

Rapport de stage PHIMECA

Kristof Attila Simady
e-mail: kristof.simady@sigma-clermont.fr

05/03/2020

Contents

1 Premiers 5 mois de stage	4
1.1 Introduction	4
1.2 Recherche bibliographique	4
1.2.1 Travaux retenus	4
1.2.2 Méthodes et outils de travail	4
1.3 Analyse de la problématique, premier tests et prise en main des outils	5
1.3.1 Mission confiée et tâches associées	5
1.3.2 Prise en main d'openTURNS et de la théorie sur les champs stochastiques	5
1.4 Exemple théorique simple, premiers codes	8
1.4.1 Choix des codes et difficultés algorithmiques	9
1.4.2 Premiers essais d'analyse de la sensibilité du modèle	12
1.4.3 Interprétation et essais avec le papier : "Time-variant global reliability sensitivity analysis of structures with both input random variables and stochastic processes"	13
1.5 Application au modèle NASTRAN d'échangeur thermique	21
1.5.1 Définition et analyse de la problématique	21
1.5.2 Mise en place de méthodologie	23
1.5.3 Application de la méthode de la déformation modale métrique pilotée par champ stochastique sur un modèle Simcenter 3D Nastran	28
1.5.4 Commentaires et mise en contexte	29
1.6 Bilan intermédiaire	29
2 Derniers 5 mois de stage	29
2.1 Intégration dans openTURNS	30
2.1.1 Specifications pour l'intégration dans openTURNS	30
2.1.2 Astuces employées pour l'intégration	30
3 Conclusions	32
Appendices	33
Appendices	33

Avant propos

Le cadre d'étude offert par SIGMA Clermont a donné la possibilité d'effectuer une année supplémentaire de stages pour étoffer nos connaissances scientifiques et techniques et découvrir le monde industriel comme des culture étrangères. De par mon appétence pour la recherche et l'informatique, j'ai eu la chance de pouvoir venir effectuer mon stage au sein de l'entreprise **PHIMECA**, située à Clermont-Ferrand, et depuis longtemps en partenariat avec SIGMA.

Il aura été l'occasion pour moi de développer un ensemble de savoir pointus sur l'analyse fiabiliste et les différents outils employés, tout comme d'avoir pu travailler sur un projet de recherche industriel complexe et s'étalant sur une multitude de domaines scientifiques. En plus d'avoir pu me servir des connaissances acquises au cours de ma formation, et approfondir ces dernières, l'utilisation de l'outil informatique et son harmonisation avec les méthodes d'analyse a permis d'acquérir de la fluidité de travail et d'ouvrir l'accès à un grand nombre d'outils, puisque plus on travaille avec des outils informatiques différents, plus l'acquisition d'une nouvelle compétence informatique devient rapide et peu coûteuse en énergie.

Bien sur, tout cela n'aurait pas été possible sans le cadre offert par Phimeca et la liberté d'action dont on jouit en travaillant ici. En effet, bien qu'ayant un sujet assez défini, la responsabilité quant à l'avancement sur le projet, ou encore le choix des outils à utiliser et la méthodologie à employer était des choix à faire soi même. Ce type de liberté d'action implique d'avoir suffisamment de connaissances pour envisager les différentes méthodes de résolution possibles et parvenir à choisir la bonne, et cela implique encore plus d'arriver à rapidement se familiariser seul avec un outil ou domaine de recherche et acquérir suffisamment d'informations sur celui-ci pour parvenir soit à l'employer, soit à juger de sa pertinence dans le projet.

En plus de ce cadre de travail et ce projet formateur, l'ambiance agréable au sein de **PHIMECA** et la présence d'une équipe assez jeune ont rendu ce stage très agréable et motivant.

Je souhaite donc avant tout remercier Mr. Nicolas GAYTON, qui a rendu ce stage possible grâce à ses nombreuses années d'échange et de collaboration avec **PHIMECA**, et la confiance qu'il a porté en mes capacités à évoluer dans ce milieu de travail et m'avoir proposé en tant que stagiaire.

Je souhaite ensuite remercier Mr. Antoine DUMAS, mon tuteur et responsable de l'équipe FIC (Fiabilité Incertitudes Cournon), qui a su trouver un sujet de stage intéressant et formateur, me mettre sur les bonnes pistes dès que des difficultés apparaissaient et qui a su enfin m'accorder suffisamment de confiance pour aller représenter l'implication de **PHIMECA** dans ce projet face aux autres acteurs impliqués.

Mes remerciements vont ensuite à Mr. Thierry YALAMAS, qui a su m'introduire à l'équipe et la philosophie d'entreprise, et qui a veillé à ce que l'ambiance de travail soit toujours la plus propice aux idées innovatrices et à l'épanouissement de

chacun, même dans le cas de force majeure qu'a été cette épidémie de COVID-19.

Je remercie enfin le reste de l'équipe, Oussama B., Fabien T., Julien S. Marc G., Rémi P., Guillaume G., Céline D., Nathalie G., et tout les autres ... (à suivre)

Note: Ce document est autant un rapport de stage qu'une note à qui voudrait se servir des différentes méthodes explorées et codes développés.

1 Premiers 5 mois de stage

1.1 Introduction

Dans le cadre d'analyse probabiliste et fiabiliste, il est d'usage d'effectuer des analyses de sensibilité sur une ou plusieurs grandeurs d'intérêt, grâce à divers moyens comme les indices de Sobol' ou les analyses de corrélation.

Ces analyses sont néanmoins souvent cantonnées à des modèles n'ayant en sortie et entrée que des grandeurs scalaires, pour lesquelles de nombreuses méthodes existent dans la littérature.

L'objectif du travail présenté ici, est d'effectuer ce type d'analyse sur des modèles ayant en entrée plusieurs champs aléatoires et des variables aléatoires scalaires et en sortie aussi des champs et grandeurs scalaires.

Pour compléter cela, il était bien sûr aussi demandé de d'abord créer un modèle simple gouverné par des champs stochastiques et de faire de l'analyse de sensibilité sur ce modèle.

Il sera d'abord présenté différentes méthodes présentes dans la littérature, avec leurs potentiels avantages et défauts.

Cette approche de l'analyse de sensibilité nous renvoie aussi sur des champs de recherche bien différents, car les méthodes peuvent autant venir du champ de la mécanique que de la géologie ou l'étude environnementale.

1.2 Recherche bibliographique

Ce travail de recherche a été l'occasion de parcourir un grand nombre de travaux, liés de plus ou moins loin à l'analyse de sensibilité sur des processus gaussiens. On notera notamment les travaux de [Lilburne and Tarantola, 2009] qui donnent dans leur travail un état de l'art complet sur les différentes méthodes utilisables pour l'analyse de sensibilité sur champs gaussiens.

D'autres techniques plus récentes ont néanmoins vu le jour, et ce sont ces dernières que nous avons explorés en premier. Notamment les travaux de [Wei et al., 2017] sur l'analyse de sensibilité de structures ayant en entrée des variables aléatoires et processus stochastiques variant dans le temps, ou encore le travail de [Pronzato, 2019] qui développe un métamodel entre l'entrée et sortie d'un modèle gouverné par des champs stochastiques.

1.2.1 Travaux retenus

Etant donné que les missions d'analyse de sensibilité et d'incertitudes se font classiquement en se basant sur la bibliothèque *openTURNS* développé par PHIMECA, le choix a été fait d'adapter techniques de recherche et méthodes déjà existantes dans l'API au problème étudié.

1.2.2 Méthodes et outils de travail

L'ensemble des codes ont été écrits dans le langage *Python*, en se servant majoritairement des bibliothèques à caractère scientifique (*SciPy*, *NumPy*) et la bibliothèque développée en partie par **PHIMECA** : *openTURNS* (*open treatment of Uncertainties and Risk & Statistics*)[Michaël Baudin and Popelin, 2015].

PHIMECA a fourni un poste de travail Linux, tout comme un accès au serveur

de calcul pour les modèles un peu plus conséquents.

1.3 Analyse de la problématique, premier tests et prise en main des outils

1.3.1 Mission confiée et tâches associées

L'objectif du stage était de développer une méthodologie pour l'analyse de sensibilité sur des modèles gouvernés par des champs aléatoires (Champs Stochastiques) et des variables aléatoires, et qui en sortie sont aussi représentables par une collection de champs et variables aléatoires.

Plus spécifiquement, cette méthodologie était à appliquer à un modèle d'échangeur de chaleur air-air commercialisé par **LIEBHERR** et utilisé pour la régulation thermique de l'air ambiant dans les avions de ligne.

Comme la mission confiée au sein de **PHIMECA** se rapprochait d'un travail de recherche, celui -ci était bien sûr accompagné de nombreuses étapes distinctes. En effet, en plus du développement des codes et de l'application analytique, une recherche bibliographique a du être menée en amont pour parvenir à lister les différentes méthodes d'analyse existantes et tester celles qui pourraient convenir au mieux. De plus, une phase d'apprentissage et de tests a du d'abord être nécessaire pour parvenir à prendre en main les différents outils, et comprendre la théorie et les mathématiques sous-jacentes.

1.3.2 Prise en main d'**openTURNS** et de la théorie sur les champs stochastiques

Champs Stochastiques

Un champ stochastique est un champ de variables aléatoires toutes corrélées entre elles, et dont l'intensité de la corrélation est déterminée par leur proximité dans l'espace. Par exemple la position de tout objet est corrélée à sa position aux intervalles de temps proches (continuité du temps), ou bien les températures au dessus d'un pays sont corrélés à faible distance mais l'effet aléatoire est plus présent lorsque les distances sont importantes.

Ce type de modélisation des champs est couramment utilisée pour modéliser des propriétés présentant des variabilités gaussiennes mais une continuité dans leur espace de définition.

Lorsque la corrélation est seulement dépendante de la distance entre deux points de l'espace et non d'une position absolue, on parle d'un champ stationnaire. Lors de cette étude, on se place dans ce cas.

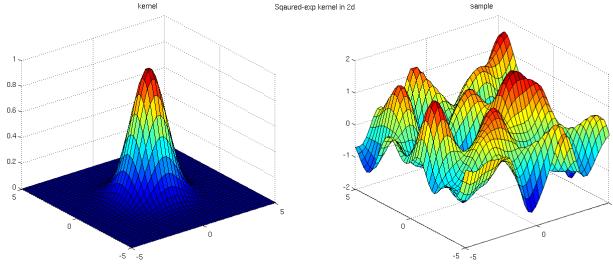


Figure 1: Réalisation d'un champ stochastique en deux dimension. Le champ est continu dans l'espace et la primitive est une gaussienne en deux dimensions.

