{U}} - \mathbf{G}\big) = \mathbf{F} \\
&\mathbf{M}(\mathbf{U} - \widetilde{\mathbf{U}}) + \mathbf{B}^{\mathrm{T}}\mathbf{\Phi} = 0 \\
&\mathbf{B}\mathbf{U} = \mathbf{G} \\
&\mathbf{P} = \mathbf{P}^n + \frac{\beta_q}{\Delta t}\mathbf{\Phi} + r\mathbf{M}_{\mathbf{p}}^{-1}\big(\mathbf{B}\widetilde{\mathbf{U}} - \mathbf{G}\big)
\end{aligned}$$

ou, de façon équivalente :

$$\begin{aligned}
&\frac{\beta_q}{\Delta t}\mathbf{M}\widetilde{\mathbf{U}} + \mathbf{A}\mathbf{U} + \mathbf{B}^{\mathrm{T}}\mathbf{P}^n + r\mathbf{B}^{\mathrm{T}}\mathbf{M}_{\mathbf{p}}^{-1}\big(\mathbf{B}\widetilde{\mathbf{U}} - \mathbf{G}\big) = \mathbf{F} \\
&\big(\mathbf{B}\mathbf{M}^{-1}\mathbf{B}^{\mathrm{T}}\big)\mathbf{\Phi} = \mathbf{B}\widetilde{\mathbf{U}} - \mathbf{G} \\
&\mathbf{P} = \mathbf{P}^n + \frac{\beta_q}{\Delta t}\mathbf{\Phi} + r\mathbf{M}_{\mathbf{p}}^{-1}\big(\mathbf{B}\widetilde{\mathbf{U}} - \mathbf{G}\big) \\
&\mathbf{M}\mathbf{U} = \mathbf{M}\widetilde{\mathbf{U}} - \mathbf{B}^{\mathrm{T}}\mathbf{\Phi}
\end{aligned}$$

▷ A ce stade, certaines modifications sont apportées au système précédent afin d'obtenir aisément une estimation de sa solution :

- $\mathbf{A}\mathbf{U}$ est approximé par $\mathbf{A}\widetilde{\mathbf{U}}$
- Dans le cas où $\mathcal{V}_h^p \subset \mathrm{H}^1(\Omega)$, $\mathbf{B}\mathbf{M}^{-1}\mathbf{B}^{\mathrm{T}}$ est approximée par $\mathbf{L}$, définie comme suit (§VII.10) :

  $$\mathbf{L}_{IJ} = \int_\Omega \frac{1}{\rho}\nabla\varphi_I^p \cdot \nabla\varphi_J^p + \int_{\Gamma_{\mathrm{N}}} \alpha\,\varphi_I^p\varphi_J^p \qquad 0 \leq I, J < N_{\mathrm{unk}}^p$$

  où α est un coefficient de pénalisation ($\alpha \gg 1$).
- La matrice masse de pression $\mathbf{M}_{\mathbf{p}}$ est approximée par sa version condensée (7.31).

Ainsi, la méthode de projection augmentée détermine $(\mathbf{U}^{n+1}, \mathbf{P}^{n+1})$ à partir de $(\mathbf{U}^n, \mathbf{P}^n)$ en résolvant en séquence les quatres équations algébriques découplées suivantes :

$$\begin{aligned}
&\Big[\frac{\beta_q}{\Delta t}\mathbf{M} + \mathbf{A} + r\mathbf{B}^{\mathrm{T}}\mathbf{M}_{\mathbf{p}\ell}^{-1}\mathbf{B}\Big]\widetilde{\mathbf{U}} + \mathbf{B}^{\mathrm{T}}\mathbf{P}^n = \mathbf{F} + r\mathbf{B}^{\mathrm{T}}\mathbf{M}_{\mathbf{p}\ell}^{-1}\mathbf{G} \\
&\mathbf{L}\mathbf{\Phi} = \mathbf{B}\widetilde{\mathbf{U}} - \mathbf{G} \\
&\mathbf{P}^{n+1} = \mathbf{P}^n + \frac{\beta_q}{\Delta t}\mathbf{\Phi} + r\mathbf{M}_{\mathbf{p}\ell}^{-1}\big(\mathbf{B}\widetilde{\mathbf{U}} - \mathbf{G}\big) \\
&\mathbf{M}\mathbf{U}^{n+1} = \mathbf{M}\widetilde{\mathbf{U}} - \mathbf{B}^{\mathrm{T}}\mathbf{\Phi}
\end{aligned}$$

Intuitivement, cette méthode consiste à introduire une prédiction $\widetilde{\mathbf{U}}$ de $\mathbf{U}^{n+1}$ que l'on calcule dans une première étape en utilisant la pression du pas de temps précédent, puis que l'on corrige pour satisfaire la contrainte $\mathbf{B}\mathbf{U}^{n+1} = \mathbf{G}$. Le terme d'augmentation de l'étape de prédiction, semblable à celui présent dans la méthode du Lagrangien augmenté (8.27), permet d'assurer une prise en compte partielle de la contrainte d'incompressiblité dès l'étape de prédiction de vitesse.

On remarque qu'aucune itération interne n'est effectuée et que l'approximation $(\mathbf{U}^{n+1}, \mathbf{P}^{n+1})$ obtenue n'est pas solution du problème complètement couplé (8.26). Cette méthode est donc moins précise que la méthode du Lagrangien augmenté, car elle introduit une erreur supplémentaire dite de fractionnement. Elle est cependant moins coûteuse en temps CPU.

VII.8 Méthode de Projection de Yosida

▷ Introduisons les inconnues auxiliaires $\widetilde{\mathbf{U}}$, $\mathbf{\Phi}$.

$(\mathbf{U}, \mathbf{P})$ est solution de (8.26) si et seulement si :

$$
\begin{aligned}
&\Big[\frac{\beta_q}{\Delta t}\mathbf{M} + \mathbf{A}\Big]\mathbf{U} + \mathbf{B}^{\mathrm{T}}\mathbf{P} = \mathbf{F}\\
&\frac{\beta_q}{\Delta t}\mathbf{M}\widetilde{\mathbf{U}} + \mathbf{A}\mathbf{U} + \mathbf{B}^{\mathrm{T}}\mathbf{P}^n = \mathbf{F}\\
&\mathbf{M}(\mathbf{U} - \widetilde{\mathbf{U}}) + \mathbf{B}^{\mathrm{T}}\mathbf{\Phi} = 0\\
&\mathbf{B}\mathbf{U} = \mathbf{G}\\
&\mathbf{P} = \mathbf{P}^n + \frac{\beta_q}{\Delta t}\mathbf{\Phi}
\end{aligned}
$$

ou, en éliminant $\mathbf{U}$ entre la troisième et la quatrième équation :

$$
\begin{aligned}
&\frac{\beta_q}{\Delta t}\mathbf{M}\widetilde{\mathbf{U}} + \mathbf{A}\mathbf{U} + \mathbf{B}^{\mathrm{T}}\mathbf{P}^n = \mathbf{F}\\
&\big(\mathbf{B}\mathbf{M}^{-1}\mathbf{B}^{\mathrm{T}}\big)\mathbf{\Phi} = \mathbf{B}\widetilde{\mathbf{U}} - \mathbf{G}\\
&\mathbf{P} = \mathbf{P}^n + \frac{\beta_q}{\Delta t}\mathbf{\Phi}\\
&\Big[\frac{\beta_q}{\Delta t}\mathbf{M} + \mathbf{A}\Big]\mathbf{U} + \mathbf{B}^{\mathrm{T}}\mathbf{P} = \mathbf{F}
\end{aligned}
$$

▷ A ce stade, certaines modifications sont apportées au système précédent afin d'obtenir aisément une estimation de sa solution :

- $\mathbf{A}\mathbf{U}$ est approximé par $\mathbf{A}\widetilde{\mathbf{U}}$
- Dans le cas où $\mathcal{V}_h^p \subset \mathrm{H}^1(\Omega)$, $\mathbf{B}\mathbf{M}^{-1}\mathbf{B}^{\mathrm{T}}$ est approximée par $\mathbf{L}$, définie comme suit (§VII.10) :

$$
\mathbf{L}_{IJ} = \int_\Omega \frac{1}{\rho}\nabla\varphi_I^p \cdot \nabla\varphi_J^p + \int_{\Gamma_\mathrm{N}} \alpha\, \varphi_I^p \varphi_J^p \qquad 0 \leq I, J < N_{\mathrm{unk}}^p
$$

où α est un coefficient de pénalisation ($\alpha \gg 1$).

Ainsi, la méthode de projection de Yosida détermine $(\mathbf{U}^{n+1}, \mathbf{P}^{n+1})$ à partir de $(\mathbf{U}^n, \mathbf{P}^n)$ en résolvant en séquence les quatres équations algébriques découplées suivantes :

$$
\begin{aligned}
&\Big[\frac{\beta_q}{\Delta t}\mathbf{M} + \mathbf{A}\Big]\widetilde{\mathbf{U}} + \mathbf{B}^{\mathrm{T}}\mathbf{P}^n = \mathbf{F}\\
&\mathbf{L}\mathbf{\Phi} = \mathbf{B}\widetilde{\mathbf{U}} - \mathbf{G}\\
&\mathbf{P}^{n+1} = \mathbf{P}^n + \frac{\beta_q}{\Delta t}\mathbf{\Phi}\\
&\Big[\frac{\beta_q}{\Delta t}\mathbf{M} + \mathbf{A}\Big]\mathbf{U}^{n+1} + \mathbf{B}^{\mathrm{T}}\mathbf{P}^{n+1} = \mathbf{F}
\end{aligned}
$$

On remarque qu'aucune itération interne n'est effectuée et que l'approximation $(\mathbf{U}^{n+1}, \mathbf{P}^{n+1})$ obtenue n'est pas solution du problème complètement couplé (8.26). Cette méthode est donc moins précise que la méthode du Lagrangien augmenté, car elle introduit une erreur supplémentaire dite de fractionnement. Elle est cependant moins coûteuse en temps CPU.

VII.9 Décomposition de Helmholtz

Propriété

- Soit Ω un ouvert connexe de $\mathbb{R}^d$ ($d = 2$ ou 3) de frontière Γ suffisamment régulière.
- Supposons que Γ admette le partitionnement :

$$\Gamma = \Gamma_{\mathrm{D}} \cup \Gamma_{\mathrm{N}} \qquad \text{avec} \quad \Gamma_{\mathrm{D}} \neq \emptyset$$

et considérons les espaces fonctionnels :

$$\begin{aligned} \mathrm{H} &= \left\{ v \in \mathrm{L}^2(\Omega)^d \;\middle|\; \nabla \cdot v = 0 \;\; \text{dans} \;\; \Omega \quad \text{et} \quad v \cdot \mathbf{n} = 0 \;\; \text{sur} \; \Gamma_{\mathrm{D}} \right\} \\ \mathrm{H}^1_{\Gamma_{\mathrm{N}}}(\Omega) &= \left\{ q \in \mathrm{H}^1(\Omega) \;\middle|\; q = 0 \;\; \text{sur} \;\; \Gamma_{\mathrm{N}} \right\} \end{aligned}$$

- Si $\Gamma_{\mathrm{N}} \neq \emptyset$, on a la décomposition orthogonale suivante :

$$\mathrm{L}^2(\Omega)^d = \mathrm{H} \oplus \nabla\big(\mathrm{H}^1_{\Gamma_{\mathrm{N}}}(\Omega)\big)$$

- Si $\Gamma_{\mathrm{N}} = \emptyset$, on a la décomposition orthogonale suivante :

$$\mathrm{L}^2(\Omega)^d = \mathrm{H} \oplus \nabla\big(\mathrm{H}^1_{\Gamma_{\mathrm{N}}}(\Omega)/\mathbb{R}\big)$$

Ainsi, tout élément $v \in \mathrm{L}^2(\Omega)$ peut être décomposé de façon unique sous la forme :

$$v = w + \nabla q \qquad \text{avec} \qquad \begin{cases} w \in \mathrm{L}^2(\Omega)^d & \nabla \cdot w = 0 \text{ dans } \Omega \;;\; w \cdot \mathbf{n} = 0 \text{ sur } \Gamma_{\mathrm{D}} \\ q \in \mathrm{H}^1(\Omega) & q = 0 \text{ sur } \Gamma_{\mathrm{N}} \end{cases}$$

Il s'agit là d'une application du théorème de la divergence. Sur la frontière de Ω, au moins l'une des deux traces de $w \cdot \mathbf{n}$ ou de q doit être nulle (condition nécessaire pour assurer l'orthogonalité L^2 de w et ∇q).

VII.10 Approximate Algebraic Darcy Problem in Projection Methods

L'étape de projection :

$$\begin{aligned} &\mathbf{M}(\mathbf{U} - \widetilde{\mathbf{U}}) + \mathbf{B}^{\mathrm{T}}\boldsymbol{\Phi} = 0 \\ &\mathbf{B}\mathbf{U} = \mathbf{G} \end{aligned}$$

présente dans les méthodes de projection discutées ici (§VII.7,§VII.8) est l'analogue algébrique du problème de Darcy (7.15). L'élimination de $\mathbf{U}$ entre ces deux équations conduit à :

$$\big(\mathbf{B}\mathbf{M}^{-1}\mathbf{B}^{\mathrm{T}}\big)\boldsymbol{\Phi} = \mathbf{B}\widetilde{\mathbf{U}} - \mathbf{G} \tag{7.32}$$

Dans le cas général, la matrice masse de vitesse $\mathbf{M}$ n'est pas diagonale, ce qui rend la manipulation directe de $\mathbf{B}\mathbf{M}^{-1}\mathbf{B}^{\mathrm{T}}$ délicate. Cependant, l'établissement du problème de Poisson (7.20) à partir du problème de Darcy (7.15) suggère le remplacement de l'équation (7.32) par :

$$\mathbf{L}\boldsymbol{\Phi} = \mathbf{B}\widetilde{\mathbf{U}} - \mathbf{G} \tag{7.33}$$

où $\mathbf{L}$ est l'opérateur discret associé au problème de Poisson (7.20) dont l'inconnue ϕ satisfait des conditions aux limites de Neumann homogènes sur Γ_{D} et des conditions aux limites de Dirichlet sur Γ_{N}. Ces dernières induisent des difficultés pratiques dans la mise en œuvre du processus d'assemblage : par exemple, l'élimination dans le système linéaire des degrés de liberté de ϕ associés à des nœuds de pression situés sur Γ_{N} conduirait à des vecteurs de tailles différentes pour $\boldsymbol{\Phi}$ et $\mathbf{P}$. Pour cette raison, la condition aux limites de Dirichlet :

$$\phi = 0 \quad \text{sur} \quad \Gamma_{\mathrm{N}}$$

est imposée par pénalisation *via* une condition aux limites de Robin :

$$\nabla\phi \cdot \mathbf{n} = \alpha\,\phi \quad \text{sur} \quad \Gamma_\mathrm{N}$$

où α est un coefficient de pénalisation ($\alpha \gg 1$).

La matrice $\mathbf{L}$ est alors définie par :

$$\mathbf{L}_{IJ} = \int_\Omega \frac{1}{\rho}\nabla\varphi_I^p \cdot \nabla\varphi_J^p + \int_{\Gamma_\mathrm{N}} \alpha\,\varphi_I^p\varphi_J^p \qquad 0 \leq I, J < N_\mathrm{unk}^p$$

Chapter VIII

Example of Application CahnHilliardNavierStokes

VIII.1 Cahn-Hilliard

VIII.1.1 Three-Phase Model

▷ Free energy:

$$\mathcal{F}^{\text{triph}}_{\mathbf{\Sigma},\varepsilon}(\mathbf{c}) = \int_\Omega \frac{12}{\varepsilon} F(\mathbf{c}) + \frac{3}{8}\varepsilon\Sigma_1\ |\nabla c_1|^2 + \frac{3}{8}\varepsilon\Sigma_2\ |\nabla c_2|^2 + \frac{3}{8}\varepsilon\Sigma_3\ |\nabla c_3|^2$$
$$\text{with} \quad \mathbf{c} = (c_1, c_2, c_3) \quad \text{et} \quad \mathbf{\Sigma} = (\Sigma_1, \Sigma_2, \Sigma_3)$$

▷ Admissible set for **c**:

$$\mathcal{S} = \left\{(c_1, c_2, c_3) \in \mathbb{R}^3 \;\middle|\; c_1 + c_2 + c_3 = 1\right\} \subset \mathbb{R}^3$$

▷ Three-phase Cahn-Hilliard system, satisfied by each order parameter c_i $(i = 1, 2, 3)$:

$$\begin{aligned} \frac{\partial c_i}{\partial t} &= \nabla \cdot \left[\frac{M_0}{\Sigma_i}\nabla\mu_i\right] \\ \mu_i &= f_i^F(\mathbf{c}) - \frac{3}{4}\varepsilon\Sigma_i\nabla^2 c_i \end{aligned}$$

with :

$$\begin{aligned} f_i^F &= \frac{4\Sigma_T}{\varepsilon}\Big[\frac{1}{\Sigma_j}(\partial_i F - \partial_j F) + \frac{1}{\Sigma_k}(\partial_i F - \partial_k F)\Big] \\ \frac{3}{\Sigma_T} &= \frac{1}{\Sigma_1} + \frac{1}{\Sigma_2} + \frac{1}{\Sigma_3} \qquad \text{(and thus: } \Sigma_1\Sigma_2 + \Sigma_1\Sigma_3 + \Sigma_2\Sigma_3 \neq 0\ ) \end{aligned}$$

VIII.1.2 Classical Part of the Energy

▷ Bulk energy:

$$\begin{aligned} F(\mathbf{c}) &= F_0(\mathbf{c}) + P(\mathbf{c}) \\ F_0(\mathbf{c}) &= \sigma_{12}c_1^2c_2^2 + \sigma_{13}c_1^2c_3^2 + \sigma_{23}c_2^2c_3^2 + c_1c_2c_3(\Sigma_1c_1 + \Sigma_2c_2 + \Sigma_3c_3) \qquad (8.1) \\ P(\mathbf{c}) &= 3\Lambda\, c_1^2c_2^2c_3^2 \end{aligned}$$

with:

$$\Sigma_1 = \sigma_{12} + \sigma_{13} - \sigma_{23} \qquad \sigma_{12} = \frac{\Sigma_1 + \Sigma_2}{2}$$
$$\Sigma_2 = \sigma_{23} + \sigma_{12} - \sigma_{13} \qquad \sigma_{23} = \frac{\Sigma_2 + \Sigma_3}{2}$$
$$\Sigma_3 = \sigma_{13} + \sigma_{23} - \sigma_{12} \qquad \sigma_{13} = \frac{\Sigma_1 + \Sigma_3}{2}$$

▷ Equivalent form for F_0 :

$$F_0(\mathbf{c}) = \sigma_{12} c_1^2 c_2^2 + \sigma_{13} c_1^2 c_3^2 + \sigma_{23} c_2^2 c_3^2 + c_1 c_2 c_3 (\Sigma_1 c_1 + \Sigma_2 c_2 + \Sigma_3 c_3) \qquad \forall \mathbf{c} \in \mathbb{R}^3 \tag{8.2}$$
$$= \frac{\Sigma_1}{2}\, c_1^2 (c_2 + c_3)^2 + \frac{\Sigma_2}{2}\, c_2^2 (c_1 + c_3)^2 + \frac{\Sigma_3}{2}\, c_3^2 (c_1 + c_2)^2 \qquad \forall \mathbf{c} \in \mathbb{R}^3 \tag{8.3}$$

and:

$$F_0(\mathbf{c}) = \frac{\Sigma_1}{2}\, c_1^2 (1 - c_1)^2 + \frac{\Sigma_2}{2}\, c_2^2 (1 - c_2)^2 + \frac{\Sigma_3}{2}\, c_3^2 (1 - c_2)^2 \qquad \forall \mathbf{c} \in \mathcal{S} \tag{8.4}$$

VIII.1.3 Time Discretisation and Variational Formulation

▷ Problem to be solved:

$$\text{Find } (c_1, \mu_1, c_2, \mu_2) \text{ such that: } \begin{cases} \dfrac{c_i - c_i^\star}{\Delta t} + \mathbf{a} \cdot \nabla c_i = \nabla \cdot \left[\dfrac{M_0(c_1, c_2)}{\Sigma_i} \nabla \mu_i \right] & \text{in} \quad \Omega \\ \mu_i = \mathrm{D}_i^F(c_1, c_2) - \dfrac{3}{4} \varepsilon \Sigma_i \nabla^2 c_i & \text{in} \quad \Omega \\ c_i = c_{i0} & \text{on} \quad \Gamma_{\mathrm{D}}^c \\ \nabla c_i \cdot \mathbf{n} = 0 & \text{on} \quad \Gamma_{\mathrm{N}}^c \\ \nabla \mu_i \cdot \mathbf{n} = 0 & \text{on} \quad \Gamma \end{cases} \tag{8.5}$$

with:

$$\mathrm{D}_i^F(c_1, c_2) = D_i^F\big((c_1^\star, c_2^\star, 1 - c_1^\star - c_2^\star)\, ,\, (c_1, c_2, 1 - c_1 - c_2) \big)$$

where $D_i^F(\mathbf{c}^\star, \mathbf{c})$ represents a semi-implicit discretization of $f^F(\mathbf{c})$ such that:

$$D_i^F(\mathbf{c}^\star, \mathbf{c}) = f^F(\mathbf{c}) \qquad \forall \mathbf{c} \in \mathcal{S}$$

Note that D_i^F is a function of the two variables c_1 and c_2 which is parametrized by $c_1^\star, c_2^\star$.

