Institut Supérieur de l'Aéronautique et de l'Espace

Développement de modèles réduits multifidélité en vue de l'optimisation de structures aéronautiques

Final Project

Internship tutors Dr. Joseph Morlier (ISAE/DMSM)

Student Rémi Vauclin (EMSE)

Résumé

Mon stage de fin d'études s'est déroulé au sein du département Mécanique des Structures et Matériaux à l'Institut Supérieur de l'Aéronautique et de l'Espace, du 1er avril au 12 septembre 2014. Celui-ci s'intègre dans une dynamique de recherche collaborative entre plusieurs acteurs du secteur aéronautique sur le sujet de l'Optimisation Multi-Disciplinaire (MDO) par l'intermédiaire de modèles réduits. Cette méthode consiste à coupler au plus tôt les différentes disciplines intervenant dans la conception de structures aéronautiques en vue d'optimiser leurs performances.

Plutôt que d'optimiser les variables de conception directement et ainsi faire appel à des simulateurs numériques coûteux, des modèles réduits sont utilisés et permettent d'accélérer substantiellement l'optimisation. Cependant, le couplage des disciplines inhérent à la MDO est un problème complexe pour l'élaboration de ces modèles. D'une discipline à l'autre, les hypothèses, les variables de conception ou encore la précision des calculs ne sont pas forcément les mêmes.

Dans ce rapport, nous nous intéressons à la problématique de construction d'un modèle réduit dans un contexte où plusieurs niveaux de précision (ou niveaux de fidélité) sont disponibles ; le couplage de variables aura déjà été réalisé en amont. On suppose par exemple que l'on dispose d'une méthode de résolution analytique formulée sous certaines hypothèses, d'une méthode semi-analytique, d'un code de calcul éléments finis plus précis mais lent à calculer, etc. L'enjeu principal du stage consiste en le développement et l'intégration en Python d'une méthode de construction de modèles réduits multifidélité puis à la recherche de stratégies d'optimisation adaptées aux problèmes des industriels.

Parmi les différents modèles existants, nous développons la méthode de krigeage (ou régression par processus gaussiens), particulièrement employée en aéronautique. Nous présentons les divers enrichissements qui existent aujourd'hui. Nous nous focalisons en particulier sur la formulation récursive du co-krigeage multifidélité décrite dans [Le Gratiet, 2013] qui présente à la fois des avantages en termes de vitesse de calcul, de lisibilité et d'adaptabilité. Nous enrichissons cette méthode en y ajoutant la possibilité d'estimer un bruit d'observationn. Nous abordons également plusieurs techniques de construction de plans d'expériences multifidélité respectant la condition d'imbrication inhérente à la formulation du co-krigeage multifidélité.

L'implémentation d'un code Python robuste et rapide est un enjeu essentiel. Les techniques classiques d'algèbre linéaire, combinées à plusieurs méthodes d'accélération de calcul, permettent de construire rapidement les modèles tout en assurant une certaine stabilité dans les résultats, en dépit de la complexité croissante des phénomènes mis en jeu vis-à-vis des enrichissements du krigeage classique.

Le co-krigeage est ensuite testé sur plusieurs cas analytiques et industriels pour définir dans quel contexte il peut améliorer significativement la précision d'un modèle par rapport au krigeage classique.

Des essais en optimisation sont construits, notamment pour étudier l'apport de la multifidélité dans le processus d'enrichissement du plan d'expériences pour optimiser la fonction objectif. Ces essais sont inachevés à l'heure où je rédige ce rapport mais pourront, je l'espère, être intégrés en partie dans la présentation finale.

Sommaire

Notations

$'$	Transposée du vecteur ou de la matrice
β	Coefficients de la régression de la tendance
β_{ρ_t}	Coefficients de régression pour $\rho_t(x)$
$\delta_{x\tilde{x}}$	Symbole de Krœnecker
det	Déterminant
ϵ	Résidus du modèle
log	Logarithme népérien
$\mathbf{1}$	Fonction indicatrice
Cov	Covariance
$\lvert\,.\,\rvert$	Valeur absolue
$\mu(x)$	Espérance du processus Z
$\odot$	Produit termes à termes
$\rho_{t-1}(x)$	Paramètre d'échelle entre deux niveaux Z_t et Z_{t-1}
σ^2	Paramètre de variance de la fonction de covariance
θ	Paramètre de portée de la fonction de covariance
d	Dimension du problème
F	Matrice contenant les valeurs de $f(X)$
$f(x)$	Fonction de régression pour la tendance de krigeage
$g(x)$	Fonction de régression pour le paramètre d'échelle $\rho(x)$
H	Matrice contenant les valeurs de $f(X)$, $g(X)$
K	Matrice de covariance, $K = \sigma^2 R$
$k(x, \tilde{x})$	Fonction de covariance, $k(x, \tilde{x}) = \sigma^2 r(x, \tilde{x})$
n	Nombre d'observations

p	Nombre de coefficients de régression pour la tendance
q	Nombre de coefficients de régression pour $\rho(x)$
R	Matrice de corrélation, $R = r(X, X)$
$r(x)$	$r(x, X)$
$r(x, \tilde{x})$	Fonction de corrélation
X	Plan d'expériences
$Z(x)$	Processus gaussien d'espérance $\mu(x)$ et de covariance $k(x, \tilde{x})$
$z(x)$	Réalisation du processus $Z(x)$
EMV	Estimation du maximum de vraisemblance
IMSE	Integrated Mean Squared Error
MDAO	Multidisciplinary Design Analysis Optimization
RMSE	Root Mean Squared Error

Partie I

Contexte du stage

I.1. Contexte de recherche

Depuis plusieurs décennies, le perfectionnement des outils informatiques permet de complexifier les processus d'optimisation industriels intervenant dans la phase de conception d'un produit. Les simulateurs numériques d'aujourd'hui sont capables de déterminer avec une excellente précision les phénomènes physiques mis en jeu. Ils permettent de s'affranchir de la majorité des tests réels, d'où une réduction considérable des coûts. Cependant, des résultats aussi complexes se font au prix de temps de calcul importants et ont tendance à transformer le simulateur en une véritable boîte noire. L'optimisation des paramètres en entrée pour minimiser la fonction objectif (telle que le poids, le coût, la déformation, etc.) devient un problème complexe qui ne peut plus être résolu de façon directe par le simulateur puisqu'un unique calcul peut prendre plusieurs heures, voire plusieurs jours, ce qui n'est pas envisageable en termes de budget.

