

## Hackathon du 21/11 : notes sur l'atelier « modèle de données »

### Éléments pré-existants

En amont du Hackathon, un modèle de données « de première intention » a été élaboré par des participants des Mines, de Safran et EDF.

Ce modèle de données se trouve dans le dépôt Git du projet, à la page suivante :

[https://github.com/frederique-pigeon/mordicus/blob/master/source/datamodel/class\\_diagram.rst](https://github.com/frederique-pigeon/mordicus/blob/master/source/datamodel/class_diagram.rst)

Pour plus de lisibilité, ce modèle de données a été divisé en 3 grandes parties : gestion des données de simulation, traitements offline et traitements online.

Cet atelier avait pour but de développer la partie « gestion des données de simulation ». Le modèle de données de première intention associé est présenté en Figure 1.

Les différents concepts présents dans ce modèle de données étant expliqués dans le glossaire :

<https://github.com/frederique-pigeon/mordicus/blob/master/source/datamodel/glossary.rst>

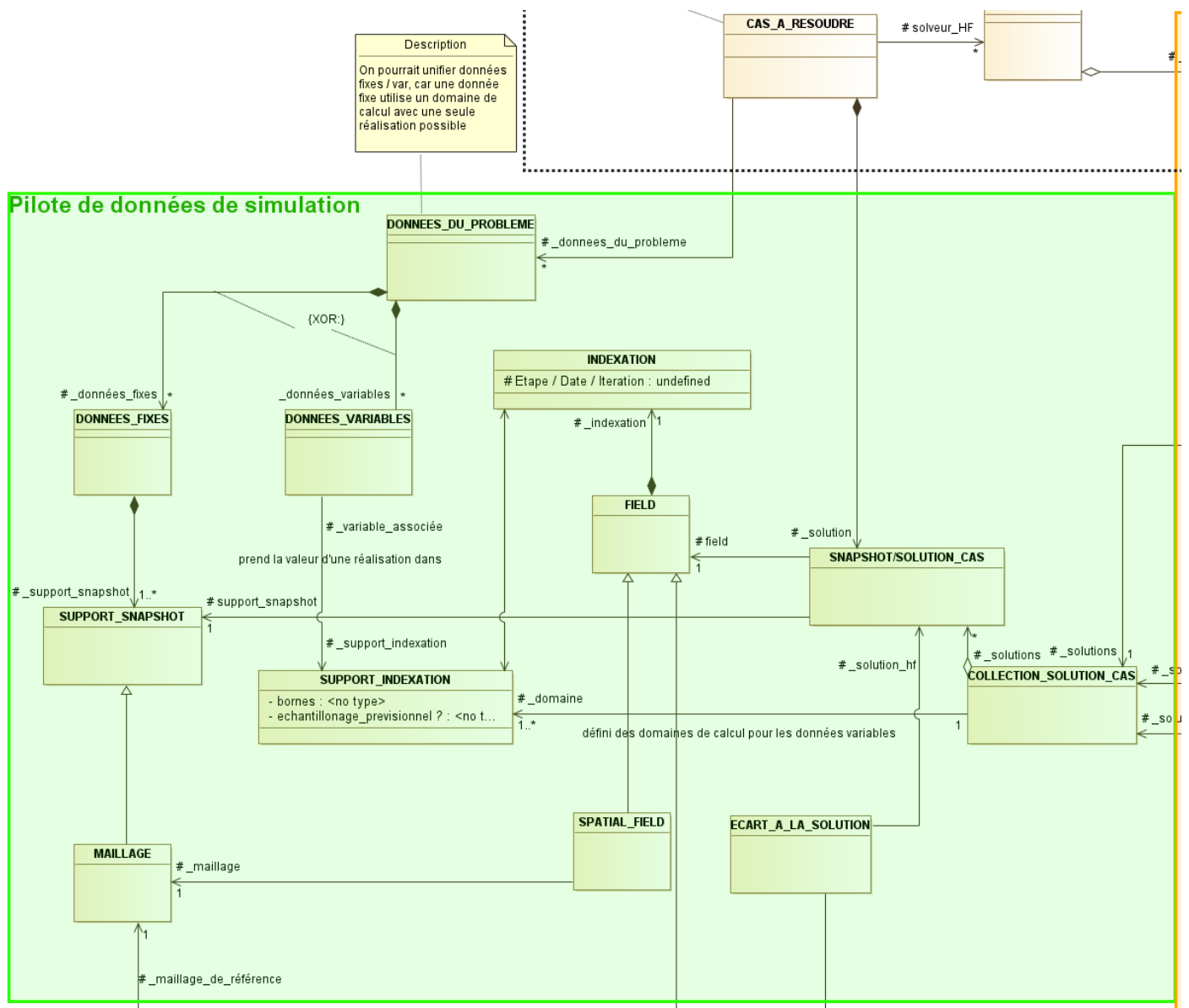


Figure 1 : modèle de données « de première intention », fait en préparation du Hackathon, pour la partie « Pilote de données de simulation ».