▷ Function spaces $\mathcal{S}^{c_i}$, $\mathcal{V}^{c_i}$, $\mathcal{S}^{\mu_i}$, $\mathcal{V}^{\mu_i}$ defined as follows :

$$\mathcal{S}^{c_i} = \left\{ c_i \in \mathrm{H}^1(\Omega) \;\middle|\; c_i = c_{i0} \text{ sur } \Gamma_{\mathrm{D}}^c \right\}$$
$$\mathcal{V}^{c_i} = \left\{ c_i \in \mathrm{H}^1(\Omega) \;\middle|\; c_i = 0 \text{ sur } \Gamma_{\mathrm{D}}^c \right\}$$
$$\mathcal{S}^{\mu_i} = \mathcal{V}^{\mu_i} = \mathrm{H}^1(\Omega)$$

▷ A variational formulation of problem (8.5) reads:

Find $(c_1, \mu_1, c_2, \mu_2) \in \mathcal{S}^{c_1} \times \mathcal{S}^{\mu_1} \times \mathcal{S}^{c_2} \times \mathcal{S}^{\mu_2}$ such that
$\forall (v^{c_1}, v^{\mu_1}, v^{c_2}, v^{\mu_2}) \in \mathcal{V}^{c_1} \times \mathcal{V}^{\mu_1} \times \mathcal{V}^{c_2} \times \mathcal{V}^{\mu_2}$:

$$\begin{aligned} \int_\Omega \Big(\frac{c_i - c_i^\star}{\Delta t} + \mathbf{a} \cdot \nabla c_i \Big)\, v^{\mu_i} + \int_\Omega \frac{M_0(c_1, c_2)}{\Sigma_i}\, \nabla \mu_i \cdot \nabla v^{\mu_i} = 0 \\ \int_\Omega \mu_i\, v^{c_i} - \int_\Omega \mathrm{D}_i^F(c_1, c_2)\, v^{c_i} - \int_\Omega \frac{3}{4} \varepsilon \Sigma_i\, \nabla c_i \cdot \nabla v^{c_i} = 0 \end{aligned} \tag{8.6}$$

VIII.1.4 Discrete Problem

▷ Finite element spaces:

$$
\begin{aligned}
X_h^c &= \operatorname{span}\{\ \varphi_k^c \ \big|\ 0 \le k < N_{\mathrm{dof}}^c\ \}\\
c_{i\mathrm{D}}(\mathbf{x}) &= \sum_{k\in\mathbb{I}^c} c_{i\mathrm{D}k}\,\varphi_k^c(\mathbf{x}) \qquad \mathbb{I}^c = \{\ k \in [0, N_{\mathrm{dof}}^c[\ \big|\ \varphi_k^c|_{\Gamma_{\mathrm{D}}^c} \not\equiv 0\ \}\\
\mathcal{S}_h^{c_i} &= c_{i\mathrm{D}h} + \mathcal{V}_h^c\\
\mathcal{V}_h^{c_1} &= \mathcal{V}_h^{c_2} = \mathcal{V}_h^c\\
\mathcal{V}_h^c &= \operatorname{span}\{\ \varphi_k^c \ \big|\ 0 \le k < N_{\mathrm{dof}}^c \ \ k \notin \mathbb{I}^c\ \} \ = \ \operatorname{span}\{\ \varphi_I^c \ \big|\ 0 \le I < N_{\mathrm{unk}}^c\ \}\\
X_h^\mu &= \operatorname{span}\{\ \varphi_k^\mu \ \big|\ 0 \le k < N_{\mathrm{dof}}^\mu\ \}\\
\mathcal{S}_h^{\mu_1} &= \mathcal{S}_h^{\mu_2} = \mathcal{V}_h^{\mu_1} = \mathcal{V}_h^{\mu_2} = \mathcal{V}_h^\mu\\
\mathcal{V}_h^\mu &= \operatorname{span}\{\ \varphi_I^\mu \ \big|\ 0 \le I < N_{\mathrm{unk}}^\mu\ \}
\end{aligned}
$$

▷ Finite element expansions:

$$
\begin{cases}
c_i \in \mathcal{S}_h^{c_i} \quad \text{and} \quad c_i(\mathbf{x}) = c_{i\mathrm{D}}(\mathbf{x}) + \displaystyle\sum_{0\le J<N_{\mathrm{unk}}^{c_i}} \mathbf{C}_{iJ}\,\boldsymbol{\varphi}_J^c(\mathbf{x})\\
c_i^\star \in \mathcal{S}_h^{c_i\star} \quad \text{and} \quad c_i^\star(\mathbf{x}) = c_{i\mathrm{D}}^\star(\mathbf{x}) + \displaystyle\sum_{0\le J<N_{\mathrm{unk}}^{c_i\star}} \mathbf{C}_{i\,J}^\star\,\boldsymbol{\varphi}_J^{c\star}(\mathbf{x})\\
\mu_i \in \mathcal{S}_h^\mu \quad \text{and} \quad \mu_i(\mathbf{x}) = \displaystyle\sum_{0\le J<N_{\mathrm{unk}}^\mu} \mathbf{M}_{iJ}\,\varphi_J^\mu(\mathbf{x})
\end{cases}
\qquad \forall \mathbf{x} \in \Omega \tag{8.7}
$$

▷ Finite dimensional problem:

Find $[\mathbf{C}_1, \mathbf{M}_1, \mathbf{C}_2, \mathbf{M}_2] \in \mathbb{R}^{N_{\mathrm{unk}}^c} \times \mathbb{R}^{N_{\mathrm{unk}}^\mu} \times \mathbb{R}^{N_{\mathrm{unk}}^c} \times \mathbb{R}^{N_{\mathrm{unk}}^\mu}$ such that:

$$
\begin{cases}
\mathbf{R}^{c_1}(\mathbf{C}_1, \mathbf{M}_1, \mathbf{C}_2, \mathbf{M}_2) = \mathbf{0}\\
\mathbf{R}^{\mu_1}(\mathbf{C}_1, \mathbf{M}_1, \mathbf{C}_2, \mathbf{M}_2) = \mathbf{0}\\
\mathbf{R}^{c_2}(\mathbf{C}_1, \mathbf{M}_1, \mathbf{C}_2, \mathbf{M}_2) = \mathbf{0}\\
\mathbf{R}^{\mu_2}(\mathbf{C}_1, \mathbf{M}_1, \mathbf{C}_2, \mathbf{M}_2) = \mathbf{0}
\end{cases} \tag{8.8}
$$

with

$$
\begin{aligned}
\mathbf{R}_I^{c_i}(\mathbf{C}_1, \mathbf{M}_1, \mathbf{C}_2, \mathbf{M}_2) &= -\int_\Omega \frac{3}{4}\varepsilon\Sigma_i\ \nabla c_i \cdot \nabla\varphi_I^c - \int_\Omega \mathrm{D}_i^F(c_1, c_2)\,\varphi_I^c + \int_\Omega \mu_{ih}\,\varphi_I^c \qquad 0 \le I < N_{\mathrm{unk}}^c\\
\mathbf{R}_I^{\mu_i}(\mathbf{C}_1, \mathbf{M}_1, \mathbf{C}_2, \mathbf{M}_2) &= \int_\Omega \frac{M_0(c_1, c_2)}{\Sigma_i}\,\nabla\mu_i \cdot \nabla\varphi_I^\mu + \int_\Omega \big(\frac{c_i - c_i^\star}{\Delta t} + \mathbf{a}\cdot\nabla c_i\big)\,\varphi_I^\mu \qquad 0 \le I < N_{\mathrm{unk}}^\mu
\end{aligned} \tag{8.9}
$$

▷ Newton method:

Step 0 : initialization with $(\mathbf{C}_1^{(0)}, \mathbf{M}_1^{(0)}, \mathbf{C}_2^{(0)}, \mathbf{M}_2^{(0)})$

Step k : $(\mathbf{C}_1^{(k)}, \mathbf{M}_1^{(k)}, \mathbf{C}_2^{(k)}, \mathbf{M}_2^{(k)})$ given by step $k-1$.

compute $(\mathbf{C}_1^{(k+1)}, \mathbf{M}_1^{(k+1)}, \mathbf{C}_2^{(k+1)}, \mathbf{M}_2^{(k+1)})$ solution of:

$$\begin{pmatrix} \mathbf{J}^{c_1c_1} & \mathbf{J}^{c_1\mu_1} & \mathbf{J}^{c_1c_2} & 0 \\ \mathbf{J}^{\mu_1c_1} & \mathbf{J}^{\mu_1\mu_1} & 0 & 0 \\ \mathbf{J}^{c_2c_1} & 0 & \mathbf{J}^{c_2c_2} & \mathbf{J}^{c_2\mu_2} \\ 0 & 0 & \mathbf{J}^{\mu_2c_2} & \mathbf{J}^{\mu_2\mu_2} \end{pmatrix} \begin{pmatrix} \mathbf{C}_1^{(k+1)} - \mathbf{C}_1^{(k)} \\ \mathbf{M}_1^{(k+1)} - \mathbf{M}_1^{(k)} \\ \mathbf{C}_2^{(k+1)} - \mathbf{C}_2^{(k)} \\ \mathbf{M}_2^{(k+1)} - \mathbf{M}_2^{(k)} \end{pmatrix} = \begin{pmatrix} -\mathbf{R}^{c_1}(\mathbf{C}_1^{(k)}, \mathbf{M}_1^{(k)}, \mathbf{C}_2^{(k)}, \mathbf{M}_2^{(k)}) \\ -\mathbf{R}^{\mu_1}(\mathbf{C}_1^{(k)}, \mathbf{M}_1^{(k)}, \mathbf{C}_2^{(k)}, \mathbf{M}_2^{(k)}) \\ -\mathbf{R}^{c_2}(\mathbf{C}_1^{(k)}, \mathbf{M}_1^{(k)}, \mathbf{C}_2^{(k)}, \mathbf{M}_2^{(k)}) \\ -\mathbf{R}^{\mu_2}(\mathbf{C}_1^{(k)}, \mathbf{M}_1^{(k)}, \mathbf{C}_2^{(k)}, \mathbf{M}_2^{(k)}) \end{pmatrix} \tag{8.10}$$

with:

$$\begin{aligned}
[\mathbf{J}^{c_ic_i}]_{IJ} &= \frac{\partial \mathbf{R}_I^{c_i}}{\partial \mathbf{C}_{iJ}}(\mathbf{C}_1^{(k)}, \mathbf{M}_1^{(k)}, \mathbf{C}_2^{(k)}, \mathbf{M}_2^{(k)}) && 0 \le I < N_{\text{unk}}^c \quad 0 \le J < N_{\text{unk}}^c \\
&= -\int_\Omega \frac{3}{4}\varepsilon\Sigma_i \, \nabla\varphi_J^c \cdot \nabla\varphi_I^c - \int_\Omega \frac{\partial \mathrm{D}_i^F}{\partial c_i}(c_1^{(k)}, c_2^{(k)}) \, \varphi_J^c\varphi_I^c \\
[\mathbf{J}^{c_i\mu_i}]_{IJ} &= \frac{\partial \mathbf{R}_I^{c_i}}{\partial \mathbf{M}_{iJ}}(\mathbf{C}_1^{(k)}, \mathbf{M}_1^{(k)}, \mathbf{C}_2^{(k)}, \mathbf{M}_2^{(k)}) && 0 \le I < N_{\text{unk}}^c \quad 0 \le J < N_{\text{unk}}^\mu \\
&= \int_\Omega \varphi_J^\mu\varphi_I^c \\
[\mathbf{J}^{c_ic_j}]_{IJ} &= \frac{\partial \mathbf{R}_I^{c_i}}{\partial \mathbf{C}_{iJ}}(\mathbf{C}_1^{(k)}, \mathbf{M}_1^{(k)}, \mathbf{C}_2^{(k)}, \mathbf{M}_2^{(k)}) && 0 \le I < N_{\text{unk}}^c \quad 0 \le J < N_{\text{unk}}^c \\
&= -\int_\Omega \frac{\partial \mathrm{D}_i^F}{\partial c_j}(c_1^{(k)}, c_2^{(k)}) \, \varphi_J^c\varphi_I^c \\
[\mathbf{J}^{\mu_ic_i}]_{IJ} &= \frac{\partial \mathbf{R}_I^{\mu_i}}{\partial \mathbf{C}_{iJ}}(\mathbf{C}_1^{(k)}, \mathbf{M}_1^{(k)}, \mathbf{C}_2^{(k)}, \mathbf{M}_2^{(k)}) && 0 \le I < N_{\text{unk}}^\mu \quad 0 \le J < N_{\text{unk}}^c \\
&= \int_\Omega \frac{1}{\Delta t}\varphi_J^c\varphi_I^\mu + \int_\Omega \big(\mathbf{a}\cdot\nabla\varphi_J^c\big)\varphi_I^\mu \\
[\mathbf{J}^{\mu_i\mu_j}]_{IJ} &= \frac{\partial \mathbf{R}_I^{\mu_i}}{\partial \mathbf{M}_{iJ}}(\mathbf{C}_1^{(k)}, \mathbf{M}_1^{(k)}, \mathbf{C}_2^{(k)}, \mathbf{M}_2^{(k)}) && 0 \le I < N_{\text{unk}}^\mu \quad 0 \le J < N_{\text{unk}}^\mu \\
&\simeq \int_\Omega \frac{M_0(c_1^{(k)}, c_2^{(k)})}{\Sigma_i} \, \nabla\varphi_J^\mu \cdot \nabla\varphi_I^\mu
\end{aligned}$$

VIII.1.5 Bulk Energy Discretization

▷ Let F and $\tilde{F}$ two functions of $\mathbb{R}^3$.

$$\Big[\forall \mathbf{c} \in \mathcal{S} \quad F(\mathbf{c}) = \tilde{F}(\mathbf{c})\Big] \Rightarrow \Big[\forall \mathbf{c} \in \mathcal{S} \quad \partial_i F(\mathbf{c}) - \partial_j F(\mathbf{c}) = \partial_i \tilde{F}(\mathbf{c}) - \partial_j \tilde{F}(\mathbf{c})\Big]$$

Hence when computing $f^F(\mathbf{c})$ for $\mathbf{c} \in \mathcal{S}$, we may use instead of F any function that is identically equal to F on $\mathcal{S}$.

▷ Using the relation (8.4), we obtain:

$$\begin{aligned} f_1^{F_0}(c_1,c_2,c_3) &= \frac{12}{\varepsilon}\Big[\Sigma_1\, c_1(1-c_1)(1-2c_1) - 2\Sigma_T\, c_1c_2c_3\Big] && \text{as soon as} \quad c_3 = 1-c_1-c_2 \\ f_2^{F_0}(c_1,c_2,c_3) &= \frac{12}{\varepsilon}\Big[\Sigma_2\, c_2(1-c_2)(1-2c_2) - 2\Sigma_T\, c_1c_2c_3\Big] && \text{as soon as} \quad c_3 = 1-c_1-c_2 \end{aligned}$$

▷ The above relations may be used directly in the derivation of expressions for $\mathrm{D}_1^F(c_1,c_2)$ and $\mathrm{D}_2^F(c_1,c_2)$. Altenatively, we may proceed as follows :

1. Rewrite F into a form that is identical to the original one of $\mathcal{S}$.

2. For $i = 1,2,3$ define a semi-implicit discretization $d_i^F(\mathbf{c}^\star,\mathbf{c})$ of $\partial_i F$ such that:

$$d_i^F(\mathbf{c},\mathbf{c}) = \partial_i F(\mathbf{c}) \qquad \forall \mathbf{c} \in \mathcal{S} \tag{8.11}$$

4. For $i = 1,2,3$ define:

$$D_i^F(\mathbf{c}^\star,\mathbf{c}) = \frac{4\Sigma_T}{\varepsilon}\Big[\frac{1}{\Sigma_j}\big(d_i^F(\mathbf{c}^\star,\mathbf{c}) - d_j^F(\mathbf{c}^\star,\mathbf{c})\big) + \frac{1}{\Sigma_k}\big(d_i^F(\mathbf{c}^\star,\mathbf{c}) - d_k^F(\mathbf{c}^\star,\mathbf{c})\big)\Big]$$

3. Restrict to $\mathcal{S}$ and consider only two independent variables:

$$\begin{aligned} \mathrm{D}_i^F(c_1,c_2) &= \frac{4\Sigma_T}{\varepsilon}\Big[\frac{1}{\Sigma_j}\big(\mathrm{d}_i^F(c_1,c_2) - \mathrm{d}_j^F(c_1,c_2)\big) + \frac{1}{\Sigma_k}\big(\mathrm{d}_i^F(c_1,c_2) - \mathrm{d}_k^F(c_1,c_2)\big)\Big] && i = 1,2 \\ \mathrm{d}_i^F(c_1,c_2) &= d_i^F\big((c_1^\star,c_2^\star,1-c_1^\star-c_2^\star),(c_1,c_2,1-c_1-c_2),\big) && i = 1,2 \end{aligned} \tag{8.12}$$

▷ The computation of $\dfrac{\partial \mathrm{D}_i^F}{\partial c_i}$ may rely on the following relation:

$$\frac{\partial \mathrm{d}_i^F}{\partial c_j}(c_1,c_2) = \Big[\frac{\partial d_i^F}{\partial c_j} - \frac{\partial d_i^F}{\partial c_3}\Big]\big((c_1^\star,c_2^\star,1-c_1^\star-c_2^\star),(c_1,c_2,1-c_1-c_2)\big) \qquad i,j = 1,2 \tag{8.13}$$

VIII.1.6 Implementation

<u>**`MI_CH3BulkEnergy`**</u>

Each class derived from **`MI_CH3BulkEnergy`** is associated to a particular function $F\colon \mathbb{R}^3 \to \mathbb{R}$. This function is devoted to be part of the three phase bulk energy (8.1) (the questionable use of the same notation F should not be confusing).

Let **`en`** be of type **`MI_CH3BulkEnergy*`**, associated to $F\colon \mathbb{R}^3 \to \mathbb{R}$.

expression	evaluation result
`en->F(c1,c2,c3)`	$F(c_1, c_2, c_3)$ c_1=**c1** c_2=**c2** c_3=**c3**
`en->diF(c1,c2,c3,c1e,c2e,c3e,i)`	$d_i^F\big((c_1^\star, c_2^\star, 1-c_1^\star-c_2^\star), (c_1, c_2, 1-c_1-c_2)\big)$ c_1=**c1** c_2=**c2** c_3=**c3** $c_1^\star$=**c1e** $c_2^\star$=**c2e** $c_3^\star$=**c3e** i=**i**
`en->DDif(c1,c2,c1e,c2e,i,eps)`	$\mathrm{D}_i^F(c_1, c_2)$ c_1=**c1** c_2=**c2** $c_1^\star$=**c1e** $c_2^\star$=**c2e** i=**i** ε=**eps**
`en->dj_DDiF(c1,c2,c1e,c2e,i,j,eps)`	$\dfrac{\partial \mathrm{D}_i^F}{\partial c_j}(c_1, c_2)$ c_1=**c1** c_2=**c2** $c_1^\star$=**c1e** $c_2^\star$=**c2e** i=**i** j=**j** ε=**eps**
`en->dj_ddiF(c1,c2,c1e,c2e,i,j,eps)`	$\dfrac{\partial \mathrm{d}_i^F}{\partial c_j}(c_1, c_2)$ c_1=**c1** c_2=**c2** $c_1^\star$=**c1e** $c_2^\star$=**c2e** i=**i** j=**j** ε=**eps**

MI_BulkChemicalPotential

The concrete class **MI_BulkChemicalPotential** provides everything concerning the bulk energy (8.1) which is relevant for the solution of the discrete problem (§VIII.1.4).

An object of type **MI_BulkChemicalPotential** is created with a hierarchical data structure made of two parts:

- **MODULE MI_CH3BulkEnergy#F0** (mandatory) for the definition of F_0.
- **MODULE MI_CH3BulkEnergy#P** (optional) for the definition of P.

Let **bcp** be of type **MI_CH3BulkChemicalPotential***, associated to the bulk energy F (8.1).

expression	evaluation result
`bcp->F(c1,c2,c3)`	$F(c_1, c_2, c_3)$ c_1=**c1** c_2=**c2** c_3=**c3**
`bcp->DDif(c1,c2,c1e,c2e,i,eps)`	$\mathrm{D}_i^F(c_1, c_2)$ c_1=**c1** c_2=**c2** $c_1^\star$=**c1e** $c_2^\star$=**c2e** i=**i** ε=**eps**
`bcp->dj_DDiF(c1,c2,c1e,c2e,i,j,eps)`	$\dfrac{\partial \mathrm{D}_i^F}{\partial c_j}(c_1, c_2)$ c_1=**c1** c_2=**c2** $c_1^\star$=**c1e** $c_2^\star$=**c2e** i=**i** j=**j** ε=**eps**

VIII.2 Navier-Stokes

VIII.2.1 Incompressible Flow

▷ Domain $\Omega \subset \mathbb{R}^d$. The canonic basis of $\mathbb{R}^d$ is denoted $(\mathbf{e}_1, \ldots, \mathbf{e}_d)$.

