ENSEIRB-MATMECA
Ingeliance

Formation Ingénieur
Filière Mathématiques et Mécanique
Spécialité Calcul Haute Performance pour la Mécanique

Rapport de Stage de fin d'études
Du 01/02/2022 au 29/07/2022

Application des méthodes de Data Sciences aux Simulations Numériques

---

Mathias Truel
Promotion 2022

Tuteur école :
Nicolas Barral

Tuteurs entreprise :
Guillaume Ruiz
Gilles Vogt

Février - Juillet 2022

Remerciements : Je souhaite remercier mes tuteurs, Gilles Vogt et Guillaume Ruiz pour l'encadrement et la confiance qu'ils m'ont apportés pendant ce stage qui clôture ma formation d'ingénieur. Je remercie également toute l'équipe calcul d'Ingeliance pour m'avoir chaleureusement accueilli.

Table des matières

Table des figures

Liste des tableaux

1 Introduction

Déjà en 1950, Alan Turing écrivait sur "l'ordinateur et l'intelligence" [1]. Mais c'est seulement depuis les années 2000 que le *machine learning* a vraiment été démocratisé. La croissance de la puissance de calcul et l'explosion de la quantité de données disponibles a permis à ces algorithmes de devenir toujours plus performants, au point de dépasser l'homme dans des domaines précis. Ces intelligences artificielles peuvent aujourd'hui assister un chirurgien [2] ou battre le meilleur joueur mondial de Go [3].

Ainsi, le domaine de la simulation numérique ne fait pas exception. Aujourd'hui, de nombreux chercheurs et entreprises cherchent à coupler les deux approches. C'est le cas d'Ingeliance qui a la volonté d'intégrer ces nouveaux outils dans ses activités. Ce stage de fin d'étude que j'ai effectué de février à juillet 2022 y participe.

1.1 Ingeliance

Ingeliance propose des services à des grands groupes dans divers domaines, notamment le spatial, l'aéronautique, le nucléaire ou encore le militaire. Parmi les 18 sites répartis en France et dans le monde, quatre d'entre eux ont une expertise dans le domaine de la simulation numérique. Deux pôles de compétences y sont développés. D'une part, la mécanique des solides qui englobe des activités de mécanique, thermomécanique, dynamique des structures, dynamique rapide et choc. D'autre part, la mécanique des fluides et les couplages multiphysiques avec des activités aérodynamiques, hydrodynamiques, thermiques et en électromagnétisme.

J'ai effectué ce stage dans l'agence de Bordeaux au sein du centre de simulation numérique. Cette agence est structurée en pôles de compétences. J'ai été encadré par Gilles Vogt qui fait partie du pôle systèmes multi-physiques et Guillaume Ruiz, responsable adjoint simulation numérique.

1.2 Contexte du stage

Une des vocations d'Ingeliance est la recherche et le développement de nouvelles méthodes de résolutions numériques pour aider à la compréhension de phénomènes physiques complexes. Cet objectif est nécessaire pour rester compétitif et proposer une valeur ajoutée vis-à-vis des autres sous-traitants. En effet, les clients souhaitent modéliser des systèmes avec une fidélité croissante, mais avec des coûts maîtrisés (temps de calcul et mise en oeuvre du modèle).

De fait, une problématique centrale est le temps de calcul. En effet, des modèles plus complexes utilisent des méthodes plus coûteuses en temps de calcul. Même quand les codes tirent profit de l'architecture parallèle des ordinateurs, les calculs peuvent durer jusqu'à plusieurs jours. De plus, des nombreuses itérations sont généralement nécessaires sur un modèle. En effet, pour chaque changement de géométrie, changement dans les conditions de sollicitations ou changement sur les hypothèses de modélisation, il faut relancer la résolution. Cet aspect n'est pas toujours compris par les clients qui peuvent ne pas comprendre le temps de calcul associé à une modification mineure du modèle.

Le *machine learning*, aujourd'hui très populaire dans de nombreux domaines, semble pouvoir aussi s'appliquer aux simulations. En effet, les modèles de simulations numériques produisent une grande quantité de résultats qui ne sont exploités que partiellement. Cette source de données pourrait donc être utilisable par des algorithmes de *machine learning*. Le potentiel serait de pouvoir accélérer les temps de calculs, limiter le nombre d'itérations sur un modèle numérique ou encore se passer complètement de la simulation. Cependant, il convient de ne pas tomber

dans le piège des *data sciences*. Mettre en place des méthodes performante n'est pas trivial et les algorithmes ne pourront pas résoudre tous les problèmes.

Ainsi, de nombreux chercheurs ont tenté de combiner les méthodes de *machine learning* aux simulations numériques. La stratégie est souvent d'entraîner un modèle de substitution moins coûteux, appelé méta-modèle, sur quelques résultats de simulations. Ce méta-modèle pourra être utilisé pour prédire de nouveaux résultats ou pour accélérer un nouveau calcul coûteux mais aussi pour faire de la propagation d'incertitudes ou pour améliorer des couplages de modèles à l'échelle globale avec des modèles à l'échelle locale. C'est dans le but d'intégrer ces nouvelles méthodes dans les travaux de l'entreprise Ingeliance que mon stage a été créé.

1.3 Objectifs et Déroulement du Stage

Chez Ingeliance, peu de travaux précèdent mon stage, c'est pourquoi mes objectifs étaient multiples. Dans un premier temps, je devais conduire une étude bibliographique sur les liens entre la simulation numérique et le machine learning et prendre en main les librairies de programmation du domaine. D'abord d'une manière générale, puis en se concentrant sur l'approximation globale de simulations par méta-modèle. Dans un second temps, je devais appliquer les méthodes les plus pertinentes sur un cas test industriel en dynamique : un ressort support d'antenne. C'était un sujet exploratoire. En effet, les objectifs et les méthodes employées allaient être déterminés à la suite de l'étude bibliographique.

J'ai finalement principalement orienté mon travail vers les problèmes d'approximation globale. Plus particulièrement en mettant en place des méthodes d'adaptive sampling et de réduction de modèles. J'ai ensuite pu appliquer ces méthodes sur le cas test du ressort mais aussi sur d'autres modèles comme une barre de Taylor et un stabilisateur pour bateau. Les modèles éléments finis m'étaient fournis par mes tuteurs et prenaient place dans des codes industriels tel que LS-DYNA. J'ai utilisé les résultats de ces modèles pour construire des bases de données utilisables par les méthodes de *machine learning* programmées en langage Python. De fait, au fil des implémentations, certaines parties des scripts que j'ai créés ont été utilisées pour faire une librairie plus simple d'utilisation. L'objectif étant de permettre aux ingénieurs d'Ingeliance de réutiliser ces outils.

2 Liens entre Simulation numérique et machine learning

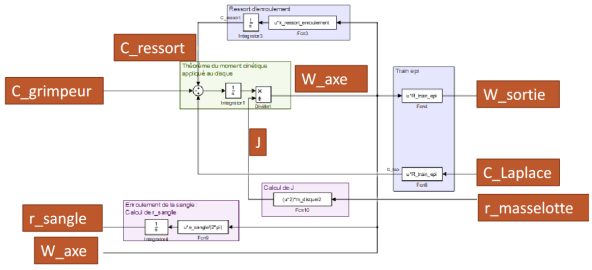
Dans cette section, les couplages existants entre le *machine learning* et la simulation numérique vont être identifiés. Ce travail bibliographique est nécessaire pour identifier les cas en liens avec les activités de simulation d'Ingeliance. J'ai conduit cette étude en me basant sur des méta-analyses [4, 5] qui synthétisent les travaux de recherche existants, puis en explorant des exemples précis plus en détails. De la sorte, on prendra une approche centrée sur le *machine learning* puis sur la simulation numérique [5]. Avant toute chose, voici une brève description de la simulation numérique et du *machine learning*.

2.1 Simulation numérique

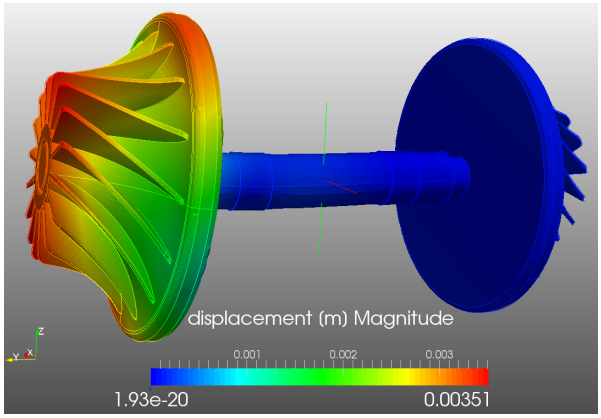
La simulation numérique a pour objectif de simuler un système réel. Un modèle numérique du système est construit à partir de modèles théoriques et empiriques. Ce modèle est ensuite résolu à l'aide de méthodes numériques.

Ainsi, un modèle numérique peut être construit en considérant les différents composants du système réel et en leur associant une loi de comportement à une échelle globale. Les relations entre les différentes lois sont ensuite définies. On parle de modèles systèmes Fig(1a).

L'autre manière de construire un modèle numérique est de considérer des modèles théoriques locaux, souvent sous forme d'équations aux dérivées partielles. Le système réel est alors discrétisé, pour former un maillage. Des méthodes numériques comme les éléments finis ou les volumes finis permettent ensuite de résoudre l'équation Fig(1b).



(a) Modèle système d'un système d'assurage automatique pour grimpeurs



(b) Modèle de simulation par éléments finis

FIGURE 1 : Exemples de modèles numériques

Ainsi, les modèles numériques, permettent d'accélérer la conception et la validation de systèmes toujours plus complexes. Des étapes qui nécessitaient autrefois un prototype et des essais expérimentaux coûteux peuvent maintenant être réalisées virtuellement. Seulement quelques essais sur le système réel peuvent être utiles pour vérifier et recaler les lois théoriques utilisées. Les calculs numériques permettent également d'avoir des informations sur des phénomènes impossibles à observer avec des capteurs. L'évènement d'intérêt peut être trop court, dans une zone inaccessible ou trop petit. En revanche, les résultats de simulations sont souvent disponibles en tous points de l'espace d'intérêt. Par ailleurs, certaines situations destructrices ne peuvent pas être réalisées par essais. C'est par exemple le cas pour la résistance de bâtiments aux séismes ou la résistance d'un stabilisateur de bateau à une explosion.

De fait, il existe une grande diversité de phénomènes étudiés, résolus avec des méthodes numériques différentes. En effet, certaines méthodes numériques seront plus pertinentes que d'autres selon le système étudié. En conséquence, de nombreux codes de calculs industriels existent et implémentent des méthodes différentes. Ainsi, pour traiter des phénomènes de mécanique des fluides, la méthode numérique des volumes finis est souvent utilisée. Un des avantages est que les bilans de masses sont respectés. Elle est notamment implémentée dans le code commercial Fluent ou le projet libre OpenFoam. D'autre part, pour traiter des phénomènes de mécanique des solides, d'éléctromagnetisme ou des couplages de physiques, c'est la méthode des éléments finis qui est généralement implémentée. La force de cette méthode est qu'elle peut s'adapter à de nombreux phénomènes. Elle est implémentée dans les codes commerciaux Abaqus, Comsol ou LS-DYNA ou dans des codes libres comme FreeFem++ ou FEniCSx. Pour effectuer des simulations par modèles systèmes, les logiciels commerciaux Matlab-simulink ou Amesim permettent de construire les modèles numériques via des interfaces graphiques. Les alternatives libres tel que XCos ne sont, à ce jour, pas aussi développées.

Ainsi, pour un sous-traitant comme Ingeliance, il est utile de maîtriser et de posséder un panel étendu de logiciels de simulation numérique. En effet, un client préférera ou imposera un logiciel qu'il utilise en interne.