Mathématiquement, cette corrélation peut être définie par différents modèles. Les deux modèles de corrélation les plus simples sont les fonctions de covariance exponentielles (1) et les exponentielles quadratiques (2).

$$C(d) = \exp(-d/V) \quad (1)$$

$$C(d) = \exp(-(d/V)^2); \quad (2)$$

Avec V étant le paramètre d'échelle et d la norme euclidienne de la distance entre deux points de l'espace considéré. (*plus le paramètre d'échelle est grand, plus le champ est lissé, tout comme le modèle quadratique est aussi plus lisse*) Une autre fonction de covariance, celle de Matérn, est aussi très utilisée, puisqu'elle présente des comportements limites similaires aux deux fonctions exponentielles présentes plus haut. La dérivabilité de ce modèle est réglable via le paramètre ν , ainsi la régularité du champ est facilement modifiable.

$$C_v(d) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{d}{\rho} \right)^\nu K_\nu \left(\sqrt{2\nu} \frac{d}{\rho} \right); \quad (3)$$

La continuité des processus gaussiens et leur comportement quasi onduelaire, permet de les traiter de manière un peu analogue à la décomposition de fourrier, et de les décomposer en une somme infinie de variables aléatoires gaussiennes décorrélées.. Cette méthode de décomposition vient du théorème de *Karhunen-Loève*.

La construction de cette série infinie se fait grâce aux vecteurs propres issus de la matrice de corrélation du modèle, et grâce à une base de vecteurs orthonormaux dans un espace Hilbertien. Les détails mathématiques de cette méthode peuvent être trouvées en [Sudret and Kiureghian, 2000].

Lors de l'approximation de *Karhunen-Loève*, la série est tronquée à l'ordre M .

$H(\mathbf{x}, \theta)$	Approx. Procéssus Gaussien
λ_i	Valeur Propre de la matrice de covariance
ξ_i	Variable Normale Centrée Réduite
$\varphi_i(\mathbf{x})$	Vecteur propre de la matrice de covariance

$$H(\mathbf{x}, \theta) = \sum_{i=1}^M \sqrt{\lambda_i} \xi_i(\theta) \varphi_i(\mathbf{x}); \quad (4)$$

Cette décomposition permet de représenter l'ensemble de variabilité du champ avec des variables gaussiennes décorrélées, et donc de faire des analyses de sensibilités sur cette nouvelle représentation du champ.

openTURNS

openTURNS est initialement un projet de bibliothèque open-source pour le traitement des incertitudes et des risques, commun à trois entreprises fondatrices, **Airbus**, **EDF** et **Phimeca Engineering**, projet ayant débuté en 2005.

Depuis, deux autres organismes, **IMACS** et **l'ONERA** ont rejoint le développement d'*openTURNS*, qui se révèle être un grand atout pour le traitement des incertitudes et l'ingénierie fiabiliste. En effet, le projet d'*openTURNS* est un regroupement d'un ensemble d'algorithmes performants écrits en C++, se basant sur la théorie développée pour les traitement des incertitudes, l'optimisation robuste, et les études de sensibilité.

Un *wrapper*¹ a été utilisée pour lier l'ensemble de la bibliothèque C++ à un module Python, et de pouvoir profiter de la facilité d'utilisation du langage Python et de sa flexibilité, tout en gardant les vitesses d'exécution du C++.

PHIMECA développe des logiciels commerciaux en se basant sur le module *openTURNS*, qui sont plus ergonomiques dans leur utilisation et automatisent certaines parties moins évidentes en utilisation directe de la librairie.

Néanmoins, la librairie reste quand même extrêmement bien documentée, avec la théorie mathématique sous-jacente à chaque algorithme d'explicitée, tout comme de nombreux exemples. La documentation présente de même des cas d'applications précis montrant la logique à avoir lors de l'écriture de codes avec *openTURNS*.

¹Un wrapper ou fonction wrapper, est un programme qui appelle un autre programme. Il permet entre autres d'utiliser du code dans un milieu hétérogène, ici, du C++ depuis Python

1.4 Exemple théorique simple, premiers codes

Pour parvenir à tester les méthodes trouvées lors de la recherche bibliographique, et avoir un modèle simple et rapide à exécuter, un exemple simple a été développé : Il s'agit d'un modèle de poutre en appui sur ses deux extrémités

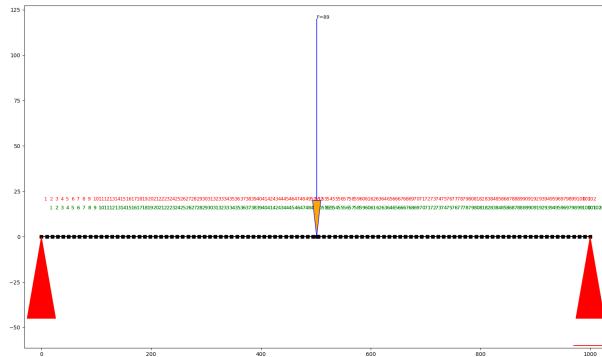


Figure 2: Modélisation d'une poutre en appui sur ses deux extrémités avec la bibliothèque anastruct

La particularité de ce modèle de poutre est, en plus d'être une aberration physique, le fait que le diamètre et le module de young sont gouvernés par un processus gaussien en une dimension, suivant la longueur de la barre. Pour parvenir à modéliser cette variation, le modèle est bien-sûr subdivisé en une centaine d'éléments finis, et le champ discréteisé sur un maillage de même longueur. En plus de ces deux grandeurs gouvernées par des champs, la densité du matériau, la position de la force, et la norme de la force sont représentés par des variables aléatoires Gaussiennes. Cette modélisations'est faite avec l'excellente bibliothèque *Anastruct* de [Vink, 2020].

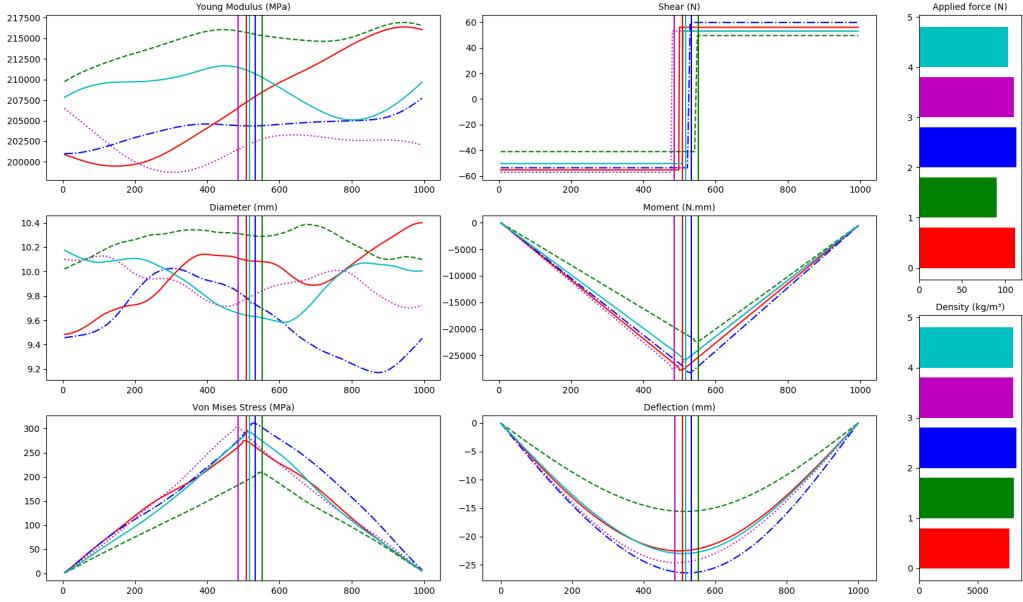


Figure 3: Échantillon de 5 réalisations de la poutre en flexion. [Sur la gauche : Processus gouvernant le module de young, Processus gouvernant le diamètre, contrainte de Von Mises] | [Sur la droite : Contrainte de cisaillement, moment de flexion, déflexion]

Les grandeurs d'étude choisies ici sont la contrainte de Von Mises, la déflexion de chaque noeud, et la déflexion maximale. On pourrait de même choisir une contrainte de Von Mises maximale pour représenter la défaillance.

Comme chacun des processus stochastiques peut être décomposé en une somme finie de variables aléatoires normales grâce à la décomposition de Karhunen-Loeve, l'on peut à l'inverse, générer des champs stochastiques à partir d'une collection de variables aléatoires, dont les paramètres sont issus de l'analyse même du champ.

On considère donc notre modèle plus comme fonction de variables aléatoires et de processus stochastiques, mais seulement comme une fonction de variables aléatoires scalaires.

Ceci permet de faire l'analyse de Sobol classique (avec l'algorithme de sensibilité de *Saltelli*) sur ces nouvelles variables aléatoires.

La difficulté ici, est d'ensuite relier les indices de sensibilité de Sobol des variables aléatoires définissant le champ au champ stochastique même. (On ne souhaite n'avoir qu'un seul indice de Sobol pour un champ.)

1.4.1 Choix des codes et difficultés algorithmiques

Au vu de choix de développer une méthode algorithmique pour l'analyse de sensibilité sur des champs stochastiques, et non pas de juste faire l'analyse de sensibilité sur une seule problématique, il y avait un besoin de robustesse supplémentaire pour l'écriture des différents codes. Ces derniers doivent pouvoir fonctionner avec tout type de fonction en python prenant en entrée des champs et variables aléatoires, et renvoyant un tuple de variables aléatoires.

Pour pouvoir tester ceci, l'exemple de la poutre en éléments fins à été codée à part, et peut être utilisée directement comme fonction, indépendamment de l'analyse de sensibilité. Cette dernière prend de même en entrée deux champs stochastiques et trois variables aléatoires, et peut renvoyer plusieurs arguments. Les tests ont été faits avec la fonction ne renvoyant qu'une collection de champs (contraintes de von mises) et avec la fonction renvoyant un tuple (collection de champs + vecteur déflexion maximale). Ce choix augmente le nombre de vérification qu'il y a à effectuer au sein du code, mais permet l'utilisation d'un plus grand nombre de fonctions, et l'analyse de sensibilité sur l'ensemble des variables de sortie en une seule fois.

Pour parvenir à bien contrôler les processus stochastiques, une classe a été écrite *NdGaussianProcessConstructor*, qui permet de définir entièrement un processus stochastique, et possède de nombreuses méthodes pour créer des échantillons, faire les décompositions de Karhunen-Loeve, ou encore reconstituer un champ à partir de la dite décomposition. Enfin, comme les échantillons sont nécessaires pour la décomposition de Karhunen-Loeve, et que ceux-ci peuvent être de taille plutôt importante, ils sont enregistrés sous form de *numpy.memmap* dans un fichier temporaire, pour relâcher un peu de pression sur la mémoire vive.

En interne, le modèle fonction de champs stochastiques, passe par un wrapper qui fait qu'on peut la considérer comme seulement fonction de variables aléatoires gaussiennes. Cela permet d'utiliser les méthodes internes à openturns comme l'algorithme de Saltelli pour estimer les indices de sobol.