## Notes sur l'atelier

3 applications identifiées pour la bibliothèque Mordicus :

- Réduction de modèles → nombre d'inconnues du « champ à réduire » fixe, car on ne peut pas faire de compression type SVD sur des champs dont la taille est variable
- Reconstruction de données manquantes (Gappy POD) → nombre d'inconnues variables
- Inférence à partir de données expérimentales → la structure de données doit permettre de traiter des données qui ne viennent pas de la simulation

### Données du problème

On a ensuite déroulé le modèle de données de première intention en développant concept par concept :

*Données du problème* ou *case data* : Tout ce qui définit le « problème » à réduire.

Néanmoins, la nécessité est apparue de distinguer les *données du problème*, notion se rapportant plus à la physique du problème d'une seconde notion : celle de *données brutes*.

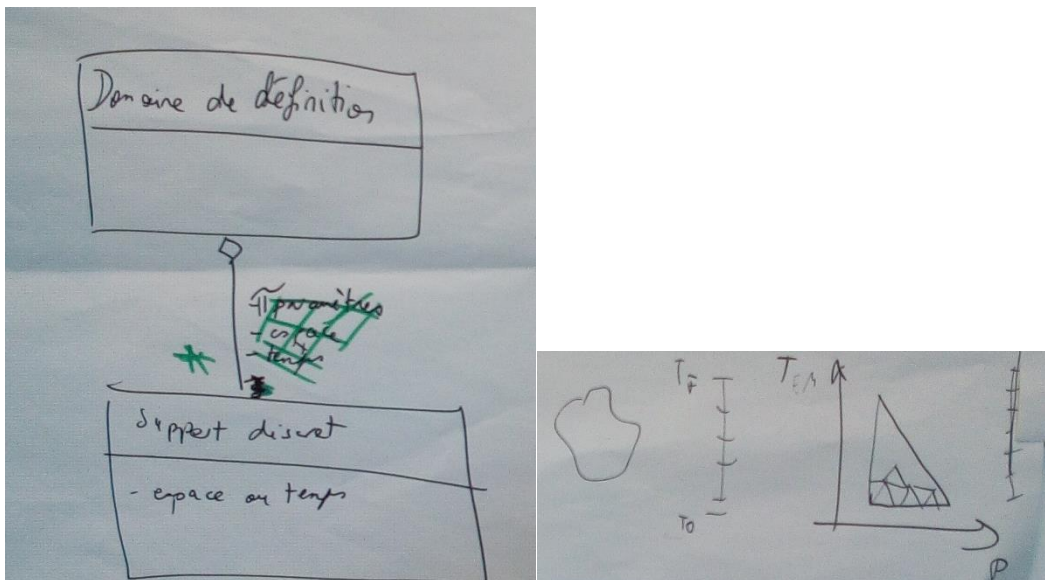
*Données brutes* : tout ce qui est nécessaire à un utilisateur « réducteur de modèles » (qui ne connaît pas nécessairement la physique) pour appliquer une méthode de réduction de modèles non-intrusive.

Proposition de définition par Mickaël : c'est composé de :