La stratégie actuelle pour réduire de façon significative le temps de calcul consiste à construire une surface de réponse représentative du modèle, appelée modèle réduit (ou métamodèle). Cette surface de réponse doit approcher au mieux la fonction objectif pour un nombre d'appels minimal au simulateur. En optimisant le métamodèle plutôt que la fonction objectif, il est alors possible de réaliser d'importantes économies en temps de calcul.

Il existe de nombreuses façons de construire un métamodèle, faisant appel à certaines hypothèses adaptées au type de données. Dans le cas de dépendances simples, des modèles linéaires ou polynômiaux peuvent être utilisés, tandis que des interactions plus complexes requièrent la construction de modèles plus élaborés, tels que les réseaux de neurones, les fonctions à base radiales, les splines ou encore la régression par processus gaussiens (également appelée krigeage).

Pour améliorer encore ses performances, l'industrie aéronautique se place aujourd'hui dans une optique d'Optimisation Multi-Disciplinaire (MDO). Le nombre de disciplines mises en jeu dans la phase de conception est important et la tendance générale jusqu'alors consistait à optimiser ces disciplines indépendamment les unes des autres. La stratégie pour le futur vise à coupler chacune des disciplines au plus tôt dans la phase de conception pour s'orienter vers un optimum global au problème.

La formulation MDO d'un problème est complexe et ne sera pas abordée dans ce rapport. Toutefois, nous nous intéressons à l'aspect des données engendrées par une telle formulation. Supposons par exemple qu'il existe plusieurs approches différentes pour résoudre un même problème,

le tout avec une précision (ou fidélité) plus ou moins bonne et un temps de calcul plus ou moins important. Nous disposons alors d'un jeu de données multifidélité qui permettra potentiellement de mieux modéliser la fonction objectif.

Le cadre de recherche du stage englobe alors l'étude et la construction de modèles multifidélité pour s'intégrer dans la problématique MDO en vogue actuellement chez les industriels.

I.2. Acteurs et environnement de travail

L'ISAE est né du groupement des écoles d'ingénieur Supaéro (École Nationale Supérieure de l'Aéronautique et de l'Espace) et l'ENSICA (École Nationale Supérieure d'Ingénieurs de Constructions Aéronautiques) sous tutelle du ministère de la Défense à Toulouse. Il est aujourd'hui un pôle au rayonnement mondial dans le domaine de l'aéronautique. Afin de garantir innovation et proximité avec les industriels, les partenariats avec les entreprises sont au coeur de sa politique de développement et de recherche. Environ 38 millions d'euros sont alloués chaque année à la recherche.

La Direction de la Recherche et des Ressources Pédagogiques (DRRP) compte environ 200 personnes réparties dans 6 départements :

- Département Aérodynamique, énergétique et propulsion (DAEP)
- Département Mécanique des structures et matériaux (DMSM)
- Département Électronique, optronique et signal (DEOS)
- Département Mathématiques, informatique, automatique (DMIA)
- Département Langues, arts, cultures et sociétés (LACS)
- Centre aéronautique et spatial (CAS)

Mon stage s'est déroulé au sein du Département Mécanique des Structures et Matériaux (DMSM) du 1er avril 2014 au 12 septembre 2014. Ce département oriente ses recherches autour des problématiques d'endommagement de structures composites, de fatigue des matériaux et structures métalliques, de dynamique vibratoire et enfin de méthodes numériques pour la mécanique.

L'ONERA (Office National d'Études et de Recherche Aérospatiale) figure parmi l'un des partenaires majeurs de l'ISAE. Il s'agit du principal centre de recherche public français à caractère industriel et commercial. Une collaboration a été initiée entre les deux acteurs sur les modèles réduits dans le domaine de l'optimisation de structures. Le projet OSYCAF (Optimisation d'un système couplé fluide-structure représentant une aile flexible) a été initié en 2010 sur le thème de l'optimisation multi-disciplinaire et pour une durée de trois ans. L'idée majeure était d'établir une méthodologie de collaboration entre mécanique des structures et mécanique des fluides pour optimiser une aile d'avion par l'intermédiaire de modèles réduits. L'optimisation s'appuyait sur un modèle éléments finis Haute Fidélité. L'optimiseur étant un solveur à base de gradient, il était suggéré d'initialiser le processus en utilisant la meilleure solution obtenue par un code de calcul semi-analytique.

Actuellement, la collaboration se tourne vers l'aspect optimisation multidisciplinaire pour les aubes de turbines (à grande dimension, thèse CIFRE de Mohamed Boulhel à la SNECMA, principal motoriste aéronautique européen) et sur l'optimisation multifidélité en avant projet avion. Dans ce cadre d'échange, mon stage a été proposé au sein du Département Mécanique des Structures et Matériaux de l'ISAE, sous tutelle de Joseph Morlier. L'intitulé est le suivant : "Développement de métamodèles multifidélité en vue de l'optimisation de structures aéronautiques". L'ONERA, par le biais de Nathalie Bartoli, Thierry Lefebvre et Rémi Lafage, et la SNECMA, par le biais de Mohamed Bouhlel et de son encadrant Abdelkader Otsmane, ont effectué un suivi régulier de mes travaux.

La problématique multifidélité de mon stage s'intègre dans une formulation MDO complexe. La plateforme OpenMDAO (*Multidisciplinary Design Analysis and Optimization*, voir [Gray et al., 2010]) est un outil développé en langage Python qui aide la génération de la formulation MDO d'un problème en combinant différentes disciplines et en gérant les interactions entre elles. Parce qu'il est libre de droit, il fournit un environnement propice à l'intégration de métamodèles multifidélité dans une dynamique collaborative. C'est pourquoi il a été choisi que la partie informatique du stage soit réalisée en Python dans le but d'apporter une pierre à cet édifice.