▷ Consider the incompressible flow of a Newtonian fluid whose constitutive law reads:

$$\boldsymbol{\sigma}(\mathbf{u},p) = -p\,\mathbf{n} + \tau(\mathbf{u})\cdot\mathbf{n} \qquad \text{and} \qquad \tau(\mathbf{u}) = \eta(\nabla\mathbf{u} + \nabla\mathbf{u}^{\mathrm{T}})$$

where the velocity $\mathbf{u}$ et the pressure p satisfy the following boundary conditions:

$$\begin{array}{rl} \Gamma_{\mathrm{D}}: & \mathbf{u} = \mathbf{u}_0 \\ \Gamma_{\mathrm{D}n}: & \mathbf{u}\cdot\mathbf{n} = 0 \quad \text{et} \quad \boldsymbol{\sigma}(\mathbf{u},p)\cdot\mathbf{n}\cdot\mathbf{t} = 0 \\ \Gamma_{\mathrm{N}}: & -p\mathbf{n} + \tau(\mathbf{u})\cdot\mathbf{n} = 0 \end{array} \tag{8.14}$$

In the definition of $\Gamma_{\mathrm{D}n}$, it is assumed that $\mathbf{n}$ colinear to one of the vectors $\mathbf{e}_i$ of the canonical basis of $\mathbb{R}^d$.

▷ Mass and momentum balance, in Ω:

$$\begin{array}{l} \rho\dfrac{\partial\mathbf{u}}{\partial t} + (\rho\mathbf{u}\cdot\nabla)\mathbf{u} - \nabla\cdot\tau(\mathbf{u}) + \nabla p = \mathbf{b} \\ \nabla\cdot\mathbf{u} = 0 \end{array} \tag{8.15}$$

Other form:

$$\begin{array}{l} \sigma\dfrac{\partial}{\partial t}(\sigma\mathbf{u}) + (\rho\mathbf{u}\cdot\nabla)\mathbf{u} + \dfrac{1}{2}\mathbf{u}\nabla\cdot(\rho\mathbf{u}) - \nabla\cdot\tau(\mathbf{u}) + \nabla p = \mathbf{b} \\ \nabla\cdot\mathbf{u} = 0 \end{array} \tag{8.16}$$

VIII.2.2 Variational Formulation

▷ A variational formulation of problem (8.15) reads:

Find $(\mathbf{u},p)$ such that: $\forall(\mathbf{v},q)$

$$\left|\begin{array}{l} \displaystyle\int_\Omega \rho\frac{\partial\mathbf{u}}{\partial t}\cdot\mathbf{v} + \int_\Omega (\rho\mathbf{u}\cdot\nabla)\mathbf{u}\ \cdot\mathbf{v} + \int_\Omega \tau(\mathbf{u})\colon\nabla\mathbf{v} - \int_\Omega p\,\nabla\cdot\mathbf{v} = \int_\Omega \mathbf{b}\cdot\mathbf{v} \\ \displaystyle\int_\Omega q\,\nabla\cdot\mathbf{u} = 0 \end{array}\right. \tag{8.17}$$

▷ A variational formulation of problem (8.16) reads:

Find $(\mathbf{u},p)$ such that: $\forall(\mathbf{v},q)$

$$\left|\begin{array}{l} \displaystyle\int_\Omega \sigma\frac{\partial}{\partial t}(\sigma\mathbf{u})\cdot\mathbf{v} + \int_\Omega \frac{1}{2}(\rho\mathbf{u}\cdot\nabla)\mathbf{u}\ \cdot\mathbf{v} - \frac{1}{2}(\rho\mathbf{u}\cdot\nabla)\mathbf{v}\ \cdot\mathbf{u} + \int_{\Gamma_{\mathrm{N}}} \frac{1}{2}(\rho\mathbf{u}\cdot\mathbf{n})(\mathbf{u}\cdot\mathbf{v}) \\ \displaystyle\qquad\qquad\qquad + \int_\Omega \tau(\mathbf{u})\colon\nabla\mathbf{v} - \int_\Omega p\,\nabla\cdot\mathbf{v} = \int_\Omega \mathbf{b}\cdot\mathbf{v} \\ \displaystyle\int_\Omega q\,\nabla\cdot\mathbf{u} = 0 \end{array}\right. \tag{8.18}$$

VIII.2.3 Time Discretization

▷ Functional spaces $\mathcal{S}^{\mathbf{u}}$, $\mathcal{S}^p$, $\mathcal{V}^{\mathbf{u}}$ et $\mathcal{V}^p$ defined as follows:

$$\begin{array}{l} \mathcal{S}^{\mathbf{u}} = \left\{\ \mathbf{v}\in\mathrm{H}^1(\Omega)^d \ \middle|\ \mathbf{v} = \underline{\mathbf{u}}_0 \text{ on } \Gamma_{\mathrm{D}} \ \text{ and } \ \mathbf{v}\cdot\mathbf{n} = 0 \text{ on } \Gamma_{\mathrm{D}n}\ \right\} \\ \mathcal{V}^{\mathbf{u}} = \left\{\ \mathbf{v}\in\mathrm{H}^1(\Omega)^d \ \middle|\ \mathbf{v} = 0 \text{ on } \Gamma_{\mathrm{D}} \ \text{ and } \ \mathbf{v}\cdot\mathbf{n} = 0 \text{ on } \Gamma_{\mathrm{D}n}\ \right\} \\ \mathcal{S}^p = \mathcal{V}^p = \mathrm{L}^2(\Omega) \end{array}$$

▷ A variational formulation of problem (8.15) reads:

$$\text{Find } (\mathbf{u},p) \in \mathcal{S}^{\mathbf{u}} \times \mathcal{S}^{p} \text{ such that: } \forall (\mathbf{v},q) \in \mathcal{V}^{\mathbf{u}} \times \mathcal{V}^{p}$$
$$\left|\begin{aligned} &\int_\Omega \rho \frac{\mathbf{u}-\mathbf{u}^\star}{\Delta t}\cdot \mathbf{v} + \int_\Omega (\rho \mathbf{u}^\star \cdot \nabla)\mathbf{u} \ \cdot \mathbf{v} + \int_\Omega \tau(\mathbf{u})\colon \nabla \mathbf{v} - \int_\Omega p \, \nabla \cdot \mathbf{v} = \int_\Omega \mathbf{b}\cdot \mathbf{v} \\ &\int_\Omega q \, \nabla \cdot \mathbf{u} = 0 \end{aligned}\right. \tag{8.19}$$

▷ A variational formulation of problème (8.16) reads:

$$\text{Find } (\mathbf{u},p) \in \mathcal{S}^{\mathbf{u}} \times \mathcal{S}^{p} \text{ such that: } \forall (\mathbf{v},q) \in \mathcal{V}^{\mathbf{u}} \times \mathcal{V}^{p}$$
$$\left|\begin{aligned} &\int_\Omega \sigma \frac{\sigma\mathbf{u}-\sigma^\star\mathbf{u}^\star}{\Delta t}\cdot \mathbf{v} + \int_\Omega \frac{1}{2}(\rho \mathbf{u}^\star \cdot \nabla)\mathbf{u} \ \cdot \mathbf{v} - \frac{1}{2}(\rho \mathbf{u}^\star \cdot \nabla)\mathbf{v} \ \cdot \mathbf{u} + \int_{\Gamma_{\mathrm{N}}} \frac{1}{2}(\rho \mathbf{u}^\star \cdot \mathbf{n})(\mathbf{u}\cdot\mathbf{v}) \\ &\qquad\qquad + \int_\Omega \tau(\mathbf{u})\colon \nabla \mathbf{v} - \int_\Omega p \, \nabla \cdot \mathbf{v} = \int_\Omega \mathbf{b}\cdot \mathbf{v} \\ &\int_\Omega q \, \nabla \cdot \mathbf{u} = 0 \end{aligned}\right. \tag{8.20}$$

VIII.2.4 Space Discretization

▷ Finite dimensional functional spaces:

$$\begin{aligned} X_h^{\mathbf{u}} &= \operatorname{span}\{ \, \boldsymbol{\varphi}_k^{\mathbf{u}} \mid 0 \le k < N_{\text{dof}}^{\mathbf{u}} \, \} \\ \mathbf{u}_{\mathrm{D}}(\mathbf{x}) &= \sum_{k \in \mathbb{I}^{\mathbf{u}}} u_{\mathrm{D}k} \, \boldsymbol{\varphi}_k^{\mathbf{u}}(\mathbf{x}) \\ \mathcal{S}_h^{\mathbf{u}} &= \mathbf{u}_{\mathrm{D}h} + \mathcal{V}_h^{\mathbf{u}} \\ \mathcal{V}_h^{\mathbf{u}} &= \operatorname{span}\{ \, \boldsymbol{\varphi}_k^{\mathbf{u}} \mid 0 \le k < N_{\text{dof}}^{\mathbf{u}} \ \ k \notin \mathbb{I}^{\mathbf{u}} \, \} \ = \ \operatorname{span}\{ \, \boldsymbol{\varphi}_I^{\mathbf{u}} \mid 0 \le I < N_{\text{unk}}^{\mathbf{u}} \, \} \\ X_h^{p} &= \operatorname{span}\{ \, \varphi_k^{p} \mid 0 \le k < N_{\text{dof}}^{p} \, \} \\ \mathcal{S}_h^{p} &= \mathcal{V}_h^{p} = \operatorname{span}\{ \, \varphi_I^{p} \mid 0 \le I < N_{\text{unk}}^{p} \, \} \end{aligned}$$

▷ When the context will remove any ambiguity, the index h will be omitted in the notation of discrete unknowns.

▷ Variational approximation of problème (8.19):

$$\text{Given } \rho \text{ and } \mathbf{u}^\star \in \mathcal{S}_h^{\mathbf{u}\star}, \text{ find } (\mathbf{u},p) \in \mathcal{S}_h^{\mathbf{u}} \times \mathcal{S}_h^{p} \text{ such that: } \forall (\mathbf{v},q) \in \mathcal{V}_h^{\mathbf{u}} \times \mathcal{V}_h^{p}$$
$$\left|\begin{aligned} &\int_\Omega \rho \frac{\mathbf{u}-\mathbf{u}^\star}{\Delta t}\cdot \mathbf{v} + \int_\Omega (\rho \mathbf{u}^\star \cdot \nabla)\mathbf{u}_h \ \cdot \mathbf{v} + \int_\Omega \tau(\mathbf{u})\colon \nabla \mathbf{v} - \int_\Omega p \, \nabla \cdot \mathbf{v} = \int_\Omega \mathbf{b}\cdot \mathbf{v} \\ &\int_\Omega q \, \nabla \cdot \mathbf{u} = 0 \end{aligned}\right. \tag{8.21}$$

where $\mathcal{S}_h^{\mathbf{u}\star} \neq \mathcal{S}_h^{\mathbf{u}}$ (because of adaptive local refinement).

▷ Variational approximation of problem (8.20):

Given σ, $\sigma^\star$ and $\mathbf{u}^\star \in \mathcal{S}_h^{\mathbf{u}\star}$, find $(\mathbf{u}, p) \in \mathcal{S}_h^{\mathbf{u}} \times \mathcal{S}_h^p$ such that: $\forall(\mathbf{v}, q) \in \mathcal{V}_h^{\mathbf{u}} \times \mathcal{V}_h^p$

$$\left|\begin{aligned} &\int_\Omega \sigma \frac{\sigma\mathbf{u} - \sigma^\star\mathbf{u}^\star}{\Delta t} \cdot \mathbf{v} + \int_\Omega \frac{1}{2}(\rho\mathbf{u}^\star \cdot \nabla)\mathbf{u} \ \cdot \mathbf{v} - \frac{1}{2}(\rho\mathbf{u}^\star \cdot \nabla)\mathbf{v} \ \cdot \mathbf{u} + \int_{\Gamma_\mathrm{N}} \frac{1}{2}(\rho\mathbf{u}^\star \cdot \mathbf{n})(\mathbf{u} \cdot \mathbf{v}) \\ &\qquad\qquad + \int_\Omega \tau(\mathbf{u}) \colon \nabla\mathbf{v} - \int_\Omega p\, \nabla \cdot \mathbf{v} = \int_\Omega \mathbf{b} \cdot \mathbf{v} \\ &\int_\Omega q\, \nabla \cdot \mathbf{u} = 0 \end{aligned}\right. \tag{8.22}$$

VIII.2.5 Augmented Lagrangian

▷ Finite element expansions:

$$\begin{cases} \mathbf{u} \in \mathcal{S}_h^{\mathbf{u}} \quad \text{and} \quad \mathbf{u}(x) = \mathbf{u}_\mathrm{D} + \displaystyle\sum_{0 \le J < N_\mathrm{unk}^{\mathbf{u}}} \mathbf{U}_J\, \boldsymbol{\varphi}_J^{\mathbf{u}}(x) \\ \mathbf{u}^\star \in \mathcal{S}_h^{\mathbf{u}\star} \quad \text{and} \quad \mathbf{u}^\star(x) = \mathbf{u}_\mathrm{D}^\star + \displaystyle\sum_{0 \le J < N_\mathrm{unk}^{\mathbf{u}\star}} \mathbf{U}_J^\star\, \boldsymbol{\varphi}_J^{\mathbf{u}\star}(x) \qquad \forall x \in \Omega \\ p \in \mathcal{S}_h^p \quad \text{and} \quad p(x) = \displaystyle\sum_{0 \le J < N_\mathrm{unk}^p} \mathbf{P}_J\, \varphi_J^p(x) \end{cases} \tag{8.23}$$

▷ Let us define, for $u, v \in \mathrm{H}^1(\Omega)^d$ and $q \in \mathrm{L}^2(\Omega)$:

$$a(u, v) = \int_\Omega \frac{\rho}{\Delta t} u \cdot v + \int_\Omega (\rho\mathbf{u}_h^\star \cdot \nabla)u \cdot v + \int_\Omega \tau(u) \colon \nabla v \tag{8.24a}$$

$$a(u, v) = \int_\Omega \frac{\rho}{\Delta t} u \cdot v + \int_\Omega \frac{1}{2}(\rho\mathbf{u}_h^\star \cdot \nabla)u \cdot v - \frac{1}{2}(\rho\mathbf{u}_h^\star \cdot \nabla)v \cdot u + \int_\Omega \tau(u) \colon \nabla v \tag{8.24b}$$

$$+ \int_{\Gamma_\mathrm{N}} \frac{1}{2}(\rho\mathbf{u}_h^\star \cdot \mathbf{n})(u \cdot v) \tag{8.24c}$$

$$b(v, q) = \int_\Omega -q\nabla \cdot v \tag{8.24d}$$

$$f(v) = \int_\Omega \mathbf{b} \cdot v + \int_\Omega \frac{\rho}{\Delta t} \mathbf{u}_h^\star \cdot v \tag{8.24e}$$

▷ Matrix formulation of problem (8.21):

Find $(\mathbf{U}, \mathbf{P}) \in \mathbb{R}^{N_\mathrm{unk}^{\mathbf{u}}} \times \mathbb{R}^{N_\mathrm{unk}^p}$ such that: $\forall(I_1, I_2) \in [0, N_\mathrm{unk}^{\mathbf{u}}[\times[0, N_\mathrm{unk}^p[$

$$\left|\begin{aligned} &\sum_{0 \le J < N_\mathrm{unk}^{\mathbf{u}}} a(\boldsymbol{\varphi}_J^{\mathbf{u}}, \boldsymbol{\varphi}_{I_1}^{\mathbf{u}})\, \mathbf{U}_J + \sum_{0 \le J < N_\mathrm{unk}^p} b(\boldsymbol{\varphi}_{I_1}^{\mathbf{u}}, \varphi_J^p)\, \mathbf{P}_J = f(\boldsymbol{\varphi}_{I_1}^{\mathbf{u}}) - a(\mathbf{u}_\mathrm{D}, \boldsymbol{\varphi}_{I_1}^{\mathbf{u}}) \\ &\sum_{0 \le J < N_\mathrm{unk}^{\mathbf{u}}} b(\boldsymbol{\varphi}_J^{\mathbf{u}}, \varphi_{I_2}^p)\, \mathbf{U}_J = b(\mathbf{u}_\mathrm{D}, \varphi_{I_2}^p) \end{aligned}\right. \tag{8.25}$$

or:

$$\text{Find } (\mathbf{U}, \mathbf{P}) \in \mathbb{R}^{N_\mathrm{unk}^{\mathbf{u}}} \times \mathbb{R}^{N_\mathrm{unk}^p} \quad \text{such that} \quad \begin{cases} \mathbf{A}\mathbf{U} + \mathbf{B}^\mathrm{T}\mathbf{P} = \mathbf{F} \\ \mathbf{B}\mathbf{U} = \mathbf{G} \end{cases} \tag{8.26}$$

with:

$$
\begin{aligned}
\mathbf{A}_{IJ} &= a(\boldsymbol{\varphi}^{\mathbf{u}}_J, \boldsymbol{\varphi}^{\mathbf{u}}_I) && 0 \le I, J < N^{\mathbf{u}}_{\text{unk}} \\
\mathbf{B}_{IJ} &= b(\boldsymbol{\varphi}^{\mathbf{u}}_J, \varphi^{p}_I) && 0 \le I < N^{p}_{\text{unk}} \quad 0 \le J < N^{\mathbf{u}}_{\text{unk}} \\
\mathbf{F}_{I} &= f(\boldsymbol{\varphi}^{\mathbf{u}}_I) - a(\mathbf{u}_{\text{D}h}, \boldsymbol{\varphi}^{\mathbf{u}}_I) && 0 \le I < N^{\mathbf{u}}_{\text{unk}} \\
\mathbf{G}_{I} &= -b(\mathbf{u}_{\text{D}h}, \varphi^{p}_I) && 0 \le I < N^{p}_{\text{unk}}
\end{aligned}
$$

▷ The system (8.25) is solved by an iterative process:

$$
\begin{aligned}
\mathbf{U} &= \lim_k \mathbf{U}^{(k+1)} \\
\mathbf{P} &= \lim_k \mathbf{P}^{(k+1)}
\end{aligned}
$$

with the following recurrence formula:

$$
\begin{aligned}
&\mathbf{A}\mathbf{U}^{(k+1)} + \mathbf{B}^{\text{T}}\mathbf{P}^{(k)} + r\mathbf{B}^{\text{T}}\mathbf{M}_{\mathbf{p}\ell}^{-1}\big(\mathbf{B}\mathbf{U}^{(k+1)} - \mathbf{G}\big) = \mathbf{F} \\
&\mathbf{P}^{(k+1)} = \mathbf{P}^{(k)} + r\mathbf{M}_{\mathbf{p}\ell}^{-1}\big(\mathbf{B}\mathbf{U}^{(k+1)} - \mathbf{G}\big)
\end{aligned}
\tag{8.27}
$$

where r is a positive real called augmentation parameter. The matrix $\mathbf{M}_{\mathbf{p}\ell}$ is the lumped pressure mass matrix, of coefficients:

$$
\mathbf{M}_{\mathbf{p}\ell\,IJ} = \delta_{IJ} \int_\Omega \varphi^p_I \qquad 0 \le I, J < N^{p}_{\text{unk}}
$$

VIII.2.6 Projection Method

Step 1

▷ Projection of $p^\star \in \mathcal{S}^{p\star}_h$ on $\mathcal{S}^{p}_h$:

Find $\widetilde{p} \in \mathcal{S}^p_h$ such that $\forall q \in \mathcal{V}^p_h$:

$$
\left| \; \int_\Omega \frac{1}{\rho} \nabla\widetilde{p}\cdot\nabla q + \int_{\Gamma_{\text{N}}} \alpha\big(\frac{1}{\rho}\widetilde{p} - \frac{1}{\sigma\sigma^\star}p^\star\big) q = \int_\Omega \frac{1}{\sigma\sigma^\star} \nabla p^\star \cdot \nabla q \right.
\tag{8.28}
$$

▷ Finite element expansion :

$$
\widetilde{p} \in \mathcal{S}^p_h \qquad \text{and} \qquad \widetilde{p}(x) = \sum_{0 \le J < N^p_{\text{unk}}} \widetilde{\mathbf{P}}_J \, \varphi^p_J(x) \qquad \forall x \in \Omega
\tag{8.29}
$$

▷ Let us define, for $p, q \in \text{H}^1(\Omega)$:

$$
\ell(p,q) = \int_\Omega \frac{1}{\rho} \nabla p \cdot \nabla q + \int_{\Gamma_{\text{N}}} \frac{\alpha}{\rho}\, p\, q
\tag{8.30a}
$$

$$
k(q) = \int_\Omega \frac{1}{\sigma\sigma^\star} \nabla p^\star \cdot \nabla q + \int_{\Gamma_{\text{N}}} \frac{\alpha}{\sigma\sigma^\star}\, p^\star\, q
\tag{8.30b}
$$

▷ Matrix formulation of problem (8.28) :

$$
\text{Find } \widetilde{\mathbf{P}} \in \mathbb{R}^{N^p_{\text{unk}}} \text{ such that: } \ \forall I \in [0, N^{\mathbf{u}}_{\text{unk}}[\qquad \sum_{0 \le J < N^p_{\text{unk}}} \ell(\varphi^p_J, \varphi^p_I)\, \widetilde{\mathbf{P}}_J = k(\varphi^p_I)
$$

or:

$$\text{Find } \widetilde{\mathbf{P}} \in \mathbb{R}^{N^p_{\text{unk}}} \text{ such that:} \qquad \mathbf{L}\,\widetilde{\mathbf{P}} = \mathbf{K}$$

with:

$$\begin{aligned} \mathbf{L}_{IJ} &= \ell(\boldsymbol{\varphi}^p_J, \boldsymbol{\varphi}^p_I) && 0 \leq I, J < N^p_{\text{unk}} \\ \mathbf{K}_I &= k(\varphi^p_I) && 0 \leq I < N^p_{\text{unk}} \end{aligned}$$

