Introduction à la Science des Données

Manet Vincent

Vincent Manet

Introduction à la Science des Données

21 juin 2019

“Insufficient data is not sufficient, Mr. Spock. You’re the Science Officer, you’re supposed to have sufficient data all the time.”
Captain Kirk, Star Trek : The Original Series, “The Immunity Syndrome”, cité dans GAITHER ET AL., *Gaither’s Dictionary of Scientific Quotations* (2007)

Table des matières

Ce document synthétise sous forme de texte à peu près autonome et compréhensible une formation déjà donnée à deux reprises en 2019. Il constitue une brève introduction à la science des données en lien avec la simulation numérique.

1 Positionnement de la problématique

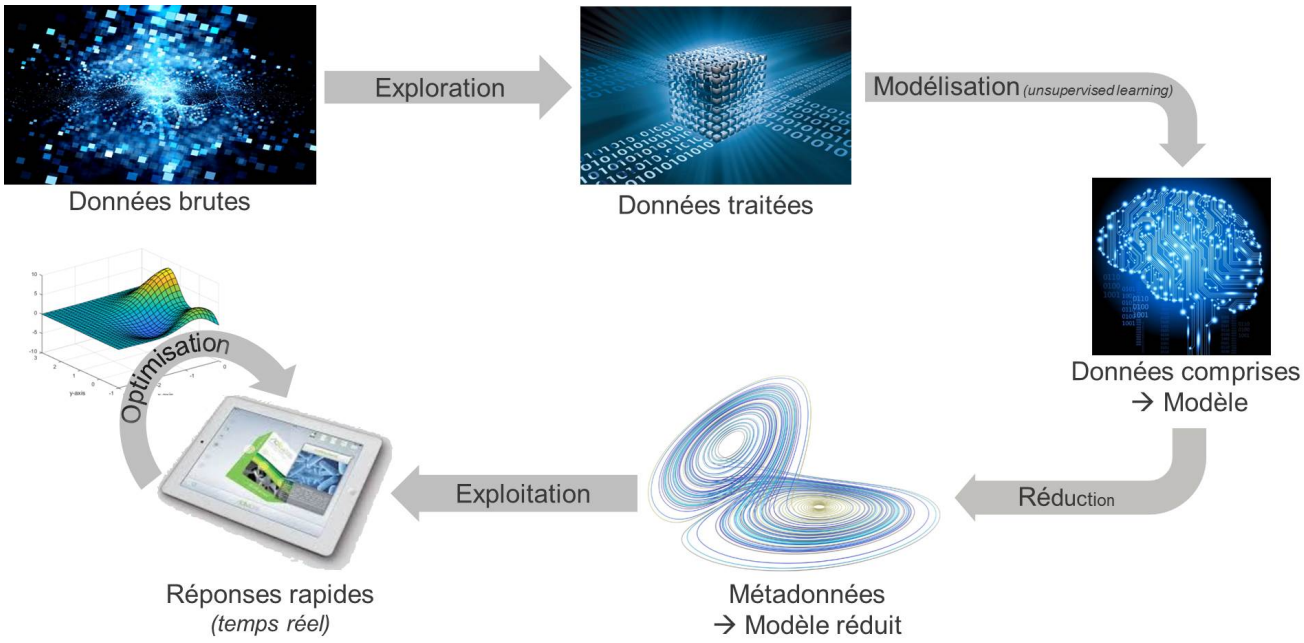

FIGURE 1: Vue d'ensemble

La figure 1 présente le contexte général dans lequel se place dans ce document d'introduction qui détaillera (un peu) les différentes étapes présentées :

- on dispose initialement de donnée brutes sur lesquelles on dispose de plus ou moins de connaissances ;
- une première étape consiste à explorer ces données, i.e. à essayer de mieux les appréhender, les connaître pour les transformer en données traitées, i.e. en données faisant sens pour nous ;
- on peut alors entreprendre de modéliser ces données pour les transformer en données comprises ce qui, pour nous, signifie que ces données peuvent être représentées par un modèle. Pour les gens du calcul, c'est souvent directement ici que le travail commence, par la création du modèle.
- Un modèle pouvant être volumineux, il peut être nécessaire de le réduire, i.e. de le décrire à l'aide d'un nombre réduit de variables que l'on appelle métadonnées et qui, elles, constituent les variables du modèle réduit ;
- enfin, disposant d'un modèle réduit, i.e. capable de fournir une bonne approximation du modèle initial dans un temps court, celui-ci peut être exploité à des fins d'optimisation par exemple, ou simplement pour répondre en temps réel dans d'autres applications (HIL par exemple).

2 Introduction à la science des données

Lorsque l'on a des données, quelle que soit leur provenance, on peut se poser 3 types de problèmes :

1. Améliorer et/ou approximer les données (le signal)
2. Estimer un modèle fonction des données [on dispose de toute une famille de points et l'on se demande où ils vivent]
3. Prédire des réponses à des questions [est-ce un chaton ?]

Le premier cas (améliorer et/ou approximer le signal) correspond au domaine appelé « traitement du signal ». La problématique rencontrée consiste à comprendre la régularité du signal x afin de pouvoir le traiter de manière parcimonieuse. Les applications sont le débruitage, les problèmes inverses, la compression...

Le second cas (estimer un modèle fonction des données) correspond au domaine de la « modélisation », ou encore, dans le contexte de la science des données à l'« apprentissage non supervisé ». Le problème à résoudre

est de trouver la probabilité $p(x)$ qu'un point appartienne ou non à l'ensemble des solutions. Si l'on dispose d'un modèle, l'application est encore une fois le traitement du signal. Si l'on ne dispose pas d'un modèle, alors les applications sont la synthèse de signaux, l'explication (physique statistique), la modélisation.

Dans ce second cas (modélisation), nous avons utilisé le terme probabilité. Il convient donc ici de faire une remarque importante sur l'utilisation d'un modèle probabiliste ou déterministe : il faut bien comprendre que cela n'a rien à voir avec le fait que le problème soit intrinsèquement aléatoire ou pas. Il s'agit ici du regard que l'on porte sur le problème, et plus particulièrement sur la manière dont on l'exprime :

— avec un modèle déterministe, on répond à la question : la donnée x est-elle dans le domaine considéré ? On cherche donc à savoir si une donnée x est dans un domaine ou non, mais l'on ne sait pas si elle est proche du domaine.

— avec un modèle probabiliste, on répond à la question : la donnée x située à cet endroit a-t-elle une forte probabilité d'être dans le domaine ? C'est donc la densité de probabilité de x, i.e. la probabilité que x soit dans un petit domaine déterminé que l'on cherche à connaître. Un tel modèle est donc plus raffiné.

Le troisième cas (prédire des réponses à des questions) correspond au domaine de l'« apprentissage supervisé ». Il s'agit cette fois de prédire la réponse $f(x)$ à la question que l'on se pose. Les applications sont extrêmement nombreuses : toute les perceptions (vision, reconnaissance de parole, de musique...), médical, sociologie, physique, neurophysiologie...

En termes de dimensions, les problèmes ne sont pas identiques :

— En traitement du signal, le signal x a une dimension comprise en 1 et 4 (espace-temps). On est donc en basse dimension.

— Pour les problématiques d'apprentissage (supervisé ou non), la dimension est celle du signal, i.e. des données. C'est donc le nombre d'échantillons ou d'exemples qui peut être grand, voire très grand. La dimension d est > 10 voire 10^6, 10^{24}...

3 Dilemme Biais-Complexité

Tous les problèmes précédents se mettent sous la forme :

$$\text{biais} + \text{variance}$$

où, d'une manière très générale, on peut dire que le terme de biais correspond à l'erreur de méthode (qui approxime le problème) et la variance aux fluctuations (des données).

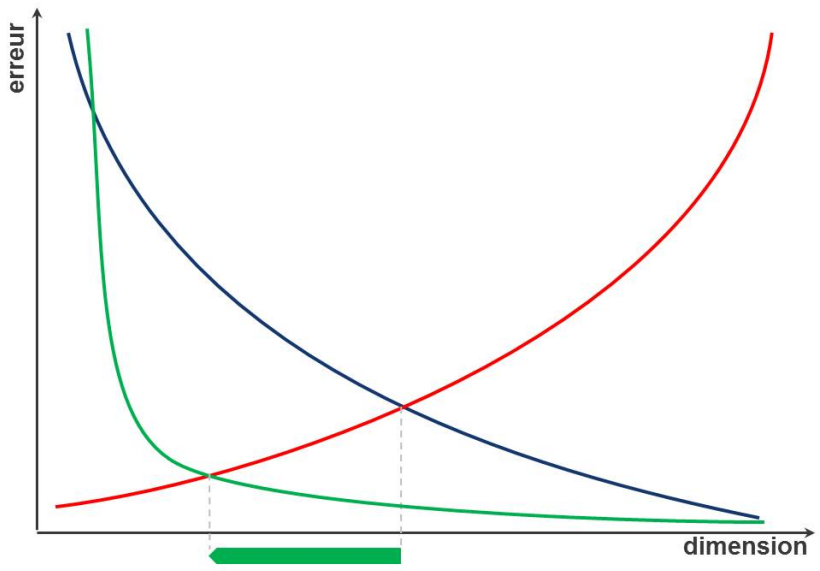

FIGURE 2: Dilemme Biais-Complexité

Graphiquement, cela se traduit par la courbe Erreur en fonction de la Dimension illustrée à la figure 2 où l'axe des abscisses représente la dimension (ou son log, ou toute mesure de la dimension) et l'axe des ordonnées représente l'erreur commise (ou son log ou toute mesure) :

- Plus on a de données, plus la fluctuation est importante... et on ne peut pas y faire grand-chose, sauf à travailler à la qualité des données (mais physiquement cela a un coût : par exemple cela peut correspondre au fait de réduire un intervalle de tolérance en fabrication...).
- Plus on a de données et plus le modèle (complet) est précis. En effet, on peut considérer que son erreur décroit avec sa dimension (penser à un modèle éléments finis).
- L'endroit où se coupent les courbes donnant la fluctuation (ou variance) et l'erreur de modèle (le biais) correspond à la dimension optimale du problème considéré : avec une dimension plus petite, on a moins de variance mais plus de biais, avec une dimension plus importante c'est le contraire.
- Par définition, un modèle réduit est un modèle qui, pour une précision donnée, fournit un résultat « en accord » avec le modèle complet mais à l'aide d'un nombre bien moins important de variables. Dans notre représentation graphique, cela signifie que son erreur diminue plus vite avec la dimension que celle du modèle complet.
- L'endroit où se coupent les courbes donnant la fluctuation et l'erreur du modèle réduit correspond à la dimension du problème considéré.

Ainsi l'on voit que l'intérêt de la RoM [1] est de réduire à la fois les effets de biais et de variance. C'est un intérêt considérable puisque l'on a gagné sur les deux tableaux, notamment sur celui de la variance qui est difficile à maîtriser.

Sur la signification des termes pour nos problèmes.
En traitement du signal, le biais est lié au choix de la base d'approximation du signal et à choix de la troncature de la base, et la variance au bruit des données.
En apprentissage, le biais est l'erreur de modèle, et la variance la fluctuation des données.

4 Méthodologie

Nous présentons succinctement ici les approches couramment mises en œuvre pour résoudre les problématiques soulevées par les différents problèmes présentés au paragraphe 2.

4.1 En traitement du signal

Nous avons mentionné qu'en traitement du signal, la problématique est de comprendre (ou supposer) la régularité du signal afin de l'approximer de manière optimale. Le but est donc de trouver une base optimale la plus réduite possible pour représenter le signal x traduisant la régularité du signal. En terme de programmation, on est ici sur de la programmation classique.

Pour procéder au choix d'une base de représentation, de nombreuses options sont possibles, parmi lesquelles les plus connues sont :

- Approche linéaire à pas fixe :
  - la plus connue de ces bases est la *base de Fourier* sur laquelle on ne s'étendra pas. Néanmoins, on peut faire quelques remarques pratiques :
    - La parcimonie de la base de Fourier correspond à une régularité classique au sens de Hölder-Sobolev du signal x ;
    - Les coefficients de Fourier traduisent la corrélation entre la sinusoïde considérée et le signal x ;
    - La vitesse de décroissance des coefficients de Fourier exprime la régularité du signal x.

1. Le terme RoM désigne la Reduction of Model

— une autre base connait un développement fort ces dernières années avec l'essor de l'ingénierie des données, bien que connue depuis très longtemps dans le champs de la statistique : il s'agit de la PCA [2]. Pour en faire une description simple, on peut dire que la PCA correspond aux directions principales de l'ellipsoïde (ou de l'hyper-ellipsoïde) qui contient tous les points considérés. Cette méthode fournit la base linéaire optimale.

— Approche adaptative :

— Linéaire à pas adaptatif : l'idée est de se dire que, tant qu'à considérer des points par lesquels faire passer la base d'approximation, autant placer ces points de manière optimale. En effet, imaginons une partie du signal linéaire, alors nul besoin de disposer de plus de 2 points pour décrire cette zone avec précision... On peut donc considérer, de manière parfaitement équivalente, que l'on place les points de manière optimale, ou que, partant d'une discrétisation fine à pas fixe, on retire les points qui n'apportent pas d'information.