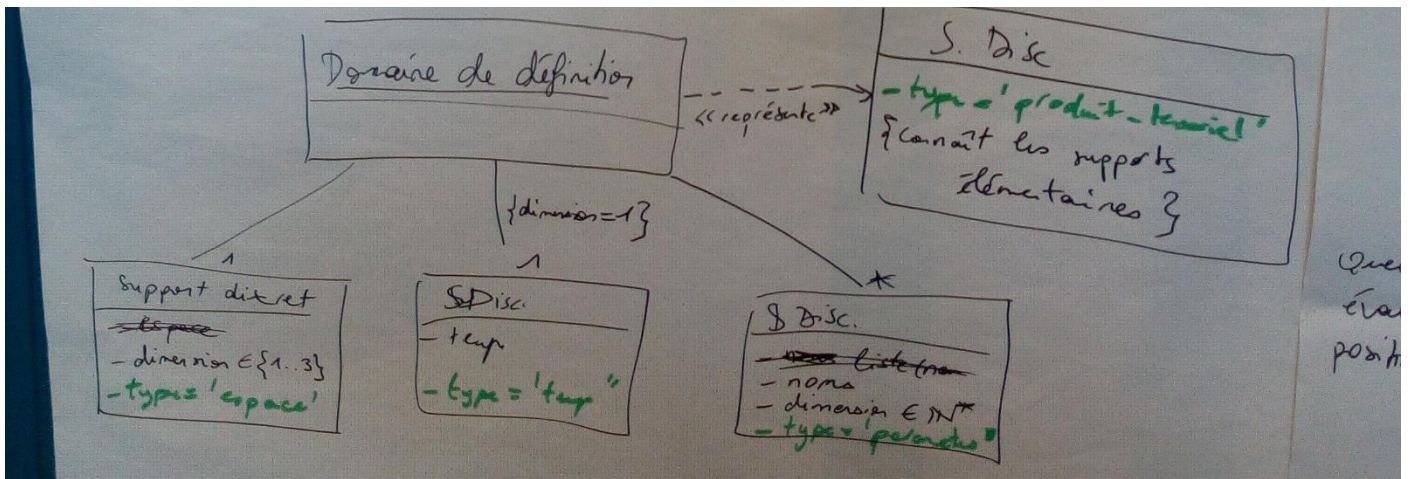
- *inconnues* ;
- *équations à résoudre* ;
- *domaine de définition* (espace – temps – paramètres)

### Domaine de définition et support discret

Ce qui a amené la définition du domaine de définition comme un produit cartésien de supports discrets :



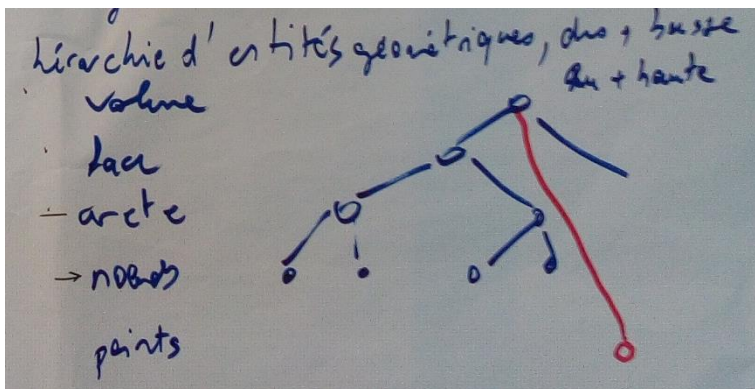
Ces supports discrets sont des instances de la même classe, c'est-à-dire qu'au niveau structure de données on ne différencie pas « maillage 1D en espace » et « discrétisation en temps » par exemple. Néanmoins, cette classe porte un attribut « type » pour dire à quelle domaine se rapporte le support : espace, temps ou paramètres. Il existe également un attribut « produit\_tensoriel » pour créer des supports dérivés par combinaison (produit cartésien) de plusieurs supports :



On arrive ainsi naturellement à la définition du *support discret* (une sorte de généralisation de la notion de *maillage*)

C'est une hiérarchie d'entités géométriques, partant d'un niveau le plus bas constitué par des *nœuds* ou des *points*. Les *nœuds* sont définis directement par leurs coordonnées, alors que les *points* le sont par une transformation à partir d'une entité de plus haut niveau : typiquement centre d'une cellule ou point de Gauss dans une face.

Les entités de niveaux supérieurs (arête, face, cellule...) sont définis exclusivement à partir des niveaux immédiatement inférieurs, formant ainsi un arbre :



Un *support discret* doit prévoir la possibilité de taguer des groupes d'objets.

Un *support discret* est à considérer surtout comme une agrégation de *supports de ddl* (dof support). Un *support de ddl* est une entité géométrique à associer avec une inconnue (1 ddl) du problème. Dans la plupart des cas, ce sont des *nœuds* (éléments finis) ou des *points* (volumes finis), mais pas nécessairement.