Step 2

▷ Finite element expansion:

$$\widetilde{\mathbf{u}} \in \mathcal{S}^{\mathbf{u}}_h \qquad \text{and} \qquad \widetilde{\mathbf{u}}(x) = \sum_{0 \leq J < N^{\mathbf{u}}_{\text{unk}}} \widetilde{\mathbf{U}}_J\, \boldsymbol{\varphi}^{\mathbf{u}}_J(x) + \mathbf{u}_{\text{D}} \qquad \forall x \in \Omega \tag{8.31}$$

▷ Matrix formulation:

$$\text{Find } \widetilde{\mathbf{U}} \in \mathbb{R}^{N^u_{\text{unk}}} \text{ such that:} \qquad \mathbf{A}\widetilde{\mathbf{U}} + \mathbf{B}^{\text{T}}\widetilde{\mathbf{P}} = \mathbf{F}$$

Step 3

▷ Computation of the pressure increment :

$$\text{Find } \phi \in \mathcal{S}^p_h \text{ such that: } \forall q \in \mathcal{V}^p_h \quad \int_\Omega \frac{1}{\rho} \nabla\phi \cdot \nabla q + \int_{\Gamma_{\text{N}}} \frac{\alpha}{\rho} \phi\, q = -\frac{1}{\Delta t} \int_\Omega q\, \nabla \cdot \widetilde{\mathbf{u}} \tag{8.32}$$

▷ Finite element expansion:

$$\phi \in \mathcal{S}^p_h \qquad \text{and} \qquad \phi_h(x) = \sum_{0 \leq J < N^p_{\text{unk}}} \mathbf{\Phi}_J\, \varphi^p_J(x) \qquad \forall x \in \Omega \tag{8.33}$$

▷ Matrix formulation of problem (8.32) :

$$\text{Find } \mathbf{\Phi} \in \mathbb{R}^{N^p_{\text{unk}}} \text{ such that: } \forall I \in [0, N^p_{\text{unk}}[$$

$$\sum_{0 \leq J < N^p_{\text{unk}}} \ell(\varphi^p_J, \varphi^p_I)\, \mathbf{\Phi}_J = -\frac{1}{\Delta t} \sum_{0 \leq J < N^{\mathbf{u}}_{\text{unk}}} \left[\int_\Omega \varphi^p_I\, \nabla \cdot \boldsymbol{\varphi}^{\mathbf{u}}_J \right] \widetilde{\mathbf{U}}_J - \frac{1}{\Delta t} \int_\Omega \varphi^p_I\, \nabla \cdot \mathbf{u}_{\text{D}} \tag{8.34}$$

or

$$\text{Find } \mathbf{\Phi} \in \mathbb{R}^{N^p_{\text{unk}}} \text{ such that:} \qquad \mathbf{L}\,\mathbf{\Phi} = \frac{1}{\Delta t}(\mathbf{B}\widetilde{\mathbf{U}} - \mathbf{G})$$

Step 4

▷ Pressure correction :

$$p = \widetilde{p} + \phi \qquad \text{or} \qquad \mathbf{P} = \widetilde{\mathbf{P}} + \mathbf{\Phi}$$

Step 5

▷ End of step velocity :

$$\text{Find } \mathbf{u} \in \mathcal{S}_h^{\mathbf{u}} \text{ such that: } \forall \mathbf{v} \in \mathcal{V}_h^{\mathbf{u}} \quad \int_\Omega \rho \frac{\mathbf{u} - \widetilde{\mathbf{u}}}{\Delta t} \cdot \mathbf{v} = -\int_\Omega \nabla \phi \cdot \mathbf{v}$$

▷ When integrating by part the right hand side and using the nullity of ϕ on Γ_{N}, the above problem reduces to:

$$\text{Find } \mathbf{u} \in \mathcal{S}_h^{\mathbf{u}} \text{ such that: } \forall \mathbf{v} \in \mathcal{V}_h^{\mathbf{u}} \quad \int_\Omega \rho \frac{\mathbf{u} - \widetilde{\mathbf{u}}}{\Delta t} \cdot \mathbf{v} = \int_\Omega \phi \nabla \cdot \mathbf{v}$$

▷ Let us define, for $u, v \in \mathrm{H}^1(\Omega)^d$:

$$m(u, v) = \int_\Omega \frac{\rho}{\Delta t} u \cdot v \tag{8.35a}$$

▷ Matrix formulation:

$$\text{Find } \mathbf{U} \in \mathbb{R}^{N^{\mathbf{u}}_{\text{unk}}} \text{ such that: } \forall I \in [0, N^{\mathbf{u}}_{\text{unk}}[$$
$$\sum_{0 \le J < N^{\mathbf{u}}_{\text{unk}}} m(\boldsymbol{\varphi}^{\mathbf{u}}_J, \boldsymbol{\varphi}^{\mathbf{u}}_I)\,(\mathbf{U}_J - \widetilde{\mathbf{U}}_J) = \sum_{0 \le J < N^{p}_{\text{unk}}} b(\boldsymbol{\varphi}^{\mathbf{u}}_I, \varphi^{p}_J)\,\boldsymbol{\Phi}_J \tag{8.36}$$

or

$$\text{Find } \mathbf{U} \in \mathbb{R}^{N^{\mathbf{u}}_{\text{unk}}} \text{ such that: } \quad \mathbf{M_u U} = \mathbf{M_u} \widetilde{\mathbf{U}} - \mathbf{B}^{\mathrm{T}} \boldsymbol{\Phi}$$

with:

$$\mathbf{M_u}_{IJ} = m(\boldsymbol{\varphi}^{\mathbf{u}}_J, \boldsymbol{\varphi}^{\mathbf{u}}_I) \qquad 0 \le I, J < N^{\mathbf{u}}_{\text{unk}}$$

Penalization

Algebraic formulation of the projection method:

$$\mathbf{L}\,\widetilde{\mathbf{P}} = \mathbf{K} \qquad \to \widetilde{\mathbf{P}} \tag{8.37}$$
$$\mathbf{A}\widetilde{\mathbf{U}} = \mathbf{F} - \mathbf{B}^{\mathrm{T}}\widetilde{\mathbf{P}} \qquad \to \widetilde{\mathbf{U}} \tag{8.38}$$
$$\mathbf{L}\,\boldsymbol{\Phi} = \frac{1}{\Delta t}(\mathbf{B}\widetilde{\mathbf{U}} - \mathbf{G}) \qquad \to \boldsymbol{\Phi} \tag{8.39}$$
$$\mathbf{P} = \widetilde{\mathbf{P}} + \boldsymbol{\Phi} \qquad \to \mathbf{P} \tag{8.40}$$
$$\mathbf{M_u U} = \mathbf{M_u}\widetilde{\mathbf{U}} - \mathbf{B}^{\mathrm{T}}\boldsymbol{\Phi} \qquad \to \mathbf{U} \tag{8.41}$$
$$\tag{8.42}$$

In the penalized version, the second step is replaced by:

$$\mathbf{A}\widetilde{\mathbf{U}} + r\mathbf{B}^{\mathrm{T}}\mathbf{M}_{\mathbf{p}\ell}^{-1}\big(\mathbf{B}\widetilde{\mathbf{U}} - \mathbf{G}\big) = \mathbf{F} - \mathbf{B}^{\mathrm{T}}\widetilde{\mathbf{P}}$$

or equivalently:

$$(\mathbf{A} + r\mathbf{B}^{\mathrm{T}}\mathbf{M}_{\mathbf{p}\ell}^{-1}\mathbf{B})\widetilde{\mathbf{U}} = F + r\mathbf{B}^{\mathrm{T}}\mathbf{M}_{\mathbf{p}\ell}^{-1}\mathbf{G} - \mathbf{B}^{\mathrm{T}}\widetilde{\mathbf{P}}$$

This leads to a modified pressure correction:

$$\mathbf{P} = \widetilde{\mathbf{P}} + r\mathbf{M}_{\mathbf{p}\ell}^{-1}(\mathbf{B}\widetilde{\mathbf{U}} - \mathbf{G}) + \boldsymbol{\Phi}$$

VIII.3 More on Cahn-Hilliard

VIII.3.1 Two-Phase Model

▷ Free energy:

$$\mathcal{F}^{\text{diph}}_{\sigma,\varepsilon}(c) = \int_\Omega \frac{12}{\varepsilon}F(c) + \frac{3}{4}\varepsilon\sigma\,|\nabla c|^2 \quad \text{with} \quad F(c) = \sigma\, c^2(1-c)^2$$

▷ Two-phase Cahn-Hilliard system, verified by the order parameter c:

$$\frac{\partial c}{\partial t} = \nabla\cdot\Big[M^{\text{diph}}\nabla\mu\Big]$$

$$\mu = \frac{12}{\varepsilon}F'(c) - \frac{3}{2}\varepsilon\sigma\nabla^2 c \quad \text{with} \quad F'(c) = \sigma\big[2c(1-c)(1-2c)\big]$$

VIII.3.2 Two-Phase Limit of the Three-Phase Model

$$c_1 = c \qquad c_2 = 0 \qquad c_3 = 1-c \qquad \sigma_{12} = \sigma$$

$$\mathcal{F}^{\text{triph}}_{\boldsymbol{\Sigma},\varepsilon}(c,0,1-c) = \mathcal{F}^{\text{diph}}_{\sigma,\varepsilon}(c) \qquad (\text{whatever } \Sigma_3)$$

$$\mu_1(c,0,1-c) = \frac{12}{\varepsilon}\Sigma_1 c(1-c)(1-2c) - \frac{3}{4}\varepsilon\Sigma_1\nabla^2 c$$

Evolution equation satisfied by $c_1 = c$:

$$\frac{\partial c}{\partial t} = \nabla\cdot\Big[\frac{M_0}{\Sigma_1}\nabla\mu_1\Big]$$

$$= \nabla\cdot\Big[M_0\nabla\big(\frac{12}{\varepsilon}c(1-c)(1-2c) - \frac{3}{4}\varepsilon\nabla^2 c\big)\Big]$$

which is formally identical to the two-phase Cahn-Hilliard equation with:

$$M^{\text{diph}} = \frac{M_0}{2\sigma}$$

c_1	c_2	c_3	σ_{12}	σ_{13}	σ_{23}	$\frac{\mu_1}{\Sigma_1} = \frac{\mu_2}{\Sigma_2}$	$\frac{M_0}{2\sigma}$
c	0	$1-c$	σ	whatever	whatever	$\frac{\mu}{2\sigma}$	M^{diph}

VIII.4 Criterion for Adaptive Local Refinement

VIII.4.1 Refinement in the Interface Neighborhood

- Given *a priori*:
  - initial conforming triangulation

  - cell size $h_{\text{interface}}$ for the interface neighborhood

- Definition of the per-cell-indicator:

$$\eta_{\mathcal{C}} = \max\Big(\frac{1}{|\mathcal{C}|} \int_{\mathcal{C}} c_1^\star \ , \ \frac{1}{|\mathcal{C}|} \int_{\mathcal{C}} c_2^\star \ , \ \frac{1}{|\mathcal{C}|} \int_{\mathcal{C}} c_3^\star \Big)$$

$$\eta_{\mathcal{C}} = 1 \quad \longrightarrow \quad \mathcal{C} \ \text{ is completely filled with a bulk phase}$$

$$\eta_{\mathcal{C}} < 1 \quad \longrightarrow \quad \mathcal{C} \ \text{ contains an interface}$$

- Refinement criterion:

$$\eta_\varphi < \eta_\varphi^{\text{ref}} \quad \text{and} \quad \operatorname{diam}(\mathcal{C}) > h_{\text{interface}} \quad \text{for at least one cell } \ \mathcal{C} \subset \operatorname{supp}[\varphi]$$

- Unrefinement criterion:

$$\eta_\varphi > \eta_\varphi^{\text{unref}}$$

VIII.4.2 `MI_CHInterfaceIndicator`

$h_{\text{interface}}$	**`h_for_interface`**
$\eta_\varphi^{\text{ref}}$	**`refinement_limit`**
$\eta_\varphi^{\text{unref}}$	**`unrefinement_limit`**

Chapter IX

Reference Solutions of Some Systems of Partial Differential Equations

IX.1 Poisson Problem I

IX.1.1 Problem Statement

- Let $\Omega =]0,1[^2 \subset \mathbb{R}^2$. The boundary of Ω is denoted by Γ.
- Let $\Gamma_1 = \{ (x,y) \in \Gamma \mid x = 1 \}$.
- Consider the following problem:

$$\text{(BVP)} \qquad \text{find } u : \Omega \to \mathbb{R} \text{ such that } \begin{cases} \nabla^2 u = 0 & \text{in } \Omega \\ u = 0 & \text{on } \Gamma \setminus \Gamma_1 \\ u = \sin \pi y & \text{on } \Gamma_1 \end{cases}$$

IX.1.2 Solution

The exact solution of (BVP) is (fig. IX.1):

$$u\colon (x,y) \mapsto \frac{\sinh \pi x}{\sinh \pi} \sin \pi y$$

IX.2 Poisson Problem II

IX.2.1 Problem Statement

- Let $\Omega =]0,1[^3 \subset \mathbb{R}^3$. The boundary of Ω is denoted by Γ.
- Let $\Gamma_1 = \{ (x,y,z) \in \Gamma \mid x = 1 \}$.
- Consider the following problem:

$$\text{(BVP)} \qquad \text{find } u : \Omega \to \mathbb{R} \text{ such that } \begin{cases} \nabla^2 u = 0 & \text{in } \Omega \\ u = 0 & \text{on } \Gamma \setminus \Gamma_1 \\ u = \sin \pi y \sin \pi z & \text{on } \Gamma_1 \end{cases}$$

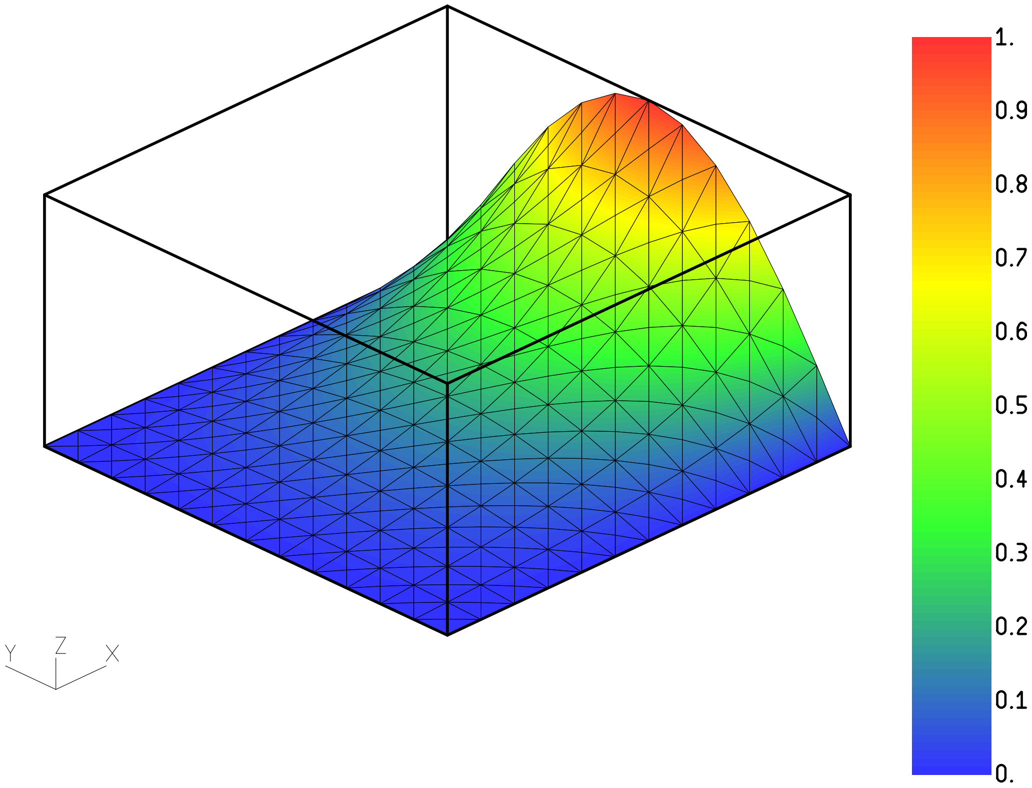


FIGURE IX.1: solution field.

IX.2.2 Solution

The exact solution of (BVP) is:

$$u\colon (x,y,z) \mapsto \frac{\sinh \sqrt{2}\pi x}{\sinh \sqrt{2}\pi} \sin \pi y \, \sin \pi z$$

IX.3 Poisson Problem with Measure Data

IX.3.1 Problem Statement

- Let $\Omega = \mathbb{R}^d$ with $d = 2$ or $d = 3$.
- Consider the following problem:

$$\text{(BVP)} \quad \text{find } u : \Omega \to \mathbb{R}^d \text{ such that } \quad -\nabla^2 u = \delta \text{ in } \Omega$$

where δ denotes the Dirac mass located at the origin.

IX.3.2 Solution

The exact solution of (BVP) is:

$$u \;:\; \begin{cases} (x,y) \mapsto \dfrac{-1}{2\pi} \ln \sqrt{x^2+y^2} & \text{for } d=2 \\ (x,y,z) \mapsto \dfrac{-1}{4\pi\sqrt{x^2+y^2+z^2}} & \text{for } d=3 \end{cases}$$

IX.4 Helmholtz Problem with Regular Solution I

IX.4.1 Problem Statement

- Let $\Omega =]0,1[^2 \subset \mathbb{R}^2$. The boundary of Ω is denoted by Γ.
- Let $\Gamma_1 = \{\, (x,y) \in \Gamma \;\big|\; x \in \{0,1\} \,\}$ and $\Gamma_2 = \{\, (x,y) \in \Gamma \;\big|\; y \in \{0,1\} \,\}$.
- Consider the following problem:

$$\text{(BVP)} \quad \text{find } u : \Omega \to \mathbb{R} \text{ such that } \begin{cases} u - \nabla^2 u = f & \text{in } \Omega \\ u = 0 & \text{on } \Gamma_1 \\ \dfrac{\partial u}{\partial \mathbf{n}} = 0 & \text{on } \Gamma_2 \end{cases}$$

with $\quad \forall (x,y) \in \Omega \quad f(x,y) = \; (2\pi^2 + 1) \sin \pi x \, \cos \pi y.$

IX.4.2 Solution

The exact solution of (BVP) is (fig. IX.2):

$$u\colon (x,y) \mapsto \sin \pi x \, \cos \pi y$$

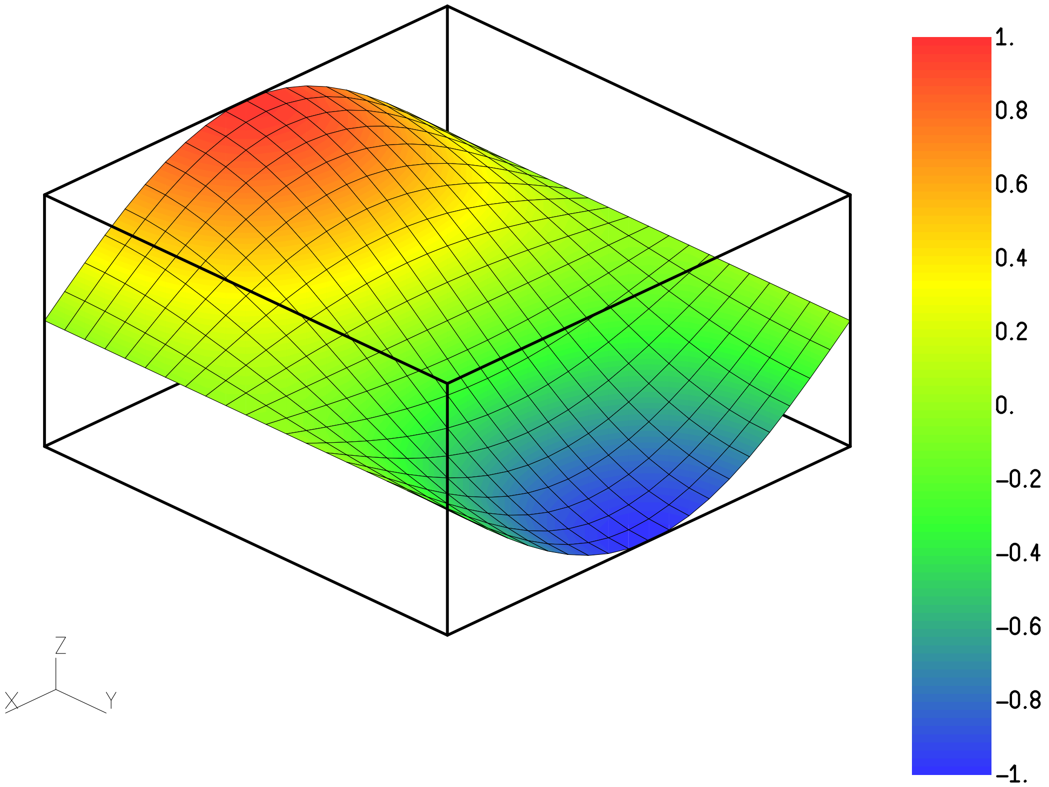


Figure IX.2: solution field.