Si l'on considère une sorte de base de Fourier à pas adaptatif, on obtient alors une *base en ondelettes*. Celle-ci dispose également de quelques intérêts pratiques que l'on peut mentionner :

— La régularité obtenue est une régularité α-Lipschitz, α étant déterminé par la méthode.

— On dispose par la même occasion d'un quantification du « taux » de discontinuité : $\alpha = 0$ échelon, $0 < \alpha < 1$ cusp, $\alpha > 1$ continue.

— Non-linéaire : il s'agit ici de méthodes dans lesquelles la base de représentation est construite (apprise) en même temps que l'on réduit. On citera ici les méthodes suivantes : RBF [3], tensorialisation, apprentissage...

Nous nous proposons d'illustrer cela maintenant à l'aide de polynômes des moindres carrées, et notamment la notion de sur-apprentissage et en quoi elle nuit à la généralisation.

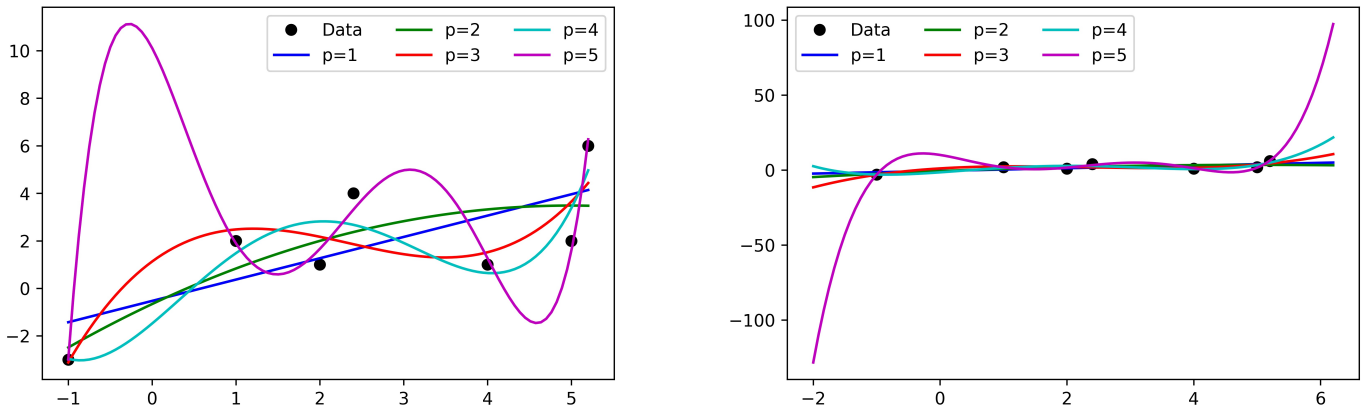

FIGURE 3: Interpolation et extrapolation : notion de sur-apprentissage

La figure 3 présente une série de points (disons de mesures) correspondant à un échantillonnage de notre signal x. Il s'agit pour nous de proposer une courbe censée représenter notre signal x « au-mieux ». Dans cet exemple, nous utilisons des polynômes des moindres carrés de différents degrés.

Il est assez évident qu'à mesure que le degré augmente, la courbe s'approche au plus près des points. En d'autres termes, l'apprentissage est de meilleure qualité... Toutefois, si l'on observe maintenant ce qu'il se passe en-dehors de la zone délimitée par les points (i.e. on fait de l'extrapolation, plus de l'interpolation), alors on voit que plus le degré augmente, plus le polynôme devient erratique... en d'autres termes, bien que l'apprentissage puisse être considéré comme meilleur concernant la capacité à interpoler, très vite la capacité à extrapoler (à généraliser) devient médiocre. On parle alors de sur-apprentissage, car finalement le modèle ne sait décrire que les données avec lesquelles il a appris.

2. Principal Component Analysis = Analyse en Composantes Principales
3. RBF = Radial Basis Function

4.2 En apprentissage non supervisé

En apprentissage non supervisé (modélisation), la problématique est d'obtenir une information sur la probabilité qu'à un point d'être solution ou non de notre problème. Le but est de trouver le modèle qui représente au mieux le problème avec le minimum de variables.

Encore une fois, on va s'intéresser à traduire la régularité de $p(x)$. C'est plus compliqué car nous sommes alors en grande dimension. Ainsi, en grande dimension, les moments statistiques d'ordre > 2 sont quasiment sans intérêt car la variance est monstrueuse : en effet, même si l'on dispose de beaucoup de points, ceux-ci ne représentent « presque rien » par rapport à tout l'espace, et l'on se retrouve dans le cas où tous les points sont très éloignés les uns des autres...

En terme de *programmation*, il est nécessaire d'aller vers des domaines de pointe : mémoire associative, mémoire distribuée, GPU... , ce qui est compliqué aussi bien théoriquement qu'informatiquement.

Un cas plus classique est celui où l'on dispose déjà d'un gros modèle (par exemple un modèle éléments finis comportant des millions de degrés de liberté). Le but reste bien de trouver une base optimale dans laquelle le modèle sera le plus réduit possible. On retombe ainsi sur la problématique évoquée au paragraphe précédent concernant le choix de la base (mais en dimension plus grande). Le problème est ici souvent bien mieux formulé et de nombreuses méthodes existent déjà depuis longtemps en fonction des problèmes considérés. Utiliser une base de vecteurs propres (correctement tronquée) est généralement une bonne idée, même si elle n'est pas forcément la base optimale.

4.3 En apprentissage supervisé

En apprentissage supervisé, on souhaite trouver la fonction $f(x)$ qui prédit correctement la réponse, et ce avec le minimum de variables.

Une fois encore, on fera en sorte que le modèle mis en place vise à traduire la régularité de la fonction $f(x)$, car celle-ci permet de définir des classes de fonctions candidates. On parlera ici de classifieur (à noyau...), de régularisation, de convexification (SVM [4]) En termes de *programmation*, on est ici sur ce que l'on appelle l'intelligence artificielle, le machine learning, qui est d'accès assez facile aujourd'hui avec les librairies python.

5 Apprentissage non supervisé (modélisation)

Revenons maintenant sur le cas le plus complexe qui est celui de l'apprentissage non supervisé, et reprenons-le dans le cadre le plus général, i.e. à partir de la donnée. Cette démarche est illustrée à la figure 4.

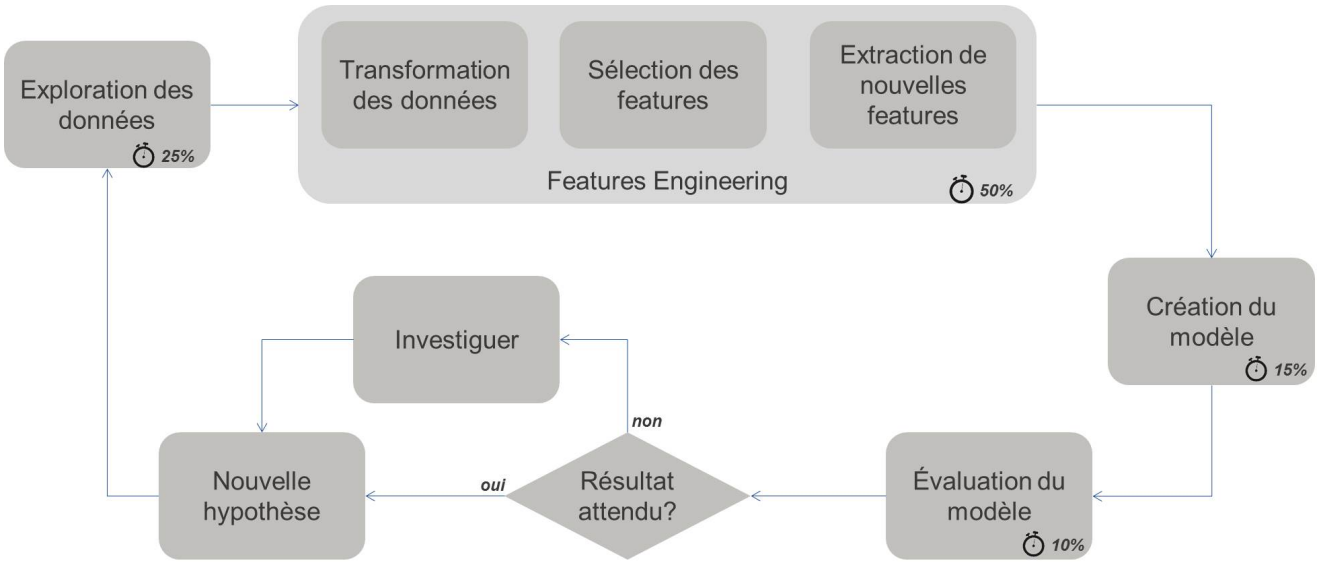

FIGURE 4: Vue générale de la démarche d'apprentissage non supervisé

4. SVM = Support Vector Machine

Détaillons chaque pas de la méthode :

— Exploration des données : il s'agit de comprendre ce que représente chaque variable. Pour ce faire on va se poser de multiples questions et utiliser de nombreux tests...
  — sait-on ce que représente la variable ?
  — analyse de la variable cible (les données sont-elles équilibrées ou pas...)
  — type des variables explicatives : discrètes, ordinales (discrète avec notion d'ordre : petit/moyen/grand), continues
  — analyse des variables explicatives : visualiser, afficher les distributions...
  — traitement des variables manquantes : que faire en cas de données manquantes ? Supprimer toute la donnée, laisser les « trous », compléter avec une valeur (0, NaN, la moyenne...) ?
  — détection des variables aberrantes : et d'abord, comment définir une variable aberrante ?
  — débruitage : les données sont-elles bruitées ou non ? Est-il intéressant de procéder à un débruitage ?
  — corrélation entre les variables
  — importance relatives des variables entre-elles
  — ...
— Features Engineering : c'est l'étape la plus importante, aussi ne faut-il pas hésiter à y passer du temps.
  Le but est de créer de nouvelles variables explicatives pour aider nos algorithmes. Par exemple, disposant des positions d'un objet dans le temps, il est possible de calculer sa vitesse, une information qui peut être précieuse selon les cas.
  Ce jeu très intéressant et subtil (et compliqué) consiste finalement en l'extraction des informations pertinentes et cachées et en la présélection de variables explicatives.
— Création du modèle : c'est la mise au point de l'algorithme lui-même. Comme on le voit sur la figure, cette tâche est très loin d'être la plus chronophage contrairement à ce que l'on pourrait croire. En général, on peut s'appuyer sur des bibliothèques qui facilitent grandement la tâche.
— Évaluation du modèle : l'étape précédente nous ayant fourni un modèle, il s'agit de vérifier que celui-ci fournit de « bonnes réponses ».
  Pour cela, on va le tester sur des données disponibles. C'est pourquoi les données doivent toujours être divisées en au moins un ensemble d'entrainement et un ensemble de validation : l'ensemble d'entrainement correspond aux données qui sont utilisées pour entrainer le modèle, i.e. pour le développer ; l'ensemble de validation à des données non utilisées pour construire le modèle et fournissant des exemples sur lesquels tester la capacité de notre modèle à généraliser, i.e. à prédire correctement des situations non apprises.
  Diviser les données permet de voir immédiatement si l'on n'a pas fait de sur-apprentissage sur les données d'entrainement, au détriment donc de la capacité de généralisation du modèle sur le nouvelles données (les données test).
  Il existe différentes stratégies pour utiliser au mieux les données disponibles, surtout si celles-ci sont peu nombreuses. Nous n'entrons pas ici dans ce genre de détail qui relève d'un cours sur l'apprentissage.
— Une boucle permet enfin de savoir si la mission est terminée ou non.

Remarque. Pourquoi passer du temps sur les données, notamment dans les deux premières phases ? il « suffirait » de tout mettre dans un réseau de neurones et d'attendre le résultat... Sauf qu'en pratique on ne le fait pas : pourquoi faire découvrir à grand renfort de calculs une variable explicative que l'on peut intuiter et calculer simplement au lieu de s'intéresser à découvrir mieux en quoi et comment elle intervient ?

Le but de ce paragraphe est d'insister sur le fait que la qualité de la modélisation ne repose pas sur le modèle lui-même mais sur la qualité des données (qu'elles soient les données initiales ou celles issues de l'étape de features engineering).

Espérer construire un modèle de qualité sur des données de mauvaise qualité n'est pas raisonnable.
S'il faut passer du temps, c'est sur la qualification des données et de leur qualité, avant tout autre chose.
En bref : les données, les données et encore les données... c'est la matière première dont nous nous servirons pour construire le modèle. Autant qu'elles soient de qualité !

6 Réduction de Modèle (RoM)

Au paragraphe 3, nous avons démontré l'intérêt de la RoM, seule manière de gagner à la fois sur le biais et sur la variance. Nous allons maintenant donner quelques compléments sur cette brique fondamentale du calcul numérique.