Exigence : il faut se réserver la possibilité d'associer des ddls avec des arêtes, faces ou volumes pour des méthodes de discrétisations particulières, par exemple éléments de Nédélec (particulièrement utilisés en électro-magnétisme).

## Champ et inconnues

On s'est ensuite posé la question de définir la notion d'*inconnues*

Plusieurs essais :

- Inconnue = variable d'un problème de dimension topologique (géométrique) donnée
- Inconnue = tenseur (ordre 0, 1, 2...) représentant une quantité physique

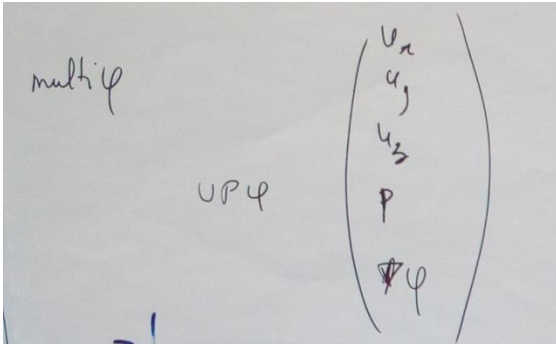
Assez rapidement, le besoin s'est fait sentir de distinguer la notion d'*inconnues* de celle de *champ* (field).

Idem, tentative pour définir la notion de *champ* :

*Champ* = répartition d'une *grandeur* (ou *quantité physique*) sur un support discret.

Exigence (notamment pour traiter le cas « reconstruction de données manquantes ») : un *champ* doit pouvoir permettre une variation du nombre de *composantes* d'un *support de ddl* à un autre.

Exigence (pour traiter les cas multiphysiques de façon monolithique, par exemple IFS) : un *vecteur d'inconnues* doit pouvoir être multi-unités :



On a finalement fini par converger :

Une *quantité physique* (ou *grandeur*) est associée à une seule unité au sens SI.

Un *champ* : valeurs portées par un *support discret*, en général sur les *nœuds* ou les *points*. Correspond à une seule *quantité physique* (éventuellement tensorielle). Peut être d'origine expérimentale ou un post-traitement, rarement la sortie brute de la simulation. Tous les *points* sur lesquels le champ porte une valeur (attention au cas données manquantes) ont le même nombre de composantes.

Définition alternative plus générale : quelque chose qu'on peut évaluer à n'importe quelle position à l'intérieur du domaine  $f(x)$ .