Néanmoins, pour pallier certaines difficultés comme les fonctions qui par moment renvoient des `np.nan` (donc lorsqu'il y a eu une erreur), on passe d'abord par une étape de correction: l'ensemble des valeurs de sortie contenant des `nan` sont regénérées, et comme les entrées de l'étude sont des combinaison de deux ensembles de variables aléatoires A et B, toutes les autres permutations des entrées ayant entraîné des erreurs sont supprimées et remplacées. Cette étape est expliquée en figure 4

Les codes sont accessibles via [github](#):
https://github.com/Kramer84/stochastic_process_analysis

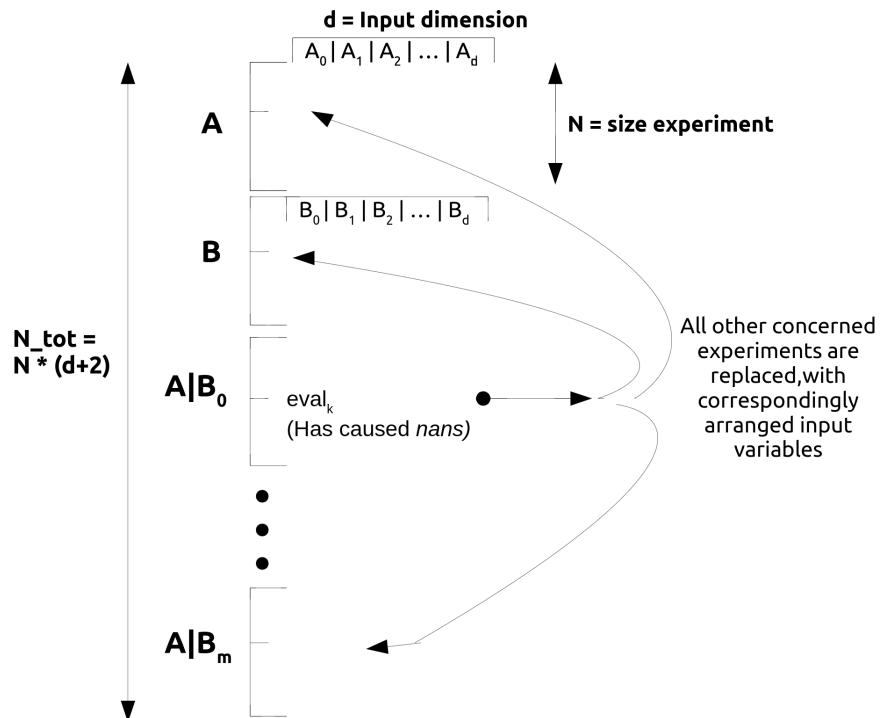


Figure 4: Organisation des variables d'entrée, expliquant l'étape de postprocessing

Une fois cette étape finie, on a accès aux indices de sobol pour chaque élément du champ.

Dans le cas de la poutre en flexion, prenant initialement 5 arguments en entrée (champ du module de young et champ du diamètre, densité, position de la force et norme de la force), la décomposition des deux champs en entrée fait que le modèle est ensuite régi par 24 variables aléatoires. Bien que cela fait augmenter la dimensionnalité du problème, nous avons désormais la possibilité de travailler seulement avec des variables aléatoires gaussiennes décorrélées, et donc faire une analyse de sensibilité classique.

La difficulté qui découle de cette analyse, est de trouver la manière qui permettra de relier les indices de sobol de ces variables aléatoires décorrélées, au champ

stochastique d'origine.



Figure 5: indices de sobol des V.A. de Karhunen-Loève pour la déflection maximale

Pour parvenir à faire ce lien, nous avions comme base différentes études sur le sujet, notamment le papier par [Wei et al., 2017], sur l'analyse de sensibilité sur des modèles ayant des paramètres dépendants du temps. (Les variables étant gouvernées par le temps sont généralement stochastiques, ex: température, effort etc.)

Mais ayant fait une analyse préliminaire de ces indices de sobol', des incohérences dans ces indices ont pu être remarqués. En effet, la somme de l'ensemble des indices du premier ordre doit être inférieure à 1 , et il ne peut y avoir d'indices négatifs. L'apparition d'indices de Sobol' négatifs peut être expliqué par un nombre d'expériences trop faible, nombre d'expériences qui lui-même est limité par la mémoire disponible dans l'ordinateur.

1.4.2 Premiers essais d'analyse de la sensibilité du modèle

Comme nous avions désormais accès aux différents indices de sobol de notre modèle basé sur la décomposition de Karhunen-Loève, le travail pouvait désormais se focaliser sur l'interprétation de ces différents indices. En effet, cette décomposition était très classique lors du travail avec des champs stochastiques, et plusieurs papiers de recherche se sont intéressés à comment faire des analyses de sensibilité de ce type de champs avec cette même décomposition. Comme mentionné précédemment, le travail de [Wei et al., 2017] offre une première approche pour l'interprétation de ces résultats, mais d'autres comme [Pronzato, 2019] ont aussi développés des méthodes pour l'analyse de ces indices issus de l'expansion de Karhunen-Loève. Dans le travail de [Shang and Yun, 2013], il est rapporté d'une méthodologie pour lier un modèle en éléments finis d'Abaqus avec des champs stochastiques générés avec MATLAB et pour l'analyse probabiliste de ce modèle. L'utilisation de l'expansion de KL y est aussi mentionnée, et ce travail offre une analyse intéressante sur la capacité de cette méthode à reproduire



Figure 6: indices de sobol des V.A. de Karhunen-Loève pour le champ des contrainte de Von Mises

le champs stochastique, en fonction de l'élément à partir duquel on troncature cette expansion.

1.4.3 Interprétation et essais avec le papier : "Time-variant global reliability sensitivity analysis of structures with both input random variables and stochastic processes"

Ce papier de recherche datant de 2017, offre une approche pour l'analyse de sensibilité de modèles définis dans le temps et leur analyse de fiabilité. En effet, de nombreuses structures mécaniques (ponts, tours etc.) sont soumises à des grandeurs variant dans le temps, et si celles se trouvent être continues dans celui ci, alors cette grandeur peut être représentée par un champ stochastique. Bien qu'il y aie une différence notable entre les grandeurs par rapport auxquelles il y a continuité (dans leur papier le temps, et nous l'espace), des analogies peuvent être faites.

L'approche se base sur la décomposition de Karhunen-Loève du champ stochastique, permettant de considérer le problème comme uniquement dépendant de variables aléatoire gaussiennes décorrélées, et de déduire la sensibilité du modèle au champ stochastique à partir de la sensibilité de chaque composant de la décomposition du champ.

La décomposition de KL est particulièrement utile pour le calcul de fonction d'enveloppe et les calculs fiabilistes. Dans ce papier, l'utilisation de l'expérience de Monte-Carlo est essentiellement utilisée pour être comparée aux autres méthodes développées (**FOEF** : First Order Envelope Function et **AK-MCS** Active learning Kriging based Monte Carlo Simulation).

L'utilité de cette approche est de pouvoir identifier les sources principales d'incertitude ayant le plus d'influence sur la fiabilité du modèle. Enfin, elle permet de classer les variables selon leur ordre d'influence.

Néanmoins, leur approche se base sur la définition d'un état limite du modèle (par exemple la différence entre la contrainte maximale et la contrainte limite) et d'une fonction indicatrice de l'état limite, alors que nous cherchons à évaluer la sensibilité d'une grandeur en sortie (par exemple la contrainte) par rapport aux variables aléatoires en entrée. Ceci revient à priori à mesurer l'impact qu'à une variables en entrée sur la variance de la grandeur en sortie.

La définition mathématique de leur problème est la suivante :

S	: Etat limite du modèle
\mathbf{X}	: Vecteur de variables aléatoires
$\mathbf{Y}(t)$: Vecteur de champs stochastiques dépendants du temps
\mathbf{Z}	: Fonction de performance
I_S	: Fonction indicatrice
R	: Probabilité de défaillance sur l'intervalle $[0,T]$

$$S = \{(\mathbf{X}, \mathbf{Y}) : Z(t) = g(\mathbf{X}, \mathbf{Y}(t), t) < 0 \forall t \in [0, T]\}; \quad (5)$$

$$I_S(t) = I_S(\mathbf{X}, \mathbf{Y}(t), T) = \begin{cases} 1 & (\mathbf{X}, \mathbf{Y}) \in S \\ 0 & (\mathbf{X}, \mathbf{Y}) \notin S \end{cases}; \quad (6)$$

$$R = R(T) = P_r(I_S = 1) = Pr(Z(t) = g(\mathbf{X}, \mathbf{Y}(t) < 0, \forall t \in [0, T]); \quad (7)$$

Les variables aléatoires et champs stochastiques ici définis sont tous indépendants entre eux.

Soient $\mu_X = (\mu_{X_1}, \dots, \mu_{X_n})$ et $\sigma_X = (\sigma_{X_1}, \dots, \sigma_{X_n})$ les vecteurs des moyennes et écarts types des variables aléatoires et $\mu_Y = (\mu_{Y_1}, \dots, \mu_{Y_n})$ et $\sigma_Y = (\sigma_{Y_1}, \dots, \sigma_{Y_n})$ les vecteurs des moyennes et écarts types des champs stochastiques.

Comme explicité lors de la définition de la décomposition de Karhunen-Loève, tout champ stochastique $Y_i(t)$ peut être écrit comme une somme de variables aléatoires gaussiennes décorrélées.

$Y_i(t)$	Approx. Procéssus Gaussien Y_i
λ_{ik}	Valeur Propre de la matrice de covariance
ξ_{ik}	Variable Normale Centrée Réduite
$\varphi_{ik}(t)$	Vecteur propre de la matrice de covariance

$$Y_i(t) = \mu_{Y_i}(t) + \sum_{k=1}^M \sqrt{\lambda_{ik}} \xi_{ik} \varphi_{ik}(t); \quad (8)$$

D'où chaque champ gaussien peut être exprimé à partir d'un ensemble de variables aléatoires gaussiennes décorrélées $\xi_i = (\xi_{i1}, \dots, \xi_{iM_i})$ et l'ensemble des champs est représenté par le vecteur $\xi = (\xi_1, \dots, \xi_m)$. ξ_i étant la représentation par des variables auxiliaires du champ Y_i .