Néanmoins, la simulation numérique présente des limites. Tout d'abord, chaque modèle numérique implique de faire des hypothèses pour le modèle théorique. Ces hypothèses impliquent des compromis qui définissent le domaine de validité du modèle. A titre d'exemple, la déformation d'une pièce avec un modèle de matériau élastique ne pourra pas capter une éventuelle rupture de la pièce dûe à des contraintes trop importantes. C'est alors à l'ingénieur de comprendre que la simulation sort de son domaine de validité. C'est pourquoi, la création de modèles numériques est un processus encore très manuel qui requiert une bonne qualification. Ensuite, pour avoir des résultats précis, il faut utiliser des maillages plus fins, ou des méthodes numériques plus complexes. Cela augmente les coûts de calculs et ainsi même en utilisant plusieurs machines, les calculs pour une simulation numérique peuvent durer plusieurs jours.

Pour ce stage, les simulations numériques ont été réalisées par mon tuteur de stage. Ainsi, je n'ai pas conçu de modèles de simulation. En revanche, j'ai dû me les approprier pour pouvoir exploiter les résultats, que ce soit pour créer des bases de données ou encore pour analyser les résultats.

2.2 Machine Learning

Par ailleurs, le *machine learning* - ou apprentissage automatique - est une sous-catégorie de l'intelligence artificielle Fig(2). Le but est d'utiliser des algorithmes pour effectuer une tâche à partir de données. Les méthodes vont utiliser des approches statistiques pour "apprendre" des données, à savoir améliorer leurs performances à résoudre les tâches sans programmation externe.

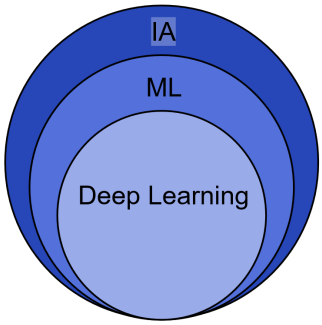


FIGURE 2 : Le *deep learning* comme sous-catégorie du *machine learning*, lui-même comme sous-catégorie de l'intelligence artificielle

Ainsi le *machine learning* est aujourd'hui utilisé dans de nombreux domaines, notamment :

- Les algorithmes de vision : reconnaissance d'images, de formes, de visages, segmentations d'images
- Le langage naturel : traduction de textes, complétion de textes
- L'aide aux diagnostics : médical, détection de fraude, cyber sécurité, analyse financière, météorologie
- Le contrôle de systèmes : voiture autonome, contrôle de centrales
- Les intelligences artificielles de jeux vidéos

En pratique, la mise en place d'une approche par *machine learning* se déploie souvent en deux phases. Une phase d'entraînement, aussi appelée apprentissage, où des données d'entraînements vont être utilisées pour régler un algorithme. Le modèle de *machine learning* va apprendre à réaliser une tâche. Cette phase précède l'utilisation de l'algorithme dans son contexte final. En effet, la deuxième phase est celle d'exploitation du modèle. De nouvelles données vont pouvoir être analysées par l'algorithme de *machine learning* et de nouveaux résultats vont pouvoir être obtenus en fonction de la tâche apprise.

Suivant la nature des données utilisées pendant la phase d'apprentissage, on distingue Fig(3) :

- Les méthodes supervisées : Les données d'entraînements sont accompagnées des réponses espérées de l'algorithme. Ces réponses sont appelées annotations. L'apprentissage se fait donc à partir d'exemples. De cette manière, pour des annotations de catégories, les modèles effectueront une classification et pour des annotations chiffrées continues, une régression.
- Les méthodes par renforcement : L'algorithme reçoit une récompense d'un agent externe pour chaque tâche effectuée. L'apprentissage est comme guidé par un tuteur.
- Les méthodes non supervisées : Les données sont dépourvues d'annotations. Le modèle de *machine learning* découvre seul les relations dans les données.

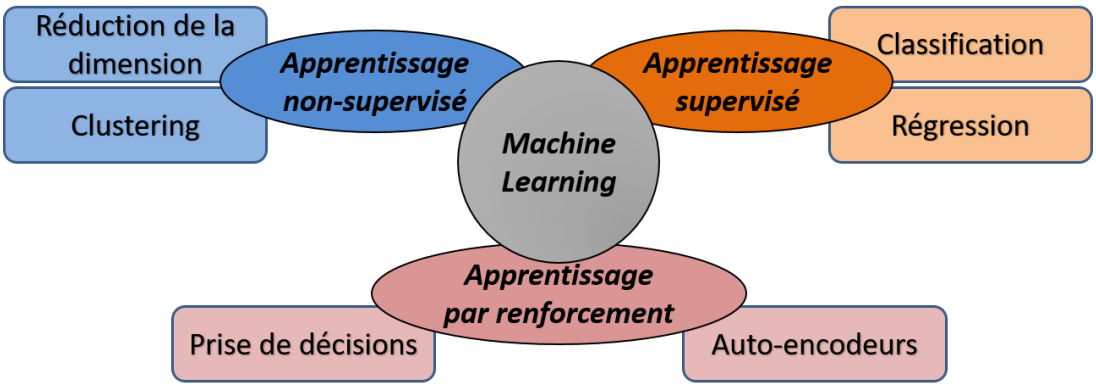


FIGURE 3 : Approches d'apprentissages par machine learning

En définitive, le *machine learning* est un outil puissant car les méthodes sont applicables dès le moment où des données sont disponibles. Dès lors, elles peuvent s'appliquer dans de très nombreux domaines. De plus, les algorithmes utilisés parviennent à tirer profit des développements récents sur les accélérateurs de calculs (GPU). Cette avancée technique contribue à l'exploitation de ces données dans un temps raisonnable. De plus, si la phase d'entraînement peut être très coûteuse en temps de calcul, la phase d'exploitation est généralement rapide. Un modèle entraîné pendant des semaines sur un serveur de calcul peut être exécuté en une fraction de seconde sur un smartphone.

Néanmoins, la performance d'une méthode dépend grandement de son apprentissage. La quantité et la qualité des données disponibles va grandement impacter la qualité des tâches effectuées. En général, il est nécessaire d'avoir beaucoup d'exemples d'apprentissage et il faut faire un long travail de nettoyage des données pour faciliter l'entraînement des algorithmes. Un autre défaut de ces méthodes est qu'elles captent les biais des données d'apprentissage. À titre d'exemple, un algorithme de reconnaissance faciale entraîné sur des visages de personnes blanches ne fonctionnera pas sur des visages de personnes noires. L'algorithme est discriminatoire car les données d'apprentissage ne prennent pas bien en compte la diversité des couleurs de peau dans la population. Il est biaisé. Enfin, les algorithmes de *machine learning* sont souvent perçus comme des boites noires. Les décisions prises par certains algorithmes peuvent être difficiles à interpréter. C'est principalement le cas des réseaux de neurones, qui comportent trop de paramètres pour donner un sens à chacun d'entre eux. Dans un contexte industriel, cet aspect freine souvent le déploiement de cette méthode car il est difficile d'identifier les incertitudes et les zones de validité du modèle.

> C'est principalement les méthodes de régression qui sont utilisées dans le contexte de la simulation numérique. Les paramètres de la simulation sont les paramètres d'entrées et les résultats de la simulation sont les annotations. Ainsi, j'ai dû me familiariser avec ces concepts et notamment mettre en place les méthodologies pour construire et comparer des modèles de régressions. Pour cela, j'ai principalement utilisé la librairie Python scikit-learn.

Maintenant que la simulation numérique et le *machine learning* ont été présentés, les manières de lier les deux domaines peuvent être abordées.

2.3 Approches combinées

Les méthodes de *machine learning* et de simulation numérique peuvent être liées de plusieurs manières. Pour balayer ces possibilités au mieux, on peut reprendre la description de l'article de

Laura von Rueden [5]. Il faut d'abord présenter les méthodes autour des modèles déterministes et des modèles basés sur des données. La simulation numérique utilise et résout des modèles déterministes pour obtenir des données. Au contraire, le *machine learning* utilise les données comme point de départ et produit un modèle. On peut ainsi décomposer les étapes clés des deux approches et distinguer des manières de coupler les approches. La première consiste à se placer d'un point de vue centré sur la simulation numérique. La seconde approche se centre autour du *machine learning*.

La figure Fig(4) résume donc les différentes étapes de chaque approche ainsi que les couplages possibles.

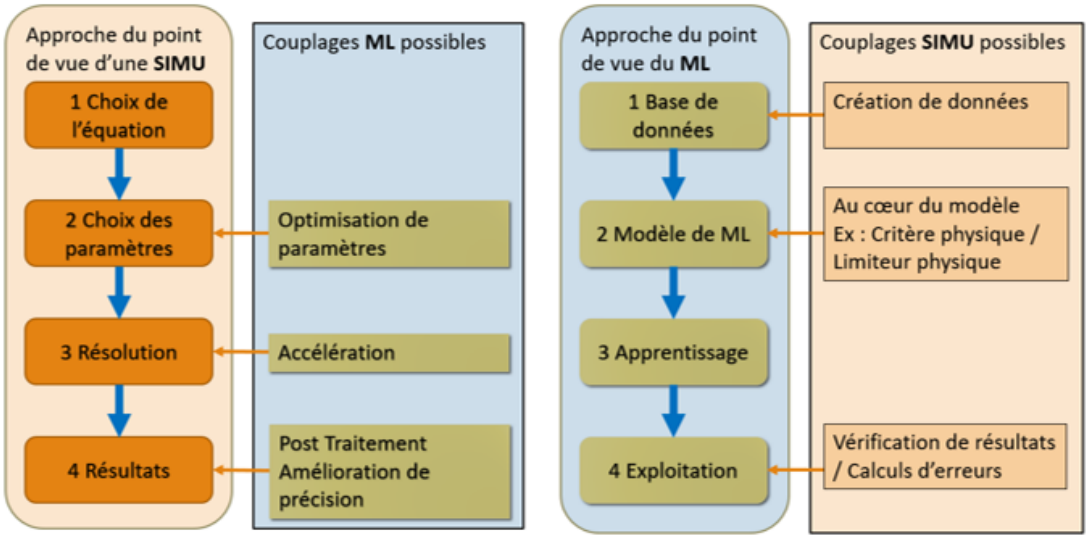


FIGURE 4 : Étapes et Couplages entre simulation et machine learning

2.3.1 Centrée sur la simulation

D'un point de vue centré sur la simulation, 3 couplages au *machine learning* sont identifiés. En premier lieu, après avoir construit un modèle numérique, il est possible de réaliser une étude paramétrique. C'est-à-dire faire varier les paramètres du modèle pour obtenir les meilleures performances. C'est un processus qui peut être long et manuel. En effet, un ingénieur doit lancer une grande quantité de simulations pour traiter tous les ensembles de paramètres. Il doit ensuite analyser les résultats pour déterminer la meilleure configuration. Dans ce cas de figure, les méthodes de *machine learning* peuvent servir à construire un modèle approché rapide (méta-modèle) qui permettra d'explorer et de converger rapidement vers un jeu de paramètres optimal sans payer le coût de toutes les simulations. Par exemple, l'optimisation des paramètres de coupes d'opérations de tournage [6]. Aussi, pour faciliter la recherche de ces paramètres optimaux, l'utilisation de méthodes pour la réduction de dimensions et le *clustering* peuvent être appliquées. Cela permet à l'ingénieur d'identifier rapidement les tendances dans les données et peut faciliter une optimisation paramétrique, comme dans le cas d'une simulation de crash de voiture [7].

Un second cas de couplage est l'accélération de convergence. Obtenir des solutions précises est parfois très long par simulation directe. C'est pourquoi les codes de simulations peuvent intégrer des modèles de régression pour approximer les termes les plus coûteux [8] ou la solution toute entière [9]. Le solveur peut également être directement optimisé, par exemple en prédisant un ordre de schéma numérique à calculer [10]. Ces modèles peuvent être préalablement entraînés sur d'autres données. Si ce n'est pas le cas, on parle alors d'online learning. C'est ainsi qu'un solveur itératif GMRES a pu être optimisé à la volée [11].

Le dernier cas de couplage identifié est l'aide au post-traitement. De cette manière, il est possible d'utiliser des méthodes de réduction de la dimension et de *clustering* pour identifier des zones d'intérêts dans la ou les solutions [7]. D'autres approches permettent aussi d'améliorer la précision d'une solution. On parle alors de super-résolution [12].