Les techniques de réduction de modèle sont utilisées dans de nombreux cas :

— La réduction de modèle est nécessaire chaque fois qu'il est compliqué et/ou long d'obtenir un résultat à partir du modèle complet. Cela peut être le cas pour des modèles dits « haute-fidélité », qui sont des modèles raffinés en 3D typiquement utilisés en mécanique des milieux continus (solides ou fluides) ; mais également dans le cas de modèles « simples » (0D-1D ou modèles nodaux) nécessitant un très grand nombre d'équations non-linéaires.

— Certains calculs imbriquent des boucles dont certaines peuvent être coûteuses. Les remplacer par un modèle réduit est alors très avantageux.

— Disposer d'un modèle moins coûteux permet également de réaliser de multiples analyses, ce qui est nécessaire lorsque l'on veut optimiser une solution.

— Un modèle réduit permet par nature d'obtenir des résultats très vite, ce qui est indispensable lorsque l'on vise des applications en temps réel.

— Quant à réduire un modèle, autant le faire dépendre de plusieurs paramètres et ainsi faire en sorte de disposer d'un modèle réduit paramétré... on peut alors l'interroger à loisir en faisant varier les paramètres et obtenir immédiatement la réponse correspondante. On dispose ainsi d'un modèle ayant une portée accrue et dont le coût de calcul est réalisé en amont de son utilisation.

Comme mentionné, la construction de ces modèles plus légers nécessite du temps de calcul, mais que celui-ci est réalisé off-line. Une fois le modèle réduit disponible, celui-ci réagit rapidement pendant la phase d'exploitation dite on-line.

Notons qu'il est tout à fait possible de mixer les approches et méthodes de réduction (mais comme nous n'avons pas encore donné de noms de méthodes...) Disons que l'on peut par exemple choisir de séparer les approximation d'espace et de temps, de séparer les coordonnées spatiales entre elles, de mixer ou de séparer les physiques, de gérer par des méthodes différentes les variables et les paramètres...

Concernant la réduction de modèle, on distingue [5] les approches suivantes :

— les méthodes de réduction de modèle ;

— les modèles de substitution ;

que nous allons brièvement détailler aux paragraphes suivants.

6.1 Méthodes de réduction de modèle

Les méthodes de réduction de modèle sont basées sur la simplification des hypothèses et la projection sur des bases réduites.

Les méthodes basées sur la simplification des hypothèses dépendent par essence fortement du problème à traiter.

On pourra citer les méthodes suivantes classiquement utilisée en mécanique du solide et implémentées dans de nombreux outils industriels : élément fini de poutre 1D sans cisaillement (Euler-Bernouilli) ou avec cisaillement (Timoshenko), élément fini de plaque mince (Kirchhoff-Love) ou épaisse, i.e. avec cisaillement (Midlin, Reissner), homogénéisation, utilisation de la base modale (pour les problèmes harmoniques, mais pas seulement), technique de réduction (super-élément : Guyan, Craig et Bampton, Mc Neal...), techniques de post-traitement...

Les méthodes de projection les plus connues sont : le développement de Karhunen-Loève, la SVD (Singular Value Decomposition), la PCA (Principal Component Analysis), la POD (Proper Orthogonal Decomposition)... elles sont peu ou proue identiques (mais portent des noms différents car redécouvertes dans des domaines différents).

5. on distingue ou l'on ne distingue pas les deux... la terminologie est floue. De toutes façons, un modèle de substitution est généralement un modèle réduit, donc une méthode de construction de modèle de substitution est bien une méthode de réduction de modèle.

Ces méthodes à postériori visent à construire une base réduite (ROB) sur laquelle on viendra projeter le problème. Cette base réduite est construite à partir de « snapshots », i.e. de résultats de calculs effectués avec le modèle complets (ou de résultats d'essais ou les deux).

Afin d'accélérer encore le calcul, il peut être intéressant d'identifier les parties linéaires (sous-espaces de Krylov) et non-linéaires, soit par la connaissance que l'on a du problème à traiter, soit par des techniques de type clustering.

Il existe de nombreuses autres méthodes de réduction parmi lesquelles certaines nous semblent assez intéressantes et que nous citerons :

— La DEIM (Discrete Empirical Interpolation Method) qui utilise la séparation de variables (comme la PGD) pour approximer le vecteur après intégration de la forme faible ; alors que la EIM construit une approximation de l'intégrande.

— La APHR (A Priori Hyper Reduction) qui utilise un domaine d'intégration tronqué.

— La APR (A Priori Reduction) qui essaye d'adapter et de corriger le modèle réduit de manière itérative. Il est possible d'enrichir la base à mesure que de nouvelles données sont disponibles.

— La PGD (Proper Generalized Decomposition) dont nous reparlerons au paragraphe 6.3

6.2 Modèles de substitution

Ces méthodes ne nécessitent pas de disposer d'un modèle numérique du problème et s'appuient uniquement sur les données (calculs et/ou mesures) disponibles.

Elles se proposent de construire une surface de réponse statistique de « grandeurs d'intérêt » (i.e. tout ou partie des données).

Par construction, ces méthodes tolèrent l'enrichissement de la base de données.

Nous citerons quelques méthodes très classiques :

— Méthodes d'interpolation telles que le krigeage.

Le krigeage est une méthode de régression par processus gaussien conditionné par les observations.

L'idée est de considérer les valeurs calculées comme ayant une composante déterministe et une composante aléatoire (et non plus uniquement déterministe)

Le krigeage peut être très intéressant dans le cadre de l'optimisation car il permet une stratégie adaptative d'amélioration du métamodèle (enrichissement). En effet, la variance générée peut donner des indications :

  — sur la zone à enrichir pour diminuer au maximum l'erreur au sens des moindres carrés ;

  — sur la zone où la probabilité d'avoir le minimum est la plus forte (Expected Improvement criterion).

Cette approche stochastique du métamodèle généré permet la prise en compte de données de qualité diverses. On parle alors de krigeage multi-fidélité.

— Modèles de régression tels que la PCE (Polynomial Chaos Expansion), les fonctions de base radiale...

Le chaos polynomial de dimension M et d'ordre p est défini comme l'ensemble des polynômes d'Hermite multidimensionnels en les M variables aléatoires gaussiennes centrées réduites de degré inférieur ou égale à p.

Lorsque l'on résout un problème stochastique, on peut arriver à exprimer la solution comme série de polynômes homogènes de variables gaussiennes centrées réduites qui forment une base, mais pas orthogonale au sens du produit scalaire défini par la moyenne. Effectuer un changement de base pour passer dans la base du chaos polynomial permet de retrouver cette orthogonalité et par suite d'obtenir des expressions simplifiées de moments statistiques d'ordre quelconque qui sont obtenus alors pour un coût numérique quasiment nul (expression analytique). Voir [15] Ch 22.

— Réseaux neuronaux (machine learning)

On rappelle que la performance d'un réseau tient essentiellement à sa structure.

D'une manière générale, à ressource identique (nombre de poids à trouver), on préfèrera un réseau moins large et plus profond au contraire.

Lorsque l'on a besoin d'un effet mémoire (modélisation de phénomène d'hystérésis...), on s'orientera vers un réseau récurrent (dans lequel les premières couches sont réinjectées plus loin) comme par exemple le LSTM (Long Short-Term Memory). Ce dernier, connu depuis 1997, est bien documenté et disponible dans des librairies.

6.3 La PGD

La PGD (Proper Generalized Decomposition) est basée sur l'hypothèse de séparation des variables (tensorialisation). Notons que séparation des variables ne veut pas forcément dire séparation de toutes les variables (tensorialisation complète). On peut très bien effectuer une séparation entre variables d'espaces d'un côté et de temps de l'autre, on peut séparer une variables d'espace des deux autres pour obtenir une théorie de plaque... bref tous les mixes sont possibles.

On ne fait pas de distinction entre variable et paramètre. Les paramètres sont ajoutés comme « coordonnées » supplémentaires. Ces paramètres, comme les variables, peuvent être de tout type, comme par exemple des conditions aux limites, des paramètres géométriques, de process...

La PGD est un algorithme glouton. Cela est appréciable car des « modes » ne sont ajoutés que tant que cela est jugé nécessaire. Par contre, ce « jugement » nécessite une mesure, typiquement un indicateur d'erreur (résidus). On peut être assez basique et suivre l'intérêt (l'apport) d'un nouveau mode. On peut aussi, selon les cas, construire des indicateurs plus fins, notamment liés à la physique que l'on cherche à capturer. On utilisera préférentiellement la méthode CRE (Constitutive Relation Error) [6].

Contrairement aux méthodes à postériori, la PGD est une méthode à priori. Cela signifie que la base dans laquelle le problème sera projeté n'est pas connue à l'avance (et donc encore moins imposée), mais qu'elle est construite au fur et à mesure par enrichissement. Il faut noter que chaque pas d'enrichissement nécessite de résoudre un problème non-linéaire.

Ainsi, la solution u de notre problème, dépendant de D variables et paramètres (nommés x_1 à x_D) est cherchée sous la forme (séparation de toutes les variables ici) :

$$u(x_1,...,x_D) = \sum_{i=1}^{N} X_i^1(x_1)X_i^2(x_2)...X_i^D(x_D) = \sum_{i=1}^{N}\prod_{j=1}^{D} X_i^j(x_j)$$

où N est le nombre de modes... sachant que celui-ci n'est pas une donnée de l'utilisateur mais est obtenu par l'algorithme glouton. Augmenter la précision demandée (i.e. diminuer le ε utilisé dans le critère de précision) permet généralement d'augmenter le nombre de modes requis.

Si l'on veut entrer un peu plus dans le détail de l'algorithme, alors on peut mentionner qu'une seconde boucle est utilisée pour approximer au mieux un mode donné. Ainsi, le calcul du mode n se fait-il à partir des $n-1$ modes connus jusque-là et en itérant p fois (avec là-aussi un critère d'arrêt basé sur la nécessiter de continuer à enrichir) selon le schéma :

$$u^{n,p} = \sum_{i=1}^{n-1}\prod_{j=1}^{D} X_i^j(x_j) + \prod_{j=1}^{D} X_{n,p}^j(x_j) = u^{n-1} + \prod_{j=1}^{D} X_{n,p}^j(x_j)$$

On en déduit que l'algorithme nécessite un « point de départ », i.e. un point d'initialisation. Celui-ci a peu d'importance : il peut être pris nul, égale au premier vecteur calculé, tiré au hasard...

Cela veut dire également qu'il est possible (mais est-ce souhaitable ?) de lancer la PGD à partir d'une base déjà existante que l'on souhaiterait conserver pour diverses raisons (par exemple pour sa signification physique connue...) et enrichir.

Les intérêts de la PGD sont les suivants :

— elle permet de réaliser un modèle réduit paramétrique. En effet, comme mentionné, on peut décider d'introduire tous les paramètres que l'on souhaite. La solution obtenue dépend alors explicitement de ceux-ci.

C'est une différence majeure avec d'autres techniques de réduction qui visent à obtenir le modèle réduit le plus compact possible, quitte à ce que celui-ci ne dépendent pas de certains paramètres ou ne dépende que de méta-paramètres sans forcément de signification physique. Malgré tout le tableau n'est pas si noir pour ces méthodes puisqu'elle procèdent en 2 étapes : la première transforme le modèle initial (qui lui dépend de tous

les paramètres physique dont on peut avoir besoin) en le modèle réduit ; quant à la second, elle fait le chemin inverse et permet de remonter in fine aux variables physiques. La PGD ne fonctionne pas comme cela : elle décrit le modèle réduit uniquement à l'aide des paramètres choisis, quitte à ce que tous les paramètres influents ne soient pas pris en compte ou que des paramètres non-influents soient considérés.

— elle permet d'avoir un modèle réduit dynamique. C'est souvent ainsi que l'on nomme un modèle réduit qui dépend du temps... il suffit de dire que le temps est l'un des paramètres du problème pour l'incorporer à la PGD.

— D'un point de vue pratique, la PGD fonctionne même avec des EDP non-linéaires couplées... et même, dirais-je, dans des cas où elle devrait être mise en défaut. On constate (ce qui n'est pas une preuve) qu'elle n'est mise en défaut que pour des non-linéarités vraiment fortes, ce qui en fait une méthode assez robuste.

— Dans le cas multi-physique et multi-échelle, on peut améliorer la PGD en remplaçant son solveur par un solveur LaTIn (on parle de LaTIn-PGD dans la littérature).

— D'autres variantes de la PGD existent, par exemple pour exploiter la sparsité des matrices...

— Pour aller au-delà du disque de convergence du développement, on peut adjoindre à la PGD des méthodes de continuation.

Dans la présentation que nous avons faite de la PGD, on peut remarquer que c'est la solution du problème à traiter qui est discrétisée sous une forme particulière. Cette solution peut donc être obtenue comme on le souhaite : soit à partir des équations de la physique (c'est souvent comme cela que c'est présenté dans la littérature), soit à partir d'une base de données (de vecteurs solution, i.e. de snapshots), soit à partir de solveurs adaptés au problème. Cette versatilité dans l'usage de la PGD est illustrée à la figure 5.