Grâce à la propriété de décomposition de la variance, on a :

$$\begin{aligned} V(I_S) &= \sum_{i=1}^n V_{X_i} + \sum_{i=1}^m V_{\xi_i} + \sum_{i=1}^n \sum_{j \neq i, j=1}^n V_{X_i X_j} \\ &\quad + \sum_{i=1}^m \sum_{j \neq i, j=1}^m V_{\xi_i \xi_j} + \sum_{i=1}^n \sum_{j=1}^m V_{X_i \xi_j} + \dots \\ &\quad + V_{X_1 \dots X_n \xi_1 \dots \xi_m} \end{aligned}$$

Figure 7: Décomposition de la variance de la fonction indicatrice

Ici, c'est la variance de la fonction indicatrice I_s en sortie qui est traitée, mais ceci est faisable pour tout modèle dépendant d'un ensemble de variables aléatoires. Il est à remarquer ici que : $V_{X_i} = V[E(I_s | X_i)]$ qui est la variance

conditionnelle de la sortie I_S lorsque la variable aléatoire X_i est connue.

Néanmoins, dans le cas du vecteur $\xi_i = (\xi_{i_1}, \dots, \xi_{i_{M_i}})$ on a:

$$V_{\xi_i} = V[E(I_S|\xi_i)] = V[E(I_S|(\xi_{i_1}, \dots, \xi_{i_{M_i}}))] - \sum_{p=1}^{M_i} V_{\xi_{i_p}} \quad (9)$$

qui est la variance conditionnelle de la sortie lorsque le vecteur aléatoire ξ_i représentatif du champ stochastique Y_i est connu.

Pour le calcul du vecteur des variances $V[E(I_S|(\xi_{i_1}, \dots, \xi_{i_{M_i}}))]$, dans le cadre d'une expérience de Monte-Carlo, ceci pourrait se faire en rajoutant dans le plan d'expérience, toutes les permutations associées à ces variables annexes aux champs stochastiques.

Avec le plan d'expérience classique proposé par openTURNS, nous sommes en capacité de facilement calculer les indices de Sobol' du 1^{er} ordre de toutes les variables S_{X_i} et $S_{\xi_{i_p}}$. Or nous cherchons S_{ξ_i} . Dans le cas d'un modèle F , prenant en entrée un vecteur de champs stochastiques $Y = (Y_1, \dots, Y_m)$ et un vecteur de variables aléatoires $X = (X_1, \dots, X_n)$ et défini comme suit:

$$\omega = F(X, Y), \omega \in \Re^N \quad (10)$$

On peut définir une fonction F' , utilisant la décomposition de Karhunen-Loève et seulement dépendante d'un ensemble de variables aléatoires décorrélées:

$$\omega = F'(X, \xi) \quad (11)$$

$$\xi = (\xi_1, \dots, \xi_m) \quad (12)$$

$$\xi_i = (\xi_{i_1}, \dots, \xi_{i_{M_i}}) \quad (13)$$

Nous savons que :

$$V_{X_i} = V[E[\omega|X_i]] \quad (14)$$

$$S_{X_i} = \frac{V_{X_i}}{V(\omega)} \quad (15)$$

$$V_{X_i X_j} = V[E[\omega|X_i X_j]] - V_{X_i} - V_{X_j} \quad (16)$$

$$S_{X_i X_j} = \frac{1}{V(\omega)} V[E[\omega|X_i X_j]] - S_{X_i} - S_{X_j} \quad (17)$$

Néanmoins, à la différence des effets d'interactions et les sensibilités qui y sont associées, dans le cas de la sensibilité du champ stochastique, il semblerait qu'on ne soustraire pas l'effet de l'interaction du premier ordre, donc on n'aurait pas :

$$S_{\xi_i} = S_{\xi_{i_1}, \dots, \xi_{i_{M_i}}} = \frac{1}{V(\omega)} V[E[\omega|\xi_{i_1}, \dots, \xi_{i_{M_i}}]] - \sum_{p=1}^{M_i} S_{\xi_{i_p}} \quad (18)$$

Mais plutôt seulement :

$$S_{\xi_i} = \frac{1}{V(\omega)} V[E[\omega | \xi_{i_1}, \dots, \xi_{i_{M_i}}]] \quad (19)$$

Ceci est intéressant dans la sens ou cela diminue grandement la complexité du calcul et qu'il ne faut plus que calculer les effets d'interaction et non plus les indices de sobol du premier ordre associés aux indices de KL des champs stochastiques.

Le calcul à posteriori de ces interactions est possible, vu que nous n'avons qu'à comparer la moyenne des variance des deux échantillons de départ (**A** et **B**) à celle lorsque tout les paramètres du champs stochastique i sont connus (V_{ξ_i}).

Néanmoins, l'on ne peut pas inclure celles-ci dans le plan d'expérience utilisé par openTURNS, vu que l'algorithme utilisé pour calculer les indices de Sobol' n'est concu que pour les calculs des indices de premier et second ordre de modèles gouvernés par un ensemble de variables aléatoires unitaires, et non des champs stochastiques. Pour parvenir à passer outre cette difficulté, un choix d'algorithme doit être fait pour le calcul de ces indices de sobol d'ordre fortement supérieur. (Vu que pour les indices d'ordre 1, le choix de l'algorithme est entièrement libre.) Au vu du travail fait par PHIMECA sur l'analyse des lois asymptotiques des estimateurs des indices de Sobol [Dumas, 2017] ,l'on prendra comme estimateur celui de Martinez, mais il sera évidemment possible de rajouter à posteriori les autres estimateurs dans le workflow.

D'autre part, pour parvenir à garder cette complexité faible, nous sommes obligés de revoir la manière dont est fait le plan d'expérience (la manière dont on mélange les différentes colonnes de nos deux samples **A** et **B**) puisque dans le cas de l'exemple de la poutre, nous aurions jusqu'à 5 fois moins de calculs à effectuer qu'en gardant tout les indices inutiles.

Génération d'expérience pour analyse de sensibilité sur champs stochastiques

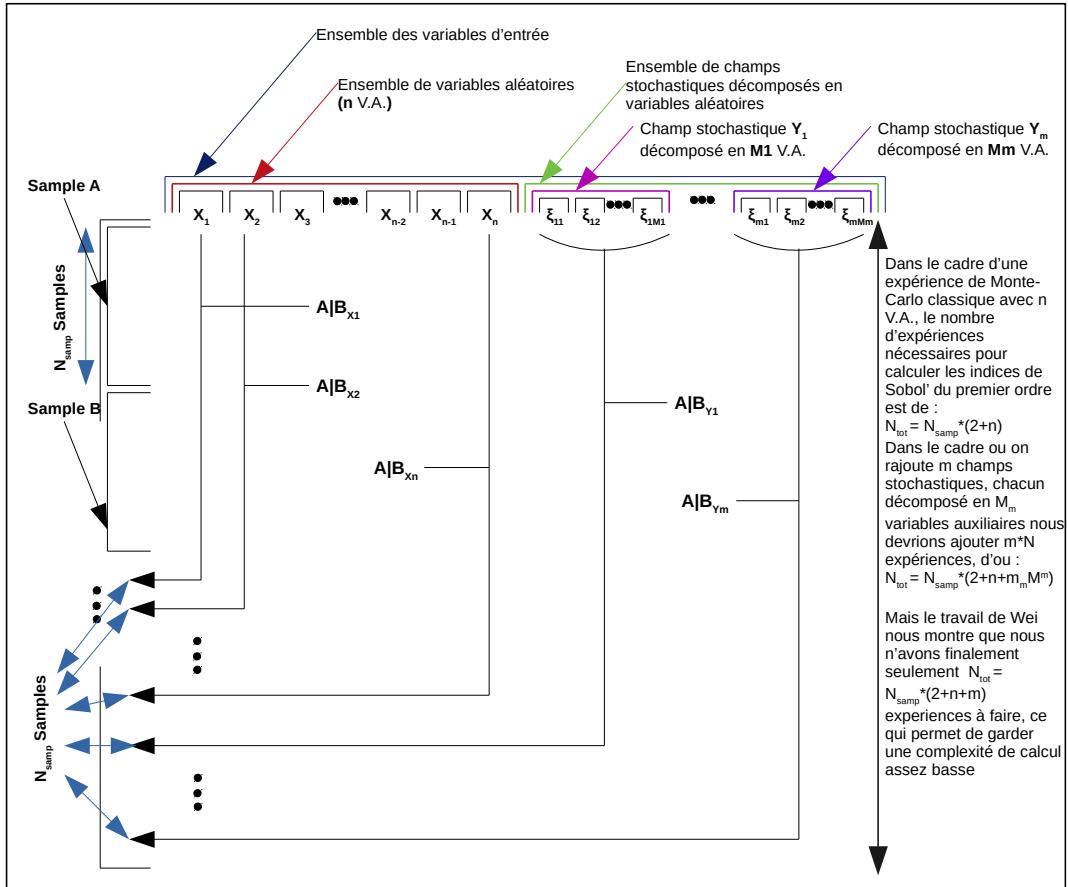


Figure 8: Génération d'expérience pour analyse de sensibilité sur champs stochastiques

Les échantillons sont mélangés comme montré en figure 8. Pour générer les samples de manière plus efficace, et être sûr de couvrir l'ensemble du domaine de variance des lois d'entrée, est inclus dans le code la possibilité de générer l'échantillon en utilisant soit l'échantillonage LHS (Latin Hypercube Sampling), soit les séries à faible divergence (Low discrepancy series), soit une version optimisée du LHS (Simulated Annealing LHS). Bien sûr une génération entièrement aléatoire reste l'option de base.

Pour le calcul formel de l'estimateur de Sobol', nous allons utilisons la méthode par Saltelli, détaillée dans le papier par [Dumas, 2017]. Ce papier détail aussi la manière de calculer l'erreur dans le calcul de l'estimateur, valeur importante pour connaître la pertinence de l'estimateur.

Détail du calcul de l'indice de Sobol' :

${}^c\mathbf{Y}^A$: Sortie du modèle associé à l'échantillon A du plan d'expérience

${}^c\mathbf{Y}^B$: Sortie du modèle associé à l'échantillon B du plan d'expérience

${}^c\mathbf{Y}^E$: Sortie du modèle associé à l'échantillon issu de A ou l'on a inseré une partie de B

S : Indice de Sobol'

Les échantillons centrées sont annotés du préscript c

$$S = \frac{\text{Cov}[{}^cY^B, {}^cY^E]}{\text{Var}[{}^cY^A]} \quad (20)$$

$$= \frac{\frac{1}{n} \sum_{k=0}^n {}^cY^B {}^cY^E - (\frac{1}{n} \sum_{k=1}^n {}^cY^B) (\frac{1}{n} \sum_{k=1}^n {}^cY^A)}{\frac{1}{n} \sum_{k=0}^n ({}^cY^B)^2 - (\frac{1}{n} \sum_{k=0}^n {}^cY^B)^2} \quad (21)$$

Comme les échantillons sont centrées par rapport à leurs moyennes respectives, le numérateur et le dénominateur se simplifient et on obtient l'estimateur suivant :

$$S = \frac{\frac{1}{n} \sum_{k=0}^n {}^cY^B {}^cY^E}{\frac{1}{n} \sum_{k=0}^n ({}^cY^A)^2} \quad (22)$$

Avec l'échantillonnage présenté précédemment, le calcul de l'estimateur de Sobol se fait très directement avec la formule de Saltelli, même lorsque la dimension de la sortie est plus grande que 1.