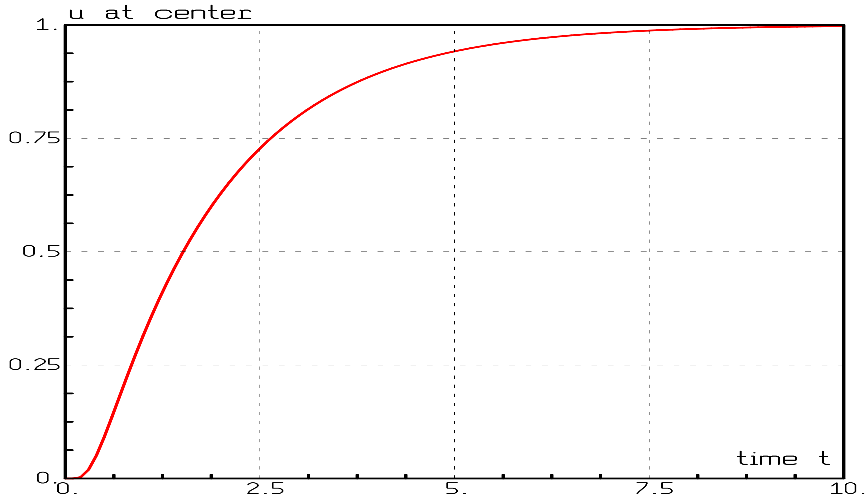


FIGURE IX.3: Solution u at $x = 0$ as a function of time for the case $L = 1$ and $a = 0.25$.

IX.5 Transient Diffusion I

IX.5.1 Problem Statement

- Let $\Omega =]-L, +L[\subset \mathbb{R}$ with $L \geq 0$.
- Let $[0, T]$ be a time interval.
- Consider the following problem:

$$\text{(IBVP)} \qquad \text{find } u : [0,T] \times \Omega \rightarrow \mathbb{R} \text{ such that } \begin{cases} \dfrac{\partial u}{\partial t} = a\nabla^2 u & \text{in } [0,T] \times \Omega \\ u = 1 & \text{on } [0,T] \times \{\pm L\} \\ u = 0 & \text{on } \{0\} \times \Omega \end{cases}$$

IX.5.2 Solution

The solution u of (IBVP) is given at the center of the domain by the expression:

$$u(t,0) = 2\sum_{n=0}^{+\infty} (-1)^n \text{erfc}\Big[\frac{2n+1}{2}\frac{L}{\sqrt{at}}\Big] \qquad \forall t \in [0,T] \tag{9.1}$$

where erfc denotes the complimentary error function. This solution is depicted in figure IX.3 for some values of the parameters.

IX.5.3 Related Expressions in PELICANS

The class `RS_TransientDiffusion1EXP` defines the expression `TransientDiffusion1` with the following characteristics.

- Argument :

Type	Value
`Double`	t

- Type : `DoubleVector`
- Value : $u(t,0)$ given by (9.1)

IX.6 Diffusion with Source Term I

IX.6.1 Problem Statement

- Let Ω be the disc of $\mathbb{R}^2$ whose center is $(0,0)$ and whose radius is $R > 0$. The boundary of Ω is denoted by Γ (unit outward normal : $\mathbf{n}$).
- Consider the following problem:

$$\text{(BVP)} \qquad \text{find } u : \Omega \to \mathbb{R} \text{ such that } \begin{cases} -\kappa\nabla^2 u = \varpi & \text{in } \Omega \\ -\kappa\nabla u \cdot \mathbf{n} = h(u - u_\infty) & \text{on } \Gamma \end{cases}$$

with $\kappa \in \mathbb{R}^+$, $\varpi \in \mathbb{R}$, $h \in \mathbb{R}^+$, $u_\infty \in \mathbb{R}^+$.

IX.6.2 Solution

The solution of (BVP) is:

$$\forall \mathbf{x} \in \Omega \qquad u(\mathbf{x}) = \frac{\varpi}{4\kappa}(R^2 - r^2) + \frac{\varpi R}{2h} + u_\infty \quad \text{with} \quad r = \|\mathbf{x}\|_2$$

IX.7 Linear Advection with Discontinuous Solution I

IX.7.1 Problem Statement

- Let $\Omega =]0,1[^2 \subset \mathbb{R}^2$. The boundary of Ω is denoted by Γ.
- Let $\Gamma_1 = \{\, (x,y) \in \Gamma \mid x = 0 \,\} \cup \{\, (x,y) \in \Gamma \mid y = 0 \text{ and } x \notin [1/3; 2/3] \,\}$
  $\Gamma_2 = \{\, (x,y) \in \Gamma \mid y = 0 \text{ and } x \in [1/3; 2/3] \,\}$
- Consider the following problem:

$$\text{(BVP)} \qquad \text{find } u : \Omega \to \mathbb{R} \text{ such that } \begin{cases} \dfrac{\partial u}{\partial t} + \mathbf{a} \cdot \nabla u = 0 & \text{in } \Omega \\ u = 0 & \text{on } \Gamma_1 \\ u = 1 & \text{on } \Gamma_2 \end{cases}$$

with $\quad \forall (x,y) \in \Omega \quad \mathbf{a} = (y, 1-x)^{\mathrm{T}}$.

IX.7.2 Solution

At steady state, *i.e.* for $t \geq \pi/2$, the exact solution is $u = 1$ in between the two circles in the upper part of figure IX.4 and zero elsewhere. For0 $\leq t \leq \pi/2$, the exact solution is depicted in the lower part of figure IX.4.

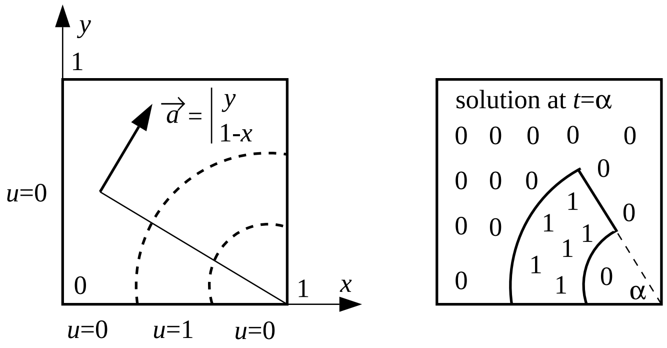


FIGURE IX.4: Linear advection around a point.

IX.8 Linear Advection with Discontinuous Solution II

IX.8.1 Problem Statement

- Let $\Omega =]-1,1[^2 \subset \mathbb{R}^2$. The boundary of Ω is denoted by Γ. Let $[0,T]$ be a time interval.
- Consider the following problem:

$$\text{(BVP)} \quad \text{find } u : \Omega \times [0,T] \to \mathbb{R} \text{ such that } \begin{cases} \dfrac{\partial u}{\partial t} + \mathbf{a} \cdot \nabla u = 0 & \text{in } \Omega \\ u = 0 & \text{on } \Gamma \end{cases}$$

with

- the advective velocity

$$\mathbf{a}(x,y,t) = (2y, -2x)^{\mathrm{T}} \quad \forall (x,y) \in \Omega \tag{9.2}$$

- the initial condition ($t = 0$):

$$\begin{cases} u(x,y,0) = 1 & \text{if } 0.1 < x < 0.6 \text{ and } -0.25 < y < 0.25 \\ u(x,y,0) = 1 - \dfrac{\sqrt{(x+0.45)^2 + y^2}}{0.35} & \text{if } \sqrt{(x+0.45)^2 + y^2} < 0.35 \\ u(x,y,0) = 0 & \text{otherwise} \end{cases}$$

IX.8.2 Solution

Denoting $x_r = x\cos(2t) - y\sin(2t)$ and $y_r = x\sin(2t) + y\cos(2t)$, the solution at any time $t \in [0,T]$ has the following form :

$$\begin{cases} u(x,y,t) = 1 & \text{if } 0.1 < x_r < 0.6 \text{ and } -0.25 < y_r < 0.25 \\ u(x,y,t) = 1 - \dfrac{\sqrt{(x_r+0.45)^2 + y_r^2}}{0.35} & \text{if } \sqrt{(x_r+0.45)^2 + y_r^2} < 0.35 \\ u(x,y,t) = 0 & \text{otherwise} \end{cases} \tag{9.3}$$

The solution at $t = k\pi$ where k is any integer is depicted in figure IX.5.

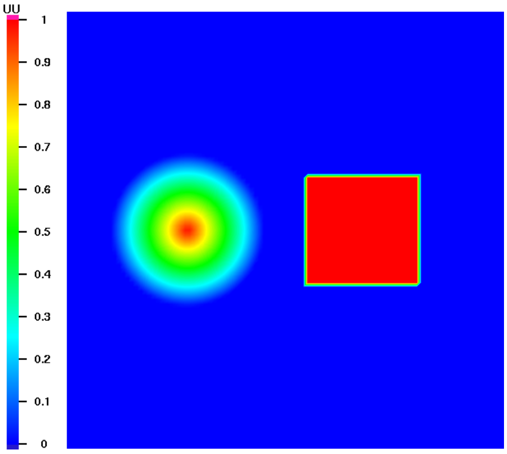


FIGURE IX.5: Pure transport of a square profile and a corned-shaped profile.

IX.8.3 Related Expressions in PELICANS

The class `RS_SolidBodyRotationEXP` defines two expressions, with the following characteristics.

	Type	Value
Argument 1	`DoubleVector`	(x, y)
Argument 2	`Double`	t

- `SolidBodyRotation_value`

Type	`Double`
Value	$u(x, y, t)$ given by 9.3

- `SolidBodyRotation_velocity`

Type	`DoubleVector`
Value	$\mathbf{a}(x, y, t)$ given by 9.2

IX.9 Advection Diffusion I

IX.9.1 Problem Statement

Consider the following problem:

$$\text{(BVP)} \qquad \text{find } u : \Omega \to \mathbb{R} \text{ such that } \begin{cases} a\dfrac{\partial u}{\partial x} - \kappa\dfrac{\partial^2 u}{\partial x^2} = 0 & \text{in }]0, 1[\\ u = u_{\text{in}} & \text{on } x = 0 \\ u = u_{\text{out}} & \text{on } x = 1 \end{cases}$$

where $a \in \mathbb{R}, a > 0$ is a constant advection speed and $\kappa \in \mathbb{R}, \kappa > 0$ is a constant diffusivity.

IX.9.2 Solution

The exact solution of (BVP) is:

$$u\colon x \mapsto u_{\text{in}} + (u_{\text{out}} - u_{\text{in}})\frac{\exp(\mathbf{Pe}\ x) - 1}{\exp(\mathbf{Pe}) - 1} \tag{9.4}$$

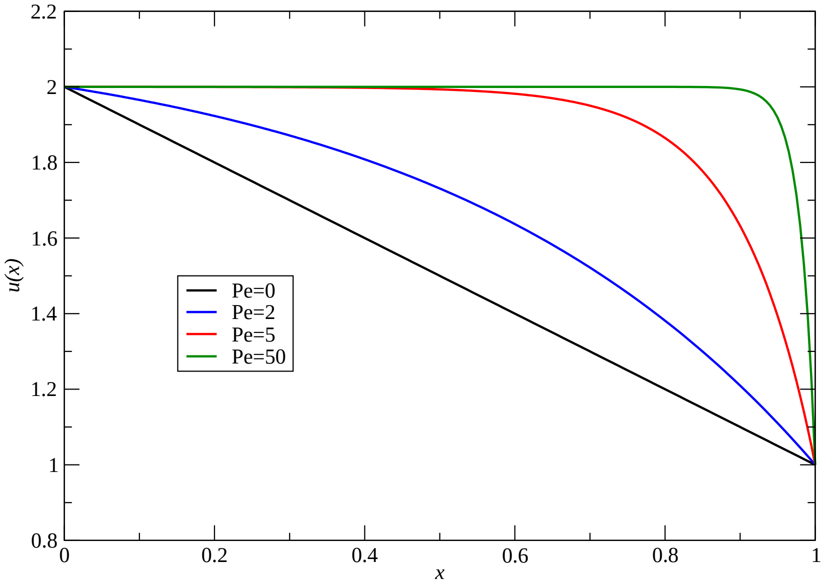


FIGURE IX.6: solution u in the case $u_{\text{in}} = 2$, $u_{\text{out}} = 1$ and for different values of **Pe**

where the Peclet number is defined by: $\mathbf{Pe} = \dfrac{a \times 1}{\kappa}$.

A strong dependence upon the Peclet number is observed: for diffusion-dominated flows, *i.e.* low **Pe**, the solution varies quasi linearly between the in- and outflow boundary values, whereas in advection-dominated situations, *i.e.* high **Pe**, a thin outflow boundary layer is present.

IX.9.3 Related Expressions in PELICANS

The class `RS_AdvectionDiffusionEXP` defines one expression `AdvectionDiffusion1_value`, with the following characteristics :

- Arguments

	Type	Value
Argument 1	`DoubleVector`	(x)
Argument 2	`Double`	a
Argument 3	`Double`	κ
Argument 4	`Double`	u_{in}
Argument 5	`Double`	u_{out}

- Type : `Double`
- Value : $u(x,t)$ given by 9.4

IX.10 Advection Diffusion II

This test describes the diffusion of an initial Gaussian pulse as it is advected.

IX.10.1 Problem Statement

- Let $\Omega =]0,2[^2 \subset \mathbb{R}^2$. The boundary of Ω is denoted by Γ. Let $[0,T]$ be a time interval.

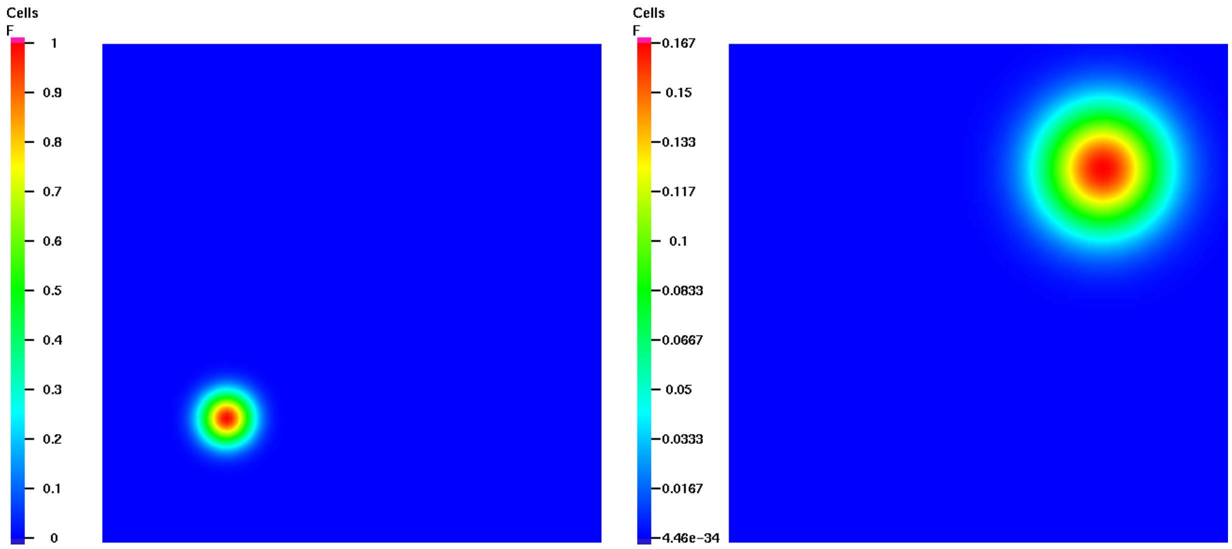


FIGURE IX.7: Advection-diffusion of a Gaussian pulse : initial condition (left) and solution at $t = 1.25$ (right).

- Consider the following problem:

$$\text{(BVP)} \quad \text{find } u : \Omega \times [0,T] \rightarrow \mathbb{R} \text{ such that } \left\{ \begin{aligned} \frac{\partial u}{\partial t} + \mathbf{a}\cdot\nabla u - \kappa_x \frac{\partial^2 u}{\partial x^2} - \kappa_y \frac{\partial^2 u}{\partial y^2} &= 0 \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \Gamma \end{aligned} \right.$$

  with
  - the advective velocity **a**, and the diffusivity coefficients k_x and k_y have to be specified while using the current expression,
  - the initial condition ($t = 0$) is a Gaussian pulse given by the solution 9.5.

IX.10.2 Solution

The solution at any time $t \in [0, T]$ has the following form :

$$u(x,y,t) = \frac{1}{4t+1} \exp\left[-\frac{(x - a_x t - x_0)^2}{\kappa_x(4t+1)} - \frac{(y - a_y t - y_0)^2}{\kappa_y(4t+1)} \right] \tag{9.5}$$

The initial condition and the solution obtained at $t = 1.25$ for :

- an advective velocity, $\mathbf{a} = (0.8, 0.8)$,
- diffusivity coefficients, $(\kappa_x, \kappa_y) = (0.01, 0.01)$,
- and an initial position for the Gaussian pulse center, $(x_0, y_0) = (0.5, 0.5)$,

are depicted respectively on the left and right parts of figure IX.7.

IX.10.3 Related Expressions in PELICANS

The class `RS_AdvectionDiffusionEXP` defines one expression `AdvectionDiffusion2_value`, with the following characteristics :

- Arguments

	Type	Value
Argument 1	`DoubleVector`	(x, y)
Argument 2	`Double`	t
Argument 3	`DoubleVector`	$\mathbf{a}$
Argument 4	`DoubleVector`	(x_0, y_0)
Argument 5	`DoubleVector`	(κ_x, κ_y)

- Type : `Double`
- Value : $u(x, y, t)$ given by 9.5

IX.11 Stokes II

IX.11.1 Problem Statement

- Let $\Omega =]0,1[^2 \subset \mathbb{R}^2$. The boundary of Ω is denoted by Γ.
- Consider the following coupled partial differential equations:

$$\begin{cases} -\nabla^2 \mathbf{u} + \nabla p = \mathbf{f} & \text{in } \Omega \\ \nabla \cdot \mathbf{u} = 0 & \text{in } \Omega \\ \mathbf{u} = 0 & \text{on } \Gamma \end{cases} \tag{9.6}$$

where $\mathbf{f} \colon \Omega \to \mathbb{R}^2$ is given *a priori*.

The unknowns are $\mathbf{u} \colon \Omega \to \mathbb{R}^2$ and $p \colon \Omega \to \mathbb{R}$.

IX.11.2 Solution

An analytical solution to (9.6) is built together with the force term $\mathbf{f}$ as follows. We choose a stream-function φ such that the homogeneous Dirichlet boundary condition holds:

$$\varphi = \frac{1}{2\pi}[\sin(\pi x)\sin(\pi y)]^2 \qquad \mathbf{u} = \begin{bmatrix} \dfrac{\partial \varphi}{\partial y} \\ -\dfrac{\partial \varphi}{\partial x} \end{bmatrix} \tag{9.7}$$

We pick an arbitrary pressure in $\mathrm{L}_0^2(\Omega)$:

$$p = \cos(\pi x)\cos(\pi y) \tag{9.8}$$

The right hand side member is then computed so as to satisfy the first equation of the Stokes problem (9.6).

IX.11.3 Related Expressions in PELICANS

The class `RS_Stokes2` defines four expressions with the following characteristics.

- They have a single argument of type `DoubleVector` and value (x, y) .
- `Stokes2_velocity`

Type	`DoubleVector`
Value	$\mathbf{u}(x, y)$ given by (9.7)

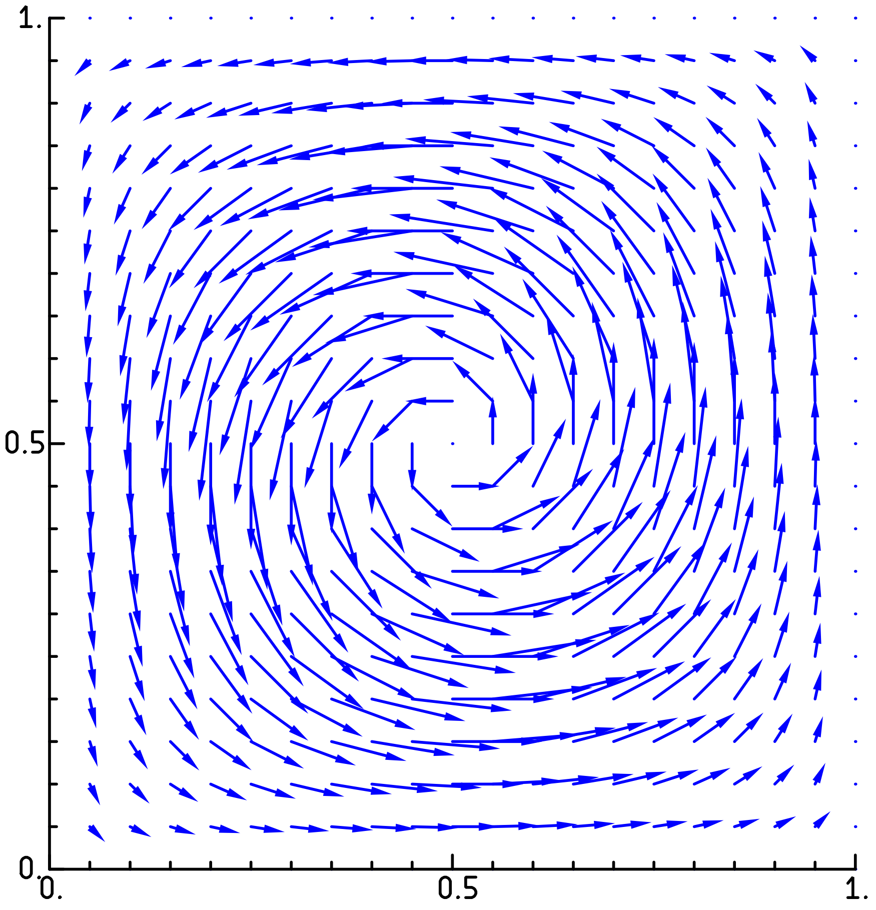


Figure IX.8: velocity field $\mathbf{u}$.