2.3.2 Centrée sur le Machine Learning

En se plaçant maintenant du point de vue du *machine learning*, on peut identifier 3 types de couplages possibles. Un d'entre eux est de faire une base de données avec les résultats d'une simulation. Il est ensuite possible d'effectuer des régressions [9] ou des analyses de données [7]. C'est le type de couplage le plus évident d'un point de vue *machine learning*. De plus, cette approche rejoint l'exploration paramétrique vu dans l'approche simulation numérique en section 2.3.1.

Le second cas d'application est d'utiliser directement une équation physique pour guider l'apprentissage d'une méthode de *machine learning*. On peut ainsi créer des "Physics Informed Neural Networks (PINNs)", qui sont des réseaux de neurones capables de respecter une équation physique [13]. L'équation est ainsi directement programmée dans le réseau de neurones.

De manière plus complexe, un réseau de neurones peut être guidé pendant son apprentissage par une simulation numérique. C'est le cas pour un apprentissage par renforcement où un "generative adversarial network" (GAN) apprend en respectant des contraintes [14]. Ce cas de couplage est différent du précédent car les contraintes sont imposées par un agent extérieur et ne sont pas intégrées dans le modèle de *machine learning*.

Finalement, dans le cadre de mon stage dont l'objectif est de réaliser des méta-modèles, le but est de faire de l'approximation globale. De fait, les simulations seront utilisées pour produire des bases de données.

2.3.3 Méthodes retenues

Pour construire des méta-modèles, deux méthodes non-intrusives ont été retenues. Le caractère non-intrusif signifie qu'il n'est pas nécessaire de modifier les méthodes de simulations numériques pour les implémenter. En effet, ces deux méthodes ont été retenues car elles peuvent se greffer autour d'un logiciel de simulation numérique existant et interpoler à partir de quelques simulations. Elles ont le potentiel d'enrichir les résultats de simulations, d'accélérer des optimisations paramétriques et de proposer une nouvelle façon de présenter des résultats numériques pour les clients. Pour faciliter ces utilisations, les implémentations de ces méthodes sont regroupées dans un module Python à destination d'Ingeliance.

D'une part, des méthodes de régressions sur des valeurs globales issues des résultats et une estimation d'erreur permettent d'explorer les paramètres de ces simulations et de construire des surfaces de réponses. Grâce à cette estimation d'erreur, il est possible d'enrichir la base de données et ainsi augmenter la précision du modèle. Ce sont les méthodes d'*adaptive sampling*.

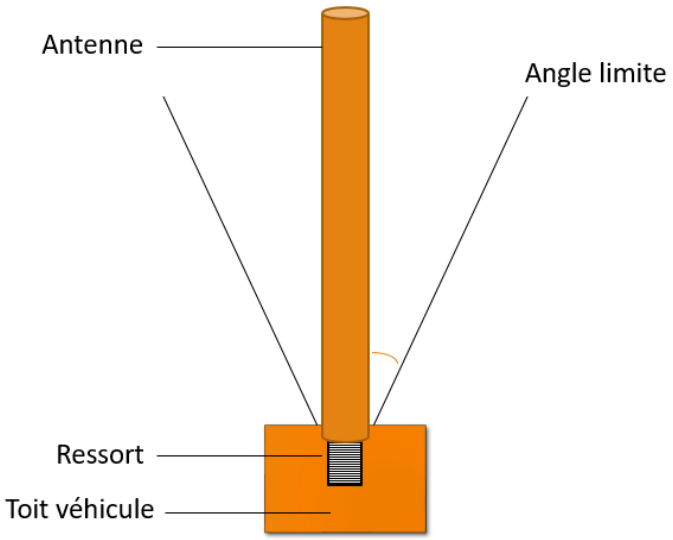
D'autre part, les solutions ont aussi été exprimées dans des dimensions plus faibles afin de faciliter les régressions sur des champs entiers de solutions. C'est de la réduction de modèles.

3 Approximation Globale

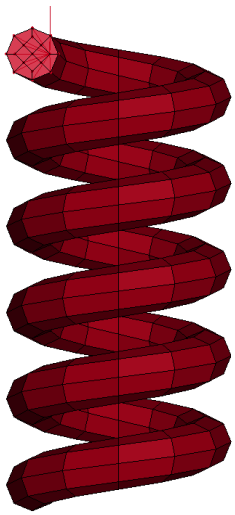
Dans un problème d'approximation globale, le but est d'approximer une fonction objectif dans tout son domaine d'évaluation. Autrement dit, on approxime la surface de réponse de cette fonction à l'aide d'un méta-modèle. Dans le cas de la simulation numérique, cette fonction objectif peut être un modèle éléments finis long à résoudre où on choisit des sorties d'intérêts. On peut donc sélectionner quelques valeurs ou sélectionner des champs de solution entiers. C'est un méta-modèle, entraîné sur quelques évaluations, qui va permettre de prédire rapidement la réponse en tout point du domaine d'évaluation.

Exemple du ressort support d'antenne : Dans cette section, les méthodes seront illustrées à l'aide d'un exemple. Pour chaque implémentation, les choix et les difficultés rencontrées seront expliquées.

Certains véhicules embarquent des antennes rigides sur leur toit. Ces antennes doivent tenir sans casser pour plusieurs types de terrains, plusieurs types de routes et se coucher au contact d'un obstacle, par exemple une branche d'arbre. De plus, elles ne doivent pas entrer en contact avec d'autres équipements fixés sur le toit. Par conséquent, elles sont montées sur un ressort qui va permettre le mouvement de l'antenne Fig(5a). Ce ressort support d'antenne doit être correctement dimensionné afin de respecter ces contraintes.



(a) Schéma du système ressort et antenne



(b) Modèle LS-DYNA paramétré du ressort

FIGURE 5 : Ressort support d'antenne

Actuellement, le client effectue des essais pour tester et trouver le bon ressort. Cette démarche est longue, coûteuse et ne garantit pas d'avoir la meilleure solution à la fin des essais. Ainsi, une démarche par simulation numérique pourrait permettre de tester les différents paramètres de ressorts Fig(5b). En revanche, le coût de calcul d'une telle exploration paramétrique est trop élevée. En effet, le temps de sollicitation du ressort est long, les méthodes numériques utilisées ne sont pas précises sur un temps long (de l'ordre de 10 min) et sont coûteuses. En conséquence, l'objectif a été de construire un méta-modèle à l'aide des méthodes d'approximation globales pour explorer les paramètres du ressort à partir d'un nombre d'essais numériques réduit. Les essais numériques sollicitent le modèle numérique du

ressort sur un temps inférieur à 10 seconde et déjà dans ce cas le temps de calcul atteint plusieurs minutes. L'objectif est d'utiliser le méta-modèle de manière temporelle pour atteindre la durée de sollicitation de 10 min.

Cette application permet de tester les méthodes décrites dans les prochains paragraphes. Le modèle est suffisamment complexe pour présenter des non-linéarités notamment liées aux contacts. Mais il est suffisamment simple pour être résolu en un temps raisonnable (10 min-20 min).

La méthodologie d'une approximation globale est explicitée par la Figure Fig(6). Généralement, la source de données va être une simulation recalée sur des essais expérimentaux (1)-(2). Les paramètres et leurs intervalles sont définis en cohérence avec cette simulation. De là, un plan d'expériences numérique peut être construit pour balayer au mieux les paramètres (3). On obtient une collection de résultats pour différents paramètres. Souvent, seulement quelques valeurs sont intéressantes parmi les nombreux champs de la solution numérique. D'où une étape de sélections des variables de sorties à étudier (4). Un modèle de machine learning va ensuite réaliser un apprentissage supervisé entre le plan d'expériences et les variables d'intérêts (5). On pourra finalement utiliser ce modèle de machine learning pour prédire rapidement des nouvelles valeurs pour des nouveaux jeux de paramètres.

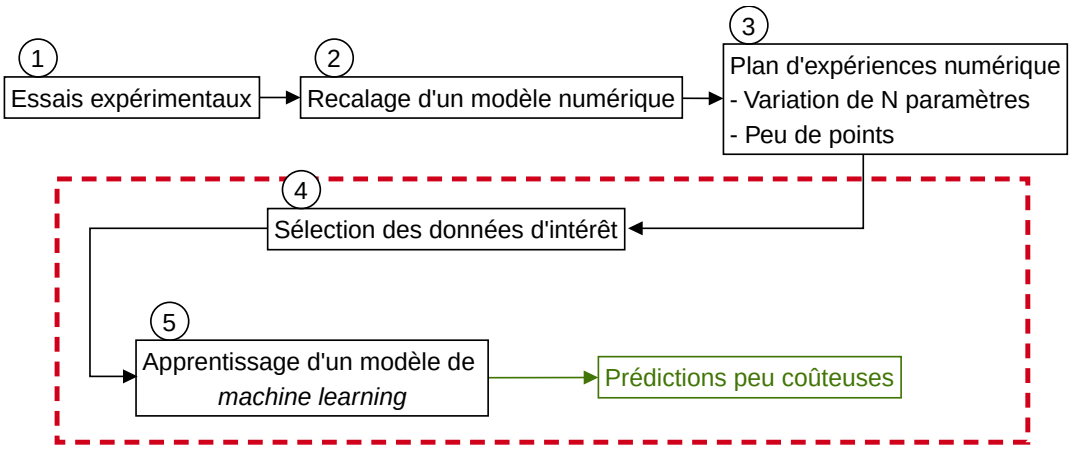


FIGURE 6 : Méthodologie d'approximation globale

Approximation globale sur le ressort : Dans le cas de l'étude d'un ressort support d'antenne pour véhicule, les lois matériaux d'un modèle éléments finis sont recalées à partir d'essais en statique sur un ressort (1)-(2). Toutefois, c'est le comportement en dynamique qui est étudié. Le ressort simulé pour l'étude sera excité à une vitesse initiale v_{ini}. Les paramètres pour le plan d'expériences peuvent être (3) :

- de paramètres géométriques avec le nombre de spires, le rayon du fil, le pas ou le rayon du ressort
- des conditions initiales en changeant la vitesse initiale ou la gravité
- le temps, qui va être interprété comme un paramètre

Les résultats qui serviront d'annotations au modèle de *machine learning* (4) peuvent

être :

- des valeurs, comme une force de rappel, le maximum des contraintes ou un déplacement maximum
- des champs scalaires entiers, comme par exemple les contraintes à chaque élément

Ensuite, un modèle de machine learning est entraîné (5).

On peut définir un cadre mathématique plus rigoureux autour de cette méthodologie. Les termes de fonction objectif et de méta-modèle vont être définis.

Fonction objectif

La fonction à approximer est la fonction objectif f. Elle est définie par :

$$\begin{aligned} f : \mathbb{X} &\to \mathbb{Y} \\ x &\mapsto f(x) = y \end{aligned}$$

$\mathbb{X} \subset \mathbb{R}^d$ est l'espace des paramètres. d est le nombre de paramètres. Ainsi x est un jeu de d paramètres.
$\mathbb{Y} \subset \mathbb{R}^{N_s}$ est l'espace des sorties. N_s est le nombre de sorties. Ainsi y est un vecteur solution.

Méta-modèle

Le méta-modèle $\hat{f}$ cherche à approcher la fonction objectif. Elle est définie par :

$$\begin{aligned} \hat{f} : \mathbb{X} &\to \mathbb{Y} \\ x &\mapsto \hat{f}(x) = \hat{y} \end{aligned}$$

$\hat{y}$ est l'approximation de $f(x) = y$ par le méta-modèle.