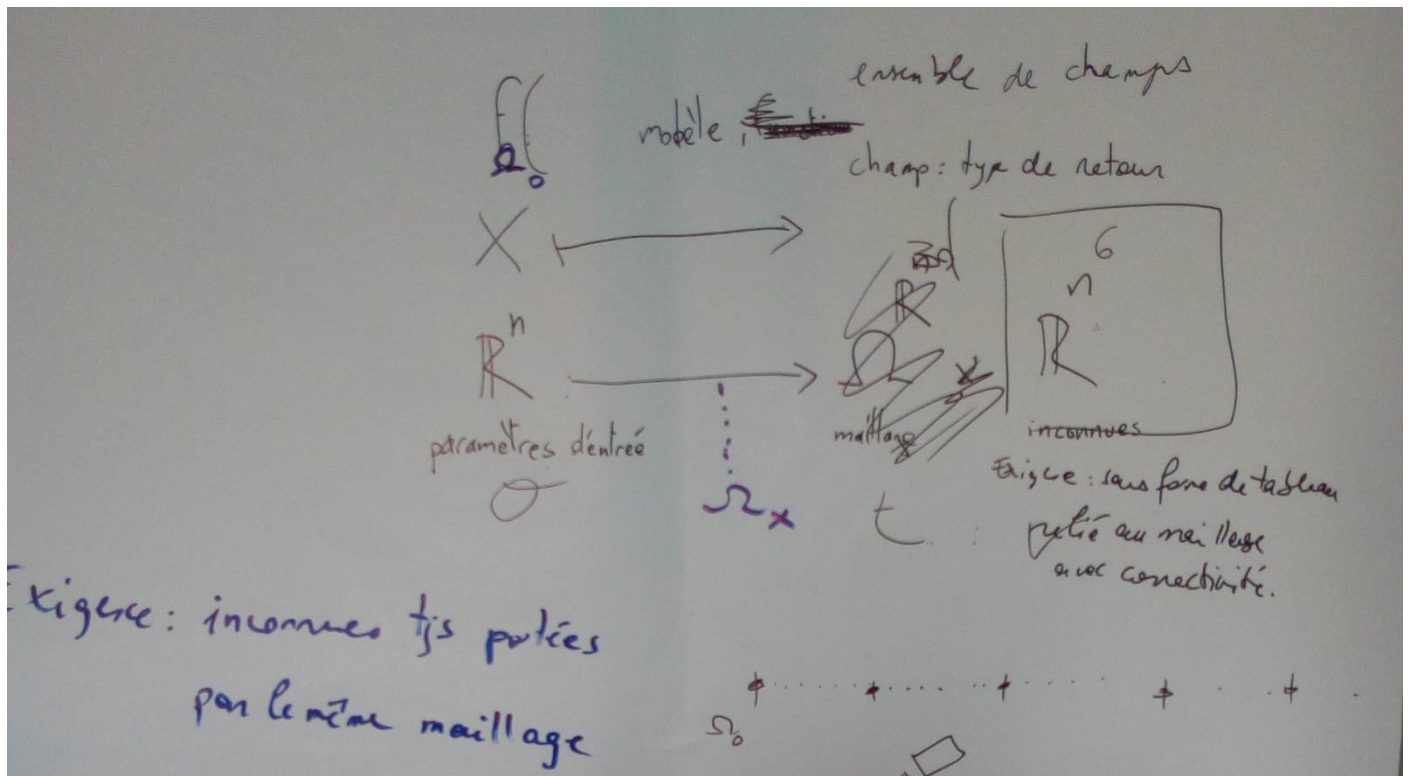
Un *vecteur d'inconnues* : C'est le vecteur d'état (vecteur des variables d'état discrètes)  $X$  que le problème doit déterminer. C'est la sortie primale de la modélisation. Il peut mélanger des inconnues de différentes unités, autrement dit mélanger différentes *quantités physiques* : déplacement, pression, multiplicateur de Lagrange...

Une *inconnue* peut être associée à un *support de ddl*. Ce n'est pas systématique (cf certains multiplicateur de Lagrange introduits de façon purement algébrique). Un *vecteur d'inconnues* est donc également associé à un *support discret*, mais de façon plus indirecte. Ce sujet sera abordé plus en détail dans le § « lien entre champ et vecteur d'inconnues ».