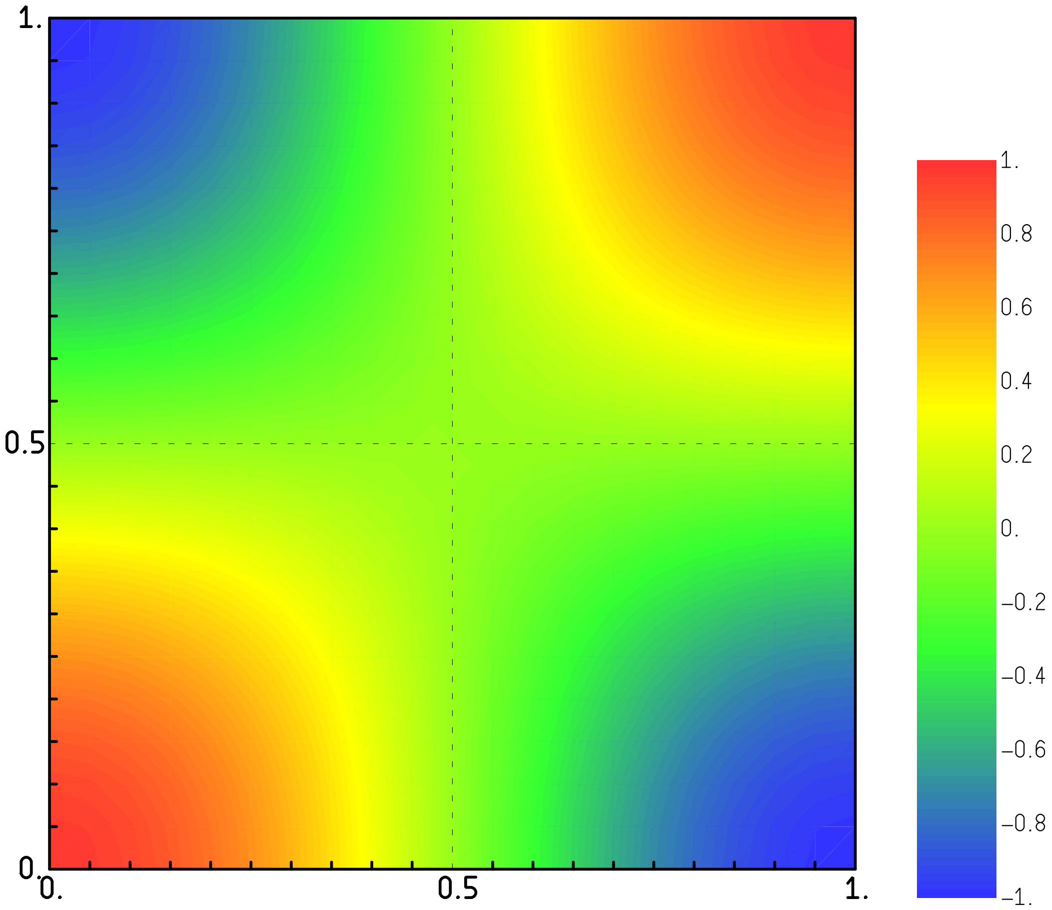


Figure IX.9: pressure field p.

- `Stokes2_pressure`

Type	`DoubleVector`
Value	$p(x, y)$ given by (9.8)

- `Stokes2_force`

Type	`DoubleVector`
Value	$\mathbf{f}(x, y)$ computed from (9.7), (9.8) and (9.6)

- `Stokes2_grad_velocity`

Type	`DoubleArray2D`
Value	$\nabla u(x, y)$ computed from (9.7)

IX.12 NavierStokes I

IX.12.1 Problem Statement

- Let $\Omega =]0,1[^2 \subset \mathbb{R}^2$. The boundary of Ω is denoted by Γ.
- Consider the following coupled partial differential equations:

$$\begin{cases} \alpha(\mathbf{u}\cdot\nabla)\mathbf{u} - \mu\nabla^2\mathbf{u} + \nabla p = \mathbf{f} & \text{in } \Omega \\ \nabla\cdot\mathbf{u} = 0 & \text{in } \Omega \\ \mathbf{u} = 0 & \text{on } \Gamma \end{cases} \tag{9.9}$$

where $\mathbf{f}\colon \Omega \to \mathbb{R}^2$ is given *a priori*.

The unknowns are $\mathbf{u}\colon \Omega \to \mathbb{R}^2$ and $p\colon \Omega \to \mathbb{R}$.

IX.12.2 Solution

An analytical solution to (9.9) is built together with the force term $\mathbf{f}$ as follows. We choose a stream-function φ such that the homogeneous Dirichlet boundary condition holds:

$$\varphi = 1000[x(1-x)y(1-y)]^2 \qquad \mathbf{u} = \begin{bmatrix} \dfrac{\partial\varphi}{\partial y} \\ -\dfrac{\partial\varphi}{\partial x} \end{bmatrix} \tag{9.10}$$

We pick an arbitrary pressure in $\mathrm{L}^2_0(\Omega)$:

$$p = 100(x^2 + y^2 - \frac{2}{3}) \tag{9.11}$$

The right hand side member is then computed so as to satisfy the first equation of the Navier-Stokes problem (9.9).

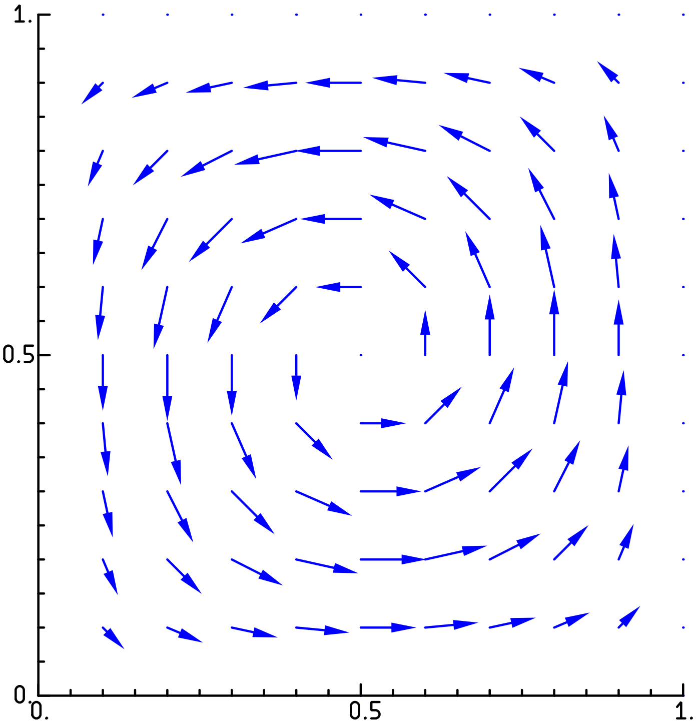


Figure IX.10: velocity field **u**.

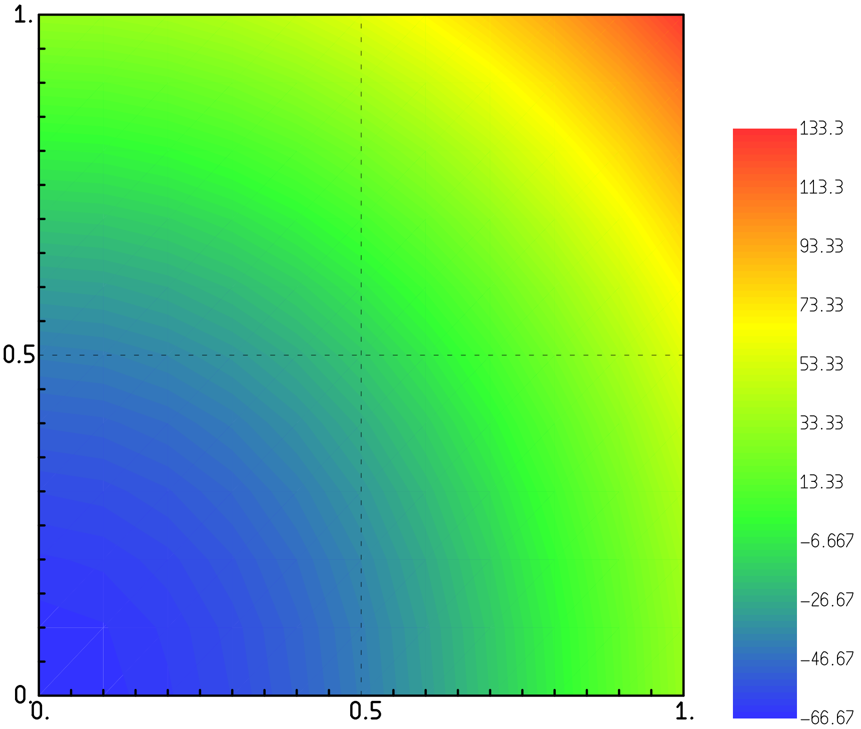


FIGURE IX.11: pressure field p.

IX.12.3 Related Expressions in PELICANS

The class RS_NavierStokes1 defines four expressions with the following characteristics.

	Type	Value
Argument 1	DoubleVector	(x,y)
Argument 2	Double	α
Argument 3	Double	μ

- NavierStokes1_velocity

Type	DoubleVector
Value	$\mathbf{u}(x,y)$ given by (9.10)

- NavierStokes1_pressure

Type	DoubleVector
Value	$p(x,y)$ given by (9.11)

- NavierStokes1_force

Type	DoubleVector
Value	$\mathbf{f}(x,y)$ computed from (9.10), (9.11) and (9.9)

- NavierStokes1_grad_velocity

Type	DoubleArray2D
Value	$\nabla u(x,y)$ computed from (9.10)

IX.13 Taylor-Green Vortices

IX.13.1 Problem Statement

- Let $[0,T]$ be a time interval.
- Consider the following coupled partial differential equations:

$$\left\{\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} - \frac{1}{\mathbf{Re}}\nabla^2\mathbf{u} + \nabla p = 0 \\ \nabla\cdot\mathbf{u} = 0 \end{aligned}\right. \quad \text{in } [0,T]\times\mathbb{R}^2 \tag{9.12}$$

with $\quad \mathbf{u}\colon [0,T]\times\mathbb{R}^2 \to \mathbb{R}^2 \qquad p\colon [0,T]\times\mathbb{R}^2 \to \mathbb{R} \qquad \mathbf{Re} > 0$

IX.13.2 Solution

The following formula define functions that are solution of (9.12):

$$\left\{\begin{aligned} \mathbf{u}(t,x,y) &= \begin{bmatrix} -\cos(2\pi x)\sin(2\pi y) \\ +\sin(2\pi x)\cos(2\pi y) \end{bmatrix} \exp(-\frac{8\pi^2}{\mathbf{Re}}t) \\ p(t,x,y) &= -\frac{\cos(4\pi x)+\cos(4\pi y)}{4}\exp(-\frac{16\pi^2}{\mathbf{Re}}t) \end{aligned}\right. \tag{9.13}$$

This solution is depicted in figures IX.12 and IX.13 for some values of the parameters.

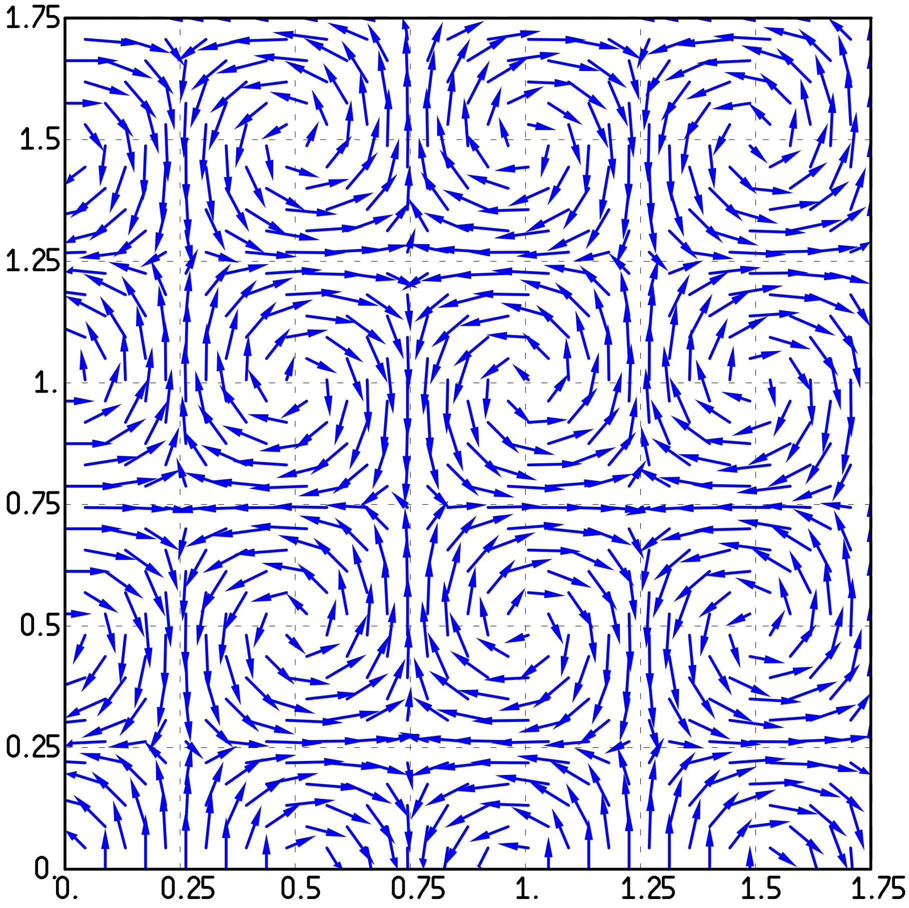


FIGURE IX.12: field $\mathbf{u}(t = 1, x, y)$ in the case $\mathbf{Re} = 100$.

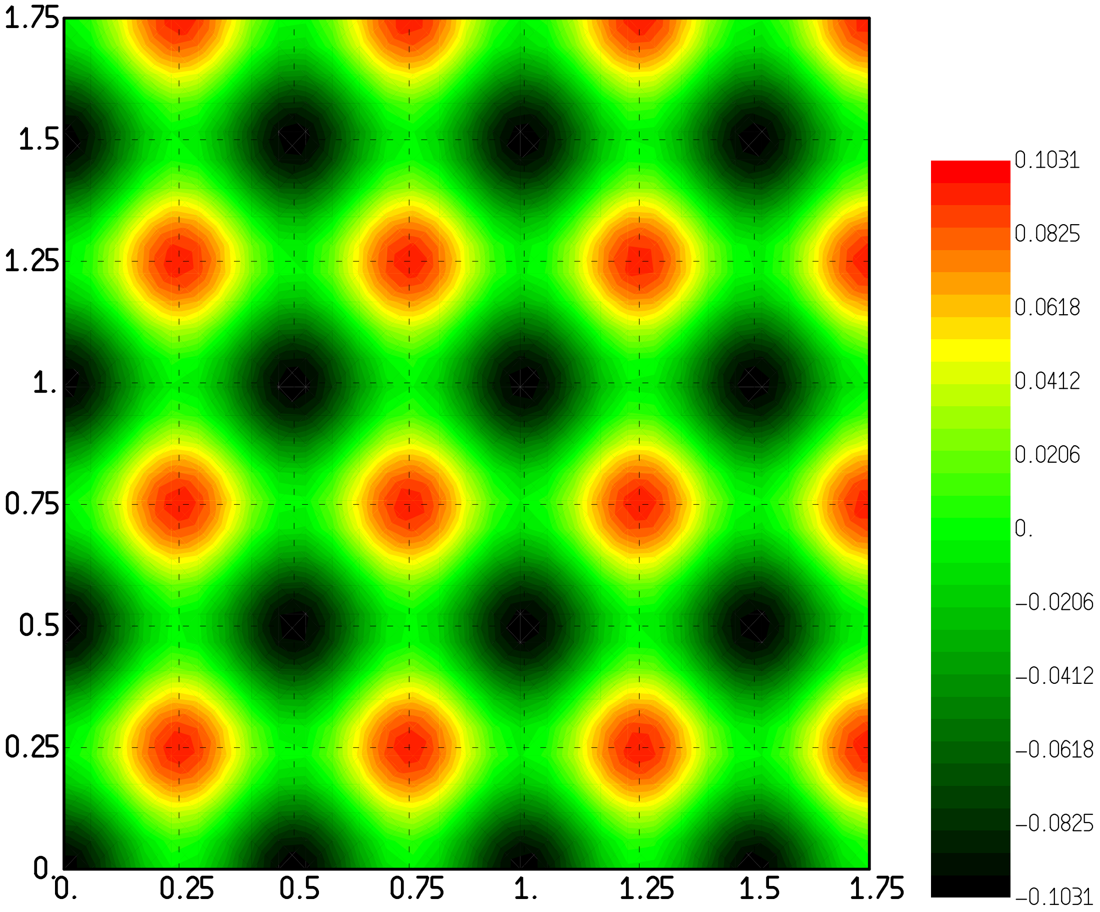


FIGURE IX.13: field $p(t = 1, x, y)$ in the case $\mathbf{Re} = 100$.

IX.13.3 Initial Boundary Value Problems with Analytical Solutions

On any domain $\Omega \subset \mathbb{R}^2$, the equations (9.12), supplemented by initial and boundary conditions defined by (9.13), form an initial boundary value problem whose analytical solution is given in $[0, T] \times \Omega$ by (9.13).

IX.13.4 Related Expressions in PELICANS

The class `RS_GreenTaylorEXP` defines three expressions, with the following characteristics.

	Type	Value
Argument 1	`DoubleVector`	(x, y)
Argument 2	`Double`	t
Argument 3	`Double`	$1/\mathbf{Re}$

- `GreenTaylor_velocity`

Type	`DoubleVector`
Value	$u(t, x, y)$ given by (9.13)

- `GreenTaylor_pressure`

Type	`DoubleVector`
Value	$p(t, x, y)$ given by (9.13)

- `GreenTaylor_grad_velocity`

Type	`DoubleArray2D`
Value	$\nabla u(t, x, y)$ computed from (9.13)

IX.14 Beltrami Flow I

In this section is stated the solution of a Beltrami flow as given in [1].

IX.14.1 Problem Statement

- Let $[0, T]$ be a time interval.
- Consider the following coupled partial differential equations:

$$\left\{ \begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\mathbf{Re}} \nabla^2 \mathbf{u} + \nabla p = 0 \\ \nabla \cdot \mathbf{u} = 0 \end{aligned} \right. \quad \text{in } [0, T] \times \mathbb{R}^3 \tag{9.14}$$

with $\quad \mathbf{u}\colon [0, T] \times \mathbb{R}^3 \to \mathbb{R}^3 \qquad p\colon [0, T] \times \mathbb{R}^3 \to \mathbb{R} \qquad \mathbf{Re} > 0$

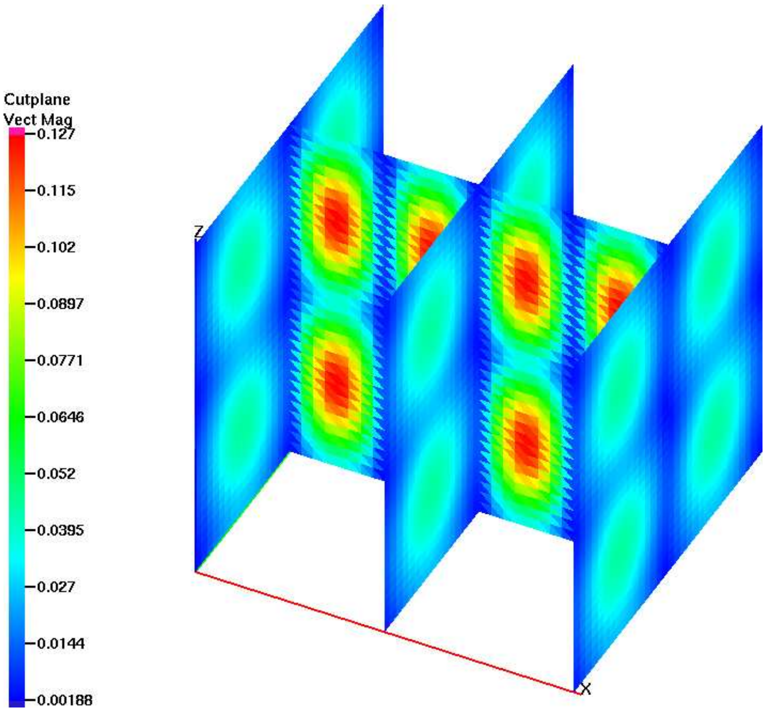


FIGURE IX.14: field $\mathbf{u}(t=1,x,y,z)$ in the case $\Omega =]0,1[^3$, $\mathbf{Re} = 100$, $\mathbf{k} = (4\pi, 2\pi, 2\pi)$, $A = 1$.