Pour pouvoir justifier de la pertinence de l'estimateur, on calcule l'intervalle de confiance à 0.975, qui permet de montrer la précision de l'estimateur et de nuancer son analyse. Pour le calcul de cet intervalle de confiance, on utilise la méthode Delta, méthode utilisée par Jansen *et al* (2014) ou de même par [Dumas, 2017]. Le détail de cette méthode est donné ci-dessous.

On pose suite à l'équation simplifiée précédente :

$$U_i = \left({}^cY^B {}^cY^E, ({}^cY^A)^2 \right)^T \quad (23)$$

Ensuite l'on défini :

$$\Psi_S(x, y) = \frac{x}{y} \quad (24)$$

Ceci implique :

$$S_N = \Psi_S(\bar{U}_N) \quad (25)$$

Grâce au théorème central limite:

$$\sqrt{N}(\bar{U}_N - \mu) \xrightarrow[N \rightarrow \infty]{\mathcal{L}} \mathcal{N}_2(0, \Gamma) \quad (26)$$

Où Γ est la matrice de covariance de U_1 et :

$$\mu = \begin{cases} Cov[Y^B, Y^E] \\ Var[Y^A] \end{cases} \quad (27)$$

De l'utilisation de la méthode delta :

$$\sqrt[4]{N}(S_N - S) \xrightarrow[N \rightarrow \infty]{\mathcal{L}} \mathcal{N}_1(0, g^T \Gamma g) \quad (28)$$

ou $g = \nabla \Psi_S(\mu)$.

Finalement, pour tout x, y tel que $y \neq 0$:

$$\nabla \Psi_S(x, y) = \left(\frac{1}{y}, \frac{-x}{y^2} \right)^T \quad (29)$$

La matrice de covariance Γ , tout comme le vecteur g se calculent bien analytiquement dans le cas de l'indicateur de Saltelli', néanmoins le calcul du gradient peut très bien se faire informatiquement, notamment dans le cas d'estimateurs plus complexes.

Pour récupérer l'intervalle de confiance de l'estimateur nous utilisons le quantile de la loi normale centrée réduite à 97.5%, multiplié par la racine de la variance calculée précédemment.

Finalement, l'on récupère les indices de Sobol' pour les différents éléments de la poutre tout comme leur intervalle de confiance:

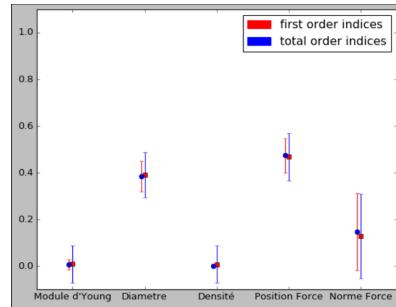


Figure 9: Sensibilité de la déflexion maximale aux champs et variables d'entrée.

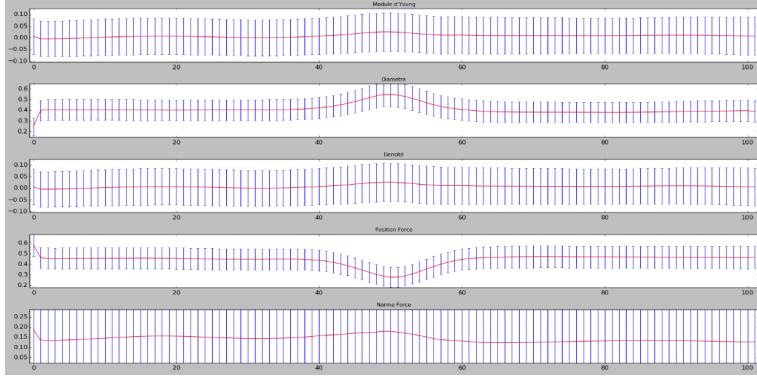


Figure 10: Sensibilité de la contrainte de Von-Mises aux champs (Module d'Young, diamètre) et variables d'entrée.(densité, position, Norme de la force)

L'on peut voir qu'il y a très peu d'écart entre les indices d'ordre total et premier ordre. Les effets d'interaction sont donc très peu présents entre les variables d'entrée.

Finalement une comparaison entre un exemple de la documentation d'*openTURNS* et les codes développés durant ce stage peut être faite, pour confirmer que les méthodes implémentées sont exemptes d'erreurs.

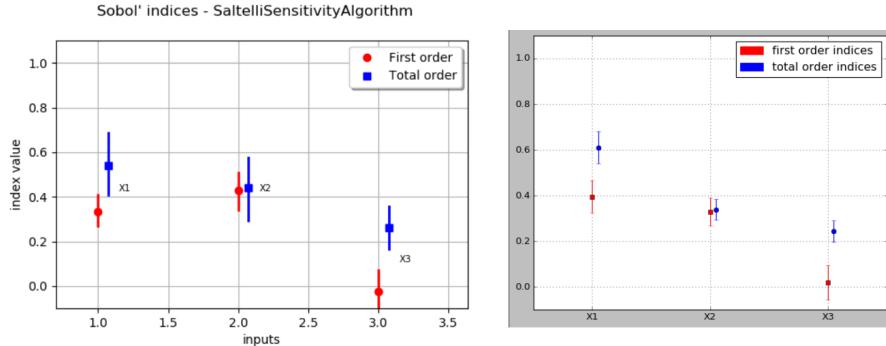


Figure 11: Comparaison entre la solution par openturns (à gauche) et la notre (à droite), pour un échantillon de taille 1000 sur la fonction d'Ishigami

L'on observe, que pour cet échantillon de 1000, l'indice de Sobol' calculé avec nos codes est relativement proche de ceux calculés avec *openTURNS*, même si des écarts subsistent, notamment au niveau de la variance. Cette variance s'explique en partie par l'incertitude dans les variables d'entrées (tout échantillon de 1000 est à priori unique).

Les codes ayant montré leur utilisabilité, il était enfin possible d'essayer d'appliquer cette méthodologie à un système plus complexe, un échangeur de chaleur air-air modelisé avec NASTRAN.

1.5 Application au modèle NASTRAN d'échangeur thermique

1.5.1 Définition et analyse de la problématique

Comme nous l'avions mentionné avant, la finalité de ce travail de quasi-recherche et développement de codes, était de parvenir à induire des défauts stochastiques dans un modèle numérique, et de mesurer la sensibilité de ce modèle à ces perturbations.

Le modèle en question est celui d'un échangeur thermique d'utilisation aéronautique. Cette pièce doit être capable de supporter et échanger la température de deux flux d'air perpendiculaires, l'un à la température de l'air en haute altitude - jusqu'à -50°- et l'autre aux températures moteurs allant jusqu'à 600C.

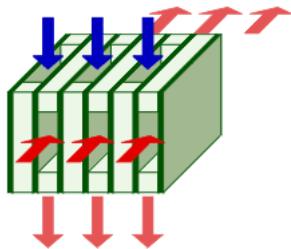


Figure 12: Principe de l'échangeur à chaleur air-air

Ce projet est financé par l'union européenne, et la fabrication est assurée par le fabricant allemand **LIEBHERR**. Ce dernier sous-traite certaines étapes de l'étude, comme les essais physiques sur des échantillons de l'échangeur, tâche réalisée par **Lortek**, la construction d'un modèle paramétrique numérique de l'échangeur, tâche réalisée par **Epsilon**, et enfin la partie analyse de sensibilité et déformation moteur réalisée par **Phimeca**.

Ce projet a donc impliqué l'étroite collaboration de ces différents acteurs, vu que pour l'analyse de sensibilité il nous fallait utiliser le modèle construit par **Epsilon**, et introduire des incertitudes dans les paramètres, dont la nature et les paramètres ont été récupérées dans les mesures faites par **Lortek**.

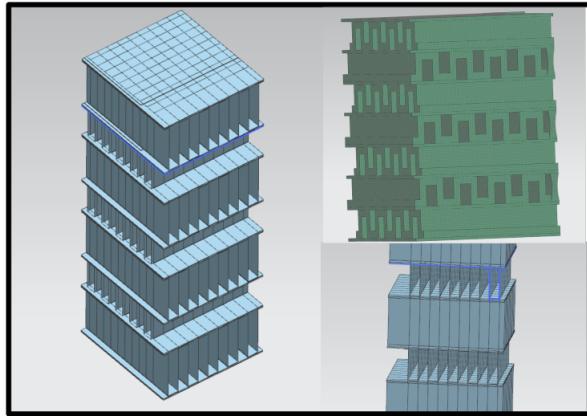


Figure 13: Modèle Nastran de l'échangeur de chaleur - Siemens Simcenter 3D

Pour bien se coordonner sur le projet, des réunions se sont régulièrement tenues, permettant de se tenir à jour sur les avancements de chacun, et de définir les prochaines étapes à atteindre.

Dans le cas de l'étude à réaliser par **Phimeca**, le sujet d'analyse était l'étude de l'influence de la variabilité dans les paramètres des ailettes de l'échangeur de chaleur sur la localisation de la défaillance. Dans le cas de l'échangeur de chaleur, la défaillance correspond à une rupture d'aillette, ou une perte d'étanchéité.

Numériquement, la rupture se traduit par une contrainte maximale supérieure à la contrainte maximale supportée par le matériau constituant de la pièce. L'analyse de la pièce se traduit donc par l'étude de l'effet de la maîtrise des paramètres de l'aillette (épaisseur, planéité, angle...) sur la localisation de la contrainte maximale.

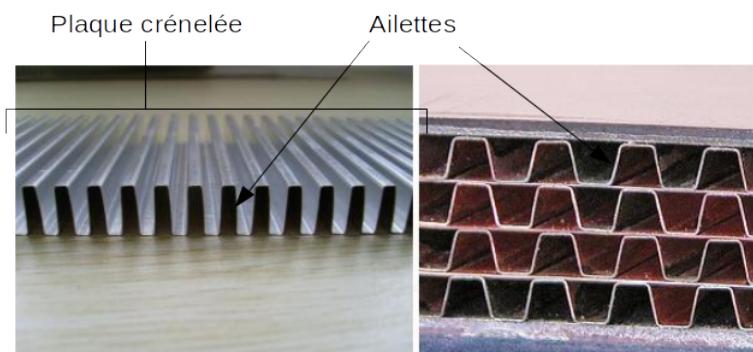


Figure 14: Ailettes d'un échangeur de chaleur

1.5.2 Mise en place de méthodologie

Pour parvenir à introduire ces variabilités différentes approches et modèles de champs stochastiques peuvent être prises en considération, puisque les réalisations peuvent fortement varier en fonction des paramètres et modèles. Il est en outre difficile de générer un champ qui serait indépendant du maillage sur lequel il est réalisé, ou du moins qui ressemblerait à un échantillon d'un champ plus grand.