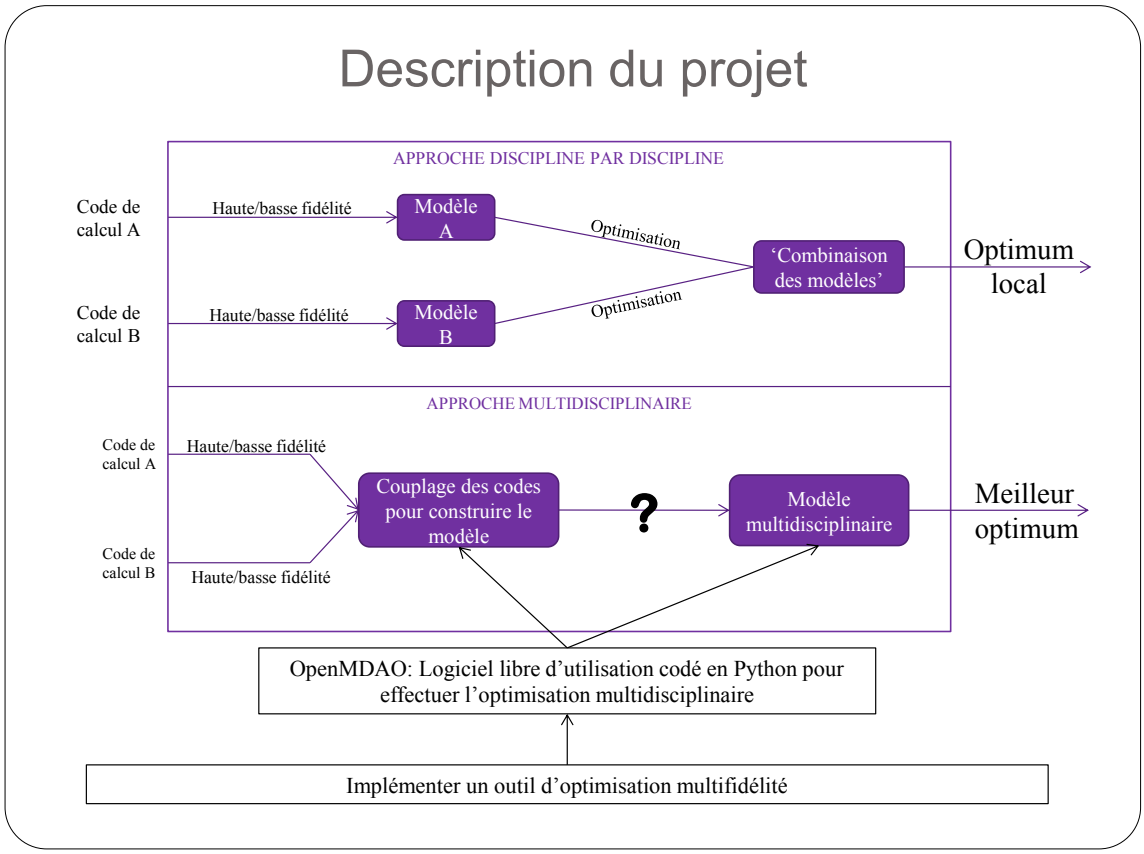


FIGURE I.1 – Intégration du projet dans le processus d'Optimisation Multi-Disciplinaire

I.3. Objectifs

Pour s'intégrer dans la problématique actuelle des chercheurs et industriels du milieu aéronautique, les enjeux de ce stage concernent les différentes phases du processus d'optimisation multifidélité, à savoir la construction d'un métamodèle et le processus d'enrichissement. Celui-ci s'est déroulé en quatre phases principales :

1. Une étude bibliographique sur les métamodèles, les plans d'expériences, critères d'enrichissement, appliqués à la problématique multifidélité et potentiellement en grande dimension
2. Le choix et le développement d'un métamodèle multifidélité en langage Python, avec des propositions de solutions aux problèmes bruités ou de grandes dimensions, ainsi que les processus d'optimisation associés
3. L'étude du comportement du métamodèle et du processus d'optimisation sur des cas tests et des cas industriels fournis par l'ONERA et la SNECMA
4. L'intégration du code implémenté en Python sous OpenMDAO

Les livrables principaux ont été les suivants :
— Présentation des premiers résultats issus du programme Python sur des cas analytiques (28 mai)
— Présentation des essais sur des cas réels ONERA et SNECMA, rédaction d'un rapport détaillé sur le comportement du modèle (11 juillet)
— Rendu du rapport ainsi que du programme final, présentation de son fonctionnement (9 septembre)

Le planning détaillé du stage est fourni en Annexe A.

Dans ce rapport, nous reprenons les fondamentaux dans la construction d'un métamodèle puis nous intéressons spécifiquement au krigeage avec des données multifidélité, en montrant ses performances et ses limites. Nous abordons le cas des problèmes à grande dimension, de plus en plus fréquents aujourd'hui, où la création d'un modèle de krigeage et le processus d'enrichissement deviennent un problème complexe. Une méthode pour construire le modèle de krigeage multifidélité en grande dimension est proposée. Nous comparons également plusieurs stratégies de construction de modèles et d'enrichissement pour minimiser la fonction objectif dans des cas réels.

Partie II

Approche théorique

Dans cette partie, nous abordons les éléments théoriques nécessaires à la construction d'un modèle réduit (ou métamodèle) multifidélité. L'orientation se fait en faveur du co-krigeage multifidélité qui s'est avéré être la méthode la plus prometteuse à l'issue de la recherche bibliographique.

Divers enrichissements de la méthode de krigeage initiale sont présentés, notamment en ce qui concerne le traitement de données bruitées (Section II.1.2.b.), l'estimation des paramètres (Section II.1.3.b.), la construction de plans d'expériences multifidélité (Section II.2.3.).

II.1. Construction d'un modèle de krigeage

II.1.1. Généralités

Nous nous plaçons dans le cas de la modélisation d'une fonction $f : \mathbb{R}^d \to \mathbb{R}$ inconnue dont nous disposons d'un vecteur d'observations $z \in \mathbb{R}^n$ limité, issu d'un jeu de données $X = (x_1, \ldots, x_n)$. d correspond à la dimension du problème. La construction du métamodèle consiste à exploiter ce jeu de données fixé pour connaitre au mieux la fonction objectif. Il existe diverses stratégies plus ou moins élaborées pour apprendre cette fonction ; nous présentons ici une liste non exhaustive des méthodes couramment utilisées aujourd'hui.