IX.14.2 Solution

For all wavenumber $\mathbf{k} = (k_x, k_y, k_z)$ and amplitude A, the following formula define functions that are solution of (9.14) (with $|\mathbf{k}|= \sqrt{k_x^2+k_y^2+k_z^2}$):

$$\left\{ \begin{aligned} \mathbf{u}(t,x,y,z) &= \begin{bmatrix} -\dfrac{A}{k_x^2+k_y^2} \begin{bmatrix} |\mathbf{k}| \; k_y \cos(k_x x)\sin(k_y y)\sin(k_z z) \\ + k_z \, k_x \; \sin(k_x x)\cos(k_y y)\cos(k_z z) \end{bmatrix} \\ -\dfrac{A}{k_x^2+k_y^2} \begin{bmatrix} |\mathbf{k}| \; k_x \sin(k_x x)\cos(k_y y)\sin(k_z z) \\ - k_z \, k_y \; \cos(k_x x)\sin(k_y y)\cos(k_z z) \end{bmatrix} \\ A \qquad \cos(k_x x)\cos(k_y y)\sin(k_z z) \end{bmatrix} \exp(-\frac{1}{\mathbf{Re}}\,|\mathbf{k}|^2\; t) \\ p &= -\frac{1}{2}\,(u_x^2+u_y^2+u_z^2) \end{aligned} \right. \tag{9.15}$$

This solution is depicted in figures IX.14 and IX.15 for some values of the parameters.

IX.14.3 Initial Boundary Value Problems with Analytical Solutions

On any domain $\Omega \subset \mathbb{R}^3$, the equations (9.14), supplemented by initial and boundary conditions defined by (9.15), form an initial boundary value problem whose analytical solution is given in $[0,T] \times \Omega$ by (9.15).

IX.14.4 Related Expressions in PELICANS

The class `RS_BeltramiEXP` defines three expressions, with the following characteristics.

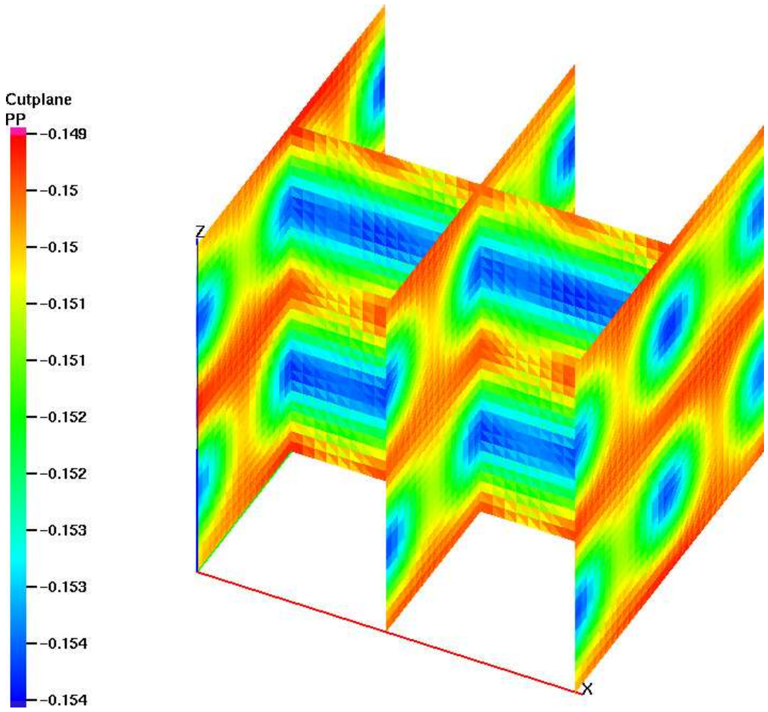


FIGURE IX.15: field $p(t = 1, x, y, z)$ in the case $\Omega =]0, 1[^3$, $\mathbf{Re} = 100$, $\mathbf{k} = (4\pi, 2\pi, 2\pi)$, $A = 1$.

	Type	Value
Argument 1	`DoubleVector`	(x, y, z)
Argument 2	`Double`	t
Argument 3	`DoubleVector`	$\mathbf{k} = (k_x, k_y, k_z)$
Argument 4	`Double`	A
Argument 5	`Double`	$1/\mathbf{Re}$

- `Beltrami_velocity`

Type	`DoubleVector`
Value	$u(t, x, y, z)$ given by (9.15)

- `Beltrami_pressure`

Type	`DoubleVector`
Value	$p(t, x, y, z)$ given by (9.15)

- `Beltrami_grad_velocity`

Type	`DoubleArray2D`
Value	$\nabla u(t, x, y, z)$ computed from (9.15)

IX.15 Lid Driven Cavity Flow

IX.15.1 Problem Statement

IX.15.2 Solution

The computed solution is compared to that of various authors concerning the horizontal velocity cut on the vertical mid-plane [2] [3] [4] and the position of the center of the three vortices [3] [5] [6] [4].

IX.16 Starting Flow in a Long Circular Tube I

IX.16.1 A Simplified PDE with Analytical Solution

Consider the following problem: find $u_\diamond(\tau,\eta)$ such that

$$\begin{cases} \dfrac{\partial^2 u_\diamond}{\partial \eta^2}(\tau,\eta) + \dfrac{1}{\eta}\dfrac{\partial u_\diamond}{\partial \eta}(\tau,\eta) - \mathbf{Re}\,\dfrac{\partial u_\diamond}{\partial \tau}(\tau,\eta) = -\mathbf{Re}\,f(\tau) & \forall \tau \in [0,\infty[\quad \forall \eta \in]0,1[\\ u_\diamond(\tau,\eta=1) = 0 & \forall \tau \in [0,\infty[\\ u_\diamond(\tau=0,\eta) = 0 & \forall \eta \in]0,1[\end{cases} \tag{9.16}$$

where $\mathbf{Re} \geq 0$ and $f(\tau)$ are given *a priori*. The solution is given by [7] [8]:

$$u_\diamond(\tau,\eta) = 2\sum_{n=1}^{\infty} \frac{J_0(\alpha_n \eta)}{\alpha_n J_1(\alpha_n)}\, h_n(\tau) \quad \text{with} \quad h_n(\tau) = \int_0^\tau f(\tau-\phi)\, e^{-\gamma_n \phi}\, d\phi \quad \text{and} \quad \gamma_n = \frac{\alpha_n^2}{\mathbf{Re}} \tag{9.17}$$

where J_0 and J_1 are Bessel functions of first kind and of order zero and one, respectively, and α_n are the zeros of J_0 (*i.e.* $J_0(\alpha_n) = 0$).

The following particular choice of $f(\tau)$ (a, c and τ_0 given *a priori*):

$$f(\tau) = \begin{cases} a\sin(\tau) + c & \tau \leq \tau_0 \\ f(\tau_0) & \tau > \tau_0 \end{cases} \tag{9.18}$$

leads to:

$$h_n(\tau) = \begin{cases} \dfrac{a}{\gamma_n^2+1}(\gamma_n \sin\tau - \cos\tau + e^{-\gamma_n \tau}) + \dfrac{c}{\gamma_n}(1-e^{-\gamma_n\tau}) & \tau \leq \tau_0 \\ \dfrac{1}{\gamma_n}(1-e^{-\gamma_n(\tau-\tau_0)})f(\tau_0) + e^{-\gamma_n(\tau-\tau_0)}h_n(\tau_0) & \tau > \tau_0 \end{cases} \tag{9.19}$$

IX.16.2 Approximate Solution to a Starting Flow

The above problem may be used to derive an approximate solution to the starting incompressible flow of a Newtonian fluid in a tube subjected to a given pressure gradient.

Let Ω be the cylinder of length L and section radius R, defined as follows in the cylindrical system of coordinates (r,θ,z):

$$\Omega = \left\{ \, (r,\theta,z) \;\middle|\; 0 \leq r \leq R \;;\; 0 \leq \theta \leq 2\pi \;;\; 0 \leq z \leq L \, \right\}$$

Let Γ_0 be the part of the boundary Γ of Ω defined by:

$$\Gamma_0 = \left\{ \, (r,\theta,z) \in \Gamma \;\middle|\; r = R \;;\; 0 \leq \theta \leq 2\pi \;;\; 0 \leq z \leq L \, \right\}$$

We consider the flow governed by the following incompressible Navier-Stokes equations:

$$\begin{cases} \varrho\big[\dfrac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\big] - \nabla\cdot\mu(\nabla\mathbf{u} + \nabla\mathbf{u}^{\mathrm{T}}) + \nabla p = \mathbf{0} & \text{in } [0,\infty[\times\Omega \\ \nabla\cdot\mathbf{u} = 0 & \text{in } [0,\infty[\times\Omega \\ \mathbf{u} = 0 & \text{on } [0,\infty[\times\Gamma_0 \\ \mathbf{u} = 0 & \text{on } \{0\}\times\Omega \end{cases} \tag{9.20}$$

with $\quad \mathbf{u}\colon [0,\infty[\times\Omega \to \mathbb{R}^3 \quad p\colon [0,\infty[\times\Omega \to \mathbb{R} \quad \varrho \geq 0 \quad \mu \geq 0 \quad$ (self significant notations).

We make the following assumptions:

- The velocity $\mathbf{u}$ and the pressure p are independant of θ (axisymmetrical conditions).
- Any plane containing the z axis is a symmetry plane for the velocity field. Hence the angular component u_θ of $\mathbf{u}$ is uniformly zero over Ω.
- The flow is *fully-developed*, in the sense that the axial component u_z of $\mathbf{u}$ does not depend on z.

With these assumptions, the incompressibility constraint together with no-slip boundary condition on Γ_0 implies that the radial component u_r of $\mathbf{u}$ is uniformly zero over Ω.

Then, the radial and angular projection of the momentum balance equation leads to $\partial p/\partial r = 0$ and $\partial p/\partial\theta = 0$ respectively. Finally the z axis projection of the momentum balance equation reduces to:

$$\varrho\frac{\partial u_z}{\partial t} = -\frac{\partial p}{\partial z} + \mu\left[\frac{\partial^2 u_z}{\partial r^2} + \frac{1}{r}\frac{\partial u_z}{\partial r}\right] \tag{9.21}$$

where u_z, which depends only on r and t, is uniformly zero at time $t = 0$ (initial condition) and vanishes at any time on $r = R$ (boundary condition).

An immediate consequence of (9.21) is that $\partial p/\partial z = 0$ does not depend on z and thus is solely a function of the time t.

It is advantageous here to use dimensionless variables. The length scale is chosen as the tube radius R, the velocity scale U will be precised later and the time scale is set as R/U. With the dimensionless variables defined as:

$$u_\diamond = \frac{u_z}{U} \qquad p_\diamond = \frac{p}{1/2\varrho U^2} \qquad \xi = \frac{z}{R} \qquad \eta = \frac{r}{R} \qquad \tau = t\frac{U}{R} \tag{9.22}$$

the equation (9.21) and the pertinent boundary conditions are rewritten as:

$$\begin{cases} \dfrac{\partial u_\diamond}{\partial \tau} = -\dfrac{1}{2}\dfrac{\partial p_\diamond}{\partial \xi} + \dfrac{1}{\mathbf{Re}}\left[\dfrac{\partial^2 u_\diamond}{\partial \eta^2} + \dfrac{1}{\eta}\dfrac{\partial^2 u_\diamond}{\partial \eta^2}\right] \qquad \text{with} \qquad \mathbf{Re} = \dfrac{\varrho U R}{\mu} \\ u_\diamond(\tau, \eta = 1) = 0 \\ u_\diamond(\tau = 0, \eta) = 0 \end{cases} \tag{9.23}$$

Now let us define the time function f from the non dimensional axial pressure gradient (which depends only on τ):

$$-\frac{1}{2}\frac{\partial p_\diamond}{\partial \xi} = f(\tau) \tag{9.24}$$

The system (9.23) is formally equivalent to the system (9.16) whose solution is given by (9.17) and by (9.19) for the particular expression (9.18) of f.

IX.16.3 Related Expressions in PELICANS

The class `RS_AvulaEXP` defines four expressions related to the above described approximate solution to a starting flow as follows.

The inflow boundary Γ_{in} and the outflow boundary Γ_{out} are respectively defined as:

$$\begin{aligned} \Gamma_{\text{in}} &= \left\{ (r,\theta,z) \in \Gamma \;\middle|\; 0 \le r \le R \;;\; 0 \le \theta \le 2\pi \;;\; z = 0 \right\} \\ \Gamma_{\text{out}} &= \left\{ (r,\theta,z) \in \Gamma \;\middle|\; 0 \le r \le R \;;\; 0 \le \theta \le 2\pi \;;\; z = L \right\} \end{aligned}$$

so that the subdivision $\Gamma = \Gamma_0 \cup \Gamma_{\text{in}} \cup \Gamma_{\text{out}}$ holds for the boundary Γ of Ω.

From an *a priori* given $T \geq 0$, we set:

$$\tau = 2\pi \frac{t}{T} \quad \text{and} \quad U = 2\pi \frac{R}{T}$$

Moreover the pressure field p is uniformly set to zero on Γ_{out} and, consistently with equation (9.24), is written in Ω:

$$p(t,r,z) = p(t,x,y,z) = p_{\text{in}}(\tau)\frac{L-z}{L} \quad \text{with} \quad p_{\text{in}}(\tau) = \begin{cases} p_1 \sin(\tau) + p_0 & \tau \leq \tau_0 = 2\pi \frac{t_{\text{cut}}}{T} \\ p_{\text{in}}(\tau_0) & \tau > \tau_0 \end{cases} \tag{9.25}$$

Hence $p_{\text{in}}(\tau)$ represents the (uniform) time dependent pressure at the inflow boundary. The constants p_1, p_0, t_{cut} are given *a priori*.

The parameters a and c involved in equation (9.18) are deduced from the relations (9.22) (9.24):

$$\big(\varrho U^2 \frac{L}{R}\big)\ a = p_1 \quad \text{and} \quad \big(\varrho U^2 \frac{L}{R}\big)\ c = p_0$$

Using the 2D axisymmetrical coordinate system (r,z), the velocity is then given by:

$$u_r(t,r,z) = 0 \quad \text{and} \quad u_z(t,r,z) = U u_\diamond(\tau,\eta) \quad \text{with} \quad \eta = \frac{r}{R} \tag{9.26}$$

Using the 3D coordinate system (x,y,z), the above expressions are rewritten:

$$u_x(t,x,y,z) = u_y(t,x,y,z) = 0 \quad \text{and} \quad u_z(t,x,y,z) = U u_\diamond(\tau,\eta) \quad \text{with} \quad \eta = \frac{\sqrt{x^2+y^2}}{R} \tag{9.27}$$

The computation of $u_\diamond(\tau,\eta)$ in `RS_AvulaEXP` is performed by truncation of the infinite sum occuring in formula (9.17) as follows:

$$u_\diamond(\tau,\eta) = 2\sum_{n=1}^{N} \frac{J_0(\alpha_n \eta)}{\alpha_n J_1(\alpha_n)}\ h_n(\tau) \quad \text{with } N \text{ the smallest index s.t.} \quad \left|\frac{J_0(\alpha_N \eta)}{\alpha_N J_1(\alpha_N)}\ h_N(\tau)\right| \leq \texttt{tol} \tag{9.28}$$

A representation of the axial velocity u_z is given in figure IX.16.

The characteristics of the four expressions defined in the class `RS_AvulaEXP` can now be stated precisely.

	Type	Value
Argument 1	`DoubleVector`	(r,z) or (x,y,z)
Argument 2	`Double`	t
Argument 3	`Double`	R
Argument 4	`Double`	L
Argument 5	`Double`	p_1
Argument 6	`Double`	p_0
Argument 7	`Double`	T
Argument 8	`Double`	t_{cut}
Argument 9	`Double`	ϱ
Argument 9	`Double`	μ
Argument 9	`Double`	`tol`

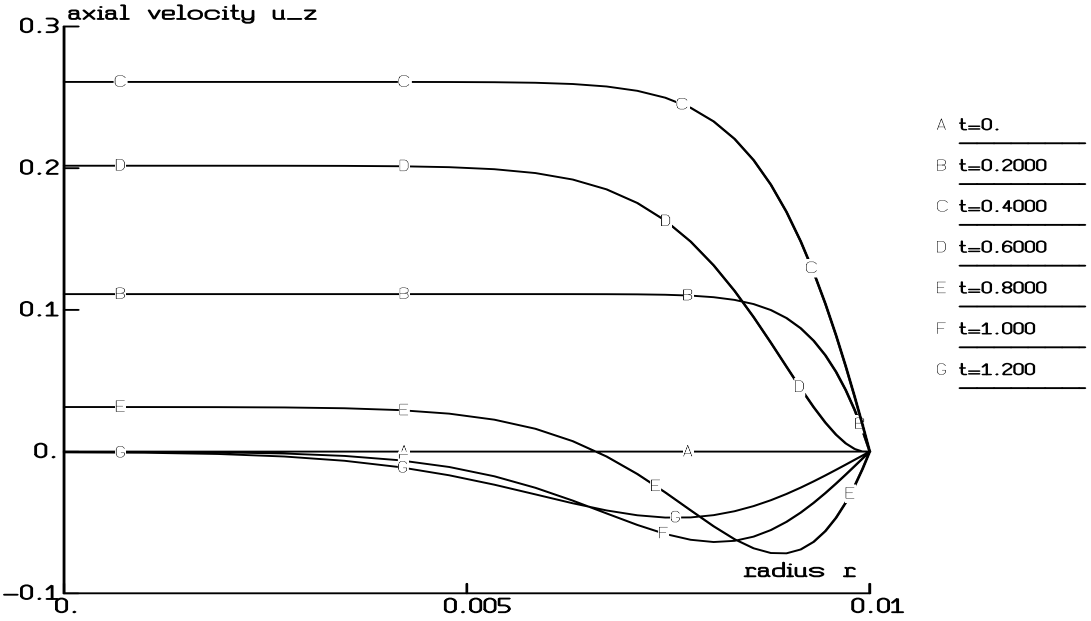


FIGURE IX.16: Axial velocity u_z computed from formula (9.26) with (S.I. units) $R = 0.01$, $L = 0.1$, $p_1 = 100$, $p_2 = 0$, $T = 0.9$, $\varrho = 1065$, $\mu = 0.039$, tol=10^{-8} .

- Avula_velocity_axi

Type	DoubleVector
Value	$\mathbf{u} = \big(u_r(t,r,z), u_z(t,r,z)\big)$ given by (9.26)

The first argument of Avula_velocity_axi represents (r, z).

- Avula_pressure_axi

Type	DoubleVector
Value	$p = p(t,r,z)$ given by (9.25)

The first argument of Avula_pressure_axi represents (r, z).

- Avula_velocity_3D

Type	DoubleVector
Value	$\mathbf{u} = \big(u_x(t,x,y,z), u_y(t,x,y,z), u_z(t,x,y,z)\big)$ given by (9.27)

The first argument of Avula_velocity_3D represents (x, y, z).

- Avula_pressure_3D

Type	DoubleVector
Value	$p = p(t,x,y,z)$ given by (9.25)

The first argument of Avula_pressure_3D represents (x, y, z).

IX.17 Variable Density Incompressible Flows I

IX.17.1 PDEs for the Simulation of Variable Density Incompressible Flows

- Let $\Omega =]0, L[\times]0,1[\subset \mathbb{R}^2$ with $L \in \mathbb{R}$.
- Let $[0, T]$ be a time interval.
- Consider the following coupled partial differential equations:

$$\begin{cases} \varrho[\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}] - \mu\nabla^2\mathbf{u} + \nabla p = \mathbf{f} \\ \frac{\partial \varrho}{\partial t} + (\mathbf{u}\cdot\nabla)\varrho = 0 \qquad \text{in } [0,T]\times\Omega \\ \nabla\cdot\mathbf{u} = 0 \end{cases} \tag{9.29}$$

with $\quad \varrho\colon [0,T]\times\Omega \to \mathbb{R} \qquad \mathbf{u}\colon [0,T]\times\Omega \to \mathbb{R}^2 \qquad p\colon [0,T]\times\Omega \to \mathbb{R} \qquad \mathbf{f}\colon [0,T]\times\Omega \to \mathbb{R}^2$

IX.17.2 Problems Statement and Solutions

Starting from (9.29), a three steps process is followed to devise Initial Boundary Value Problems with analytical solutions.

- First, arbitrary spatio-temporal expressions for $\mathbf{u}$ and p are assumed (yet ensuring $\nabla\cdot\mathbf{u} = 0$).
- Then, $\mathbf{f}$ and ϱ are calculated so as to satisfy the first two equations of (9.29).
- Finally, the system (9.29) is supplemented with initial and boundary conditions inferred from the postulated expressions of $\mathbf{u}$ and p.