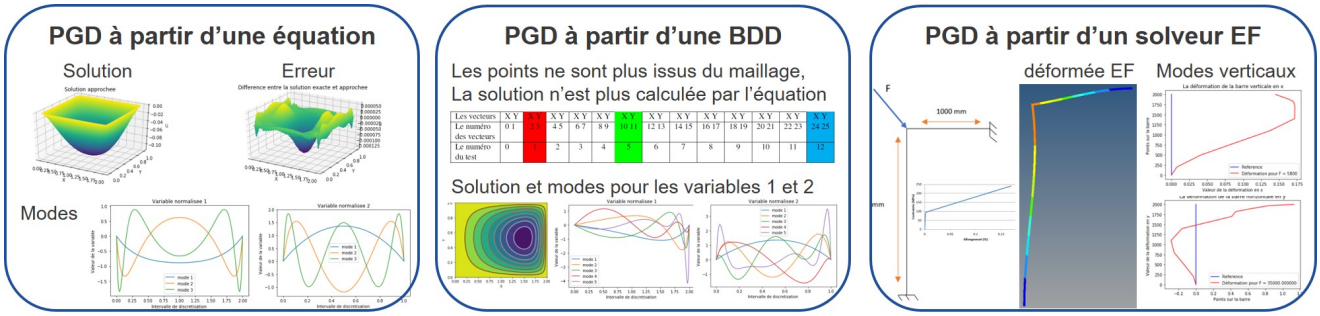

FIGURE 5: Exemples de cas d'usage d'une PGD

On peut noter que certains industriels commencent à regarder cette technique pour l'intégrer au cœur de leurs solveurs (comme c'est le cas pour Siemens avec Samcef pour les calculs non-linéaires) et diminuer ainsi les temps de calcul.

6.4 Trucs & astuces en numérique

Nous en profitons ici pour glisser quelques compléments « numériques » qui peuvent être utiles et/ou sont utilisés dans les méthodes présentées ou pour en étendre la portée.

Lorsque l'on parle de calcul numérique, il ne faut pas négliger le travail à réaliser sur le préconditionnement des systèmes à résoudre. Un préconditionnement simple peut par exemple rendre un système symétrique, ce qui a de nombreux avantages. On n'oubliera pas la rank-revealing factorization.

On pourra se servir de la structure algébrique des équations à traiter pour effectuer des découplages et changements de variables adéquats (comme cela est fait par exemple dans le cadre des équations différentielles algébriques)...

Tous les algorithmes qui arrivent à préserver la structure des matrices mises en jeu (lorsque celle-ci est intéressante évidemment), sont à regarder avec intérêt.

Le truc du noyau

Pour les espaces de Hilbert à noyau reproduisant (espace de Hilbert de fonctions pour lequel toutes les applications sont des formes linéaires continues), on dispose d'un principe (kernel trick) qui confère un caractère non-linéaire à nombre de méthodes originellement linéaires :

— L'idée est de projeter le problème non-linéaire dans un espace plus grand qu'initialement, jusqu'à ce qu'il s'exprime linéairement dans cet espace plus grand.

— On procède ensuite à une réduction de dimension sur ce modèle linéaire.

D'un point de vue pratique :

— Si le problème s'exprime uniquement en fonction de produits scalaires des observations, alors il suffit de remplacer chacun de ces produits scalaires par un noyau non-linéaire.

— Ainsi la structure des algorithmes demeure-t-elle inchangée et le surcoût calculatoire dû à l'évaluation des noyaux négligeable.

On peut toujours trouver un espace plus grand pour linéariser notre problème... quitte à ce que celui-ci soit de dimension infinie. On utilisera alors le mot clef « Opérateur de Koopman [6] » pour en apprendre plus.

Dans la pratique, les méthodes « étendues » par l'utilisation du truck du noyau (kernel trick) sont dénommées k-X où X est le nom de la méthode initiale. On parle ainsi de k-PCA, de k-PGD...

Non-linéarité d'ordre élevé

D'une manière générale, il est plus difficile d'exploiter des exposants élevés dans les méthodes numériques. On aime bien se retreindre au degré 2 (surtout si l'on veut travailler sur des notions de convexité).

Une technique qui fonctionne bien est d'introduire des variables complémentaires afin d'abaisser les degrés mis en jeux dans les équations.
Ainsi, plutôt que de gérer $u = x^3$, on posera une nouvelle variable $v = x^2$ et on aura $u = vx$ qui est de degré 2.

Lifting : transformations algébriques

Cette manière de gérer les degrés élevés nous amène tout naturellement à des techniques assez anciennes mais que l'on appelle parfois « lifting » dans la littérature contemporaine, notamment en MDO [4].

L'idée est non seulement de se limiter aux polynômes de degré 2, mais également d'exprimer des fonctions sous forme de tels polynômes... en les décrivant par leurs propriétés algébriques (équations fonctionnelles).

Exemple : si l'on a une équation de type $f(u) = a\sin(\Omega t)$ alors on peut la transformer en le système : $\begin{cases} f(u) = z \\ z'' + \Omega^2 z = 0 \end{cases}$ faisant ainsi « disparaître » la fonction sinus pour la remplacer par une équation polynomiale en les variables (qui viennent d'être augmentées de la variable z). Un tel système présente des avantages en termes de résolution numérique.

7 MDA-MDO

À ce stade du jeu, comme indiqué sur la figure 1, nous avons traité les données pour réaliser un modèle (dit modèle complet) que nous avons réduit. Nous disposons donc d'un modèle réduit capable de nous fournir rapidement une réponse précise... il ne nous reste plus qu'à l'exploiter. Dans ce paragraphe, on s'intéresse au cas où des modèles réduits (méta-modèles) sont utilisé pour réaliser une optimisation multi-disciplinaire.

6. et il est bien linéaire car il appartient à l'espace dual. Il est en effet défini comme le dual de l'opérateur de Perron-Frobenius.

7.1 Analyse du problème

Avant même de pouvoir envisager explorer l'espace de conception à des fins d'optimisation (MDO), il convient de commencer par réaliser une analyse du problème (MDA). Cette dernière consiste à identifier les disciplines mises en jeu, leurs couplages, les variables de conception et à déterminer les objectifs et contraintes à prendre en compte.

Afin de définir la stratégie la plus pertinente, il convient de réaliser cette analyse avec suffisamment de précision :

- Quel est le niveau de détail nécessaire pour chaque discipline ? Est-il le même sur l'intégralité du système ? Combien de phénomènes doit-on inclure ? Quelles sont les entrées et sorties ?...
- Quel type de dépendance existe-t-il entre les disciplines et les données : quelles sont les données globales et locales ? la portée des degrés de liberté ? Y a-t-il des sous-séquences de calcul (voire d'optimisation) ?...
- Quelles sont les variables de couplage ? Sont-elles toutes globales ? Appartiennent-elles au même espace pour les disciplines qu'elles couplent ?...
- Les variables de conception sont-elles continues ou discrètes ? macroscopiques ou locales ? dépendantes de la qualité de la représentation de la géométrie ?...
- Les objectifs nécessitent-ils d'être optimisés simultanément, ou certains peuvent-ils être optimisés séparément ?...

De plus, on peut distinguer deux contextes d'optimisation : un contexte déterministe abordé au paragraphe 7.2 et un contexte probabiliste ou stochastique dont nous parlerons au paragraphe 7.3 ; ainsi que deux grandes familles d'algorithmes d'optimisation (avec et sans gradient) qui seront détaillées au paragraphe 7.2.3.

7.2 Optimisation en contexte déterministe

Ce paragraphe se base essentiellement sur [5] complété par [11].

7.2.1 Formalisation du problème

De manière générale, le problème à traiter peut se formuler de la manière suivante :

$$\begin{cases} \text{Minimiser la fonction } f(x,y,z) \text{ par rapport à } x \text{ de sorte que :} \\ g(x,y,z) \leq 0 \\ h(x,y,z) = 0 \\ \text{et vérifiant :} \\ \forall i, \forall j \neq i, \qquad y_i = \left\{ c_{ji}(x_j, y_j, z_j) \right\} \\ \forall i, \qquad R_i(x_i, y_i, z_i) = 0 \end{cases} \tag{1}$$

où x, y et z désignent les variables de conception, de couplage et d'état respectivement :

- *les variables de conception* x sont les degrés de liberté que l'on s'accorde et sur lesquels on joue pour trouver l'optimum. Elles peuvent intervenir dans un ou plusieurs sous-systèmes.
- *les variables de couplage* y sont utilisées pour coupler les différents sous-systèmes et pour évaluer la consistance de la solution par rapport aux couplages.
- *les variables d'état* z sont constitutives des disciplines considérées. Attention, elles peuvent varier pendant l'analyse, car elles sont liées par exemple à l'état d'équilibre de la discipline concernée. Ainsi, ces variables n'entrent dans l'optimisation que comme variables couplées aux variables de conception x dont elles dépendent, généralement de manière implicite. Une boucle d'optimisation locale peut ainsi être requise afin de déterminer ces variables d'état au sein d'une discipline donnée pour une valeur donnée des variables de conception.

et où les autres termes représentent :

- $f(x,y,z)$: la fonction objectif à optimiser (minimiser) ;
- $g(x,y,z)$ et $h(x,y,z)$: des fonctions traduisant des contraintes sur les variables de conception. Ces contraintes se présentent sous la forme d'inégalités (fonction g) ou d'égalités (fonction h). Ces contraintes sont souvent regroupées au sein d'un unique vecteur (fonction vectorielle) ;

— c_{ij} représente le couplage entre les sous-systèmes i et j ;

— $R_i(x_i, y_i, z_i)$ est le résidu du système i. Selon la stratégie choisie, 1) le sous-système peut résoudre complètement l'équation relative aux résidus, ou 2) simplement les évaluer, ou alors 3) une analyse multidisciplinaire peut être réalisée qui résout les équations relatives aux couplages et aux résidus.

Dans ce qui précède, les quantités sont vectorielles et on pose : $x \in \mathscr{X}$ où le domaine $\mathscr{X} \subset \mathbb{R}^n$, $y \in \mathbb{R}^c$, $z \in \mathbb{R}^s$, $f \in \mathbb{R}^d$, $g \in \mathbb{R}^i$ et $h \in \mathbb{R}^e$, soit en d'autres termes n le nombre de variables de conception, c le nombre de couplages, s le nombre de variables d'état, d la dimension de l'optimisation, i et e le nombre de contraintes d'inégalité et d'égalité.

Une manière plus compacte de présenter ça est de considérer $x = (x, y, z)$ (i.e. on concatène les vecteurs x, y et z dans le vecteur x), $g = (g, h)$ (i.e. on concatène les fonctions vectorielles g et h dans g), ce qui conduit à :

$$\begin{cases} \text{Minimiser la fonction } f(x) \text{ par rapport à } x \text{ de sorte que :} \\ g(x) \leq 0 \\ \text{et vérifiant } x \in \mathscr{X} \end{cases} \tag{2}$$

les relations de couplage et sur les résidus étant contenues de manière implicites dans le fait que x doit rester sur la variété solution $\mathscr{X}$.

Remarque. On rappelle qu'il peut être intéressant de coupler un problème de minimisation avec un problème de pénalisation : cela permet de minimiser le nombre de coefficient non-nuls et de pouvoir sélectionner directement les coefficients les plus importants et influents pour la minimisation de l'erreur.

7.2.2 Architectures

On distingue essentiellement deux types de méthodes pour attaquer ce problème d'optimisation :

— les méthodes à un niveau d'optimisation : MDF (MultiDiscipline Feasible), IDF (Individual Discipline Feasible), AAO (All At Once)...

  — Dans la méthode MDF, la variable d'état x est considérée comme une fonction des variables de conception z pour des couplages c donnés.

  — Dans la méthode IDF, la variable d'état x est considérée comme une fonction des variables de conception z et de couplage y.

  — Dans la méthode AAO, on utilise la formulation générale.

— les méthodes à plusieurs niveaux d'optimisation : CO (Collaborative Optimization), BLISS (Bi-Level Integrated System Synthesis), CSSO (Concurrent Sub-Space Optimization), ATC (Analytical Target Cascading)...

  — Dans la méthode CO, l'optimisation principale porte sur $f(x, z)$ et on évalue les contraintes de couplages sous forme de problème adjoint. Pour chaque sous-système on minimise le problème adjoint et les contraintes sur g et h.

  — Dans la méthode BLISS, on optimise dans la boucle MDA et on vérifie la convergence. Une étude de sensibilité permet la mise à jour des variables pour la boucle suivante.

Les deux cas extrêmes et opposés suivants sont bien évidemment à éviter :

— optimiser tout avec tous les couplages possibles (approche MDF) : on génère alors un espace énorme dans lequel il peut être très difficile de converger, surtout s'il faut calculer des gradients [7] ;

— optimiser chaque sous-système séparément et essayer de les coupler à postériori : chaque discipline génère des effets de bords qui doivent être absorbés par les autres disciplines, généralement au détriment de la solution globale.