Pour entraîner le méta-modèle, une base de données est créée. Elle est composée de N paires (jeu de paramètres/vecteur solution) (x, y). Pour ce faire, la fonction objectif est appelée avec plusieurs jeux de paramètres :

$$\mathcal{X} = \{x^i, i = 1, \ldots, N\}$$

Les résultats sont collectés dans un vecteur :

$$\mathcal{Y} = \{y^i, i = 1, \ldots, N\}$$

Le méta modèle $\hat{f}$ est entraîné à partir de ces deux ensembles. Pour obtenir la surface de réponse, on construit une grille fine :

$$\mathcal{X}^\star = \{x^i, i = 1, \ldots, N_{fine}\} \text{ avec } N_{fine} \gg N$$

Le méta-modèle est évalué en chaque point de cette grille donnant la surface de réponse approchée : $\hat{\mathcal{Y}}^\star$.

$$\hat{\mathcal{Y}}^\star = \hat{f}(\mathcal{X}^\star)$$

Le symbole $\star$ souligne la facilité d'obtention de cette surface qui a une haute résolution. En effet, évaluer le méta-modèle est peu coûteux.

Tout compte fait, suivant les applications plusieurs méthodes d'approximation globale sont possibles. Dans le cadre du stage, deux cas ont été envisagés. Dans le premier, limiter le nombre d'échantillons était primordial et il n'y avait qu'une sortie. La solution a alors été de construire le plan d'expériences numérique de manière itérative et intelligente jusqu'à avoir une surface de réponse suffisamment précise. C'est l'*adaptive sampling*. Dans le second cas, pour avoir une meilleure compréhension des résultats, le but est de prédire des champs entiers de solutions, et donc un grand nombre de sorties. La solution a été d'utiliser de la réduction de modèles non-intrusive.

3.1 Adaptive Sampling

Dans un souci de réduire les coûts de calculs, le plan d'expériences numérique doit contenir le moins de points possibles. Une manière est d'utiliser des échantillonnages de l'espace qui balayent au maximum les paramètres. Les méthodes Latin Hypercube Sampling (LHS) sont déjà efficaces pour cela. Cependant, elles ne prennent pas en compte les variations de la solution au fur et à mesure du calcul des points. L'*adaptive sampling* le permet. En effet, le plan d'expériences peut être construit progressivement. A chaque itération le prochain point de calcul sera choisi selon un critère dépendant de la prédiction du méta-modèle. Il est ainsi possible d'ajouter des points jusqu'à avoir une solution satisfaisante.

La figure Fig(7) reprend la méthodologie d'approximation globale avec les spécificités de l'*adaptive sampling*. La méthode est initialisée avec très peu de points dans le plan d'expériences (3). Les simulations sont lancées pour chaque point. Ensuite, commence la boucle d'enrichissement. Elle est composée de trois étapes :

1. En premier lieu, le modèle de machine learning est entraîné à partir des résultats des simulations numériques (5) du plan d'expériences.
2. Dans un second temps, un estimateur d'erreur estime l'erreur du méta-modèle sur l'ensemble des paramètres possibles. Cet estimateur permet d'avoir un volume autour de la surface d'erreur (6).
3. Enfin, l'utilisation d'une fonction d'acquisition permet de choisir un nouveau point à ajouter dans le plan d'expérience. Ce nouveau point est choisi de manière à réduire au maximum l'erreur du méta-modèle (7). Ce nouveau point est simulé et les résultats sont réutilisés à l'étape 1.

Cette boucle d'enrichissement du plan d'expérience est réalisée jusqu'à ce qu'un critère d'arrêt soit atteint. Ce peut être un nombre d'itérations maximum, une estimation d'erreur à atteindre ou encore un temps de calcul maximal. Une fois cette boucle réalisée, le méta-modèle et l'estimateur d'erreur peuvent être utilisés pour prédire de nouveaux résultats et donner un intervalle de confiance.

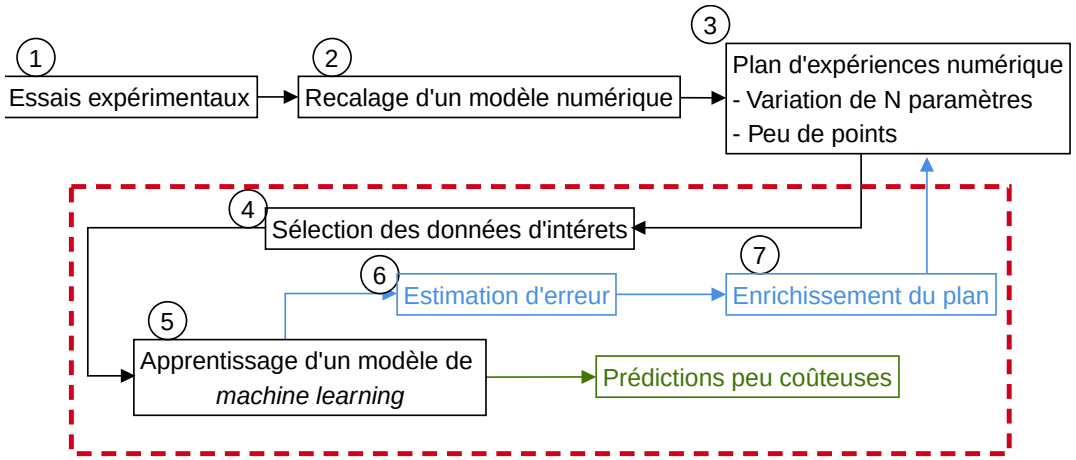


FIGURE 7 : Méthodologie d'adaptive sampling

Pour régler cette méthode, les outils sur lesquels il est possible d'influer sont encadrés en rouge pointillé Fig(7). Il est ainsi possible de choisir le modèle de régression, l'estimateur d'erreur et la fonction d'acquisition. Cependant, nous nous concentrerons uniquement sur le choix des deux derniers outils.

3.1.1 Calcul d'erreur

Bien évaluer la performance d'un modèle de machine learning n'est jamais simple. La méthodologie classiquement utilisée pour attribuer un score à un modèle est la validation croisée. Elle permet de donner un score de performance globale à un modèle. Pour ce faire, les données sont séparées en deux groupes. Un groupe réservé pour l'entraînement du méta-modèle et un groupe de test réservé pour l'évaluation. Cette méthode ne permet pas d'avoir une erreur pour chaque prédiction. De plus, elle implique de laisser de côté des données test pendant l'apprentissage.

Pour effectuer de l'adaptive sampling, il faut une estimation d'erreur associée à chaque prédiction.

Estimateur d'erreur

L'estimateur d'erreur E utilise la surface de réponse du méta-modèle et les données d'entraînement pour estimer l'erreur du méta-modèle sur la surface de réponse approchée $\hat{\mathcal{Y}}^\star$.

$$E\colon \mathcal{X}_{fine}, \hat{\mathcal{Y}}^\star, \mathcal{X}, \mathcal{Y} \mapsto \hat{\mathcal{E}}$$

$\hat{\mathcal{E}}$ est une estimation de l'erreur $\mathcal{E}$. L'erreur $\mathcal{E}$ est coûteuse à évaluer car elle implique d'évaluer la fonction objectif.

Il existe alors plusieurs stratégies. La première est d'utiliser un méta-modèle qui donne une telle estimation par construction. Les processus Gaussiens en font partie. La seconde est d'utiliser plusieurs méta-modèles et de les comparer pour identifier les zones où leurs réponses divergent. Ces méthodes se rapprochent ainsi du *bagging* ou de la validation croisée par Leave One Out.

Processus Gaussiens : Les processus Gaussiens ou Krigeage sont des estimateurs qui incorporent cette estimation d'erreur. En effet, la prédiction d'un processus Gaussien est une distribution gaussienne. On peut ainsi prendre la moyenne et l'écart type de cette distribution pour

avoir respectivement une valeur et un intervalle. La figure Fig(8) représente les points d'apprentissages, la distribution gaussienne pour une prédiction, la moyenne et l'intervalle de confiance pour un intervalle.

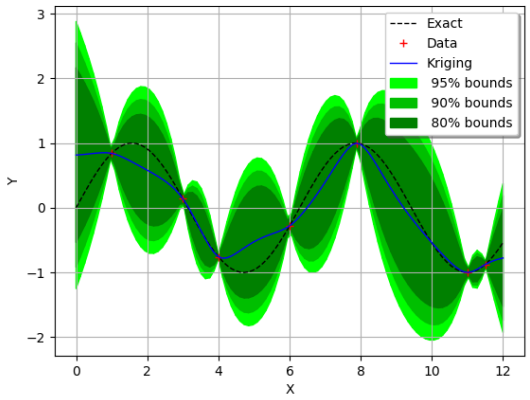


FIGURE 8 : Exemple de processus Gaussien en 1D

Mathématiquement, les caractéristiques de cette méthode font qu'elle garantit le minimum de variance pour chaque estimation. Pour cela, elle tient compte de la corrélation au sein des données mais aussi de la corrélation entre les données et le point d'estimation. On remarque aussi que les points de mesures sont interpolés.

Chaque modèle de processus gaussiens est associé à un modèle de covariance. Ce modèle de covariance utilise la distance entre les données pour estimer leur ressemblance. Le choix et le réglage de cette composante dirige en grande partie la réponse du modèle. Dans les utilisations pour des mesures géostatistiques, où le Krigeage est utilisé classiquement, ce modèle de covariance est réglé en analysant les données dans leur contexte, notamment pour tenir compte de biais ou de tendances. Cependant, pour créer un méta-modèle d'une fonction objectif quelconque, il n'est pas toujours possible d'avoir des informations sur cette fonction. De plus, les compétences pour bien réaliser le réglage limitent la prise en main de la méthode.

De fait, les méthodes qui ont été utilisées ici ont été réglées automatiquement en minimisant la fonction log-vraisemblance[1] du modèle. C'est de cette manière que les processus gaussiens sont implémentés dans la librairie scikit-learn. Le modèle de covariance RBF a été utilisé, principalement pour sa simplicité : $k_{RBF} = \sigma^2 exp\left(-\frac{(x-x')^2}{2l^2}\right)$

Une limite des processus gaussiens est qu'ils sont souvent implémentés pour ne prédire qu'une seule variable à la fois. Ainsi pour estimer plusieurs variables il faudrait construire plusieurs méta-modèles en parallèles. De cette manière, on ne prend pas en compte les corrélations qu'il existe entre ces variables. Cela est dû à la difficulté de définir une distance pour plusieurs variables. Plusieurs approches existent dans la littérature. J'ai ainsi testé la définition décrite dans l'article de Bonilla [15] implémentée dans le module Python GPyTorch. Une seconde limite des processus gaussiens est qu'ils utilisent toutes les données pour chaque prédiction. De fait, quand le nombre de points de mesure devient important, les calculs deviennent très gourmands en mémoire. Pour pallier cette limite, il est possible d'utiliser seulement un échantillon des données. Ici encore, une implémentation existe dans GPyTorch. Toutefois, une autre façon de s'affranchir des limitations des processus gaussiens est d'utiliser un autre estimateur d'erreur. Dans ce cas, on peut aussi

[1]La vraisemblance décrit la plausibilité de la valeur du paramètre. Le logarithme de cette fonction est appelé log-vraisemblance. Il permet de simplifier les calculs.

utiliser un autre modèle de machine learning. Par exemple, on pourra utiliser des forêts aléatoires ou des réseaux de neurones.

Validation croisée : La validation croisée consiste à entraîner un modèle de machine learning plusieurs fois avec des jeux de données légèrement différents. On obtient ainsi plusieurs modèles aux réponses différentes. En comparant ces réponses, on peut identifier les zones où elles divergent, ce qui implique un manque d'informations à cet endroit. Une façon de définir une telle estimation d'erreur peut être [16] :

$$E_{LOOCV}(x_i) = \sum_{k}^{N} |\hat{f}(x_i) - \hat{f}_{-k}(x_i)|$$

où $\hat{f}_{-k}$ est le méta modèle entraîné sur les données privées du k-ième élément. J'ai pu la mettre en place pendant le stage, mais je n'ai pas eu le temps d'analyser les résultats.

Cet estimateur d'erreur permet d'utiliser n'importe quel algorithme de machine learning. C'est donc une méthode souple. Cependant, elle est plus complexe à mettre en place que les processus gaussiens.