With the idea of representing some pulsating flow in an horizontal duct, we set:

$$\begin{aligned} \mathbf{u}(t,x,y) &= \begin{bmatrix} \alpha y(1-y)(2+\cos\beta t) \\ 0 \end{bmatrix} \qquad \forall (t,x,y) \in [0,T]\times\Omega \\ p(t,x,y) &= -2\mu\alpha(2+\cos\beta t)(x - L/2) \end{aligned} \tag{9.30}$$

with $\alpha, \beta \in \mathbb{R}$. Note that:

$$\int_\Omega p = 0$$

Given $\mathbf{u}$, the evolution of ϱ is governed by a pure advection equation that can be solved with the characteristic method, provided that the initial spatial dependence of ϱ is known. Assume that:

$$\varrho(t=0,x,y) = \varrho_0(x) \overset{\text{def}}{=} C\big[1 + \frac{(x-x_0)(-2x+3x_1-x_0)}{(x_1-x_0)^3}\big] \qquad \forall (x,y) \in \Omega \tag{9.31}$$

with $C > 0$, $x_0, x_1 \in]0, L[$ and $x_0 < x_1$. Then:

$$\varrho(t,x,y) = \varrho_0(X) \quad \text{with} \quad X = x - \alpha y(1-y)(2t + \frac{\sin\beta t}{\beta}) \qquad \forall (t,x,y) \in [0,T]\times\Omega \tag{9.32}$$

Note that the initial ϱ field satisfies:

$$\varrho(t=0,x_0,y) = C \qquad \varrho(t=0,x_1,y) = 2C \qquad \forall y \in]0,1[$$

The expressions of $\mathbf{u}$ and p lead to:

$$(\mathbf{u}\cdot\nabla)\mathbf{u} = \mathbf{0} \qquad -\mu\nabla^2\mathbf{u} + \nabla p = \mathbf{0}$$

so that, due to (9.29):

$$\mathbf{f} = \varrho\frac{\partial \mathbf{u}}{\partial t} \tag{9.33}$$

IX.17.3 Summary

The following Initial Boundary Value Problems have been devised:

- solution : (9.30), (9.31), (9.32) ;
- partial differential equations : (9.29), (9.33) ;
- initial and boundary conditions consistent with (9.30), (9.32) ;
- adjustable parameters : α, β, μ, x_0, x_1, C, L.

IX.17.4 Related Expressions in PELICANS

The class `RS_VariableDensityFlow1EXP` defines nine expressions, with the following characteristics.

	Type	Value
Argument 1	`DoubleVector`	(x, y)
Argument 2	`Double`	t
Argument 3	`Double`	α
Argument 4	`Double`	β
Argument 5	`Double`	μ
Argument 6	`Double`	x_0
Argument 7	`Double`	x_1
Argument 8	`Double`	C
Argument 9	`Double`	L

- `VariableDensityFlow1_velocity`

Type	`DoubleVector`
Value	$\mathbf{u}(t, x, y)$ given by (9.30)

- `VariableDensityFlow1_grad_velocity`

Type	`DoubleArray2D`
Value	$\nabla\mathbf{u}(t, x, y)$ computed from (9.30)

- `VariableDensityFlow1_pressure`

Type	`DoubleVector`
Value	$p(t, x, y)$ given by (9.30)

- `VariableDensityFlow1_rho`

Type	`DoubleVector`
Value	$\varrho(t, x, y)$ given by (9.32)

- `VariableDensityFlow1_rhodt`

Type	`DoubleVector`
Value	$\frac{\partial \varrho}{\partial t}(t, x, y)$ computed from (9.32)

- `VariableDensityFlow1_grad_rho`

Type	`DoubleArray2D`
Value	$\nabla\varrho(t,x,y)$ computed from (9.32)

- `VariableDensityFlow1_rho_velocity`

Type	`DoubleVector`
Value	$\varrho\mathbf{u}(t,x,y)$ computed from (9.32), (9.30)

- `VariableDensityFlow1_grad_rho_velocity`

Type	`DoubleArray2D`
Value	$\nabla(\varrho\mathbf{u})(t,x,y)$ computed from (9.32), (9.30)

- `VariableDensityFlow1_rhsf`

Type	`DoubleVector`
Value	$\mathbf{f}(t,x,y)$ given by (9.33)

Bibliography

[1] Alan Shapiro. The use of an exact solution of the navier-stokes equations in a validation test of a three-dimensional nonhydrostatic numerical model. *Monthly Weather Review*, 121:2420–2425, 1993.

[2] T. Tanahashi, H. Okanaga, and Saito T. GSMAC finite element method for unsteady incompressible Navier-Stokes equations at high Reynolds numbers. *International Journal for Numerical Methods in Fluids*, 11:479–499, 1990.

[3] U. Ghia, K.N. Ghia, and C.T. Shin. High-Re solutions for incompressible flow using the Navier-Stokes equation and multigrid methods. *Journal of Computational Physics*, 48:387–411, 1982.

[4] Marcela A. Cruchaga and Eugenio Oñate. A finite element formulation for incompressible flow problems using a generalized streamline operator. *Computer Methods in Applied Mechanics and Engineering*, 143(1-2):49–67, 1997.

[5] M. Bercovier and M. Engelman. A finite element for the numerical solution of viscous incompressible flows. *Journal of Computational Physics*, 30:181–201, 1979.

[6] M. Fortin and F. Thomasset. Mixed finite element methods for incompressible flow problems. *Journal of Computational Physics*, 31:113–145, 1979.

[7] Xavier J. Avula. A combined method for determining velocity of starting flow in a long circular tube. *Journal of the Physical Society of Japan*, 27(2):497–502, 1969.

[8] William M. Deen. *Analysis of Transport Phenomena*. Oxford University Press, 1998. 0195084942.

License for PELICANS

CeCILL-C FREE SOFTWARE LICENSE AGREEMENT

Notice

This Agreement is a Free Software license agreement that is the result of discussions between its authors in order to ensure compliance with the two main principles guiding its drafting:

- firstly, compliance with the principles governing the distribution of Free Software: access to source code, broad rights granted to users,
- secondly, the election of a governing law, French law, with which it is conformant, both as regards the law of torts and intellectual property law, and the protection that it offers to both authors and holders of the economic rights over software.

The authors of the CeCILL-C (for Ce[a] C[nrs] I[nria] L[ogiciel] L[ibre]) license are:

Commissariat à l'Energie Atomique - CEA, a public scientific, technical and industrial research establishment, having its principal place of business at 25 rue Leblanc, immeuble Le Ponant D, 75015 Paris, France.

Centre National de la Recherche Scientifique - CNRS, a public scientific and technological establishment, having its principal place of business at 3 rue Michel-Ange, 75794 Paris cedex 16, France.

Institut National de Recherche en Informatique et en Automatique - INRIA, a public scientific and technological establishment, having its principal place of business at Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay cedex, France.

Preamble

The purpose of this Free Software license agreement is to grant users the right to modify and re-use the software governed by this license.

The exercising of this right is conditional upon the obligation to make available to the community the modifications made to the source code of the software so as to contribute to its evolution.

In consideration of access to the source code and the rights to copy, modify and redistribute granted by the license, users are provided only with a limited warranty and the software's author, the holder of the economic rights, and the successive licensors only have limited liability.

In this respect, the risks associated with loading, using, modifying and/or developing or reproducing the software by the user are brought to the user's attention, given its Free Software status, which may make it complicated to use, with the result that its use is reserved for developers and experienced professionals having in-depth computer knowledge. Users are therefore encouraged to load and test the suitability of the software as regards their requirements in conditions enabling the security of their systems and/or data to be ensured and, more generally, to use and operate it in the same conditions of security. This Agreement may be freely reproduced and published, provided it is not altered, and that no provisions are either added or removed herefrom.

This Agreement may apply to any or all software for which the holder of the economic rights decides to submit the use thereof to its provisions.

Article 1 - DEFINITIONS

For the purpose of this Agreement, when the following expressions commence with a capital letter, they shall have the following meaning:

Agreement: means this license agreement, and its possible subsequent versions and annexes.

Software: means the software in its Object Code and/or Source Code form and, where applicable, its documentation, ”as is” when the Licensee accepts the Agreement.

Initial Software: means the Software in its Source Code and possibly its Object Code form and, where applicable, its documentation, ”as is” when it is first distributed under the terms and conditions of the Agreement.

Modified Software: means the Software modified by at least one Integrated Contribution.

Source Code: means all the Software’s instructions and program lines to which access is required so as to modify the Software.

Object Code: means the binary files originating from the compilation of the Source Code.

Holder: means the holder(s) of the economic rights over the Initial Software.

Licensee: means the Software user(s) having accepted the Agreement.

Contributor: means a Licensee having made at least one Integrated Contribution.

Licensor: means the Holder, or any other individual or legal entity, who distributes the Software under the Agreement.

Integrated Contribution: means any or all modifications, corrections, translations, adaptations and/or new functions integrated into the Source Code by any or all Contributors.

Related Module: means a set of sources files including their documentation that, without modification to the Source Code, enables supplementary functions or services in addition to those offered by the Software.

Derivative Software: means any combination of the Software, modified or not, and of a Related Module.

Parties: mean both the Licensee and the Licensor.

These expressions may be used both in singular and plural form.

Article 2 - PURPOSE

The purpose of the Agreement is the grant by the Licensor to the Licensee of a non-exclusive, transferable and worldwide license for the Software as set forth in Article 5 hereinafter for the whole term of the protection granted by the rights over said Software.

Article 3 - ACCEPTANCE

3.1 – The Licensee shall be deemed as having accepted the terms and conditions of this Agreement upon the occurrence of the first of the following events:

(i) loading the Software by any or all means, notably, by downloading from a remote server, or by loading from a physical medium;

(ii) the first time the Licensee exercises any of the rights granted hereunder.

3.2 – One copy of the Agreement, containing a notice relating to the characteristics of the Software, to the limited warranty, and to the fact that its use is restricted to experienced users has been provided to the Licensee prior to its acceptance as set forth in Article 3.1 hereinabove, and the Licensee hereby acknowledges that it has read and understood it.

Article 4 - EFFECTIVE DATE AND TERM

4.1 EFFECTIVE DATE – The Agreement shall become effective on the date when it is accepted by the Licensee as set forth in Article 3.1.

4.2 TERM – The Agreement shall remain in force for the entire legal term of protection of the economic rights over the Software.

Article 5 - SCOPE OF RIGHTS GRANTED

The Licensor hereby grants to the Licensee, who accepts, the following rights over the Software for any or all use, and for the term of the Agreement, on the basis of the terms and conditions set forth hereinafter.

Besides, if the Licensor owns or comes to own one or more patents protecting all or part of the functions of the Software or of its components, the Licensor undertakes not to enforce the rights granted by these patents against successive Licensees using, exploiting or modifying the Software. If these patents are transferred, the Licensor undertakes to have the transferees subscribe to the obligations set forth in this paragraph.

5.1 RIGHT OF USE – The Licensee is authorized to use the Software, without any limitation as to its fields of application, with it being hereinafter specified that this comprises:

1. permanent or temporary reproduction of all or part of the Software by any or all means and in any or all form.
2. loading, displaying, running, or storing the Software on any or all medium.
3. entitlement to observe, study or test its operation so as to determine the ideas and principles behind any or all constituent elements of said Software. This shall apply when the Licensee carries out any or all loading, displaying, running, transmission or storage operation as regards the Software, that it is entitled to carry out hereunder.

5.2 RIGHT OF MODIFICATION – The right of modification includes the right to translate, adapt, arrange, or make any or all modifications to the Software, and the right to reproduce the resulting software. It includes, in particular, the right to create a Derivative Software.

The Licensee is authorized to make any or all modification to the Software provided that it includes an explicit notice that it is the author of said modification and indicates the date of the creation thereof.

5.3 RIGHT OF DISTRIBUTION – In particular, the right of distribution includes the right to publish, transmit and communicate the Software to the general public on any or all medium, and by any or all means, and the right to market, either in consideration of a fee, or free of charge, one or more copies of the Software by any means.

The Licensee is further authorized to distribute copies of the modified or unmodified Software to third parties according to the terms and conditions set forth hereinafter.

5.3.1 DISTRIBUTION OF SOFTWARE WITHOUT MODIFICATION – The Licensee is authorized to distribute true copies of the Software in Source Code or Object Code form, provided that said distribution complies with all the provisions of the Agreement and is accompanied by:

1. a copy of the Agreement,
2. a notice relating to the limitation of both the Licensor's warranty and liability as set forth in Articles 8 and 9,

and that, in the event that only the Object Code of the Software is redistributed, the Licensee allows effective access to the full Source Code of the Software at a minimum during the entire period of its distribution of the Software, it being understood that the additional cost of acquiring the Source Code shall not exceed the cost of transferring the data.

5.3.2 DISTRIBUTION OF MODIFIED SOFTWARE – When the Licensee makes an Integrated Contribution to the Software, the terms and conditions for the distribution of the resulting Modified Software become subject to all the provisions of this Agreement.

The Licensee is authorized to distribute the Modified Software, in source code or object code form, provided that said distribution complies with all the provisions of the Agreement and is accompanied by:

1. a copy of the Agreement,
2. a notice relating to the limitation of both the Licensor's warranty and liability as set forth in Articles 8 and 9,

and that, in the event that only the object code of the Modified Software is redistributed, the Licensee allows effective access to the full source code of the Modified Software at a minimum during the entire period of its distribution of the Modified Software, it being understood that the additional cost of acquiring the source code shall not exceed the cost of transferring the data.

5.3.3 DISTRIBUTION OF DERIVATIVE SOFTWARE – When the Licensee creates Derivative Software, this Derivative Software may be distributed under a license agreement other than this Agreement, subject to compliance with the requirement to include a notice concerning the rights over the Software as defined in Article 6.4. In the event the creation of the Derivative Software required modification of the Source Code, the Licensee undertakes that:

1. the resulting Modified Software will be governed by this Agreement,
2. the Integrated Contributions in the resulting Modified Software will be clearly identified and documented,
3. the Licensee will allow effective access to the source code of the Modified Software, at a minimum during the entire period of distribution of the Derivative Software, such that such modifications may be carried over in a subsequent version of the Software; it being understood that the additional cost of purchasing the source code of the Modified Software shall not exceed the cost of transferring the data.

5.3.4 COMPATIBILITY WITH THE CeCILL LICENSE – When a Modified Software contains an Integrated Contribution subject to the CeCILL license agreement, or when a Derivative Software contains a Related Module subject to the CeCILL license agreement, the provisions set forth in the third item of Article 6.4 are optional.

Article 6 - INTELLECTUAL PROPERTY

6.1 OVER THE INITIAL SOFTWARE – The Holder owns the economic rights over the Initial Software. Any or all use of the Initial Software is subject to compliance with the terms and conditions under which the Holder has elected to distribute its work and no one shall be entitled to modify the terms and conditions for the distribution of said Initial Software.

The Holder undertakes that the Initial Software will remain ruled at least by this Agreement, for the duration set forth in Article 4.2.

6.2 OVER THE INTEGRATED CONTRIBUTIONS – The Licensee who develops an Integrated Contribution is the owner of the intellectual property rights over this Contribution as defined by applicable law.

6.3 OVER THE RELATED MODULES – The Licensee who develops a Related Module is the owner of the intellectual property rights over this Related Module as defined by applicable law and is free to choose the type of agreement that shall govern its distribution under the conditions defined in Article 5.3.3.

6.4 NOTICE OF RIGHTS – The Licensee expressly undertakes:

1. not to remove, or modify, in any manner, the intellectual property notices attached to the Software;
2. to reproduce said notices, in an identical manner, in the copies of the Software modified or not;
3. to ensure that use of the Software, its intellectual property notices and the fact that it is governed by the Agreement is indicated in a text that is easily accessible, specifically from the interface of any Derivative Software.

The Licensee undertakes not to directly or indirectly infringe the intellectual property rights of the Holder and/or Contributors on the Software and to take, where applicable, vis-à-vis its staff, any and all measures required to ensure respect of said intellectual property rights of the Holder and/or Contributors.

Article 7 - RELATED SERVICES

7.1 – Under no circumstances shall the Agreement oblige the Licensor to provide technical assistance or maintenance services for the Software.

However, the Licensor is entitled to offer this type of services. The terms and conditions of such technical assistance, and/or such maintenance, shall be set forth in a separate instrument. Only the Licensor offering said maintenance and/or technical assistance services shall incur liability therefor.

7.2 – Similarly, any Licensor is entitled to offer to its licensees, under its sole responsibility, a warranty, that shall only be binding upon itself, for the redistribution of the Software and/or the Modified Software, under terms and conditions that it is free to decide. Said warranty, and the financial terms and conditions of its application, shall be subject of a separate instrument executed between the Licensor and the Licensee.

Article 8 - LIABILITY

8.1 – Subject to the provisions of Article 8.2, the Licensee shall be entitled to claim compensation for any direct loss it may have suffered from the Software as a result of a fault on the part of the relevant Licensor, subject to providing evidence thereof.

8.2 – The Licensor's liability is limited to the commitments made under this Agreement and shall not be incurred as a result of in particular: (i) loss due the Licensee's total or partial failure to fulfill its obligations, (ii) direct or consequential loss that is suffered by the Licensee due to the use or performance of the Software, and (iii) more generally, any consequential loss. In particular the Parties expressly agree that any or all pecuniary or business loss (i.e. loss of data, loss of profits, operating loss, loss of customers or orders, opportunity cost, any disturbance to business activities) or any or all legal proceedings instituted against the Licensee by a third party, shall constitute consequential loss and shall not provide entitlement to any or all compensation from the Licensor.

Article 9 - WARRANTY

9.1 – The Licensee acknowledges that the scientific and technical state-of-the-art when the Software was distributed did not enable all possible uses to be tested and verified, nor for the presence of possible defects to be detected. In this respect, the Licensee's attention has been drawn to the risks associated with loading, using, modifying and/or developing and reproducing the Software which are reserved for experienced users.

The Licensee shall be responsible for verifying, by any or all means, the suitability of the product for its requirements, its good working order, and for ensuring that it shall not cause damage to either persons or properties.

9.2 – The Licensor hereby represents, in good faith, that it is entitled to grant all the rights over the Software (including in particular the rights set forth in Article 5).

9.3 – The Licensee acknowledges that the Software is supplied ”as is” by the Licensor without any other express or tacit warranty, other than that provided for in Article 9.2 and, in particular, without any warranty as to its commercial value, its secured, safe, innovative or relevant nature.

Specifically, the Licensor does not warrant that the Software is free from any error, that it will operate without interruption, that it will be compatible with the Licensee's own equipment and software configuration, nor that it will meet the Licensee's requirements.

9.4 – The Licensor does not either expressly or tacitly warrant that the Software does not infringe any third party intellectual property right relating to a patent, software or any other property right. Therefore, the Licensor disclaims any and all liability towards the Licensee arising out of any or all proceedings for infringement that may be instituted in respect of the use, modification and redistribution of the Software. Nevertheless, should such proceedings be instituted against the Licensee, the Licensor shall provide it with technical and legal assistance for its defense. Such technical and legal assistance shall be decided on a case-by-case basis between the relevant Licensor and the Licensee pursuant to a memorandum of understanding. The Licensor disclaims any and all liability as regards the Licensee's use of the name of the Software. No warranty is given as regards the existence of prior rights over the name of the Software or as regards the existence of a trademark.

Article 10 - TERMINATION

10.1 – In the event of a breach by the Licensee of its obligations hereunder, the Licensor may automatically terminate this Agreement thirty (30) days after notice has been sent to the Licensee and has remained ineffective.

10.2 – A Licensee whose Agreement is terminated shall no longer be authorized to use, modify or distribute the Software. However, any licenses that it may have granted prior to termination of the Agreement shall remain valid subject to their having been granted in compliance with the terms and conditions hereof.

Article 11 - MISCELLANEOUS

11.1 EXCUSABLE EVENTS – Neither Party shall be liable for any or all delay, or failure to perform the Agreement, that may be attributable to an event of force majeure, an act of God or an outside cause, such as defective functioning or interruptions of the electricity or telecommunications networks, network paralysis following a virus attack, intervention by government authorities, natural disasters, water damage, earthquakes, fire, explosions, strikes and labor unrest, war, etc.

11.2 – Any failure by either Party, on one or more occasions, to invoke one or more of the provisions hereof, shall under no circumstances be interpreted as being a waiver by the interested Party of its right to invoke said provision(s) subsequently.

11.3 – The Agreement cancels and replaces any or all previous agreements, whether written or oral, between the Parties and having the same purpose, and constitutes the entirety of the agreement between said Parties concerning said purpose. No supplement or modification to the terms and conditions hereof shall be effective as between the Parties unless it is made in writing and signed by their duly authorized representatives.

11.4 – In the event that one or more of the provisions hereof were to conflict with a current or future applicable act or legislative text, said act or legislative text shall prevail, and the Parties shall make the necessary amendments so as to comply with said act or legislative text. All other provisions shall remain effective. Similarly, invalidity of a provision of the Agreement, for any reason whatsoever, shall not cause the Agreement as a whole to be invalid.

11.5 LANGUAGE – The Agreement is drafted in both French and English and both versions are deemed authentic.

Article 12 - NEW VERSIONS OF THE AGREEMENT

12.1 – Any person is authorized to duplicate and distribute copies of this Agreement.

12.2 – So as to ensure coherence, the wording of this Agreement is protected and may only be modified by the authors of the License, who reserve the right to periodically publish updates or new versions of the Agreement, each with a separate number. These subsequent versions may address new issues encountered by Free Software.

12.3 – Any Software distributed under a given version of the Agreement may only be subsequently distributed under the same version of the Agreement or a subsequent version.

Article 13 - GOVERNING LAW AND JURISDICTION

13.1 – The Agreement is governed by French law. The Parties agree to endeavor to seek an amicable solution to any disagreements or disputes that may arise during the performance of the Agreement.

13.2 – Failing an amicable solution within two (2) months as from their occurrence, and unless emergency proceedings are necessary, the disagreements or disputes shall be referred to the Paris Courts having jurisdiction, by the more diligent Party.

Version 1.0 dated 2006-09-05.

Contents