7. Avec les nouveaux algorithmes en provenance du big data, cette solution pourrait ne plus être hors de portée...

Il n'existe pas de « meilleure méthode » dans l'absolu. La méthode la plus adaptée est fonction du problème et notamment de la dimension de l'espace de conception, du nombre de couplages, des objectifs, de la disponibilité ou non des dérivées/sensibilités...

Tout mettre dans un seul niveau d'optimisation ou imbriquer les boucles, tel est le dilemme. Il reste bon de se rappeler que l'on peut toujours se ramener à un seul niveau d'optimisation... même si ce n'est pas forcément optimal d'un point de vue calculatoire.

Deux plateformes sont utilisées et développées sur Toulouse :

— OpenMDAO : plateforme libre développée par la NASA initialement. L'ONERA et l'ISAE sont des acteurs majeurs de son développement. La plateforme est basée sur une architecture MDF. L'optimisation est basée sur le gradient via l'« automatic computation of total derivatives accross multidisciplinary models ».

— MDA-MDO : plateforme développée à l'IRT Saint-Exupéry. Le cœur (GEMS) passera en Open Source en 2020. L'architecture est de type BLISS. Pour l'optimisation, l'option gradient-free est préférée. Airbus et très impliqué, l'ONERA également.

Nous présentons à la figure 6 l'architecture de la plateforme MDA-MDO de l'IRT afin de fixer un peu le vocabulaire. On voit bien sur cette figure que les algorithmes d'optimisation et la mise à disposition de méta-

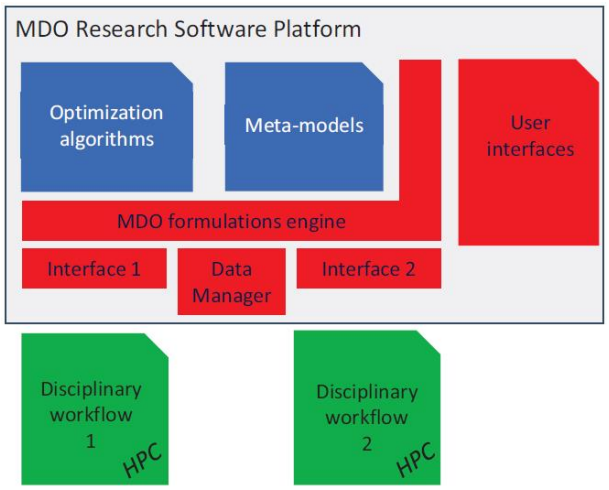

FIGURE 6: Architecture de la plateforme MDA-MDO de l'IRT

modèles font partie des briques essentielles. Les stratégies possibles seront détaillées dans les paragraphes suivants. Notons que sur la plateforme de l'IRT, ces méthodes ne sont pas figées dans l'outil.

Toujours sur cette figure apparaissent des « disciplinary workflow » qui correspondent à des worflows déjà existants liés à des problématiques métier (par exemple CFD ou mécanique [8]). Ces worflows peuvent également faire appel à des outils divers couplés entre eux (au sens chaînage numérique), à de l'optimisation... et ils peuvent être importés comme une brique à part entière avec laquelle on dialoguera (échanges par les entrées et/ou sorties). Notons enfin que rien n'empêche ces workflows de déjà mettre en œuvre des physiques couplées.

On pourra donc traiter des problèmes localement couplés dans les workflows, mais on pourra également les considérer au travers de méta-modèles. On se référera à [16] pour voir un exemple de méta-modèle (modèle réduit par PGD) correspondant à un problème d'interaction fluide/structure sous couplage fort.

L'un des intérêts de la plateforme MDA-MDO réside dans l'analyse des couplages à réaliser : une analyse des variables en entrée et en sortie permet d'identifier les couplages, qui, assemblés sous forme de graphe, sont simplifiés afin de réduire la portée des boucles nécessaires. Ce point est très intéressant et apporte un réel plus.

8. Pour une problématique structurelle, un worflow métier pourrait correspondre à un équipement (par exemple un échangeur) tel qu'il peut ou pourrait être considéré de manière globale : définition du faisceau, homogénéisation du faisceau, assemblage des divers éléments du faisceau, simulation de l'emboutissage des boîtes, simulation de la soudure des boîtes, assemblage complet du faisceau dont on s'intéressera au comportement statique et dynamique. Dans une telle approche, le workflow consiste à chaîner des solveurs et à faire passer les champs d'intérêt (déformations et contraintes résiduelles) entre eux.

7.2.3 Algorithmes d'optimisation

Les algorithmes d'optimisation sont l'un des principaux ingrédients de la MDO. Une bonne architecture permet normalement de pouvoir intégrer les algorithmes nécessaires au sujet à traiter. Certaines architectures permettent un choix automatique de l'algorithme adapté, quitte même à ce que ce choix évolue au cours du traitement du problème (par exemple pi-Seven).

On distingue classiquement deux grandes familles d'algorithmes :

— gradient-based : ce sont les algorithmes les plus classiques d'optimisation. Ils différencient la fonction objectif et les contraintes afin d'ajuster les variables, ce qui implique une certaine régularité de ces fonctions.

Les méthodes basées sur le gradient se servent de celui-ci pour déterminer la direction dans laquelle aller chercher un point fournissant une meilleure valeur de la fonction objectif. Évidemment, ce type d'approche a ses limites, par exemple si la fonction possède plusieurs minima locaux, est non-convexe ou si le domaine de recherche n'est pas continu.

Quelques pièges classiques sont les minima locaux, les minima en fond d'une vallée étroite... et de nombreuses variantes aux algorithmes de gradient linéaires ou quadratiques existent, comme par exemple l'algorithme BFGS (Broyden-Fletcher-Goldfarb-Shanno), l'algorithme des faisceaux (subgradient method), l'algorithme de Frank-Wolfe, la méthode des plans sécants...

— gradient-free : ils présentent un vrai intérêt dans le cadre de la MDO qui utilise généralement une plateforme généraliste permettant de faire appel à tout type de codes, dont certains n'ont pas été prévus pour fournir des informations de dérivées ou de sensibilités.

De plus, ces algorithmes permettent également de traiter le cas de fonctions non différentiables (liées à certains phénomènes physiques, donc impossible à éviter selon les cas à traiter), non-convexes, ou définies sur des domaines non continus. On peut également les mettre en œuvre si le coût d'obtention du gradient est élevé.

Les algorithmes les plus répandus dans cette famille sont les algorithmes génétiques, l'algorithme de Nelder et Mead, le recuit simulé, cobyla et associés, les essaims particulaires (particle swarm), la SPSA (Simultaneous Perturbation Stochastic Approximation) et par suite tous les algorithmes probabilistes (randomized algorithms) qui peuvent être incroyablement performants (mais non dénués de limitations).

Nous avons cité quelques noms d'algorithmes, nous allons très brièvement en détailler certains.

Algorithmes à gradient

— Méthode du gradient ou algorithme de la plus forte pente : l'idée est bien évidemment de se servir du gradient pour trouver la direction de recherche optimale. Toutefois, il faut noter que ces méthodes de gradient convergent souvent très lentement, et leur principal intérêt est théorique. On passera souvent sur des méthodes de type gradient stochastique surtout si l'on est en grande dimension. D'autres raffinements sont évidemment possibles...

— Méthode du gradient conjugué : il s'applique au cas particulier de systèmes linéaires (même très grands) décrits par une matrice symétrique et définie-positive (comme en mécanique linéaire). On peut également tirer partie de la nature éparse de la matrice (comme en éléments finis). Enfin, une version non-linéaire existe.

— L'algorithme de Frank–Wolfe, ou méthode du gradient conditionnel, est un algorithme d'optimisation itératif du premier ordre applicable aux problèmes contraints convexes. L'idée est de considérer une approximation linéaire de la fonction objectif et de se déplacer vers un minimiseur de celle-ci.

Algorithmes sans gradient

— L'exploration aléatoire de l'espace est la méthode la plus basique. Des variantes existent qui consistent souvent à améliorer (réduire) le diamètre de l'hypersphère dans laquelle la recherche s'effectue. Le Cuckoo search, comme d'autre méthodes (Random walks), peuvent également se classer dans ces méthodes.

— L'optimisation Bayesienne est une stratégie de minimisation globale sans dérivée qui considère la fonction objectif comme inconnue, et l'approche comme étant une fonction aléatoire dont on cherche la distribution de probabilité. On initialise la recherche par une estimation initiale de cette distribution, avant toute autre connaissance sur celle-ci. Puis, grâce au théorème de Bayes on améliore cette estimation à mesure que des données (valeurs de la fonction) sont connues.

— La méthode de descente selon les axes (coordinate descent) consiste à considérer les variables les unes après les autres, une à la fois : on part de la première variable et on minimise la fonction par rapport à elle, puis, à partir de ce point, on fait varier la deuxième coordonnée...

— Les algorithmes génétiques utilisent des opérateurs de mutation, crossover et sélection. A partir d'une population initiale de solution générée aléatoirement, on définit la population suivante (génération suivante) à partir de la précédente à partir des individus les « meilleurs », i.e. ceux ayant la meilleure valeur de la fonction objectif : certains par sélection (ils sont meilleurs), d'autres par mutation (on les modifie aléatoirement), soit en échangeant une partie de leur génome (crossover)... et ainsi de suite. La mutation permet, on l'espère, de disposer d'individus suffisamment éloignés des autres pour échapper à un minimum local. Le problème de cette démarche est de définir un critère d'arrêt : une nouvelle génération est généralement « meilleure » que la précédente, mais l'on a pas de mesure absolue, uniquement un critère relatif entre les individus.

— Optimisation par essaims particulaires (Particle swarm optimization) : comme les algorithmes génétiques, cette méthode se base sur la nature et notamment sur la collaboration des individus entre eux. Comme pour les fourmis, l'idée est qu'un groupe d'individus peu intelligents peut posséder une organisation globale complexe. On initialise l'algorithme avec une population tirée au hasard. On laisse ensuite chaque particule évoluer en fonction de sa vitesse, de sa valeur de la fonction, de la meilleure valeur de la fonction dans son voisinage. Cette méthode fonctionne évidemment mieux pour des variables continues, mais peut converger vers des minima locaux.

— La méthode COBYLA (Constrained optimization by linear approximation) est une méthode itérative qui se propose, à chaque itération, de résoudre une approximation du problème par un problème de programmation linéaire. On obtient donc à chaque itération un candidat pour la solution optimale que l'on évalue sur la fonction objectif d'origine, ce qui donne le point suivant pour l'algorithme qui se termine lorsque l'écart entre deux itérations est petit.

— Les méthodes de sous-gradient (Subgradient method) permettent de traiter des problèmes d'optimisation non-convexe. Ces méthodes convergent même lorsque le fonction objectif n'est pas différentiable (et lorsqu'elle l'est, les méthodes de sous-gradient dans le cas non contraint utilisent les mêmes directions de recherche que la méthode de la plus grande descente). Des variantes et extensions existent.

Autres cas

On peut se dire que l'on aimerait bien utiliser une méthode à gradient, d'autant plus qu'un tel gradient existe peut-être pour le problème considéré, mais que l'on n'en dispose pas. Comment faire alors ?

Même si l'on ne dispose pas de gradients (parce que ce n'est pas prévu par les outils utilisés par exemple), il est possible de les obtenir par diverses méthodes. L'article de Hwang et Martins [10] en mentionne certaines dont la Modular Analysis and Unified Derivatives (MAUD) développée par de John Hwang en 2014, qui permet le calcul parallèle, et est implémentée dans la plateforme de la NASA OpenMDAO choisie par L'ONERA. On se méfiera quand même des algorithmes de différentiation automatique et on vérifiera avant de les utiliser qu'ils sont applicables aux fonctions que l'on considère...

Nous avons beaucoup insisté dans les premiers paragraphes sur la régularité du signal ou du modèle que l'on cherche à créer. Cette information, bien que physiquement connue, n'est pas toujours correctement introduite dans les algorithmes. Classiquement, on est face à deux cas :

— Cas 1 : on a plus de régularité qu'on ne le pense.

  Exemple : ce n'est pas parce que l'on s'intéresse à un système que l'on déclenche par un on/off que sa réponse n'est pas continue. Même si la sigmoïde qui la représente est raide, elle l'est toujours infiniment moins d'un Heaviside.

  Dans ce cas-là, il ne faut pas hésiter à construire un modèle continu puisque c'est la réalité. Avec un peu de chance, des informations de type gradient seront accessibles pour un coût raisonnable, et alors c'est gagné.

— Cas 2 : on n'a pas de régularité, mais ce n'est pas si grave.

Lorsque l'on a une variable discrète, on peut choisir d'approximer sa réponse par une fonction continue, sachant pertinemment que seules les valeurs en des points discrets auront une signification physique. Ce faisant, on dispose alors de la possibilité d'utiliser le gradient.

C'est une technique souvent utiliser que de créer une surface de réposne qui possède une régularité plus grande que celle du modèle physique. L'optimisation du système est alors faite sur ce modèle de substitution (SBO = Surrogate-based Optimization), avec des techniques de gradient si on le peut, et on termine en regardant les points physiquement admissibles autour de l'optimum calculé.