3.1.2 Fonctions d'acquisitions

La fonction d'acquisition a pour objectif de choisir un nouveau point à partir d'un critère de raffinement basé sur une estimation d'erreur. Il existe plusieurs stratégies, l'article de Fuhg [16] décrit et compare plusieurs façons de choisir un nouveau point de calcul. On peut caractériser ces méthodes par leur tendance à explorer les paramètres ou à raffiner la réponse du méta-modèle. L'exploration est caractérisée par le choix de nouveaux calculs ou l'estimation d'erreur est grande. Le raffinement est caractérisé par le choix de nouveaux points où il y a de fortes discontinuités. Une démarche trop exploratrice peut ne pas être assez précise proche des zones à forts gradients. A l'opposé, un critère qui raffine va placer les points proches des forts gradients de la solution pour mieux les capter. Le risque d'une fonction qui raffine est de trop se concentrer sur une zone et de passer à côté d'une caractéristique de la fonction objectif.

Le choix du critère de raffinement est donc déterminant pour obtenir le meilleur méta-modèle possible à l'issue de l'*adaptive sampling*.

Critère de raffinement

Le critère de raffinement RC utilise une estimation d'erreur $\hat{\mathcal{E}}^\star$, les données d'entraînement et/ou les estimations pour donner une mesure du potentiel d'information pour chaque point d'une grille fine $\mathcal{X}_{fine}$.

$$\mathcal{X}_{fine}, \hat{\mathcal{Y}}^\star, \hat{\mathcal{E}}^\star, \mathcal{X}, \mathcal{Y} \mapsto \mathcal{RC}$$

Dans l'article [16], un état de l'art des méthodes d'adaptive sampling pour des processus gaussiens est réalisé. Une quinzaine de critères de raffinements sont décrits et comparés sur des cas test. Il a donc été choisi de se baser sur cet article pour choisir une fonction d'acquisition adaptée.

3.1.3 *Réalisation d'un méta-modèle par adaptive sampling*

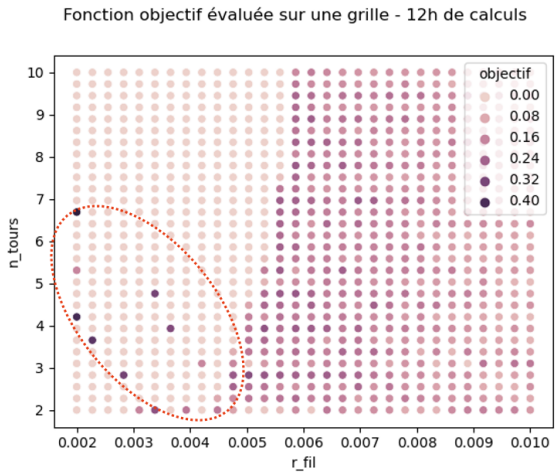
Optimisation de forme du ressort Pour le cas test du ressort, une méthode d'adaptive sampling a été réalisée pour optimiser les paramètres de forme du modèle. Ces paramètres de forme sont :

- Le rayon du fil variant de $2mm$ à $10mm$
- Le nombre de spires variant de 2 tours à 10 tours

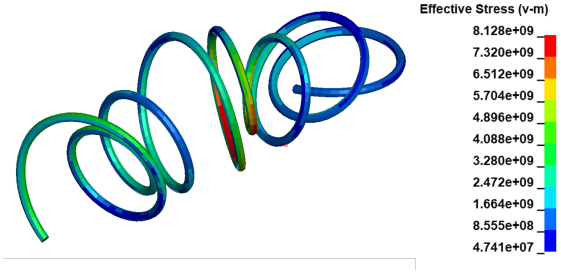
Le pas du ressort est lié au rayon du fil de manière à obtenir $0.1mm$ de jeu entre chaque spire. Le rayon du ressort est fixé à $25mm$. Le but sera de limiter l'accélération maximale en tout point du ressort en assurant une contrainte sur l'angle d'inclinaison du ressort. Cet angle ne devra pas dépasser 45°. Le ressort est excité par vitesse initiale $v_{ini} = 25m.s^{-1}$. Cet objectif est traduit par la fonction objectif à maximiser :

$$f(a_{max}, \theta_{max}) = \frac{1000}{1000 + a_{max}} \left(1 - \frac{1}{1 + e^{45-\theta_{max}}}\right)$$

Cette étude n'est pas liée à un cas d'application réel. De fait, les critères et valeurs sont arbitraires. Cependant, j'ai choisi des valeurs pouvant correspondre à un impact avec une branche d'arbre à $90km/h$. J'ai considéré le diamètre du ressort comme fixe, comme s'il était imposé par un socle préexistant. La valeur limite 45°est arbitraire mais permet d'ajouter une contrainte à l'optimisation.



(a) Grille d'exploration paramètrique



(b) Solution non physique

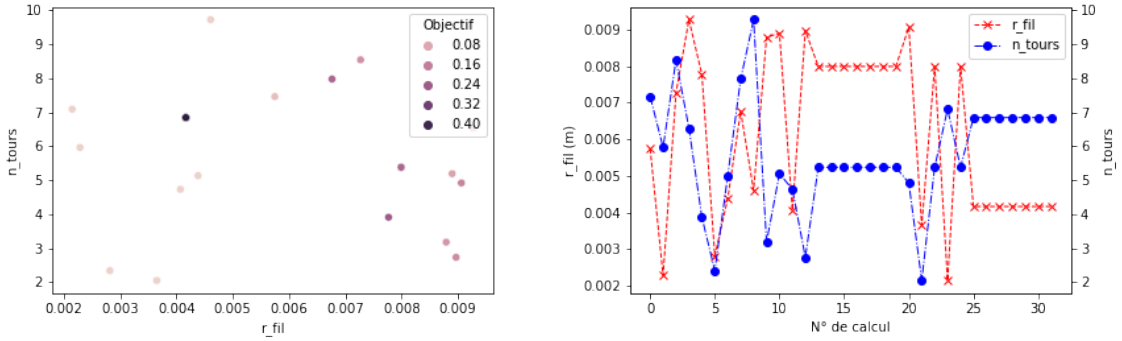
FIGURE 9 : Exploration paramétrique du ressort support d'antenne

Une surface de réponse de référence de la fonction objectif a d'abord été créée. Pour cela, une grille uniforme de 300 points a été réalisée. La figure Fig(9a) présente les résultats obtenus. On remarque que la fonction objectif est nulle sur un domaine qui correspond à la contrainte sur l'angle. Néanmoins certaines configurations font exceptions et obtiennent un meilleur score. Elles sont entourées en pontillés. En observant les résultats pour cette configuration Fig(9b), on remarque que ce sont des solutions qui ne sont pas physiquement possibles. En effet, les contraintes de Von-Mises atteignent $8GPa$, bien au dessus des résistances mécaniques des aciers. Pour garantir la qualité des solutions, il faudrait ajouter un critère physique à la fonction objectif. Par exemple en prenant en compte le maximum des

contraintes de Von-Mises. Toutefois, pour cette fonction objectif, la configuration optimale semble être parmi ces points non physiques.

Pour réaliser l'adaptive sampling, j'ai choisi d'utiliser un modèle de processus gaussiens car il n'y a qu'une seule valeur à prédire. Le critère de raffinement choisi est la borne haute du processus gaussien. C'est un critère simple qui est adéquat pour une recherche de maximum. Le plan d'expérience initial est composé de 10 points.

La figure Fig(10a) présente les points qui ont été sélectionnés après l'adaptive sampling. La figure Fig(10b) montre l'évolution des paramètres de forme du ressort au cours de l'adaptive sampling.



(a) Positions des points

(b) Evolution des paramètres

FIGURE 10 : Adaptive sampling du ressort support d'antenne

On constate que la méthode a calculé plusieurs fois les mêmes jeux de paramètres. Cela est dû à un mauvaise réglage du processus gaussien. En effet, dans la figure Fig(11) on remarque que la réponse du méta-modèle est irrégulière et ne ressemble pas à la fonction objectif Fig(9a). Cette réponse contient des maximums proches des points d'entrainement. La méthode d'adaptive sampling est ainsi "piègée" dans cette zone.

Ce comportement a pu être reproduit en 1D avec des données test. Dans la figure Fig(12), le processus gaussiens doit approximer la fonction objectif en pointillées pour 3, 4, 5 et 6 points d'entraînement. La réponse du méta-modèle est tracé en ligne pleine bleu et l'estimation d'erreur est représenté par l'intervalle orange. On remarque que quand le nombre de points est insuffisant, la réponse du méta-modèles reste nulle loin des points d'entraînement. Ce phénomène est contrôlé par les réglages du modèle de covariance. Notamment un paramètre d'échelle.

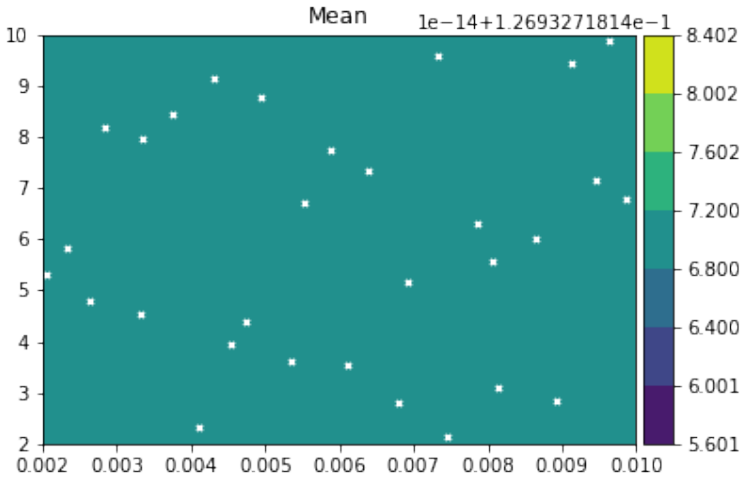


FIGURE 11 : Mauvaise prédiction du processus gaussien

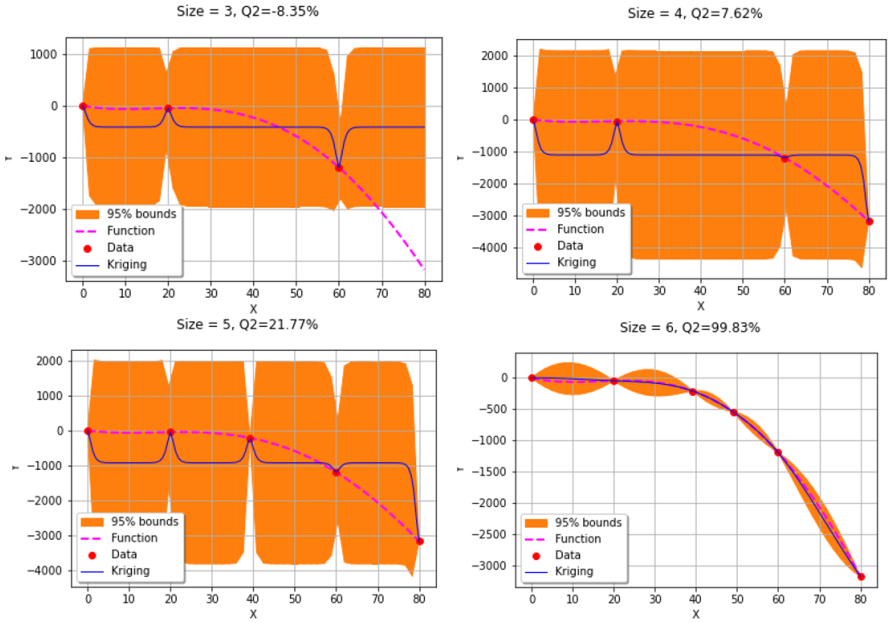


FIGURE 12 : Adaptive sampling en 1D sur un cas test

Finalement, je n'ai pas pu régler correctement les processus gaussiens et relancer les calculs avant la fin de mon stage. Il a donc été conclu que ces méthodes d'adaptive sampling étaient prometteuses, car elles ont déjà été utilisées dans la littérature [16]. Néanmoins, elles sont complexes à implémenter et un travail supplémentaire pour être utilisables pour un cas d'application commercial.