Analyse de l'influence du paramètre v du modèle de covariance Matérn, sur un champ rectangulaire de taille 200 x 7.

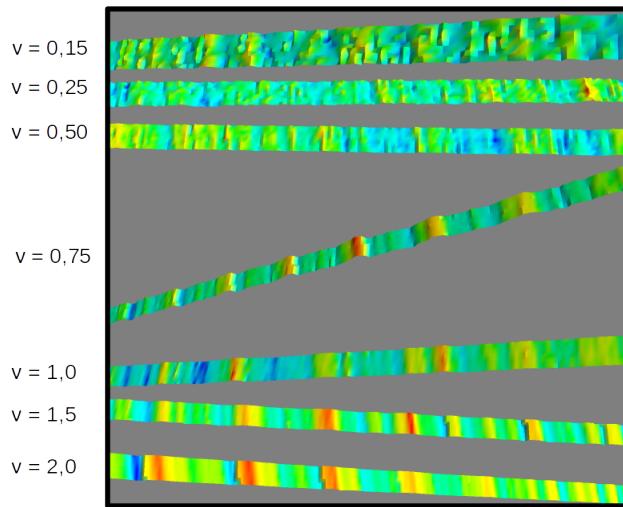


Figure 15: Impacte d'un paramètre sur, un champ

En deux dimensions, un champ stochastique peut être vu comme un maillage uniforme, où une valeur est associée à chaque noeud, et où la valeur à chaque noeud est corrélée à la distance et valeur des nœuds proches. La première idée aurait donc été d'appliquer directement une déformation générée par un champ de la même taille que le maillage d'une ailette aux nœuds constituant l'aillette, et donc de simuler une déformation stochastique par ailette, mais en écartant du coup toute corrélation inter-aillettes. La déformation se serait fait par translation de chaque nœud suivant l'axe normal au plan de l'aillette. Cette méthode a vite été écartée, puisque la génération de champs sur des maillages sous forme de long rectangle étiré était bien moins précise et contrôlables, allant jusqu'à des réalisations quasi constantes ou périodiques. De plus le nombre de champs générés par passe serait très important, alors qu'il serait plus simple de ne générer qu'un seul champ par plaque et de paramétriser les déformations.

D'après le travail réalisé par [Goka, 2019], une méthodologie pour modéliser au mieux les imperfections de conception d'une pièce et se rapprocher de défauts réalistes existe. Dans son travail de thèse, [Goka, 2019] analyse les différentes manières de représenter les défauts de forme d'une pièce issues des imperfections de fabrication. Chaque méthode qu'elle soit paramétrique (Courbes de Bézier, B-Spline, NURBS), aléatoire (Bruit Gaussien), issue de morphing, d'un Skin-

Model ou encore basé sur des modes (FFT, Polynômes, PCA, Décomposition modale métrique, champs aléatoires...) est analysée et jugée par rapport à 4 critères :

- capacité à représenter les défauts de forme et les ondulations
- compatibilité avec une représentation probabiliste du champ des défauts
 \Leftrightarrow amplitude des défauts de méthode doivent être modélisées par des lois de probabilité.
- compatibilité avec les méthodes d'analyse probabiliste existantes, et interprétabilité physique (les coefficients contrôlant l'amplitude des défauts et leur distribution doivent représenter une grandeur physique mesurable) .
- doit être capable de modéliser des contacts locaux entre pièces et de retrouver les points de contact

L'applicabilité de ces méthodes a notre problème dépend aussi de la réponse à ces critères, et elles ont été développées au cours de cette thèse en vue de leur utilisation sur des projets d'analyse comme celui-ci. La méthode retenue par Goka au cours de son analyse est la méthode de la décomposition métrique modale. Cette méthode ne s'appuie ni sur des équations du mouvement de la mécanique vibratoire, ni sur la méthode des éléments finis. Elle permet d'introduire des défauts dans un modèle en appliquant une translation à chaque noeud du maillage de la surface. Cette translation est appliquée à l'aide de ce qu'on appelle les vecteurs modaux. Ces vecteurs modaux représentent un défaut élémentaire de la surface. Il existe différents types de déformations élémentaires, et chacun d'eux est à rapporter avec une forme élémentaire. De même ces vecteurs modaux sont généralement représentés dans la base qui est la plus adaptée pour représenter la pièce à laquelle on applique ces défauts. Par exemple les défauts élémentaires d'une surface plane cylindrique.

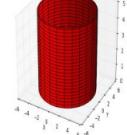
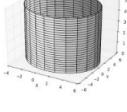
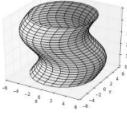
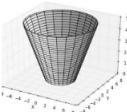
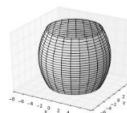
Surface nominale	Vecteur modal	Déformation élémentaire
	$\mathbf{f}_{\text{elliptique}}(x, y, z) = k * \begin{pmatrix} \cos \theta * \cos \theta \\ \sin \theta * \cos \theta \\ 0 \end{pmatrix}$	
	$\mathbf{f}_{\text{sinusoidal}}(x, y, z) = \begin{pmatrix} \sin\left(\frac{2\pi z}{h}\right) \\ \sin\left(\frac{2\pi z}{h}\right) \\ 0 \end{pmatrix}$	
	$\mathbf{f}_{\text{Conique}}(x, y, z) = k * \begin{pmatrix} 2 * \cos \theta * \left(\frac{2}{5}z - 1\right) \\ 2 * \sin \theta * \left(\frac{2}{5}z - 1\right) \\ 0 \end{pmatrix}$	
	$\mathbf{f}_{\text{tomeuse}} = k \begin{bmatrix} \cos(\theta) * \left[1 - 4\left(\frac{2}{5}z - 1\right)^2\right] \\ \sin(\theta) * \left[1 - 4\left(\frac{2}{5}z - 1\right)^2\right] \\ 0 \end{bmatrix}_{(x,y,z)}$	

Figure 16: Vecteur des déformations modales pour un cylindre

Analytiquement, un champ modal est défini de la manière suivante (en coordonnées cartésiennes ici):

$$\mathbf{f}(x, y, z) = \begin{bmatrix} \mathbf{f}_x(x, y, z) \\ \mathbf{f}_y(x, y, z) \\ \mathbf{f}_z(x, y, z) \end{bmatrix}_{(x,y,z)} \quad (30)$$

ce vecteur $\mathbf{f}(x, y, z)$ est appliqué à chaque point de la surface de la pièce que l'on souhaite déformer. Le champ modale représente les défauts de forme de la surface, ou encore l'écart entre les positions théoriques parfaites et les positions réelles des points.

On peut écrire que pour chaque point de la surface nominale $M_n(x_n, y_n, z_n)$ on applique ce vecteur $\mathbf{f}(x, y, z)$ pour obtenir un nouveau point $M_f(x_f, y_f, z_f)$

$$\begin{pmatrix} x_f \\ y_f \\ z_f \end{pmatrix} = \begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix} + \begin{pmatrix} \mathbf{f}_x(x_n, y_n, z_n) \\ \mathbf{f}_y(x_n, y_n, z_n) \\ \mathbf{f}_z(x_n, y_n, z_n) \end{pmatrix}_{(x,y,z)} \quad (31)$$

Le champ modal total représentant les défauts de forme de la surface est défini par la somme pondérée des modes élémentaires appliqués à chaque noeud de la surface :

$$\mathbf{f}_{pa,A_0} = \begin{pmatrix} \mathbf{f}_{xpa,A_0} \\ \mathbf{f}_{ypa,A_0} \\ \mathbf{f}_{zpa,A_0} \end{pmatrix}_{(x,y,z)} = \sum_{j=1}^n \lambda_{pa,j} \times \mathbf{f}_{pa,j,A_0} \quad (32)$$

Ici, \mathbf{f}_{pa,A_0} est le champ modal total de la surface a de la pièce p et exprimé au point A_0 ; \mathbf{f}_{pa,j,A_0} est le vecteur modal correspondant au j^{me} défaut de forme élémentaire écrit en un point A_0 de la surface a de la pièce p ; $\lambda_{pa,j}$ représente le coefficient d'amplitude du j^{me} défaut de forme élémentaire de la surface a de la pièce p . Les vecteurs modaux dépendent de la classe et géométrie de la surface, tout comme du moyen de fabrication utilisé.

Les vecteurs modaux sont ensuite normés, pour que les coefficient d'amplitude $\lambda_{pa,j}$ représentent bien une grandeur physique mesurable (métrique). Le calcul de la norme se limite à calculer la norme maximale du champ vectoriel modal pour tous les points de la surface, avant de normer ce maximum à 1mm :

$$\max(\|\mathbf{f}_{(M)}\|)_{1 \leq j \leq n} = 1 [mm] \quad (33)$$

L'on obtient les différents modes en étudiant les caractéristiques intrinsèques et les degrés d'invariance de la surface nominale. Quatre types de modes peuvent être distingués :

- Les modes rigides qui ne changent pas la nature de la surface: les modes de translation et de rotation de la surface correspondent au premier modes.
- Les modes dits de "dilatation" qui correspondent à des variations des caractéristiques de la surface.
- Les modes d'ondulation définis par des déviations sinusoidales le long des *Eléments Géométriques de Référence Minimum*² (EGRM) de la surface.
- Les modes des sections définissant des variations de caractéristiques intrinsèques en utilisant des coordonnées polaires (*Ref?*). Ces modes font varier le profil ou la géométrie de la section droite d'une surface. Dans le cas d'une ailette d'une passe de l'échangeur, cela revient à faire varier le profil de l'aillette en fonction de la direction du flux d'air. Ceci donne à l'aillette une sorte de géométrie de "ruban".
- Les modes quelconques sont ceux n'appartenant pas aux autres catégories.

Le choix final des modes de déformation se fait finalement à partir des données issues d'expériences ou connaissances en fabrication. En fonction du procédé de fabrication certains défauts sont en effet récurrents. L'amplitude des défauts est déterminée grâce à des mesures prises sur les pièces fabriquées.

Le choix de cette méthode semblait donc très adapté, puisque l'utilisation de défauts paramétrisés permet de facilement contrôler des défauts complexes avec qu'un seul paramètre, et évite de devoir générer un grand nombre de déformations stochastiques qui sont bien moins contrôlables et plus coûteuses

²Les éléments géométriques de référence sont des formes géométriques (plans, cylindre, axe) parfaites qui servent de base pour mesurer les écarts. Dans le cas du modèle de l'échangeur de chaleur, ces plans de référence sont les plans d'origine.

en temps de calcul à générer. De plus l'utilisation de cette méthode permet de garder une grande maîtrise des défauts et des corrélations inter-défaut.