7.2.4 Dérivées

Au paragraphe précédent, nous avons évoqué les deux grandes familles d'algorithme : avec ou sans gradient. Nous sortons de ce cadre pour considérer ici le gradient sous un nouvel angle... plus exactement, en fonction d'où se situe l'optimisation à réaliser dans l'architecture MDO, il peut être intéressant de distinguer plusieurs « types de dérivées ».

Ainsi, du point de vue algorithmique, on distingue les *trois types suivants de dérivées* concernant les modèles de substitution :

— « Prediction derivatives » : ce sont les dérivées des sorties du modèle par rapport aux entrées correspondantes du modèle. Elles sont nécessaires pour réaliser une optimisation à gradient (gradient-based optimization) à partir du modèle réduit.

— « Training derivatives » : ce sont les dérivées des sorties d'entrainement du modèle par rapport aux entrées d'entrainement. Elles permettent d'ajouter des données d'entrainement supplémentaires afin d'accroître la précision du modèle réduit, si la méthode choisie le permet bien évidemment. Si de plus on utilise la méthode adjointe pour calculer ces dérivées, alors on peut obtenir un modèle réduit de grande qualité pour un coût très raisonnable puisque ces dérivées sont obtenues à un coût indépendant du nombre d'entrées.

— « Output derivatives » : ce sont les dérivées des sorties du modèles (prédictions) par rapport aux données de sorties utilisées pour l'entrainement. Elles interviennent lorsque le modèle réduit est réactualisé ou reconstruit pendant les itérations de l'optimisation à gradient.

7.3 Optimisation en contexte incertain

Ce paragraphe se base essentiellement sur [9, 12, 1].

7.3.1 Analyse du problème

Il existe bien des manières de considérer l'aspect stochastique d'un problème. En termes d'objectifs déjà, selon que l'on cherche à déterminer la probabilité globale de défaillance, à déterminer la loi de cette probabilité de défaillance, à avoir des indices sur la sensibilité aux paramètres, à connaître les bornes dans lesquelles la solution restera, ou à étudier l'évolution du comportement en fonction de perturbations, les méthodes mises en œuvre diffèrent radicalement. Il convient également de se demander si les variabilités envisagées doivent être traitées au niveau global ou si elles peuvent rester locales. Ainsi on commencera toujours par se demander si l'on souhaite seulement tester la robustesse de la solution obtenue face à des variations stochastiques des paramètres ou si l'on souhaite réellement réaliser une optimisation stochastique.

Commençons par quelques définitions :

— Robustesse : un système est dit robuste si ses performances varient peu lorsqu'il est soumis à des variations. Réduire la variation des performances augmente la robustesse ;

— Fiabilité : capacité d'un système à remplir les fonctions requises dans les conditions spécifiées pendant une période donnée. Déplacer la performance moyenne permet d'augmenter la fiabilité ;

— Fiabilité et robustesse : en adaptant la moyenne et l'écart-type de la performance, on peut améliorer à la fois la fiabilité et la robustesse.

Les méthodes d'optimisation en contexte incertain peuvent se classer en quatre familles :

— Optimisation dans le cas pire (WCO) (Worst Case Optimization) : il s'agit d'étudier les incertitudes autour des points de conception de sorte que celles-ci n'induisent aucune violation des contraintes du problème. L'optimum du cas pire est donc moins optimal que l'optimum déterministe : il garantit qu'aucun cas n'est à danger, et peut-être vu comme de la sur-qualité ;

— Optimisation robuste (RDO) (Robust Design Optimization) : d'après la définition donnée de la robustesse, on en déduit que cette méthode recherche une solution au problème d'optimisation dont la variance est minimale ;

— Optimisation fiable (RBDO) (Reliability Based Design Optimization) : repose sur le calcul de la probabilité de défaillance. Le point de conception (à trouver) doit être tel que l'application de la densité de probabilité des variables d'entrée à ce point ne viole les contraintes qu'avec une probabilité inférieure ou égale à la probabilité de défaillance cible ;

— Optimisation robuste et fiable (RBRDO) (Reliability Based Robust Design Optimization) : combinaison de robustesse et de fiabilité. Cela signifie que l'on veut à la fois être suffisamment éloigné de l'état limite des contraintes (fiabilité), tout en ayant une faible variabilité (robustesse).

Nous ré-insistons encore une fois sur le fait que de telles approches ne sont pas envisageables avec des modèles complexes (3D ou haute-fidélité) et qu'il est alors indispensable de disposer de modèles réduits.

7.3.2 Formalisation du problème

On ne s'intéressa pas ici à l'optimisation WCO, mais aux trois autres cas (RDO, RBDO, RDRBO) qui font intervenir des notions de probabilités et/ou de moments.

Considérons le problème général tel qu'il a été formulé en contexte déterministe à la relation (1). En présence de paramètres incertains ξ (environnementaux, matériaux, géométriques...), les fonctions objectif f et de contraintes g et h dépendent désormais également de ces paramètres ξ. On remplace alors $f(x,y,z,\xi)$, $g(x,y,z,\xi)$ et $h(x,y,z,\xi)$ par des mesures de robustesse et de fiabilité $\rho_f(x,y,z)$, $\rho_g(x,y,z)$ et $\rho_h(x,y,z)$ respectivement. Le problème d'optimisation stochastique s'écrit alors :

$$\left\{\begin{array}{l} \text{Minimiser } \rho_f(x,y,z) \text{ par rapport à } x \text{ de sorte que :} \\ \rho_g(x,y,z) \leq 0 \\ \rho_h(x,y,z) = 0 \\ \text{et vérifiant :} \\ \forall i, \forall j \neq i, \qquad y_i = \left\{c_{ji}(x_j,y_j,z_j)\right\} \text{ (plus ou moins exactement ou en probabilité)} \\ \forall i, \qquad R_i(x_i,y_i,z_i) = 0 \end{array}\right. \tag{3}$$

où les quantités ont été explicitées précédemment, et où $\rho_f \in \mathbb{R}^d$, $\rho_g \in \mathbb{R}^i$ et $\rho_h \in \mathbb{R}^e$.

On peut là-aussi réécrire cette formulation sous forme compacte :

$$\left\{\begin{array}{l} \text{Minimiser la fonction } \rho_f(x) \text{ par rapport à } x \text{ de sorte que :} \\ \rho_g(x) \leq 0 \\ \text{et vérifiant } x \in \mathscr{X} \end{array}\right. \tag{4}$$

Les formulations pour ρ_f et ρ_g sont nombreuses. Toutefois, on se sert souvent des moments statistiques (robustesse). Cela conduit alors, pour ρ_f au problème :

$$\left\{\begin{array}{rl} \text{Moyenne} & \rho_f = \mathbb{E}_\xi\left[f(x,\xi)\right] \\ \text{Variance} & \rho_f = \mathbb{V}_\xi\left[f(x,\xi)\right] \\ \text{Optimisation} & \rho_f = \min_\xi\left[f(x,\xi)\right] \text{ ou } \max_\xi\left[f(x,\xi)\right] \\ \text{Quantile} & \rho_f = q^p_\xi\left[f(x,\xi)\right], p \in [0,1] \end{array}\right. \tag{5}$$

Notons que le calcul précis de ces quantités (intégrales) est un vrai problème que l'on résout souvent en approximant ces mesures statistiques par la méthode de Monte Carlo sur des modèles de substitution.

7.3.3 Architectures et Algorithmes

Nous présentons ici très brièvement les approches classiques utilisées pour les différents types d'optimisation en contexte incertain.

Analyse de sensibilité

Que l'on se place dans le contexte de l'optimisation ou hors de ce contexte, il peut être intéressant de simplement connaître la sensibilité d'une variable de sortie à une variable d'entrée. Cette information est donnée par les indices de Sobol.

Dans le cas d'un modèle ayant pour sortie la variable aléatoire Y correspondant aux variables aléatoires d'entrée X_i, le i^e indice de sensibilité de premier ordre est donné par :

$$S_i = \frac{\mathbb{V}[\mathbb{E}[Y|X_i]]}{\mathbb{V}[Y]} \tag{6}$$

Les indices de Sobol se généralisent aux ordres supérieurs (et quantifient alors la variance attribuée à l'interaction entre paramètres) ainsi qu'au cas de paramètres dépendants.

Dans le cas où le modèle est exprimé dans une base de chaos polynomial (voir plus bas), les indices de Sobol peuvent être déduits des PCE comme montré dans [2, 3].

Optimisation WCO

- la manière la plus directe d'aborder le problème est d'avoir deux boucles imbriquées : la boucle principale itère sur la fonction objectif, la seconde sur les contraintes stochastiques pour trouver le pire cas. Le temps de calcul est important pour une telle approche ;
- Dans l'approche WWCO (Worst-vertex WCO), seules les bornes de chaque variable dans l'ensemble d'incertitude sont évaluées. On détermine ensuite les directions dans lesquelles les valeurs des fonctions g et h augmentent et on suppose que le sommet situé dans ces directions correspond au pire cas. Dans [13] on trouvera un exposé plus complet sur les intervalles d'incertitude, mais dans un cadre plus complexe de SORA (voir plus bas RBDO)
- L'approche GWCO est la version gradient-based de la WCO. On utilise le gradient et un développement de Taylor au premier ordre pour estimer (par Cauchy-Schwarz) une majoration du pire cas. Bien que la plus rapide des trois méthodes, la majoration obtenue peut être hors du domaine d'incertitude.

La version WWCO présente donc un bon compromis entre rapidité et précision.

Optimisation RDO

Comme on cherche à minimiser la fonction objectif ainsi que ses moments (sa variance), on a un problème multi-objectifs :

- Approches mono-objectif : ces approches visent à adjoindre le second problème au premier pour transformer le problème en une optimisation mono-objectif. Les méthodes diffèrent par les poids qu'elles accordent à la moyenne et à l'écart-type ;
- Approches multi-objectif : elles traitent séparément la moyenne et l'écart-type (donc 2 objectifs) pour obtenir un front de Pareto :
  - la méthode « ε-contrainte » transforme le problème multi-objectif en une série de problèmes mono-objectif avec une contrainte supplémentaire. Cette méthode trouve toujours un front ;
  - la méthode « pondérée » combine les deux objectifs à l'aide d'un poids dont la valeur varie. Cette méthode ne fonctionne qu'avec un front convexe ;
  - la méthode réellement « multi-objectifs », plus lente que les précédentes.

La méthode « ε-contrainte » est généralement bien adaptée.

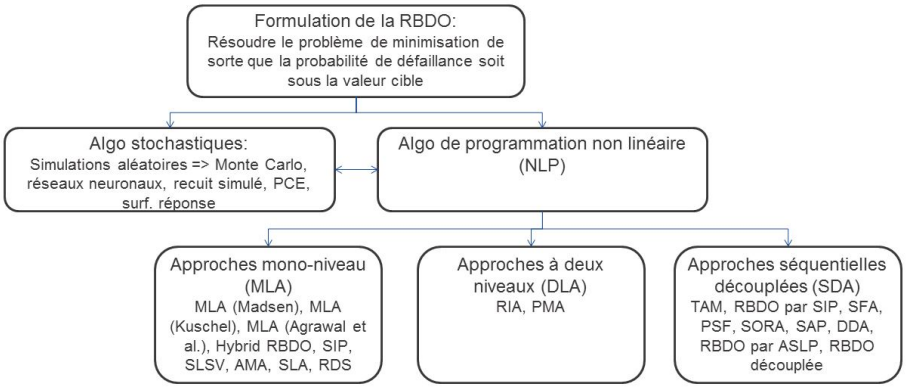

FIGURE 7: Méthodes RBDO

Optimisation RBDO

On rappelle que l'on cherche à atteindre une probabilité fixée. La figure 7 illustre des approches couramment utilisées en RBDO [9] :

— les algorithmes stochastiques :

— Monte Carlo : cette méthode consiste à réaliser les tirages nécessaires (à balayer l'espace) puis à faire des statistiques sur les résultats. Évidemment au cours du temps de nombreuses variantes ont été développées afin de mieux localiser les tirages, diminuer leur nombre, améliorer leur variance (échantillonnage préférentiel...). Elle reste la méthode de référence pour valider les autres méthodes mais nécessite alors un très grand nombre de tirages.

— PCE (Polynomial Chaos Expansion) : l'idée initiale telle qu'utilisée par Ghanem et Spanos était de décomposer la solution dans une base de polynômes orthogonaux pour des variables gaussiennes. Les polynômes auxquels on pense alors immédiatement sont les polynômes d'Hermite sous forme probabiliste... et c'est ce choix qui a été fait. Une présentation en est faite dans [15] §22.3.3.

L'avantage des PCE est qu'ils sont applicables à toute formulation, même non-linéaire. Leur inconvénient est leur taille qui augmente extrêmement vite, puisque la dimension de la base du chaos polynomial de dimension M (nombre de variables aléatoires) et d'ordre p (degré max du développement) est donnée par $\binom{M+p}{p}$, soit 10 pour M=3 et p=2, 56 pour M=5 et p=3, 210 pour M=6 et p=4.