L'idée générale dans la construction d'un métamodèle consiste à supposer que la fonction objectif peut être décomposée en une somme de fonctions de base. La régression linéaire est la méthode la plus simple. Elle consiste à supposer qu'il existe une dépendance linéaire entre les paramètres $X = (x_1, \ldots, x_n)$ en entrée et la sortie z. Le modèle est construit de la façon suivante :

$$z = X\beta + \epsilon$$

où ϵ correspond aux résidus du modèle.

Pour approcher au mieux la fonction, il convient de minimiser ces résidus, c'est-à-dire de minimiser la quantité $z - X\beta$. La stratégie la plus employée, appelée méthode des moindres carrés, consiste à minimiser le carré des résidus (RSS, Residual Sum Squares), i.e la quantité

$$RSS = (z - X\beta)'(z - X\beta)$$

En dérivant par rapport à β, on obtient

$$\beta = \left(X'X\right)^{-1} X'z$$

À noter qu'il est possible de transformer les entrées X pour ne pas se restreindre à des interactions strictement linéaires. Les transformations usuelles sont les fonctions $x \to x^k, k = 0 \dots n-1$, permettant la construction de surfaces polynomiales, ou encore les fonctions $\log(.)$, $\sqrt{(.)}$, etc. Il est également possible d'étudier les interactions entre paramètres d'entrées en ajoutant à la matrice X les termes correspondants.

L'avantage d'un tel modèle est qu'il est très simple à construire et à interpréter. De plus, il fournit d'excellents résultats à coût réduit si l'hypothèse de linéarité entre entrées et sortie du système est vraie. Toutefois, il ne permet pas de construire des surface des réponses très complexes, du fait de l'hypothèse assez restrictive sur les données.

Il existe plusieurs généralisations à la régression linéaire qui cette fois, permettent d'observer des interactions bien plus complexes. Elles consistent par exemple à effectuer des régressions polynômiales morceau par morceau (splines) ou encore à supposer que la sortie est une combinaison linéaire de fonctions de base elles-mêmes construites par linéarité selon les entrées ou par d'autres fonctions. Le choix des fonctions de base est très important ; leurs propriétés conduiront à des analyses très différentes. À titre d'exemple, le choix de fonctions périodiques conduit à l'analyse de Fourier, le choix de fonctions d'activation (sigmoïde, échelon, etc.) conduit à un réseau de neurones tandis que le cas d'un unique "neurone" conduit au modèle de régression linéaire décrit dans le paragraphe précédent. Le choix de fonctions de base radiale comme fonctions d'activation aura également des propriétés très intéressantes.

La régression par processus gaussiens, aussi appelée krigeage, reprend les concepts décrits ci-dessus en employant des fonctions particulières associées à la corrélation spatiale entre les données en entrée. Le krigeage a été choisi comme modèle de développement ; nous présentons ci-après sa construction en détail.

II.1.2. Formulation du modèle de krigeage

II.1.2.a. Krigeage simple et krigeage universel

Le krigeage est une méthode qui a été développée par [Matheron, 1962] en géostatistique et repris dans tous les domaines grâce à ses qualités de modélisation, entre autres dans l'aéronautique. Pour construire le modèle, on fait l'hypothèse que la sortie $z(x)$ étudiée est la réalisation d'un processus gaussien Z. La réponse est modélisée comme un processus gaussien d'espérance $\mu(x)$ et de structure de covariance (ou noyau) :

$$k : (x, \tilde{x}) \to (\mathrm{Cov}(Z(x), Z(\tilde{x})) \tag{II.1}$$

La structure de corrélation spatiale aura une importance majeure dans la sortie du modèle.

Par exemple, si des données proches dans l'espace sont fortement corrélées, la fonction en sortie sera lisse et possédera des propriétés intéressantes en dérivation. En revanche, si la corrélation est mauvaise, le processus modélisé sera supposé plus chaotique.

Le noyau de covariance a la particularité d'être défini positif, ce qui aura une importance majeure par la suite. Dans le cas de notre étude, nous nous concentrons exclusivement sur des fonctions de covariance stationnaires, i.e. de la forme $k(x, \tilde{x}) = k(x - \tilde{x})$ par abus de notation. De plus, nous supposerons qu'elles ont la forme

$$k(x, \tilde{x}; \sigma^2, \theta) = \sigma^2 r(x - \tilde{x}; \theta) \tag{II.2}$$

Dans le cas d'un processus Z centré sur 0, la surface de réponse est conditionnée par les observations de la façon suivante :

$$\mu(x) = r'(x)R^{-1}z \tag{II.3}$$

où $R = r(X, X)$, $r(x) = r(x, X)$ pour simplifier la notation.

On peut lui associer une variance :

$$\mathrm{Var}(x) = \sigma^2 \left(1 - r'(x)R^{-1}r(x)\right) \tag{II.4}$$

Cette formulation correspond à l'appellation de krigeage simple.

Il est possible d'enrichir cette modélisation, en supposant cette fois que Z a une tendance (ou dérive) $m(x)$. Dans le cas où $m(x)$ est connue, on modifie le calcul de la moyenne de krigeage de la façon suivante :

$$\mu(x) = m(x) + r'(x)R^{-1}\left(z - m(x)\right) \tag{II.5}$$

La variance de krigeage reste inchangée.