3.2 Réduction de modèles

L'autre approche d'approximation globale qui été envisagée pour ce stage est la réduction de modèles. Cette méthode permet de prédire des champs entiers de solutions. Il est ainsi possible

de mieux comprendre et interpréter les résultats prédits. Avoir des solutions complètes a aussi un intérêt commercial car il permet de faire des visualisations rapides pour des nouveaux paramètres. Ainsi, un rapport d'étude pourrait être complété par un modèle paramétré capable de rejouer les résultats indépendamment de tout logiciel commercial.

Plus précisément, une base réduite composée de vecteurs solutions est d'abord construite. C'est la phase d'entraînement, mais elle est appelée phase hors ligne dans la littérature dans le cas de la réduction de modèles. Cette base réduite doit être composée d'un nombre minimum de vecteurs et doit capter au maximum les variations des solutions en fonction des paramètres. Ce qui veut dire qu'en faisant des combinaisons linéaires des vecteurs de la base, on doit pouvoir obtenir toutes les solutions possibles de l'espace paramétrique. Utiliser une base réduite permet de tirer profit des corrélations entre les données du vecteur solution. En effet, sur une pièce mécanique par exemple, deux points proches sont susceptibles d'avoir des contraintes internes similaires. De ce fait, le nombre de variables objectifs pour la méthode de régression est réduit.

La deuxième phase est celle d'exploitation. On l'appelle phase en ligne pour la réduction de modèles. Cette étape consiste à projeter des nouveaux paramètres dans la base réduite. Traditionnellement, cette projection utilise l'équation utilisée dans le modèle de simulation numérique. C'est le cas par exemple de la projection de Galerkin. Ces méthodes sont intrusives, car il faut modifier les implémentations du modèle numérique pour les mettre en place. Ces méthodes sont donc difficiles à mettre en place, surtout quand on utilise des codes de calculs du commerce.

Pour s'affranchir du caractère intrusif, des méthodes dites non intrusives ont été développées. L'objectif est de capter la relation entre les paramètres et les coordonnées des solutions projetées dans la base réduite. Ainsi, une étape d'entraînement du méta-modèle est ajoutée à la phase hors-ligne. Ensuite, pour prédire une nouvelle solution, on utilise le méta-modèle et on réalise une combinaison linéaire entre les coordonnées prédites et les vecteurs de la base réduite. C'est cette approche qui est utilisée pour Ingeliance. En effet, elle permet l'utilisation de résultats de simulation provenant de n'importe quel logiciel de simulation numérique.

Le schéma Fig(13) résume les étapes de la réduction de modèle.

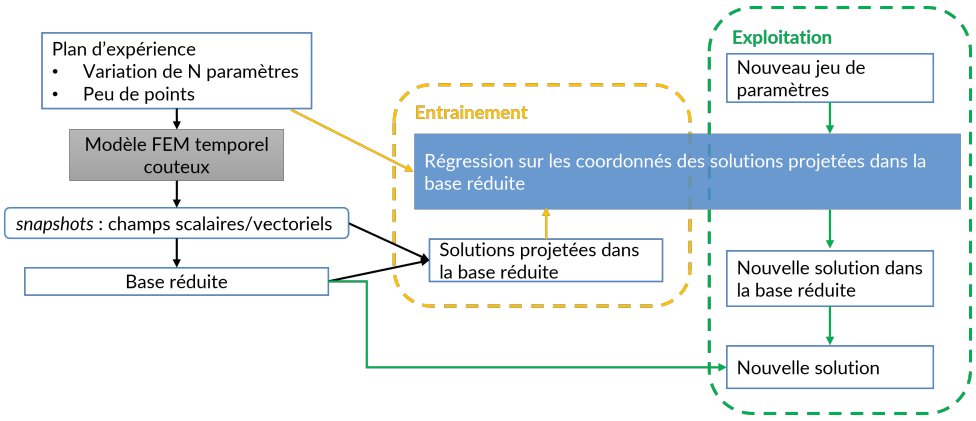


FIGURE 13 : Méthodologie de réduction de modèles non intrusive

3.2.1 Création de bases réduites par SVD

La base réduite doit permettre de construire une solution à approximer à partir d'une collection de solutions qu'on appelle *snapshots*. Cette collection de solutions doit correctement balayer les paramètres du problème. Un algorithme s'appuiera sur les *snapshots* pour construire des vecteurs représentatifs des solutions. C'est la construction de la base réduite. Les solutions peuvent être

projetées dans la base réduite. Ainsi une solution exprimée par un grand nombre de valeurs va être exprimée par une collection de coefficients spécifiant la contribution de chaque vecteur de la base réduite.

L'algorithme utilisé pour construire la base réduite est la Décomposition en Valeurs Singulières (SVD). Cet algorithme factorise une matrice en vecteurs singuliers. Ces vecteurs ont une valeur singulière associée. Plus la valeur singulière est élevée, plus le vecteur associé permet de reconstruire les colonnes de la matrice.

En résumé, on peut reprendre la méthode avec des notations mathématiques. Tout d'abord, les solutions sont collectées dans $\mathcal{Y}$ comme cela a été décrit en fin de section 3. Ces vecteurs solutions sont organisés sous forme de matrice Y de taille (N_s, N) avec N_s la taille de la solution.

$$Y = [y_1|y_2|\dots|y_N]$$

La SVD est appliquée à la matrice Y.

$$Y = W\Sigma Z$$

avec W la matrice des vecteurs singuliers, Σ la matrice diagonale des valeurs singulières décroissantes, Z la matrice adjointe qui n'est pas utilisée ici.

En ne gardant que les vecteurs singuliers associés aux valeurs singulières les plus élevées, on peut créer une base réduite capable de reconstruire la majorité des colonnes de la matrice. C'est la *Proper Orthogonal Décomposition* (POD). Le critère de troncature peut être un ratio vis-à-vis de la première valeur singulière. La base réduite créée est notée $\mathbb{V}$. Elle est formée des premières colonnes de W et ainsi $\mathbb{V} \approx W$. Cette opération est présentée dans la figure Fig(14) avec k valeurs tronquées.

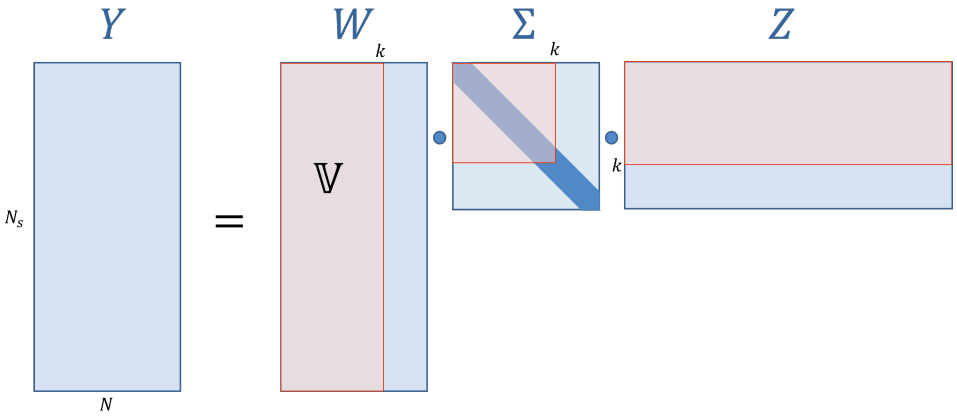


FIGURE 14 : Matrices de la POD

Enfin, pour une solution donnée, on peut calculer les coefficients réduits u_r associés en la projetant dans l'espace de la base réduite.

$$u_r = \mathbb{V}^T y$$

Il existe plusieurs façons de créer des bases réduites. Toutes utilisent une collection de solutions numériques comme données. J'ai utilisé la méthode de la POD car elle est simple à implémenter. Néanmoins, pour mon code, j'ai suivi la méthode par double POD de l'article [17]. Cette méthode permet de traiter des solutions temporelles rapidement. La première POD, est faite sur chaque matrice de solutions temporelles. Puis une seconde POD est faite vis-à-vis des paramètres. Cela permet de traiter deux matrices de tailles raisonnables rapidement.

3.2.2 Régressions entre les paramètres et les coordonnées réduites

Pour pouvoir prédire de nouvelles solutions, il faut capter les relations entre les paramètres et les solutions. Cependant, les méthodes de régressions deviennent complexes et coûteuses à entraîner quand il faut prédire un grand nombre de sorties. C'est pourquoi l'outil de régression va seulement apprendre une relation entre les coefficients associés à chaque vecteur de la base réduite. Chaque coefficient va indiquer la contribution d'un vecteur de la base réduite à la solution. Ce sont les coefficients réduits.

Ainsi, l'outil de régression $\hat{f}$ va prédire un vecteur de coefficients réduits u_r associé à un jeu de paramètres x.

La base de donnée d'apprentissage utilise les solutions calculées $\mathcal{Y}$ à partir des paramètres $\mathcal{X}$, pour calculer une collection de vecteurs de coefficients réduits $\mathcal{U}_r$. Ainsi, l'apprentissage supervisé se fait avec $(\mathcal{X},\mathcal{U}_r)$.

Pendant la phase d'exploitation, un nouveau jeu de paramètres est donné à l'outil de régression qui prédit les valeurs des coefficients réduits. La solution est obtenue en utilisant la base réduite.

4 Applications

Les méthodes d'approximation globale développées précédemment sont non intrusives, elles peuvent ainsi être appliquées à partir du moment où des résultats de simulation numérique sont disponibles. Ainsi, des cas d'applications tests ont été réalisés, d'une part pour tester les méthodes, d'autre part pour créer des exemples utiles à la réalisation future de nouveaux cas d'applications.

4.1 Outils développés

Toutes les méthodes de *machine learning* ont été réalisées dans le langage Python. Tout au long du stage, j'ai implémenté les différentes méthodes via des programmes *notebook jupyter*. Une fois une méthode suffisamment maîtrisée, elle est ajoutée à une librairie locale. L'utilisation de cette librairie permet ensuite de cacher les lignes de code qui n'ont pas d'apports pour l'approximation globale. Ainsi, le code montre uniquement les paramètres et étapes des méthodes. De fait, la librairie permet de :

- Réaliser des plans d'expériences numérique
- Créer des bases de données
- Créer des bases de données par adaptive sampling
- Créer des modèles réduits à partir d'une base de données

Un guide d'utilisation de la librairie avec des exemples sous formes de *notebook* Jupyter à été fourni à Ingeliance.

4.2 Barre de Taylor

Le cas d'application de la barre de Taylor consiste à projeter un cylindre métallique contre un plan incliné infiniment rigide et indéformable Fig(15). Comme la vitesse d'impact est élevée, la barre se déforme plastiquement en une "patte d'éléphant". Le modèle est paramétré par l'angle de la plaque θ et la vitesse initiale du cylindre v_{ini}.

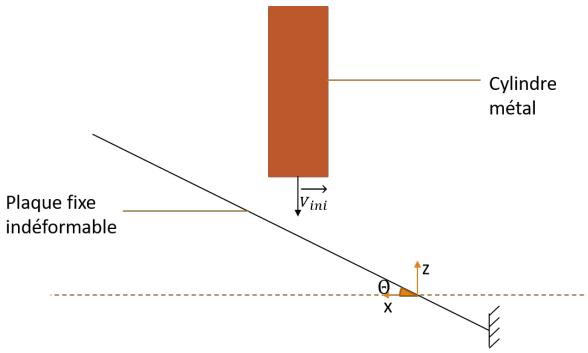


FIGURE 15 : Schéma du cas test de la barre de Taylor

Pour ce cas test, l'objectif est de réaliser quatre modèles réduits. Trois sur les champs de déplacements et un sur le champ des contraintes de Von-Mises. L'objectif est de pouvoir rejouer les résultats rapidement pour différentes vitesses et différents angles.

Dans la figure Fig(16), les quatre champs de solutions sont utilisés pour reconstruire la barre de Taylor. On observe de droite à gauche pour un angle $\theta = 30°$ au dernier pas de temps : la

solution de référence, la solution prédite et l'erreur de la prédiction par rapport à la référence. L'apprentissage à été réalisé sur 10 solutions avec θ variant de 0° à 80°.