En tant qu'illustration, un schéma détaillant la méthode pour la déformation d'une passe de l'échangeur de chaleur avec la méthode de la déformation modale métrique est donné ci dessous. Sur ce schéma, le coefficient d'amplitude $\lambda_{pa,j}$ est donné par un champ stochastique défini sur un maillage de la même taille que la passe, et l'élément géométrique minimal est l'ensemble d'arêtes perpendiculaires aux deux flux d'air.

La valeur du champ stochastique en un point donne l'écart maximal entre la surface nominale et la surface "réelle", et la déformation en elle-même suit un polynôme du second ordre s'annulant aux bornes inférieures et supérieures de la passe.

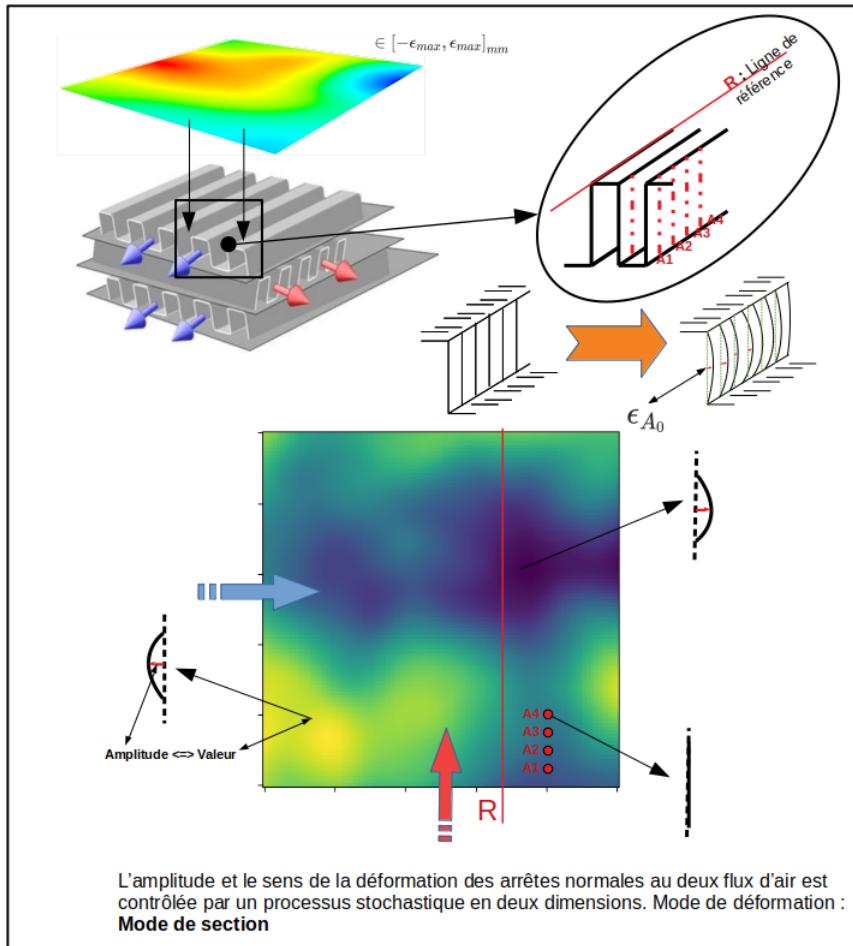


Figure 17: Schéma de l'influence entre la réalisation d'un champ stochastique et la déformation modale de la passe de l'échangeur

Cette méthode était hautement fonctionnelle, car les déformations sont toutes corrélées au sein d'une même passe, avec même la possibilité de parvenir à une

corrélation tri-dimensionnelle, si l'on génère un champ 3D .

1.5.3 Application de la méthode de la déformation modale métrique pilotée par champ stochastique sur un modèle Simcenter 3D Nastran

Bien sûr, bien que théoriquement simple, l'implémentation de cette méthode au sein de Simcenter NX Nastran était plus complexe, car l'on devait pouvoir garder le contrôle total sur la géométrie nominale de la pièce - peu importe quels paramètres géométriques originaux on imposait - et aussi un contrôle total sur quel types de passes/ plaques de séparation on introduisait des défauts, et enfin un contrôle total sur les processus stochastiques, leurs fonctions de covariance, et bien sûr les modes de déformation modale et leur nombre.

Néanmoins plusieurs difficultés ont été rencontrées lors de l'écriture des codes, puisque les scripts python devaient interagir avec le logiciel de Siemens. Simcenter NX possède un langage de "scripting" appelé NX, qui permet d'utiliser les fonctions de Simcenter depuis un interpréteur et d'automatiser des tâches dans ce qu'on appelle des journaux. L'accès au capacités de scripting de Simcenter peut se faire à partir de différents langages, dont le Visual Basic ou Python, bien que très peu de tutoriels existent pour Python.

Mais c'est bien le Python qui a été choisi au final de par sa plus grande simplicité par rapport au Visual Basic et l'existence d'une documentation fournie.

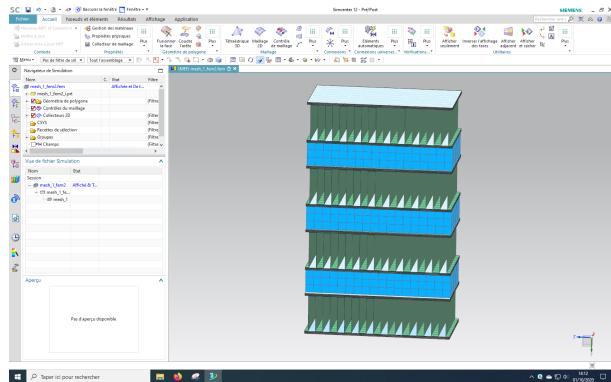


Figure 18: Interface du logiciel Simcenter 12 - vue active du modèle FEM

Néanmoins, l'utilisation de NX dans un journal python ne peut se faire presque uniquement que depuis l'interpréteur Python 3.64 intégré dans Simcenter et non un interpréteur configuré personnellement. Ceci veut dire que l'on a pas accès depuis Simcenter à toutes les autres bibliothèques externes, dont **NumPy** ou **openTURNS**, composants néanmoins nécessaires pour la génération de champs stochastiques et l'analyse de la pièce.

Pour contourner cette difficulté, le choix a été fait de scinder le programme en deux parties, une pour interagir directement avec Simcenter et le modèle de la

pièce, faire l'analyse , exporter les résultats de l'analyse dans un fichier externe, et ensuite faire appel à un programme Python exécutable qui lui possède toutes les bibliothèques externes et génère les champs stochastiques et les déformations en fonction des données exportées. Ce second script enregistre à son tour toutes les données sur les déformation qui sont réimportés dans le premier script et les déformations appliqués.

Un schéma explicatif de la méthodologie et de l'architecture des codes se trouve en annexe.

1.5.4 Commentaires et mise en contexte

Bien que entièrement fonctionnel, la méthodologie mise en place se heurte encore à des limites externes. En effet, au cours du projet, des questionnements ont eu lieu quant à l'outil à utiliser pour les simulations thermo-mécaniques, et l'utilisation de Simcenter a été remise en question au profit du logiciel Altair. Dans le cas où le choix retomberait sur Altair, la méthodologie développée ne pourra plus être gardée, au vue des différences probablement importantes entre le langage de scriptage d'Altair et de Simcenter.

1.6 Bilan intermédiaire

Au vu des complications ayant étées entraînées par la pandémie de COVID-19, un ralentissement notable dans le secteur de l'aviation a eu des conséquences néfastes sur le projet, et celui a perdu de son importance immédiate chez Phimeca, comme chez les autres entreprises collaborant sur ce projet. Ce faisant, l'absence de données concernant l'échangeur de chaleur ont entravé sa modélisation, et des discussions avec les collaborateurs des autres entreprises ont soulevé des problèmes dans la modélisation initialement envisagée de l'échangeur. En effet, sans informations supplémentaires ni sur les moyens de fabrication employés, ni les matériaux, ni sur les efforts exercés sur la pièce lors de son utilisation , nous ont empêchés d'inclure certaines grandeurs physiques et efforts sur lesquels nous n'avions pas d'à-priori, notamment la masse intrinsèque au plaques, et la pose des plaques crenelées, qui, se faisant séquentiellement, vont graduellement augmenter les efforts dans les niveaux de plaques inférieurs, et donc provoquer des déformation au fur et à mesure que le système est construit.

Nous avons donc décidé de mettre en pause cette partie du projet tant que des informations supplémentaires n'étaient pas obtenues, et de se concentrer sur l'intégration des codes et méthodes développés dans *openTURNS*, comme de faire une étude plus en détail de l'exemple de la poutre en flexion, et de voir l'implication des différents paramètres de modélisation en jeu lors d'une telle analyse.

2 Derniers 5 mois de stage

Comme mentionné précédemment, ces derniers mois de stage nous on permis de nous focaliser sur l'intégration des méthodes dans *openTURNS*, tout comme d'analyser plus en détail cette méthode.

2.1 Intégration dans openTURNS

2.1.1 Specifications pour l'intégration dans openTURNS

Le but de la partie intégration, était de réécrire les codes de manière à ce qu'il y aie le plus possible de fonction internes à *openTURNS* qui soient utilisées, de ne pas inclure d'autres librairies autres qu'*openTURNS* et la librairie standard de **Python**, tout comme de rendre des codes documentés et lisibles, facilitant la transcription ensuite des codes dans le lanqage **C++**.

2.1.2 Astuces employées pour l'intégration

Puisque le choix a été fait de travailler dans un cas général de système pouvant prendre en entrée une combinaison arbitraire de lois aléatoires scalaires et de champs aléatoires a dimensions variables, et de retransformer le système de manière à ce qu'il ne prenne en entrée qu'un vecteur aléatoire normal centré réduit, l'utilisation des méthodes incluses dans *openTURNS* était indispensable, notamment pour pouvoir traiter l'entierté des lois aléatoires présentes dans *openTURNS*, tout comme tout type de champ stochastique, qu'il soit issu de lois de covariance, ou bien à partie d'une regression sur des données.