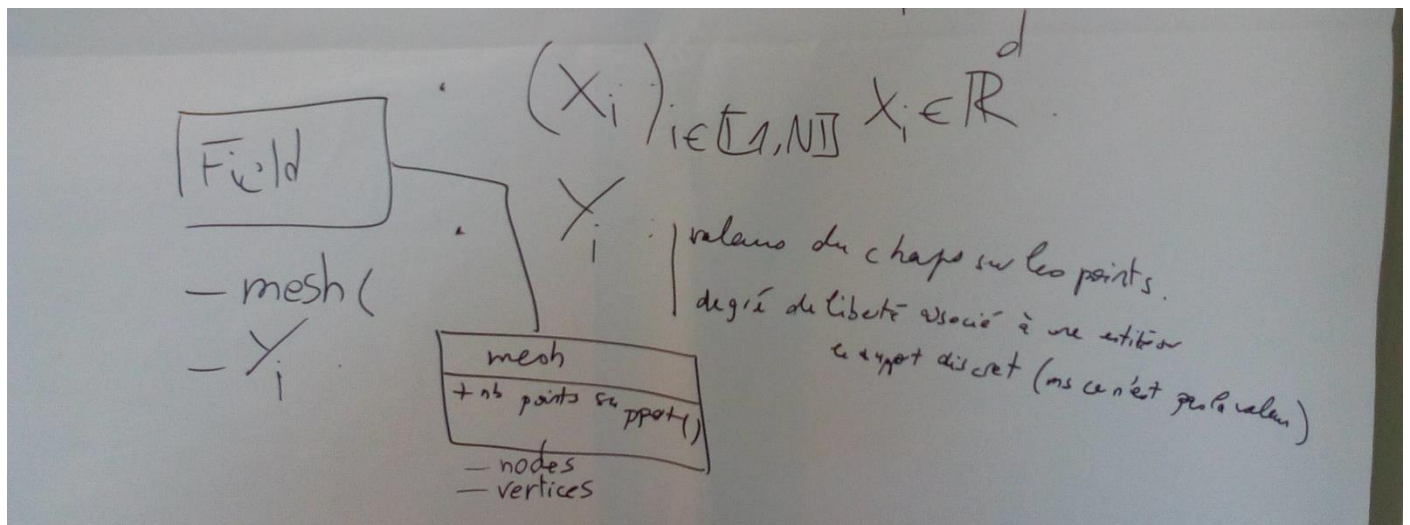
## Modèle

Vision Phiméca : un *modèle* est une fonction du domaine paramètres x temps, qui donne en retour un champ. S'il peut faire appel à des maillages variables  $\Omega_x$  en interne (selon la valeur du paramètre ou du temps, par exemple), tous les champs produits au final doivent se rapporter à un unique maillage « de référence »  $\Omega_0$ . La transformation entre les deux est masquée à l'intérieur de la fonction en quelque sorte :





Chaque degré de liberté est associé à une entité du *support discret* mais ce n'est pas nécessairement la *valeur* à cet endroit :

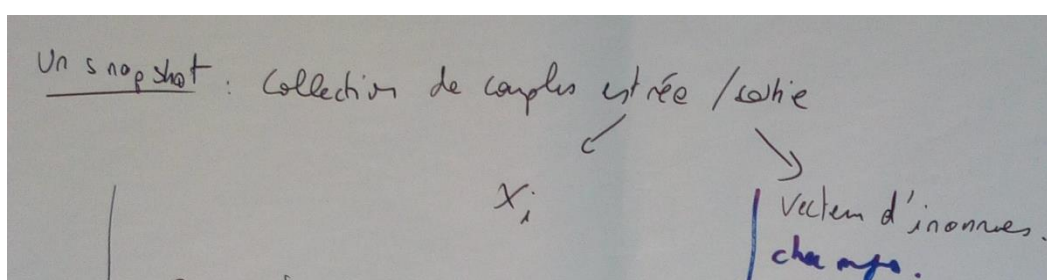


Question : le type de retour est-il vraiment un *champ*, n'est-ce pas plutôt un *vecteur d'inconnues* ?

Ce qui a amené la question suivante : Un *snapshot* est-il un *champ* ou un *vecteur d'inconnues* ?

### Snapshot

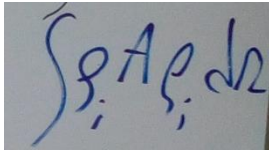
Définition d'un *snapshot* :



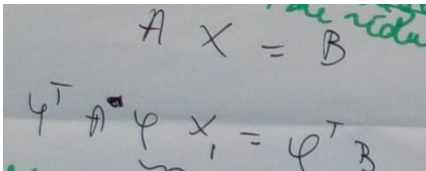
Il a donc été choisi de tolérer les deux types de sorties (*champs* ou *vecteur d'inconnues*) pour un *snapshot*.

En effet, un snapshot sert à produire une base de modes empiriques  $\varphi$ , base qui servira **d'une façon ou d'une autre** à projeter les équations du problèmes. Or il y a deux « façons » de projeter :

- en écrivant la projection de façon intégrale (approches intrusives ou « recodées »), ou plus généralement avec une méthode de réduction qui oblige à calculer des intégrales :


$$\int \varphi_i A \varphi_j d\Omega$$

- en écrivant la projection de façon algébrique :


$$A x = B$$
$$\varphi^T A \varphi x = \varphi^T B$$

Dans le 1<sup>er</sup> cas, on a besoin d'une base empirique sous forme de *champs*  $f(x)$  pour pouvoir écrire des intégrales.

Dans le 2<sup>nd</sup> cas, on a besoin d'une base empirique sous formes de *vecteurs d'inconnues*, pour avoir des opérations algébriques compatibles (en dimension) avec celles du problème complet.