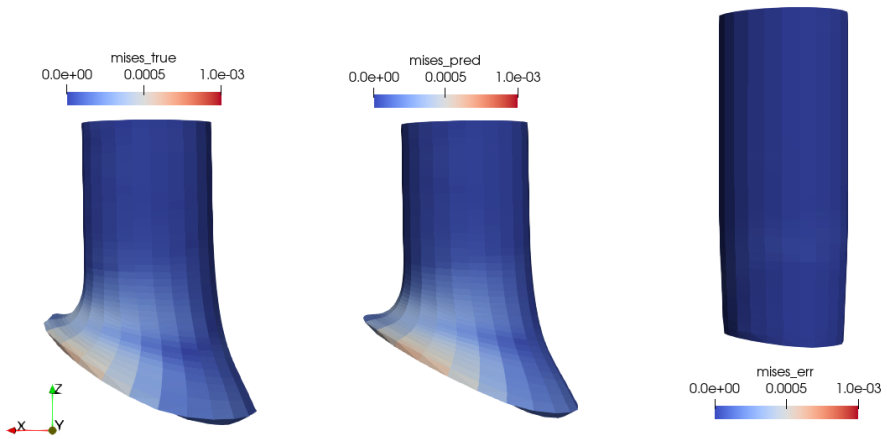
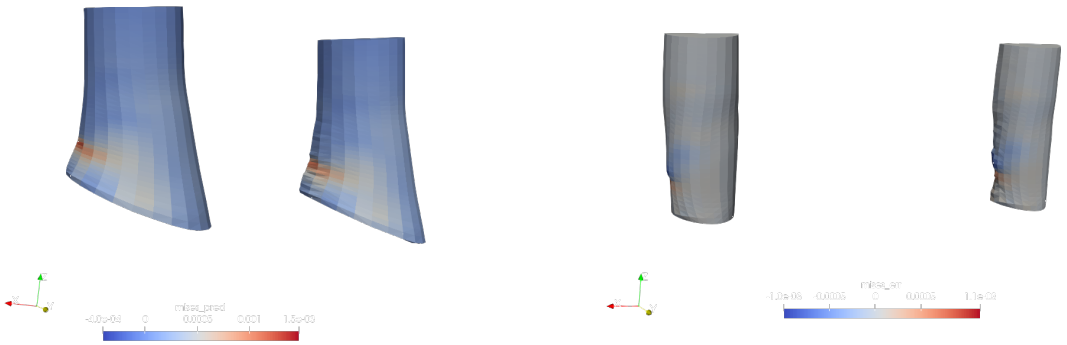


FIGURE 16 : Schéma du cas test de la barre de Taylor

On constate que la barre est correctement prédite pour ce pas de temps. En effet, la visualisation de l'erreur, montre une barre très peu déformée. La solution de référence et la solution prédite sont donc très proches.

Dans la figure Fig(17), la solution et l'erreur pour deux méthodes de régressions différentes sont comparées. La RandomForest prend en compte les corrélations entre les données alors que la RandomForest Multiouput ne les prend pas en compte. On remarque que l'erreur est moins régulière dans ce cas. Ce comportement est cohérents et montre l'importance de prendre en compte la corrélation au sein des données.



(a) Solution pour le dernier pas de temps

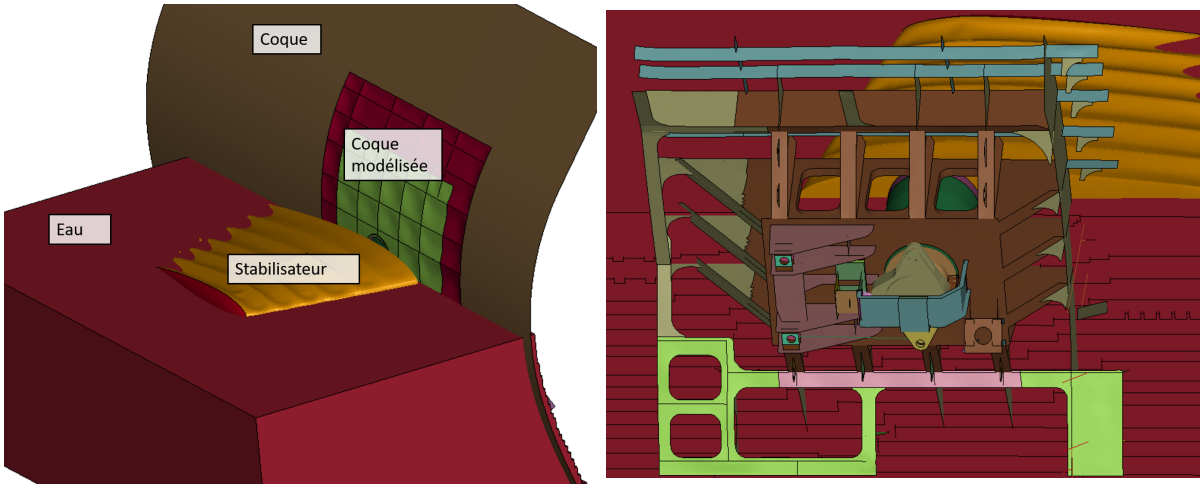
(b) Erreur pour le dernier pas de temps

FIGURE 17 : Comparaison entre les modèles RandomForest et RandomForest Multioutput

4.3 Stabilisateur

Ce cas d'application est un stabilisateur de navire. Cet élément permet au navire de contrôler son assiette et ainsi de se déplacer dans le plan vertical.

Un modèle élément fini sous LS-DYNA a été créé pour simuler une explosion proche d'un stabilisateur. L'objectif est de vérifier que la structure résiste et qu'aucune fuite n'apparaisse. Il est composé d'une partie mécanique et d'un domaine fluide immergé (le fluide traverse le stabilisateur). Les éléments fluides à l'intérieur du stabilisateur et de la coque sont composés d'air. Les autres éléments sont composés d'eau. Sur les figures Fig(18a) et Fig(18b), on observe le stabilisateur, les éléments mécaniques et une section du domaine fluide.



(a) Parties du modèle

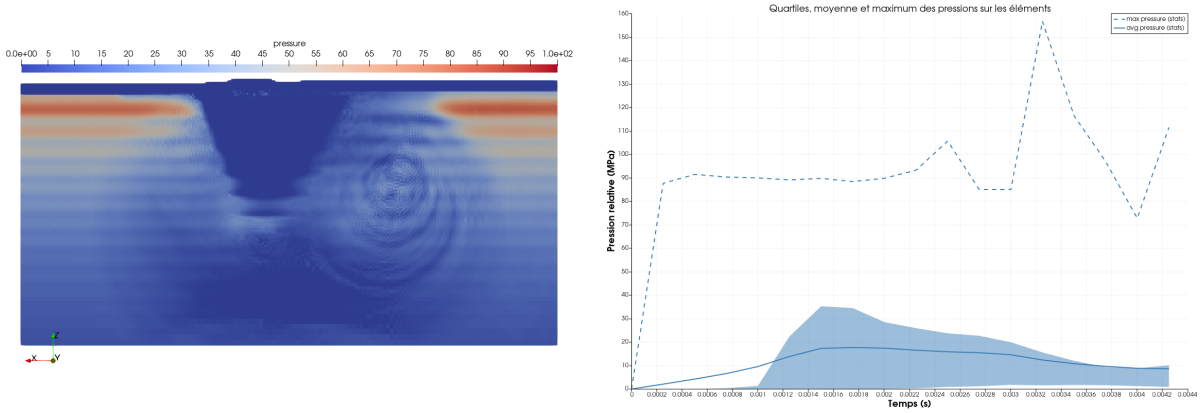
(b) Mécanique du stabilisateur

FIGURE 18 : Modèle LS-DYNA du stabilisateur

L'intérêt de ce cas test est le nombre d'éléments. En effet, le domaine fluide est composé de 3 millions d'éléments. La résolution du modèle est donc assez longue (16 heures sur 4 cœurs pour 18 pas de temps). L'objectif est donc de tester la méthode de réduction de modèle pour une taille de solution conséquente. Seule la pression dans le domaine fluide va être étudiée. Le temps est traité comme l'unique paramètre de la simulation. On observera dans un premier temps la capacité du modèle réduit à interpoler des pas de temps. Puis on étudiera l'influence du nombre de vecteurs dans la base réduite.

Pour cette réduction de modèles, les résultats ont été récupérés pour les 18 pas de temps. Le modèle de régression choisi est l'ExtraTreeRegressor de la librairie *scikit-learn*. C'est une forêt aléatoire qui permet de réduire la variance des prédictions et la moyenne. Les méta-paramètres de cet outil de régression sont optimisés par une optimisation paramétrique bayésienne.

La solution obtenue au bout de $0,003$ s est visible figure Fig(18a). En rouge, le front de l'onde de choc a atteint la paroi et est en train d'être réfléchi. On distingue la forme du stabilisateur au milieu de la figure. Les cercles concentriques à droite du stabilisateur sont des artefacts numériques. Dans la figure Fig(19b), des statistiques sur la pression dans les éléments du domaine fluide sont tracées. L'aire représente le premier et le dernier quartile. La ligne foncée représente la moyenne des pressions et la ligne pointillée le maximum. L'onde de choc entre en contact avec le stabilisateur à $t = 0.001s$. Cet évènement est identifiable via la première croissance du dernier quartile. L'onde est réfléchie sur la paroi à $t = 0.0035s$, ce qui est identifiable via le pic de maximum.

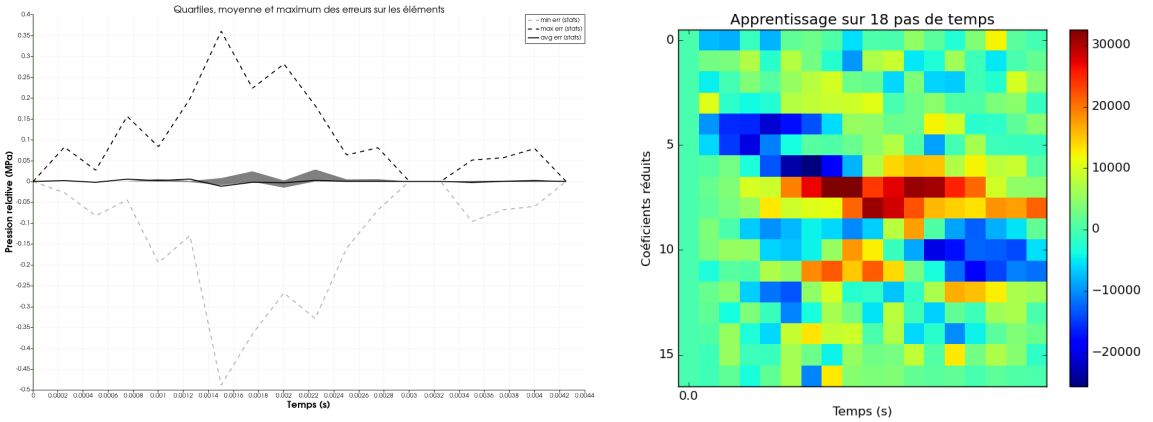


(a) Pression dans une coupe du domaine fluide à $t = 0.003$

(b) Évolution de la pression dans les éléments dans la coupe

FIGURE 19 : Résultats de la simulation

En utilisant tous les pas de temps pour l'apprentissage et en optimisant les méta-paramètres du méta-modèle, on obtient un modèle réduit capable de rejouer exactement les résultats. L'erreur est alors dans le pire des deux cas d'un ordre de grandeur de 1% Fig(20a). En moyenne elle est d'un ordre de grandeur de $1E-2$ soit 0.01%. Pour mieux comprendre comment les vecteurs de la base réduite contribuent à chaque solution prédite, on peut représenter les coefficients réduits pour chaque solution Fig(20b). Ainsi pour une prédiction aux 18 pas de temps d'apprentissage avec une base réduite de 18 vecteurs, on observe que les vecteurs 4 à 11 contribuent à la solution de manière successive. Cette succession laisse supposer que la propagation de l'onde est captée par les combinaisons de ces vecteurs.