Supposons maintenant la tendance $m(x)$ inconnue. Elle peut être supposée constante ou bien avoir la forme d'une régression linéaire de la forme $m(x) = F(x)\beta$, où $F = (f_1, ..., f_p)$, les vecteurs f_i étant les transformées de l'entrée x selon des fonctions de régression de base, $\beta = (\beta, ..., \beta_p)$ étant le vecteur des paramètres de la régression. Contrairement à la régression linéaire présentée ci-dessus, les résidus sont supposés corrélés selon une structure de covariance déterminée. Cette modélisation porte le nom de krigeage universel. La formulation du meilleur prédicteur et de la variance associée est alors la suivante :

$$\mu(x) = f'(x)\beta + r'(x)R^{-1}\left(z - F\beta\right) \tag{II.6}$$

$$\begin{aligned}\mathrm{Var}(x) = \sigma^2\Big[1 &- r'(x)R^{-1}r(x) \\ &+ \left(f'(x) - r'(x)R^{-1}F\right)\left(F'R^{-1}F\right)^{-1}\left(f'(x) - r'(x)R^{-1}F\right)'\Big]\end{aligned} \tag{II.7}$$

Dans le cas où on considère uniquement une tendance constante inconnue, les formules sont les suivantes :

$$\mu(x) = \frac{\mathbf{1}R^{-1}z}{\mathbf{1}R^{-1}\mathbf{1}} + r'(x)R^{-1}(z - \frac{\mathbf{1}R^{-1}z}{\mathbf{1}R^{-1}\mathbf{1}}) \tag{II.8}$$

$$\mathrm{Var}(x) = \sigma^2\left(1 - r'(x)R^{-1}r(x) \; + \; \frac{(\mathbf{1} - r(x)\mathbf{1})'(\mathbf{1} - r(x)'R^{-1}\mathbf{1})}{\mathbf{1}R^{-1}\mathbf{1}}\right) \tag{II.9}$$

Les formules présentées ci-dessus dénotent les propriétés particulières du krigeage. On peut remarquer que le meilleur prédicteur interpole les données et que la variance de prédiction est nulle en ces points. De plus, la variance de prédiction ne dépend pas de la valeur des observations.

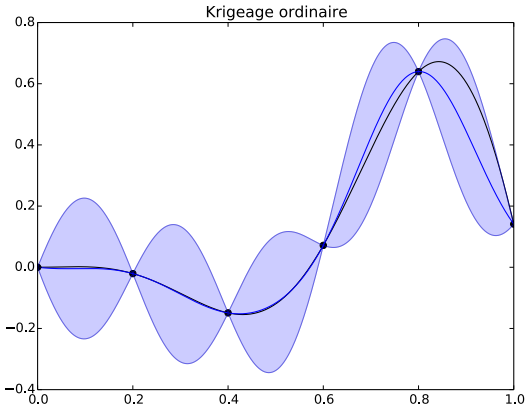


FIGURE II.1 – Allure classique d'un modèle de krigeage. Le modèle (en bleu) interpole les données en noir et la variance est nulle en ces points (en bleu pâle, les intervalles de confiance à 95%).

Remarque : Certains auteurs utilisent le gradient de la fonction objectif pour améliorer la construction du modèle. [Liu, 2003] propose cette technique tout en redéfinissant un algorithme d'optimisation des paramètres (cf. Section II.1.3.). Il construit également des modèles réduits avec des réseaux de neurones utilisant l'information du gradient. Ces techniques ne sont pas abordées dans ce rapport.

II.1.2.b. Krigeage "régressant"

Les propriétés interpolantes du krigeage sont intéressantes dans le cas où les données observées sont déterministes et non-bruitées. Toutefois, avec des possibles incertitudes de mesures expérimentales ou encore l'apparition de simulateurs stochastiques, il est possible que les données comportent un bruit d'observation, connu ou inconnu. Ceci est communément appelé *Effet pépite* en référence aux prélèvements de géostatisticiens qui peuvent tromber sur ladite pépite. Un modèle interpolant peut alors s'avérer de piètre qualité, comme on le voit sur la figure II.2.

Il existe une méthode assez simple pour pallier à cet effet, elle consiste à modifier la structure de covariance du modèle de krigeage. On peut écrire :

$$k(x, \tilde{x}; \theta, \sigma^2, \sigma_\epsilon^2) = \sigma^2 r(x, \tilde{x}; \theta) + \sigma_\epsilon^2 \delta_{x\tilde{x}} \tag{II.10}$$

où δ est le symbole de Krœnecker.

Ainsi, le krigeage devient "régressant" ; le meilleur prédicteur n'interpole plus les données et la variance n'est plus nulle aux points d'observation. L'utilisation de cette structure stabilise grandement la construction du modèle dans le cas où les données sont bruitées et que le conditionnement de la matrice est mauvais. Cette méthode correspond au procédé de régularisation de Tikhonov (voir [Tikhonov, 1963]).

Il est possible de fixer la valeur de σ_ϵ^2 pour améliorer le conditionnement de la matrice de corrélation, mais [J. Forrester et al., 2006] propose aussi une méthode qui permet de l'estimer. On écrit la matrice de covariance sous la forme

$$K = \sigma^2 (R + \lambda I) \tag{II.11}$$

où λ correspond au paramètre de bruit ; il est optimisé avec les paramètres θ lors de la maximisation de la vraisemblance.

On obtient les formules suivantes pour la moyenne et la variance de krigeage :

$$\mu(x) = f'(x)\beta + r'(x)\left(R + \lambda I\right)^{-1}(z - F\beta) \tag{II.12}$$

$$\begin{aligned}\mathrm{Var}(x) = \sigma^2 \Big[& 1 + \lambda - r'(x)\left(R + \lambda I\right)^{-1} r(x) \\ & + \left(f'(x) - r'(x)\left(R + \lambda I\right)^{-1} F\right)\left(F'\left(R + \lambda I\right)^{-1} F\right)^{-1}\left(f'(x) - r'(x)\left(R + \lambda I\right)^{-1} F\right)' \Big]\end{aligned} \tag{II.13}$$

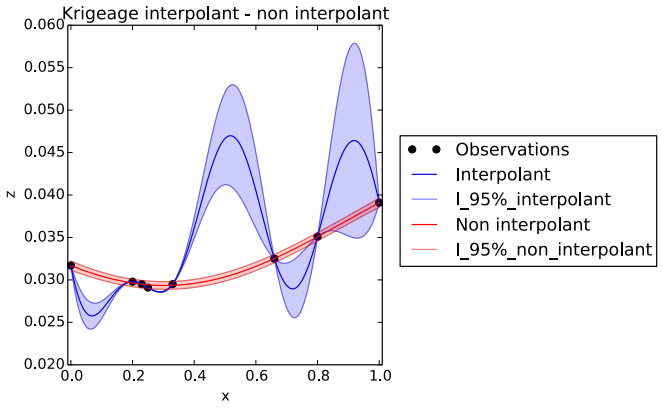


FIGURE II.2 – Comparaison entre un modèle de krigeage interpolant et un modèle non-interpolant sur des observations bruitées.