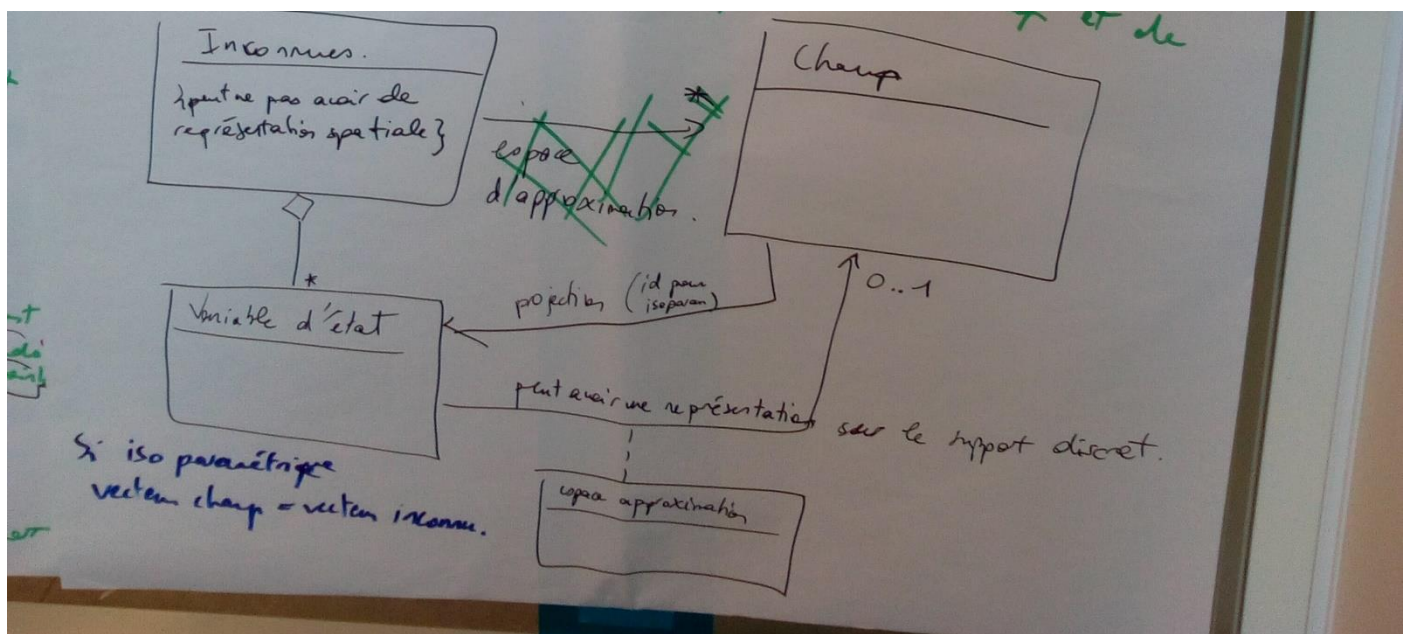
On peut aussi avoir une sortie du modèle d'un type (*champ* ou *vecteur d'inconnue*) et une méthode de projection d'un autre. Il faut alors pouvoir convertir entre les deux. On rejoint la problématique du lien entre *champ* et *vecteur d'inconnues*.

### Lien entre champ et vecteur d'inconnues

Comme on l'a dit, un *vecteur d'inconnues* agrège plusieurs *variables d'état* (déplacement, multiplicateurs...), dont certaines peuvent ne pas avoir de représentation spatiale.