(a) Erreur maximum, minimum, moyenne et quartiles sur la pression dans les éléments

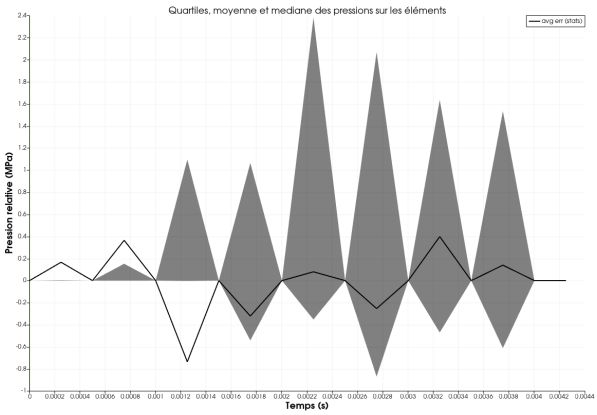
(b) Visualisation des valeurs des coefficients réduits prédit par l'outil de régression

FIGURE 20 : Résultats du modèle réduit pour rejouer les données d'apprentissage

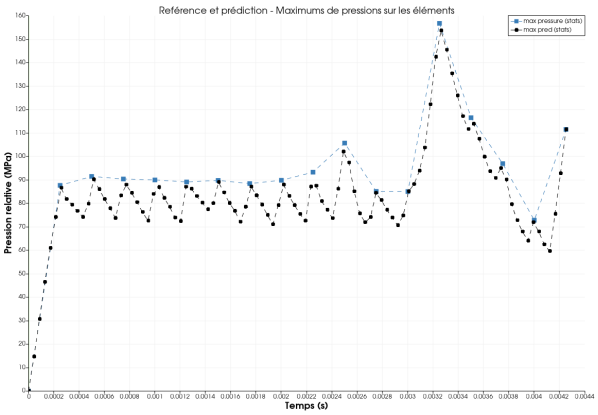
Interpolation de pas de temps : Pour tester la capacité du modèle réduit à interpoler des pas de temps, certains pas de temps des résultats ne sont pas utilisés pendant la création de la base réduite et pour l'apprentissage du modèle de régression. La solution est ensuite prédite pour tous les pas de temps et on peut observer l'erreur pour des pas de temps qui n'ont pas été utilisés pendant l'apprentissage.

Dans la figure Fig(21a), on trace les statistiques sur l'erreur entre la pression prédite par

le modèle réduit et la pression simulée. On remarque qu'elle est plus élevée aux pas de temps non appris. En plaçant 100 pas de temps dans l'étendue des 18 pas de temps d'apprentissage Fig(21b), on constate que le modèle réduit reste précis aux pas de temps d'apprentissage. En effet, la pression maximum prédite (pointillés) s'éloigne de la pression maximum (aire claire).



(a) Erreur moyenne et quartiles sur la pression dans les éléments - Interpolation aux pas de temps impairs



(b) Pression prédite et simulée - Pour 100 pas de temps de prédiction et 18 pas de temps simulés et d'apprentissage

FIGURE 21 : Résultats pour des interpolations

Finalement, je constate que ce modèle réduit n'est pas capable d'interpoler correctement la solution entre des pas de temps d'apprentissage. En effet, si certaines valeurs intégrées, comme la moyenne ou les quartiles, sont bien prédites, la propagation de l'onde de choc n'a pas été captée. C'est un résultat cohérent avec la construction du modèle réduit. La méthode ne fait qu'ajouter ou soustraire des vecteurs solutions bien choisis. De fait, elle ne pourra pas prédire l'onde de choc dans une zone où elle n'a jamais été présente.

Pour mieux capter cette propagation, des méthodes non-linéaires comme par exemple des auto-encodeurs sont peut-être plus appropriées.

5 Conclusion

L'objectif de ce stage était de réaliser des méta-modèles de simulations numériques. Pour cela, j'ai réalisé une étude bibliographique des différents couplages entre *machine learning* et simulation numérique. A la suite de laquelle, nous avons choisi de se concentrer sur des problématiques d'approximation globale. D'abord, pour approximer des surfaces de réponses, les méthodes d'*adaptive sampling* réduisent le nombre de simulations réalisées pour explorer un espace paramétrique. Ensuite, j'ai implémenté des méthodes de réduction de modèles non intrusives via POD pour de prédire des champs entiers de solutions. Ces méthodes ont été implémentées dans une librairie Python dans le but d'être réutilisée par Ingeliance. Cette librairie a ensuite été utilisée pour réaliser des cas d'applications tests.

Finalement, les deux méthodes étudiées ont permis d'obtenir des résultats encourageants. Les deux méthodes sont non intrusives et sont ainsi adaptables à tous les codes de simulation numérique existants. La méthode d'adaptive sampling permet d'obtenir des plans d'expériences plus petits et captant mieux les comportements de certaines valeurs intégrées sur les résultats. Cependant, le réglage des processus gaussiens et son influence sur le choix de différents critères de raffinement n'a pas été abordé.

D'autre part, les méthodes de réduction de modèles, permettent de rejouer les résultats et même d'interpoler les résultats pour différentes valeurs de paramètres. Cependant, si des valeurs résumant les résultats sont correctement prédites, la méthode n'est pas capable de capter des phénomènes physiques comme la propagation d'une onde. Pour adresser cette limite, une piste serait de s'intéresser aux méthodes de créations de bases non linéaires, comme par exemples les réseaux Auto Encodeurs.

La réalisation des cas test et la facilité de prise en mains des outils livrés montre un potentiel fort pour intégrer ces méthodes dans les méthodologies de simulations numériques. L'intérêt est donc d'accélérer une optimisation paramétrique, de capitaliser sur des résultats incomplets ou encore de livrer un modèle réduit aux clients pour présenter d'une nouvelle façon les résultats.

D'un point de vue personnel, ce stage m'a en premier lieu apporté des compétences techniques, notamment en Python et en *machine learning*. De plus, j'ai découvert le monde du service en ingénierie et par la même occasion la relation clients. Enfin, ce stage débouche sur un projet de thèse CIFRE avec Ingeliance et en lien avec l'INRIA dans le but d'approfondir le sujet de réduction de modèles.

Références

[1] A. M. Turing. "I.— Computing Machiney and Intelligence". In : *Mind* LIX.236 (oct. 1950), p. 433-460. ISSN : 0026-4423. DOI : 10.1093/mind/LIX.236.433. eprint : https://academic.oup.com/mind/article-pdf/LIX/236/433/30123314/lix-236-433.pdf. URL : https://doi.org/10.1093/mind/LIX.236.433.

[2] Scott McKinney et al. "International evaluation of an AI system for breast cancer screening". In : *Nature* 577 (jan. 2020), p. 89-94. DOI : 10.1038/s41586-019-1799-6.

[3] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In : *Nature* 529 (jan. 2016), p. 484-489. DOI : 10.1038/nature16961.

[4] Ricardo Vinuesa et Steven L. Brunton. "The Potential of Machine Learning to Enhance Computational Fluid Dynamics". In : (2021). DOI : 10.48550/ARXIV.2110.02085. URL : https://arxiv.org/abs/2110.02085.

[5] Laura von Rueden et al. "Combining Machine Learning and Simulation to a Hybrid Modelling Approach : Current and Future Directions". In : *Advances in Intelligent Data Analysis XVIII*. Sous la dir. de Michael R. Berthold, Ad Feelders et Georg Krempl. Cham : Springer International Publishing, 2020, p. 548-560. ISBN : 978-3-030-44584-3.

[6] Kaveh Amouzgar et al. "Metamodel-based multi-objective optimization of a turning process by using finite element simulation". In : *Engineering Optimization* 52.7 (2020), p. 1261-1278. DOI : 10.1080/0305215X.2019.1639050. eprint : https://doi.org/10.1080/0305215X.2019.1639050. URL : https://doi.org/10.1080/0305215X.2019.1639050.

[7] Bastian Bohn et al. "Analysis of Car Crash Simulation Data with Nonlinear Machine Learning Methods". In : *Procedia Computer Science* 18 (2013). 2013 International Conference on Computational Science, p. 621-630. ISSN : 1877-0509. DOI : https://doi.org/10.1016/j.procs.2013.05.226. URL : https://www.sciencedirect.com/science/article/pii/S1877050913003694.

[8] Jonathan Tompson et al. "Accelerating Eulerian Fluid Simulation With Convolutional Networks". In : *CoRR* abs/1607.03597 (2016). arXiv : 1607.03597. URL : http://arxiv.org/abs/1607.03597.

[9] Amitava Choudhury. "Random Forest Regression-Based Machine Learning Model for Accurate Estimation of Fluid Flow in Curved Pipes". In : *Processes* 9 (nov. 2021). DOI : 10.3390/pr9112095.

[10] Alexandre Bourriaud, Raphaël Loubère et Rodolphe Turpault. "A Priori Neural Networks Versus A Posteriori MOOD Loop : A High Accurate 1D FV Scheme Testing Bed". In : *Journal of Scientific Computing* 84.2 (août 2020). DOI : 10.1007/s10915-020-01282-1. URL : https://hal.archives-ouvertes.fr/hal-03084463.

[11] Kevin Luna, Katherine Klymko et Johannes Blaschke. "Accelerating GMRES with Deep Learning in Real-Time". In : (mars 2021).

[12] Kai Fukami, Koji Fukagata et Kunihiko Taira. "Super-resolution reconstruction of turbulent flows with machine learning". In : *Journal of Fluid Mechanics* 870 (mai 2019), p. 106-120. DOI : 10.1017/jfm.2019.238. URL : https://doi.org/10.1017%2Fjfm.2019.238.

[13] Stefano Markidis. *The Old and the New : Can Physics-Informed Deep-Learning Replace Traditional Linear Solvers ?* 2021. DOI : 10.48550/ARXIV.2103.09655. URL : https://arxiv.org/abs/2103.09655.

[14] Hongyu Ren et al. *Adversarial Constraint Learning for Structured Prediction*. 2018. DOI : 10.48550/ARXIV.1805.10561. URL : https://arxiv.org/abs/1805.10561.

[15] Edwin V Bonilla, Kian Chai et Christopher Williams. "Multi-task Gaussian Process Prediction". In : *Advances in Neural Information Processing Systems*. Sous la dir. de J. Platt et al. T. 20. Curran Associates, Inc., 2007. URL : https://proceedings.neurips.cc/paper/2007/file/66368270ffd51418ec58bd793f2d9b1b-Paper.pdf.

[16] Jan Nikolas Fuhg, Amélie Fau et Udo Nackenhorst. “State-of-the-art and Comparative Review of Adaptive Sampling Methods for Kriging”. In : *Archives of Computational Methods in Engineering* (août 2020). doi : 10.1007/s11831-020-09474-6. url : https://hal.archives-ouvertes.fr/hal-02919440.

[17] Qian Wang, Jan Hesthaven et Deep Ray. “Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem”. In : (juin 2018).

Abstract

The simulation division of Ingeliance, a company specialized in numerical simulations studies for other corporations, aims to leverage the assets of machine learning algorithms. During this internship, I explored the possibilities of couplings between numerical simulations and machine learning methods. The first objective was to conduct a study on existing methods to focus on those relevant to Ingeliance. Consequently, the work was centered on building a global approximation of an objective function. In other words, train meta-models on a finite number of call of a computationally expensive numerical simulation.

Two approaches were conducted. Adaptive sampling methods and non-intrusive reduced order models. Adaptive sampling methods relies on a meta-model, an estimation of errors and an acquisition function. The meta-model is first constructed with a minimal number of objective function evaluations. Then, new evaluations are computed sequentially to most improve the meta-model response. This iterative enrichment phase is halted when a precision criterion is reached or computational budget is exhausted. Non-intrusive reduced order models allow the prediction of complete fields of simulation results. Multiple calls to the objective function are used to construct a reduced basis. Then, a meta-model learns the relationship between the parameters and results represented in the reduced basis.

Finally, a meta-model will be constructed to explore parameters for the spring mount of an antenna on a vehicle's roof. Both implemented methods can thus be tested on an industrial application.