II.1.3. Estimation des paramètres

II.1.3.a. Calcul de la vraisemblance

Le krigeage fait intervenir un lot de paramètres inconnus : θ, σ^2 et β. La méthode du maximum de vraisemblance se prête particulièrement bien pour les calculer : on obtient une solution analytique simple pour σ^2 et β tout en garantissant rapidité et précision. Elle consiste à maximiser la fonction de vraisemblance

$$L(z_1, \ldots, z_n) = \prod_{i=1}^{n} f(z_i) \tag{II.14}$$

où f est la fonction de densité de probabilité du processus Z. Dans le cas du krigeage universel, par hypothèse on a :

$$f(z_i; \theta, \sigma^2, \beta) = \frac{1}{\sqrt{2\pi\sigma^2}^n \sqrt{\det(R)}} \exp\left(-\frac{1}{2\sigma^2}(z_i - F\beta)' R^{-1} (z_i - F\beta)\right) \tag{II.15}$$

Il est naturel de chercher à minimiser l'opposé de la log-vraisemblance, qui s'écrit (à une constante près) :

$$-\,l(z; \theta, \sigma^2, \beta) = \Big(n \log(\sigma^2) + \log\left(\det(R)\right)\Big)\left(-\frac{1}{2\sigma^2}(z - F\beta)' R^{-1} (z - F\beta)\right) \tag{II.16}$$

Il est possible de déterminer les paramètres β et σ^2 de façon analytique. En dérivant la formule par rapport à β, on obtient :

$$\beta = \left(F' R^{-1} F\right)^{-1} F' R^{-1} z \tag{II.17}$$

Ce qui permet d'obtenir σ^2 de la même façon :

$$\sigma^2 = \frac{(z - F\beta)' R^{-1} (z - F\beta)}{n} \tag{II.18}$$

En incluant (II.17) et (II.18) dans (II.16), on obtient alors :

$$-\,l(z_1, \ldots, z_n; \theta) = n \log(\sigma^2) + \log\left(\det(R)\right) \tag{II.19}$$

À noter que [Patterson and Thompson, 1971] remarque que la formule (II.19) peut induire un biais particulièrement important dans le cas où le nombre de degrés de libertés du modèle est petit. Pour cette raison, nous utilisons la log-vraisemblance restreinte qui s'écrit de la façon suivante :

$$-\,l(z_1, \ldots, z_n; \theta) = (n - p) \log(\sigma^2) + \log\left(\det(R)\right) \tag{II.20}$$

avec β et σ^2 déterminés de la façon suivante :

$$\beta = \left(F' R^{-1} F\right)^{-1} F' R^{-1} z \tag{II.21}$$

$$\sigma^2 = \frac{(z - F\beta)' R^{-1} (z - F\beta)}{n - p} \tag{II.22}$$

où p est la taille de β.

Il reste à estimer les paramètres $\theta = (\theta_1, \ldots, \theta_d)$ pour minimiser la log-vraisemblance. Il n'y a pas de solution analytique à ce problème ; il faut utiliser un optimiseur. On note dans la formule (II.20) qu'il faut inverser la matrice de corrélation R et calculer son déterminant, ce qui a une complexité algorithmique en $\mathcal{O}(n^3)$ où n est le nombre de points observés. Cette étape est la plus longue dans la construction du modèle ; il faut définir judicieusement la méthode d'optimisation.

Remarque : L'utilisation de la formule (II.20) peut avoir une conséquence importante sur l'estimation des coefficients.

II.1.3.b. Optimisation de la vraisemblance

La fonction de vraisemblance définie dans la formule (II.20) est relativement difficile à minimiser. Elle est fréquemment multimodale ; il est impossible d'utiliser un algorithme d'optimisation locale. À noter qu'avec la montée en dimension, ce problème devient rapidement très difficile et coûteux en temps, ce qui nuit à la performance du krigeage. Une alternative simple consiste à supposer que la fonction objectif est isotrope et alors $\theta_1 = \ldots = \theta_d$, d'où une optimisation en dimension 1. Ceci réduit grandement les performances de prédiction du modèle dans le cas (courant) où l'hypothèse d'isotropie est violée.

Il est possible d'obtenir une formule analytique des dérivées de la fonction de vraisemblance (II.20). Le détail des calculs est disponible dans [Toal et al., 2009] ou encore dans [Toal et al., 2011], en sachant que dans ces deux références, les auteurs utilisent la différenciation automatique (*Algorithmic differentiation*, voir [Griewank et al., 1989]) pour accélérer le calcul. Cette technique permet de calculer une dérivée au sein même du programme informatique en différenciant chacune des étapes élémentaires de calcul de la fonction. Le gradient peut alors être calculé extrêmement rapidement. De plus, le résultat énoncé dans [Toal et al., 2011] s'affranchit de la technique de différenciation automatique, ce qui est très intéressant en termes d'implémentation (celle-ci étant assez lourde à mettre en place) et peut provoquer une accélération substantielle pour tout algorithme d'optimisation globale utilisant basiquement une approximation du gradient de la fonction.

Les résultats sont définis pour une structure de covariance de type exponentiel généralisé de la forme :

$$k(x, \tilde{x}) = \sigma^2 \exp\left(- \sum_l 10^{\theta_l} |x_l - \tilde{x}_l|^{p_l} \right) \tag{II.23}$$

Pour un tel noyau, le gradient de la log-vraisemblance négative concentrée (II.20) notée ϕ selon un paramètre de portée θ_l s'énonce de la façon suivante :

$$\frac{\partial \phi}{\partial \theta_l} = - \log 10 \sum_{i,j} 10^{\theta_l} |x_{i,l} - x_{j,l}|^{p_l} R_{ij} \bar{R}_{ij} \tag{II.24}$$

Le gradient selon p_l s'écrit :

$$\frac{\partial \phi}{\partial p_l} = -\sum_{i,j} 10^{\theta_l} |x_{i,l} - x_{j,l}|^{p_l} \log |x_{i,l} - x_{j,l}| R_{ij} \bar{R}_{ij} \tag{II.25}$$

où la matrice "adjointe" $\bar{R}$ est calculée par :

$$\bar{R} = -\frac{1}{2}\left(R^{-1}(z - F\beta)(y - F\beta)' R^{-1} - R^{-1}\right) \tag{II.26}$$

Dans le cas où on procède à l'estimation d'un bruit (voir formule (II.11)), le gradient selon λ s'écrit :

$$\frac{\partial \phi}{\partial \lambda} = 10^{\lambda} \log 10 \sum_i \bar{R}_{ii} \tag{II.27}$$

Remarque : Le choix d'utiliser 10^{θ_l} et 10^{λ} plutôt que θ_l et λ est détaillé dans la Section III.1.2.a..