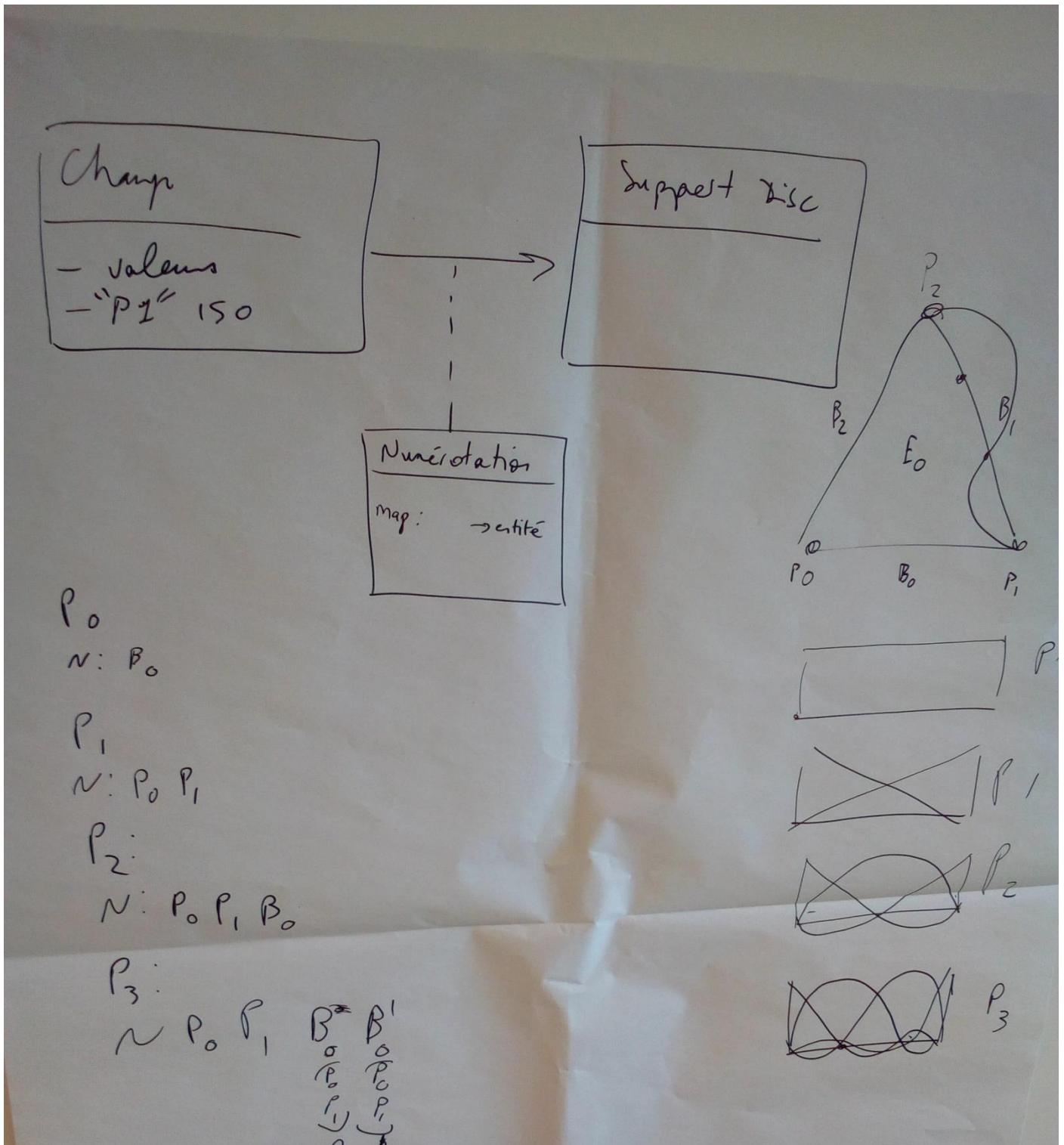
Une *variable d'état* peut se représenter comme un champ via un *espace d'approximation*, lequel s'appuie sur le *support discret*. Exemples d'espaces d'approximation : espace polynomiaux par éléments associées aux éléments finis de Lagrange, de Hermite...

A l'inverse, un champ se représente comme variable d'état par une méthode d'*interpolation* visant à définir un vecteur à partir d'opérations (éventuellement intégrales) sur les valeurs du champ :





Chaque entrée du vecteur est associée à un *support de ddl* via un *système de numérotation*. En général, un *support de ddl* est constitué d'une seule *entité géométrique*. Cependant, pour certain type de discrétisations (par exemple éléments finis sous-paramétriques), il peut être nécessaire de définir un *support de ddl* comme combinaisons d'entité géométriques pour avoir une définition univoque. Cf figure suivante, demander à Felipe Bordeu pour les détails :



### Conclusion :

Au final les discussions se sont éloignées du modèle de données « de première intention ». **C'est une bonne chose !** (le modèle initial voulait initier la discussion, surtout pas la brider)

Néanmoins, on retrouve des concepts similaires entre le **modèle initial** et les **concepts discutés lors de l'atelier** :

- *données du problème* (dans les deux) ;
- *champ / field* (dans les deux) ;
- *snapshot* (dans les deux) ;
- *support discret* vs *support\_snapshot* et *support\_indexation*.

On remarque également que l'atelier a prévu de faire émerger de nouveaux concepts pas prévus dans le modèle de données original : ceux de *domaine de définition*, de *modèle*, de *vecteur d'inconnues*, de *support de ddl*, de *points*...

**Il est donc nécessaire d'amender et enrichir le modèle de données original, et donc aussi le glossaire, avec les concepts introduits ou modifiés lors de l'atelier.**