L'utilisation des PCE permet de remonter aux indices de Sobol [2, 3], mais également aux moments statistiques de la réponse [15] §22.4, de sorte que l'on arrive à rentabiliser leur coût de calcul initial.

— Réseaux de neurones : nous n'entrerons pas dans le détail de ces algorithmes, bien trop nombreux. Toutefois, on trouve de nombreux exemples aussi bien en réduction de modèle qu'en optimisation stochastique, notamment dans des cas complexes (dynamique chaotique) et sur des variétés. Leur programmation est assez aisée avec les librairies actuellement disponibles. Ce sont par conséquent de très bons candidats, même s'ils nécessitent une taille importante de données.

— les algorithmes de programmation non-linéaire (NLP) :

— méthodes de « double-boucle » ou méthodes à deux niveaux (DLA) : la boucle principale (extérieure) vise à trouver la probabilité optimale en se basant sur les probabilités calculées par la boucle intérieure. Cette dernière (boucle intérieure) utilise généralement la formulation FORM (1er ordre) pour calculer

9. Quelques rappels d'abréviations :
FORM = First Order Reliability Method, SORM = Second Order Reliability Method, PCE = Polynomial Chaos Expansion
MLA = Mono-Level Approaches : SIP = Semi-Infinite Programming, SLSV = Single Loop Single Variable, AMA = Approximate Moments Approach, SLA = Single Loop Approach, RDS
DLA = Double-Level Approaches : RIA = Reliability Index Approach, PMA = Performance Measure Approach
SDA = Sequential Decoupled Approaches : TAM = Traditionnal Approximation Method, SFA = Safety Factor Approach, PSF = Probabilistic Sufficiency Factor, SORA = Sequential Optimization and Reliability Assessment, SAP = Sequential Approach Programming, DDA = Direct Decoupling Approach, ASLP = Adaptive Sequential Linear Programming

la probabilité de défaillance. [10]

Parmi les méthodes double-boucle, citons la RIA (reliability index approach) et la PMA (Performance Measure Approach). La RIA cherche le point de conception ayant la plus forte probabilité de défaillance dans la boucle intérieure, alors que la PMA cherche, pour un indice de fiabilité donné, le point ayant la meilleure performance (contraintes max).

— méthodes « simple boucle » ou méthodes « d'approximation » ou méthodes mono-niveau (MLA) : pour lesquelles la probabilité de défaillance est approximée et non plus calculée dans la boucle intérieure (qui n'est donc plus une boucle d'optimisation).

Parmi les méthodes simple boucle citons AMA (Approximate Moments Approach) et SLA (Single Loop Approach). La SLA évalue les contraintes à l'approximation du point de performance maximale alors que l'AMA utilise une approximation de premier ordre de la contrainte autour de la valeur moyenne.

— méthodes « séquentielles découplées » (SDA) : elles visent à changer le problème initial en une série de cycles d'optimisation. Chaque cycle séquentiel comprend une optimisation déterministe et une analyse de fiabilité.

En d'autres termes, dans les approches SDA, le problème de la RBDO est transformé en plusieurs sous-problèmes d'optimisation déterministes résolus séquentiellement. Le lien entre la fiabilité cible et l'optimum déterministe est assuré par des approximations ou des analyses de fiabilité effectuées séparément, i.e. à l'extérieur de chaque sous-problème d'optimisation déterministe.

Parmi les méthodes séquentielles découplées, citons la méthode SORA (Sequential Optimization and Reliability Assessment) et la méthode SAP (Sequential Approximate Programming) :

— La méthode SORA, transforme le problème de la RBDO en plusieurs cycles d'optimisations déterministes reliées au point de performance minimal cible (Minimum Performance Target Point) obtenu par une analyse de fiabilité inverse. Cette analyse inverse étant séparée du processus déterministe, elle peut être faite à chaque convergence du cycle d'optimisation déterministe, permettant ainsi d'actualiser le point cible.

— La méthode SAP diffère de SORA en ce qu'elle utilise des approximations et est donc plus rapide que SORA, mais elle peut ne pas converger si le problème est complexe.

— Dans [13] on trouvera une approche visant à déterminer les intervalles d'incertitude dans le cadre de la méthode SORA.

La méthode SORA est généralement la méthode préférée (plus rapide que les méthodes double-boucle et plus précise que les méthodes simple boucle).

Notons que l'on parle ici de la phase d'optimisation globale de la MDO. D'autres approches existent dédiée à des « problématiques métier » qui peuvent être traitée localement dans les workflows (pseudo-inverse et réanalyse : [15] §13.6, formulations stochastiques aux frontières, utilisation d'éléments finis stochastiques : [15] §22.3, ...).

Optimisation RBRDO

On cherche à atteindre un objectif à la fois robuste (RDO) et fiable (RBDO). Comme pour la RDO, les approches sont mono- ou multi-objectifs :

— Approches mono-objectif : voir ci-dessus ;

— Approches multi-objectifs : en fonction des objectifs de conception, trois stratégies existent :

— Nominal-the-best : il s'agit de trouver la solution dont la valeur de l'objectif est la plus proche d'une valeur cible tout en minimisant l'écart-type de l'objectif.

— Smaller-the-better : il s'agit de trouver la solution dont la valeur de l'objectif minimise à la fois la moyenne et l'écart-type de l'objectif.

— Larger-the-better : il s'agit de trouver la solution dont la valeur de l'objectif maximise la moyenne et minimise l'écart-type.

10. Si la taille des données est vraiment grande, il faut passer par une phase dite de « construction de scénarii » et des algorithmes comme CPLEX, GLPK, Gurobi... Pour réduire la taille de l'ensemble des scénarii, on peut utiliser la méthode de Monte-Carlo, les méthodes d'inférence statistique, mais également la « Multistage portfolio optimization ».

En considérant un nombre fini de scénarii, les algorithmes linéaires stochastiques à deux niveaux peuvent être considérés comme de gros problèmes de programmation linéaire. Cette formulation est souvent appelée équivalent déterministe linéaire.

Remarques

Dans ce paragraphe, nous faisons quelques remarques sur les méthodes présentées.

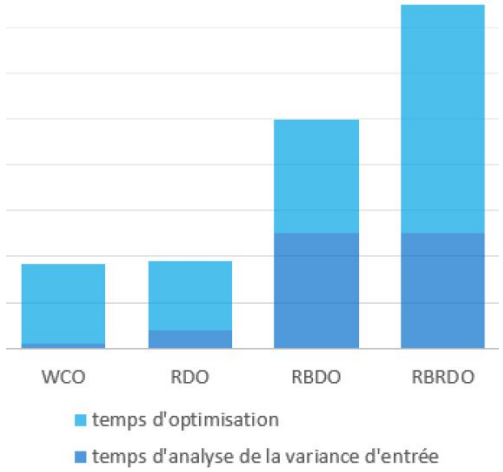

FIGURE 8: Comparaison des méthodes en terme de temps

La figure 8 compare les méthodes précédentes en termes de temps nécessaire. Évidemment, on trouve que WCO < RDO < RBDO < RBRDO.

Cette figure présente le temps d'optimisation et le temps d'analyse de la variance d'entrée, i.e. l'analyse des extrema, des moments statistiques ou des densités de probabilité qui sont nécessaires aux optimisations spécifiques.

Si l'échantillonnage de variables d'entrée du système prend du temps, on s'orientera vers les méthodes qui ne nécessitent pas un trop grand échantillon pour l'analyse des variances.

Toutefois, ces différentes méthodes ne répondent pas aux mêmes questions :

— WCO : cette approche est la plus simple et la plus conservative pouvant conduire à un surdimensionnement ;

— RDO : cette approche se focalise sur la stabilité de l'optimum (insensibilité aux incertitudes) ;

— RBDO : cette approche vise à obtenir une probabilité de défaillance donnée. Une telle approche n'est envisageable que si cela a du sens, i.e. si l'on travaille dans un contexte de grande série. Ainsi, cette méthode est adaptée à une conception en grande série à moindre coût avec un taux de défaillance faible.

— RDRBO : cette approche vise à obtenir une conception à la fois robuste et fiable.

Bien que l'approche RDRBO soit la plus générale, on se contente souvent d'une approche RBDO.

7.4 Optimisation multi-fidélité et Model Management

Ce paragraphe est basé sur [7].

Nous ne rentrons pas dans le détail, il s'agit ici de quelques remarques complémentaire par rapport à ce qui précède. Toutefois, il est important d'avoir en tête qu'il est possible de travailler avec des modèles de qualités différentes (modèle complet et plusieurs modèles réduits de qualités différentes les uns des autres), tout comme il est possible de travailler avec des données de qualités différentes (issues d'un modèle numérique fin et issues de mesures en quelques points par exemple).

La boucle externe d'optimisation, illustrée à la figure 9, peut être réalisée sur :

— le modèle complet (haute-fidélité) ;

— le modèle réduit (basse-fidélité) ;

— un mixe des deux (enrichissement du modèle réduit pendant l'optimisation). Cela est alors géré par le Model Management.

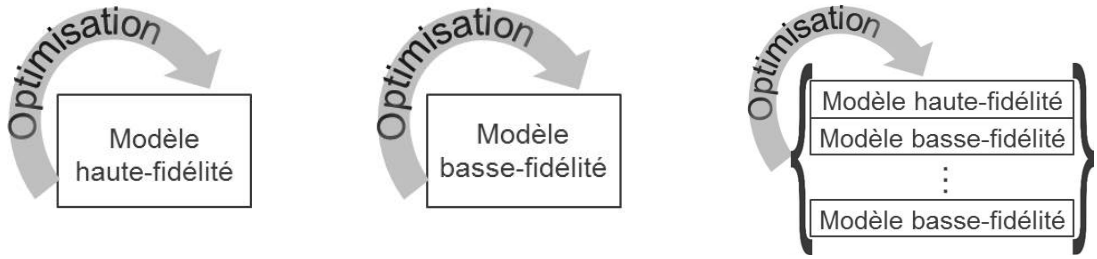

FIGURE 9: Management d'une optimisation multi-fidélité

On distingue 3 approches de **Model Management** :

— Adaptation : l'optimum est recherché à partir du modèle réduit (phase d'exploitation) qui est raffiné par rapport au modèle complet (phase d'exploration).

Un des algorithmes les plus connu dans ce contexte est l'algorithme EGO (Efficient Global Optim) où le modèle réduit est issu de krigeage, dont il existe des variantes (Super-EGO...) et qui est également développé à l'ONERA et à l'ISAE.

— Fusion : combinaison des résultats des modèles réduit et complet. Dans ce cas, tout ce joue évidemment au niveau de la stratégie de combinaison

— Filtrage : le modèle complet n'est utilisé que lorsque le modèle réduit n'est pas précis. On met en œuvre des techniques de tirage d'importance ou de région de confiance (approximation quadratique du krigeage).

Incertitudes

Elles sont introduites dans la fonction objectif haute-fidélité (ex : perfo + variance perfo) :

— Chaque itération d'optimisation inclut une boucle de *quantitifation d'incertitude* ;

— On peut remarquer que les évaluations des itérations précédentes forment un bon modèle réduit (ou utiliser un développement sur la base du chaos polynomial).

7.4.1 Apprentissage fédéré

L'optimisation « mixte » peut faire penser à l'apprentissage fédéré.

L'apprentissage fédéré correspond aux cas où l'on souhaite optimiser (ou apprendre) conjointement (i.e. entre divers partenaires) mais sans partager les données source ou les modèles (problèmes de confidentialité, de propriété intellectuelle...) :

— Chaque partenaire dispose de ses données et modèles, qui restent chez lui, inaccessibles aux autres ;

— Seules les informations nécessaires à l'algorithme d'optimisation ou d'apprentissage sont envoyées « au système », i.e. à la boucle générale d'optimisation. Ces informations ne sont pas les données (qui restent confidentielles), mais des grandeurs issues du traitement des données (la moyenne par exemple).

— Cela permet de réaliser la tâche globale...

Cette approche est actuellement utilisée dans les domaines de la défense et de la santé.

7.5 Ce qui n'a pas été abordé

Comme son titre l'indique, ce document n'est qu'un survol, et de nombreux points n'ont donc pas été abordés. Toutefois, nous souhaitions faire quelques remarques sur les points importants suivants :

— L'un des point importants non abordé concerne le contrôle de l'erreur que nous avons mentionné sous forme de résidu. Nous n'entrons pas en détail car cela peut être très lié aux physiques choisies et aux modes de couplage. En mécanique et particulièrement dans l'optique de la PGD, la CRE (l'Erreur en Relation de Comportement ou Constitutive Relation Error) est une méthode bien adaptée. Dans le contexte du machine learning, on parle plutôt de risque empirique et de fonction de perte (loss function).

— Nous avons également utilisé le raccourci « x doit rester sur la variété solution $\mathscr{X}$ ». Lorsque la variété en question a une topologie assez « gentille », les techniques classiques (comme l'espace euclidien tangent au point d'intérêt) sont satisfaisantes. De nombreux travaux portent sur l'optimisation sur des variétés Riemannienne, de Stiefeld, de Grassmann, d'Hadamard... et dans ce cas, on regardera la littérature sur le sujet pour trouver la méthode la plus adaptée (souvent à gradient, mais pas uniquement).