Remarque 2 : La formule (II.24) peut être extrapolée à d'autres types de noyaux. Pour un noyau de type $Matérn\frac{3}{2}$, on a :

$$\frac{\partial \phi}{\partial \theta_l} = -3 \log 10 \sum_{i,j} 10^{2\theta_l} \frac{|x_{i,l} - x_{j,l}|^2}{1 + \sqrt{3} 10^{\theta_l} |x_{i,l} - x_{j,l}|} R_{ij} \bar{R}_{ij} \tag{II.28}$$

Pour un noyau de type $Matérn\frac{5}{2}$, on a :

$$\frac{\partial \phi}{\partial \theta_l} = -\frac{5}{3} \log 10 \sum_{i,j} 10^{2\theta_l} \frac{|x_{i,l} - x_{j,l}|^2 \left(1 + \sqrt{5} 10^{\theta_l} |x_{i,l} - x_{j,l}|\right)}{1 + \sqrt{5} 10^{\theta_l} |x_{i,l} - x_{j,l}| + \frac{5}{3} 10^{\theta_l} |x_{i,l} - x_{j,l}|^2} R_{ij} \bar{R}_{ij} \tag{II.29}$$

Les formules sont légèrement plus compliquées dans ces deux cas mais font intervenir des termes relativement simples à calculer.

II.1.4. Krigeage et données multifidélité

Plaçons-nous dans le cas où plusieurs sources de données $z_1, \ldots, z_s$ sont disponibles pour un même problème. Celles-ci ont des niveaux de précision et un temps de calcul différents. On supposera que les codes les plus précis sont les plus lourds à évaluer et que l'on dispose de $n_1, \ldots, n_s$ échantillons avec $n_1 > \ldots > n_s$. Afin de réduire les budgets computationnels, on souhaite incorporer les données moins précises pour construire le modèle. Ces données peuvent être par exemple les résultats partiellement convergés d'un simulateur, les résultats d'un modèle numérique simplifié ou encore d'un modèle physique analytique.

II.1.4.a. Krigeage régressant avec différents niveaux de bruit

Une première approche pour résoudre le problème de données aux différentes fidélités consiste à supposer que les sources de données sont bruitées et que chacun des niveaux a un niveau de bruit différent. On suppose qu'elles peuvent toutes être modélisées par le même processus mais sont plus ou moins biaisées. On construit un unique vecteur $z = (z_1', \ldots, z_s')'$ puis on génère un vecteur d'incertitudes

$$\lambda = (\underbrace{\lambda_1, \ldots, \lambda_1}_{n_1}, \ldots, \underbrace{\lambda_s, \ldots, \lambda_s}_{n_s})'$$

avec (en théorie) $\lambda_1 > \ldots > \lambda_s$. On redéfinit alors la matrice de covariance :

$$K = \sigma^2 (R + \lambda I) \tag{II.30}$$

Les valeurs $\lambda_1, \ldots, \lambda_s$ sont estimées par maximum de vraisemblance avec les autres paramètres.

Cette méthode possède l'avantage d'augmenter assez peu le nombre de paramètres à estimer, en dépit de l'apport de niveaux de codes supplémentaires. En revanche, la matrice de covariance est maintenant de taille $(n_1 + \ldots + n_s)$ ce qui alourdit considérablement le temps de calcul de son inversion et de son déterminant pour la maximisation de la vraisemblance. La matrice a aussi plus de chances de présenter des singularités dans le cas où les données sont proches spatialement et l'inversion pourra poser davantage de problèmes. Il faut alors que les points des différents niveaux soient suffisamment distants les uns des autres.

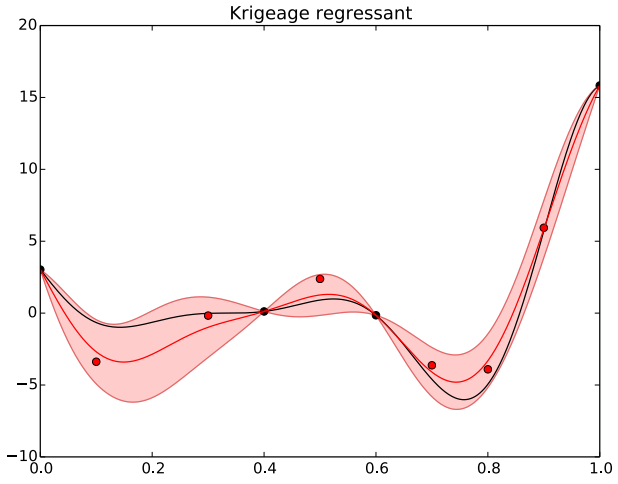


FIGURE II.3 – Illustration d'un cas avec 4 points estimés avec certitude (en noir) et 6 points où la mesure n'est pas bonne (en rouge, en ajoutant une perturbation uniforme sur $[-3; 3]$). Le modèle de krigeage "régressant" interpole les points noirs et essaie d'approcher au maximum les points rouges, tout en ayant une variance non nulle en ces points.

Elle aura un intérêt par exemple si pour un jeu de données, certaines n'ont pas été aussi bien mesurées que les autres (voir Figure II.3). En revanche, elle ne permettra pas de cerner d'interactions d'un autre type qu'un bruit pur (comme un biais systématique par exemple). Nous présentons dans la Section IV.1. un cas où cette situation ce présente.