- Pour permettre l'utilisation de tout type de champ, peu importe l'origine, l'idée était de travailler non pas sur le procéssus stochastique (qui est un objet à part dans *openTURNS*) mais sur l'objet contenant la décomposition de Karhunen Loeve, appelé **KarhunenLoeveResults** dans l'API.
Cet objet permet de passer directement du vecteur centrée normal réduit au champ et inversemment.
- Pour les lois aléatoires scalaires ceci est encore plus simple, puisque tout les lois scalairs dans *openTURNS* possèdent les méthodes **.IsoProbabilisticTransformation** et **.InverseIsoProbabilisticTransformation** qui permettent de passer d'une loi quelconque à une loi normale centrée réduite et inversment.
- Ces objets et méthodes combinées ont permis de créer le premier objet nécessaire à l'analyse de sensibilité sur champs stochastiques et a été apelé ici **AggregatedKarhunenLoeveResults**. Cet objet est crée en lui fournissant une liste ordonnée de **KarhunenLoeveResults** et de lois de distribution scalaires (il n'est pas possible ici d'utiliser des lois composées, des vecteurs aléatoire ou encore des décompositions de procéssus stochastiques composés, puisque l'utilisation de tels objets ammenerait trop d'ambiguités). Ensuite cet objet offre la possibilité de transformer des vecteurs aléatoires centrés normaux réduits (ou ensemble de vecteurs appelés **samples**) en une liste de champs stochastiques et de variables scalaires, ou de faire cette transformation inverse. *Cette modélisation soulève de nombreux problèmes pour le portage de ces codes dans le langage C++, en effet dans ce langage il n'existe pas de conteneur semblable à une liste pouvant contenir des objets de différents types. Cette contrainte impliquera qu'il faudra par-exemple séparer les différents objets et les indexer à part, et reconstruire la liste python dans une sur-couche adaptée.* Cet objet serait entièrement nouveau dans l'environnement *openTURNS*.

- Vu qu'une analyse de sensibilité s'appuie toujours sur une combinaison particulière de lois aléatoires et d'un système à analyser, il fallait un moyen simple pour permettre de facilement changer des paramètres dans les lois aléatoires, sans pour autant compromettre l'interaction lois/modèle. En effet, changer des paramètres comme le seuil d'approximation (*threshold*) de la méthode de Karhunen-Loeve sur un champ stochastique a de très importants effets sur le nombre de modes à utiliser pour représenter le champ avec une telle précision (Dans le pire des cas, le nombre de modes serait égal au nombre d'élément contenus dans l'espace discréteisé sur lequel on représente le champ). Ceci implique donc que le vecteur aléatoire qui pilote nos champs et variables scalaires est de taille variable, ce qui rend difficile, voire impossible de manuellement réécrire la fonction représentant notre système pour qu'elle prenne en entrée seulement le vecteur aléatoire. Ceci devient vraiment visible lorsqu'on cherche à faire une analyse de l'influence des paramètres des lois aléatoires sur l'analyse de Sobol', vu qu'à priori, à *threshold* constant, tout combinaison différente de paramètres des champs stochastiques va entraîner une longueur différente du vecteur aléatoire censé piloter notre système. Ceci fait que dans la philosophie de cette méthode, on travaille surtout avec des fonctions prenant en entrée une liste de champs (appelés **Field** dans *openTURNS*) et de variables scalaires, ou encore prenant en entrée une liste d'échantillons de champs et d'échantillons de points, si la fonction a été écrite pour traiter de nombreux cas en parallèle par exemple. Pour supprimer ce besoin de réécrire la fonction manuellement pour qu'elle prenne en entrée le vecteur aléatoire, on a créé un *wrapper* qui permet de transformer la fonction du système réel en une nouvelle fonction ne prenant en entrée que ce vecteur aléatoire. Cet objet s'appelle **KarhunenLoeveGeneralizedFunctionWrapper** et prend en entrée le modèle "réel" et l'objet **AggregatedKarhunenLoeveResults** défini précédemment.
- Pour la partie analyse de sensibilité, nous utilisons la méthode classique basé sur une expérience de monte-carlo, comme introduit dans la méthode issue du papier de [Wei et al., 2017]. La différence avec la méthode classique basée sur monte-carlo pour le calcul des indices de Sobol' est la manière de générer la matrice des mélanges. Le schéma explicatif de ce mélange se trouve en figure 8. L'implémentation de ce mélange se fait simplement, et peut facilement se traduire en C++. Pour la construction de l'objet permettant la génération d'échantillons, nous passons l'objet **AggregatedKarhunenLoeveResults**, la taille de l'échantillon et le booléen indiquant si l'on souhaite calculer les indices du second ordre ou non. Cette classe est appellée ici **KarhunenLoeveSobolIndicesExperiment**. Cette classe est conforme à la classe **SobolIndicesExperiment** dans *openTUNRS*, pour un fonctionnement identique des codes.
- Enfin, pour le calcul même des indices de Sobol', différentes méthodes existent. Pour n'en citer que celles qui sont implémentées dans *openTURNS*, il y a la méthode de *Martinez*, de *Saltelli*, de *Jansen* ou encore de *Mauntz Kucherenko*. Comme ces méthodes utilisent uniquement la réponse du modèle aux valeurs contenues dans la matrice de mélange il n'y a à-priori pas besoin d'avoir des informations concernant la matrice de mélange. Néanmoins, les classes pour le calcul des indices de Sobol'

d'openturns, requierent en entrée la matrice de mélange, premièrement pour récupérer le nom des différentes variables, et vérifier si la dimension de la matrice de mélange est bien cohérente avec le nombre d'élément contenus dans la réponse du modèle, en se basant sur la formule $N(2 + d)$, N étant la taille de l'échantillon, et d la dimension du problème. Néanomins, avec la méthode impliquant la décomposition de Karhunen-Loeve, la dimension de la matrice de mélange

3 Conclusions

References

- [Dumas, 2017] Dumas, A. (2017). Lois asymptotiques des estimateurs des indices de sobol. *confidentiel EDF*.
- [Goka, 2019] Goka, E. (2019). Analyse des tolérances des systèmes complexes – Modélisation des imperfections de fabrication pour une analyse réaliste et robuste du comportement des systèmes.
- [Lilburne and Tarantola, 2009] Lilburne, L. and Tarantola, S. (2009). Sensitivity analysis of spatial models. *International Journal of Geographical Information Science*, 23(2):151–168.
- [Michaël Baudin and Popelin, 2015] Michaël Baudin, Anne Dutfoy, B. I. and Popelin, A.-L. (2015). Openturns: An industrial software for uncertainty quantification in simulation.
- [Pronzato, 2019] Pronzato, L. (2019). Sensitivity analysis via Karhunen–Loève expansion of a random field model: Estimation of Sobol' indices and experimental design. *Reliab. Eng. Syst. Saf.*, 187:93–109.
- [Shang and Yun, 2013] Shang, S. and Yun, G. J. (2013). Stochastic finite element with material uncertainties: Implementation in a general purpose simulation program. *Finite Elements in Analysis and Design*, 64:65–78.
- [Sudret and Kiureghian, 2000] Sudret, B. and Kiureghian, A. D. (2000). Stochastic Finite Element Methods and Reliability A State-of-the-Art Report. *ResearchGate*.
- [Vink, 2020] Vink, R. (2020). A nonlinear water accumulation analysis in Python - Ritchie Vink. [Online; accessed 2. Mar. 2020].
- [Wei et al., 2017] Wei, P., Wang, Y., and Tang, C. (2017). Time-variant global reliability sensitivity analysis of structures with both input random variables and stochastic processes. *Struct. Multidiscip. Optim.*, 55(5):1883–1898.

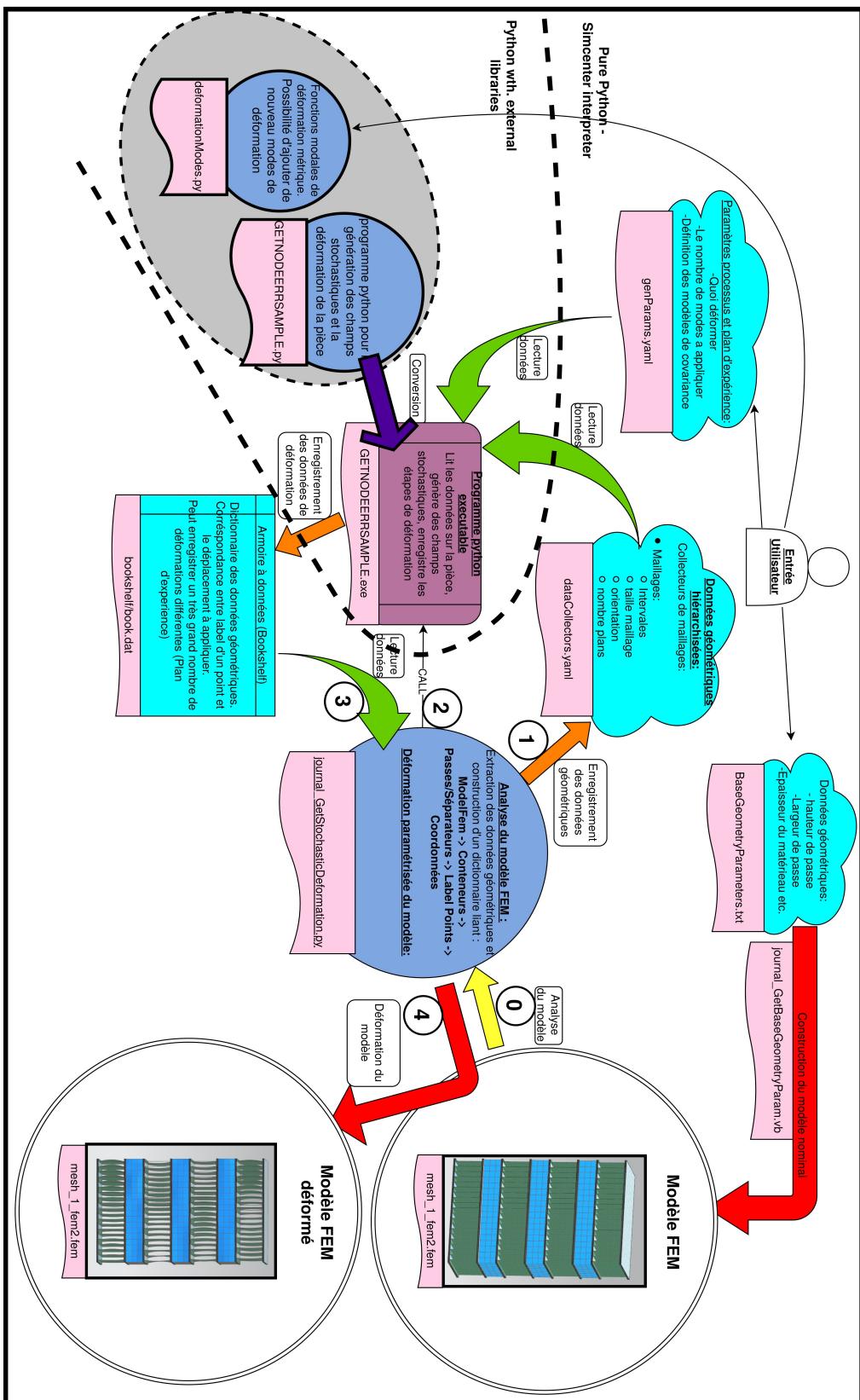


Figure 19: Architecture des programmes servant à introduire des déformations pilotées par champ stochastique dans l'échangeur de chaleur