  Toutefois, dans la pratique, nos paramètres de conception nécessitent parfois d'approximer cette variété au-delà de cette gentille zone. Il convient alors de recourir à des méthodes plus complexes. On peut par exemple utiliser des approches algébriques pour imposer des contraintes, s'interroger sur la séparabilité des variables (tensorialisation), proposer différents ansatz selon les problèmes considérés, tenter une approche par variété centrale ou inertielle...

  Dans le cas de la dynamique non-linéaire, Wang et al.[14] réussissent à capturer la dynamique sur la variété inertielle grâce à un réseau de neurones récurrent (RNN) régularisé par un LSTM (long short-term memory) en exploitant des théorèmes de plongement de Whitney (1936) et Takens (1981) sur la dynamique complémentaire [11] (donc après avoir séparé la dynamique en deux parties orthogonales).

  Cette condition d'appartenance à une variété apparaît également dans le cas stochastique. Pour une formulation dans ce cadre-là, on peut se reporter à de récents travaux de Soize et Ghanem [8].

— Les aspects multi-échelles n'ont pas été abordés ici, et on les trouvera dans le document [16]. Généralement ils se situent au niveau métier, donc dans les sous-systèmes et les méthodes pour les traiter sont déjà bien établies. Dans le cadre de la PGD, les problèmes non-linéaires à résoudre peuvent se servir de la méthode LaTIn comme solveur, randant la méthode multi-physique et multi-échelle.

— Enfin, nous noterons également sur la figure 6 que le terme HPC apparaît. Nous n'avons pas abordé ici l'aspect scalabilité des algorithmes car cela nécessite de rentrer dans le détail. Le document [16] donne quelques éléments sur le sujet. Si des algorithmes dédiés peuvent toujours être optimisés voire réécrits pour profiter du parallélisme, une méthode simple et générale consiste à utiliser la décomposition de domaine pour réaliser la séparation des solveurs.

  Toujours dans [16], nous montrons comment l'utilisation de domaines fictifs permet à la fois d'assurer la convergence des sous-domaines et d'accélérer cette convergence. Du point de vue des calculs de structures, cette méthode nous semble particulièrement intéressante. Elle serait également adaptée au cas de la CFD.

8 Une alternative : la Décomposition de Domaine par Domaine Fictif

La décomposition de domaine a été mentionnée au paragraphe précédent, aussi allons-nous en dire quelques mots car elle est très intéressante. Nous nous placerons dans le cadre de la décomposition de domaine par domaine fictif qui nous semble encore plus prometteuse.

L'idée est de disposer d'« un modèle pour les gouverner tous » :

— Le modèle global est linéaire. Il garantit et améliore la convergence des domaines fictifs .

— Les domaines fictifs (patchs) correspondent à des modèles locaux et peuvent :

  — être de toute taille, forme...

  — faire intervenir n'importe quelle physique ;

  — être résolus par n'importe quelle méthode.

— On apportera du soin à la méthode de passage entre domaines : collocation, mortar (multiplicateur de Lagrange), LaTIn

Voici quelques exemple d'utilisation des domaines fictifs :

— Utilisation d'une méthode XFEM dans une zone locale afin d'y étudier la propagation de fissure ;

— Traitement d'un comportement local fortement non-linéaire ;

11. ce qui est une manière « classique » d'écrire les choses (somme directe) et surtout qui permet de gérer plus aisément le problème du contrôle de l'erreur (résidu). Pour la CFD, on se rappellera de la projection de Leray (pseudo-opérateur différentiel) qui permet de simplifier le champ de pression...

— Raccord entre type d'éléments (classiques avec NURBS...) ;

— Interaction fluide-structure (même forte et multi-échelle : fluide visqueux dans squelette souple...) ;

— Soudure...

8.1 Mise en œuvre des domaines fictifs

On considère la structure globale, dans laquelle on a identifié une ou des zones nécessitant un traitement particulier :

— On garde d'un côté l'ensemble de la structure (composée de la partie « simple » et de la partie nécessitant un traitement particulier)

— et on ajoute un « double » de la zone nécessitant un traitement particulier (patch).

L'intérêt de la méthode réside dans le fait que l'opérateur K défini sur la structure entière n'est pas modifié :

— Le maillage global reste inchangé (ainsi que la matrice de raideur globale), ce qui est particulièrement intéressant pour les structures de grande dimension ;

— Comportement local non-linéaire : on reste sur un solveur linéaire pour le modèle global (grande taille) et on utilise un solveur dédié (non-linéaire) uniquement pour le modèle local (petite taille) ;

— La prise en compte du modèle local est effectuée par une correction du modèle global appliquée sous forme de second membre (effort d'interface) : il est possible de coupler de manière non-intrusive plusieurs codes ou logiciels différents.

8.2 Algorithme de couplage non-intrusif

La décomposition de domaine permet donc un couplage non-intrusif de plusieurs codes et entre plusieurs zones. Toutefois, notons que la vitesse de convergence de ce couplage non-intrusif entre plusieurs zones dépend :

— de l'écart de rigidité entre les zones. Cela peut être contourné en augmentant la taille du domaine d'intérêt, mais cela alourdit en même temps le calcul ;

— ainsi que de la géométrie du patch.

Notons qu'il existe une formulation incrémentale mieux adaptée aux techniques d'accélération (méthode de relaxation...) et à l'étude de la convergence.

Si l'on a plusieurs patchs sur la même structure :

— L'échange d'information sur les interfaces entre le modèle global et chaque patch se fait de manière indépendante : il n'y a jamais d'échange entre deux patchs locaux (même s'ils partagent une interface commune) ;

— Si les maillages sur l'interface ne sont pas conformes, les informations globales (déplacement) sont transférées aux modèles locaux en utilisant l'opérateur de projection voulu (mortar, LaTIn...). Le maillage global (dont les mailles sont supposées plus grandes que celles des patchs) agit comme un filtre passe-bas sur les informations à l'interface (champ de déplacement d'interface).

Pour le couplage non-intrusif, chaque patch peut être traité de manière indépendante des autres, puisque seul le déplacement issu du modèle global à l'itération précédente est nécessaire à sa résolution, en d'autres termes il est possible d'effectuer de tels calculs locaux en parallèle [12].

12. L'utilisation « brutale » de la décomposition de domaine est souvent utilisée uniquement à des fins de parallélisation d'un calcul.

9 Conclusion

Nous terminons ici notre rapide survol de la science des données, qui sera à compléter par des cours plus pointus sur les différents domaines présentés.

Les données peuvent conduire aux modèles, mais les modèles existants peuvent s'enrichir des données. Cet aller-retour entre modèles et données nous semble particulièrement intéressant et important.

Pendant longtemps, la seule interaction entre modèles et données a consisté en ce que l'on appelle le recalage de modèle. Cela sous-entend que le modèle, correspondant à une discrétisation numérique de la réalité physique connue a travers ses lois, détenait la vérité et qu'éventuellement il convenait de recaler ça et là quelques petits paramètres afin que mesures et simulations coïncident.

Aujourd'hui nous en sommes bien au-delà. D'une part, on construit des modèles sur la seule base de données afin de modéliser des phénomènes pour lesquels on ne dispose pas de la physique (ou avec par exemple insuffisamment de précision). D'autre part, on peut enrichir des modèles afin d'étendre leur plage de validité (par exemple compléter un modèle fluide par des points de mesure dans des zones de pompage ce qui est très difficilement modélisable, surtout à des coûts raisonnables).

Nous avons essayé, à travers ce document, de dresser un panorama de cette symbiose entre science des données et simulation numérique.

10 Bibliographie

[1] Younes AOUES. « Reliability-based design and maintenance optimization of structures ». Thèse de doctorat. Université Blaise Pascal - Clermont-Ferrand II, jan. 2008. URL : https://tel.archives-ouvertes.fr/tel-00726003 (cf. page 20).

[2] Bruno SUDRET. « Global sensitivity analysis using polynomial chaos expansions ». In : *Reliability Engineering & System Safety* 93.7 (2008). Bayesian Networks in Dependability, pages 964 –979. ISSN : 0951-8320. DOI : https://doi.org/10.1016/j.ress.2007.04.002. URL : http://www.sciencedirect.com/science/article/pii/S0951832007001329 (cf. pages 22, 23).

[3] Géraud BLATMAN et Bruno SUDRET. « Adaptive sparse polynomial chaos expansion based on least angle regression ». In : *Journal of Computational Physics* 230.6 (2011), pages 2345 –2367. ISSN : 0021-9991. DOI : https://doi.org/10.1016/j.jcp.2010.12.021. URL : http://www.sciencedirect.com/science/article/pii/S0021999110006856 (cf. pages 22, 23).

[4] C. GU. « QLMOR : A Projection-Based Nonlinear Model Order Reduction Approach Using Quadratic-Linear Representation of Nonlinear Systems ». In : *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 30.9 (2011), pages 1307–1320. DOI : 10.1109/TCAD.2011.2142184 (cf. page 14).

[5] S. DEFOORT, M. BALESDENT, P. KLOTZ, P. SCHMOLLGRUBER, J. MORIO, J. HERMETZ, C. BLONDEAU, G. CARRIER et N. BÉREND. « Multidisciplinary Aerospace System Design : Principles, Issues and Onera Experience. » In : *AerospaceLab* 4 (mai 2012), pages 1–15. URL : https://hal.archives-ouvertes.fr/hal-01184311 (cf. page 15).

[6] Pierre LADEVÈZE et Ludovic CHAMOIN. « Toward garanteed PGD-reduced models ». In : *CIMNE*. Barcelona, Spain, juil. 2012. URL : https://www.researchgate.net/publication/295103983_TOWARD_GUARANTEED_PGD-REDUCED_MODELS (cf. page 12).

[7] Benjamin PEHERSTORFER, Karen WILLCOX et Max GUNZBURGER. « Survey of Multifidelity Methods in Uncertainty Propagation, Inference and Optimization ». In : *SIAM Review* 60.3 (2016), pages 550–591. DOI : 10.1137/16M1082469. URL : https://epubs.siam.org/doi/abs/10.1137/16M1082469 (cf. page 25).

[8] Christian SOIZE et Roger GHANEM. « Polynomial chaos representation of databases on manifolds ». In : *Journal of Computational Physics* 335 (2017), pages 201–221. DOI : 10.1016/j.jcp.2017.01.031. URL : https://hal-upec-upem.archives-ouvertes.fr/hal-01448413 (cf. page 27).

[9] Siyang DENG. « Methods for robust and reliability-based design optimization of electromagnetic devices ». Thèse de doctorat. Ecole Centrale de Lille, jan. 2018. URL : https://tel.archives-ouvertes.fr/tel-01905925 (cf. page 20).

[10] John T. HWANG et Joaquim R.R.A. MARTINS. « A Computational Architecture for Coupling Heterogeneous Numerical Models and Computing Coupled Derivatives ». In : *ACM Trans. Math. Softw.* 44.4 (juin 2018), 37 :1–37 :39. ISSN : 0098-3500. DOI : 10.1145/3182393. URL : mdolab.engin.umich.edu/sites/default/files/paper_mdolab_5.pdf (cf. page 19).

[11] Yves LE GUENNEC, Jean-Patrick BRUNET, Fatima Zohra DAIM, Ming CHAU et Yves TOURBIER. « A parametric and non-intrusive reduced order model of car crash simulation ». In : *Computer Methods in Applied Mechanics and Engineering* (2018). URL : https://hal.archives-ouvertes.fr/hal-01485276 (cf. page 15).

[12] Mickael RIVIER et Pietro Marco CONGEDO. *Surrogate-Assisted Bounding-Box Approach for Optimization Problems with Approximated Objectives*. Research Report RR-9155. Inria, fév. 2018, pages 1–35. URL : https://hal.inria.fr/hal-01713043 (cf. page 20).

[13] Zhong Yi WAN, Pantelis R. VLACHAS, Petros KOUMOUTSAKOS et Themistoklis P. SAPSIS. « Data-assisted reduced-order modeling of extreme events in complex dynamical systems ». working paper or preprint. Mar. 2018. URL : https://arxiv.org/abs/1803.03365 (cf. pages 22, 24).

[14] Lei WANG, Chuang XIONG, Juxi HU, Xiaojun WANG et Zhiping QIU. « Sequential multidisciplinary design optimization and reliability analysis under interval uncertainty ». In : *Aerospace Science and Technology* 80 (2018), pages 508 –519. ISSN : 1270-9638. DOI : https://doi.org/10.1016/j.ast.2018.07.029. URL : http://www.sciencedirect.com/science/article/pii/S127096381731903X (cf. page 27).

[15] Vincent MANET. *La méthode des éléments finis : vulgarisation des aspects mathématiques, illustration des capacités de la méthode*. URL : https://hal.archives-ouvertes.fr/cel-00763690/ (cf. pages 11, 23, 24).

[16] Vincent MANET. *Stratégie de calcul et Méthodes de réduction de modèles* (cf. pages 17, 27).