II.1.4.b. Co-krigeage

Une deuxième approche consiste à utiliser le co-krigeage pour utiliser des données multi-fidélité. Le co-krigeage est une méthode qui permet d'approcher une fonction à partir de plusieurs jeux de données différents mais corrélés entre eux (voir par exemple [Vauclin et al., 1983]). Cette technique est fréquemment utilisée en géostatistique, où par exemple la présence de bassins aquifères peut être repérée à la fois par des prélèvements (coûteux et destructifs) et par des données sismiques plus faciles à obtenir.

Il sera alors intéressant de reconsidérer la modélisation, en réécrivant la réponse $(z_1, \dots, z_m)$ sous la forme d'un unique vecteur $Z = (z_1', \dots, z_s')'$ qui est la réalisation d'un processus gaussien $\tilde{Z}$ à construire. Il convient alors de réécrire la matrice de covariance du processus, de la forme :

$$K = \begin{pmatrix} k_1(x,x) & k_{12}(x,x) \\ k_{12}(x,x) & k_2(x,x) \end{pmatrix} \tag{II.31}$$

où k_{12} représente une structure de covariance croisée entre les deux sorties, à définir.

Ceci a une application tout à fait intéressante dans le cas où l'on dispose de plusieurs jeux de données aux fidélités différentes pour simuler une même réponse. Par hypothèse, il existe une corrélation entre chacune des réponses (étant donné qu'elles décrivent le même objet) et il est alors possible d'utiliser le co-krigeage pour tirer parti au maximum de toutes les informations.

De la même façon que dans la méthode décrite ci-dessus, la restructuration de la matrice de covariance alourdit de façon significative le temps de calcul du modèle de co-krigeage. En effet, il convient d'estimer simultanément les paramètres $(\theta, \sigma^2, \beta)$ de chacuns des modèles et d'inverser une matrice de covariance de taille $(n_1 + \ldots + n_s)$ dans la boucle d'optimisation de la vraisemblance, ce qui peut être extrêmement coûteux en temps.

Toutefois, le co-krigeage a fait l'objet de nombreux perfectionnements. Nous nous plaçons dans le cas où nous disposons de deux réponses z_1, z_2 issues de deux simulateurs aux précisions différentes, avec z_1 la réponse "basse fidélité" peu coûteuse en temps de calcul et z_2 la réponse "haute fidélité" plus lente à calculer. Elles sont modélisées par deux processus Z_1 et Z_2. L'idée du co-krigeage avec données multi-fidélité consiste à apprendre la relation entre les deux niveaux de code pour pouvoir exploiter au mieux les données basse fidélité, plus rapides à calculer.

Il est possible de considérer une erreur additive $err = z_2 - z_1$, multiplicative $err = \frac{z_2}{z_1}$, etc. Ceci permet d'identifier des relations plus complexes que dans la section précédente.

[Kennedy and O'Hagan, 2000] propose d'utiliser une formulation autorégressive d'ordre 1 (AR1) de la forme

$$Z_2(x) = \rho Z_1(x) + \delta(x) \tag{II.32}$$

où ρ est un paramètre d'échelle scalaire entre les deux niveaux de code et δ est l'erreur résiduelle. Avec cette formulation, il fait l'hypothèse que

$$\mathrm{Cov}(Z_2(x), Z_1(\tilde{x})|Z_2(x)) = 0 \ \forall x \neq \tilde{x} \tag{II.33}$$

Cela correspond à dire qu'un point basse fidélité n'apporte aucune information supplémentaire au modèle aux endroits où la réponse haute fidélité est connue. Alors les paramètres à estimer sont ceux de Z_1 et δ.

Sous cette hypothèse, il est montré que les paramètres $(\theta_1, \sigma_1^2, \beta_1)$ peuvent être estimés indépendamment de $(\theta_2, \sigma_2^2, \beta_2)$. La matrice de covariance peut alors être restructurée en conséquence ; le lecteur est amené à lire [Kennedy and O'Hagan, 2000] pour obtenir un descriptif détaillé.

Récemment, [Le Gratiet, 2013] a enrichi ce modèle et propose d'utiliser un paramètre d'échelle $\rho(x)$ de la forme d'une régression linéaire, dont les paramètres peuvent être déterminés analytiquement. [Qian and Wu, 2008] propose également d'utiliser une dépendance spatiale pour ρ, de telle façon qu'il est lui-même la réalisation d'un processus gaussien. Dans ce cas, il n'existe plus de formule analytique pour obtenir les paramètres. Ceux-ci doivent être calculés par analyse bayésienne en utilisant des méthodes de type MCMC (Monte Carlo Markov Chain) qui alourdissent considérablement le temps de calcul du modèle.

II.1.4.c. Co-krigeage avec formulation récursive

[Le Gratiet, 2013] reprend le concept du modèle AR1 pour un nombre quelconque de niveaux de fidélité et montre qu'il est possible de calculer la moyenne et la variance de krigeage de façon récursive. Ainsi, non seulement les paramètres de chacun des niveaux de codes peuvent être estimés indépendamment, mais en plus il n'est plus nécessaire de reconstruire une matrice de covariance de taille $(n_1 + \ldots + n_m)$ qui devra être inversée dans la boucle d'optimisation. Construire un modèle de krigeage avec s niveaux de fidélité devient aussi coûteux en temps que construire s modèles séparément. Il suffit simplement d'inverser des matrices de tailles n_i, ce qui en réduit grandement la complexité temporelle (de $\mathcal{O}((n_1 + \ldots + n_m)^3)$ à $\mathcal{O}(n_1^3 + \ldots + n_m^3)$) et spatiale (de $\mathcal{O}((n_1 + \ldots + n_m)^2)$ à $\mathcal{O}(n_1^2 + \ldots + n_m^2)$).

Supposons maintenant que nous disposons d'un nombre s de niveaux de codes. Soit $X_1, \ldots, X_s$ les plans d'expériences associés à chaque niveau. Pour tout $t \in [1, \ldots, s]$ les formules du co-krigeage simple (tendance $m_t(x)$ et paramètre d'échelle $\rho_{t-1}(x)$ connus) deviennent au niveau t :

$$\begin{aligned} \mu_t(x) = \rho_{t-1}(x)\mu_{t-1}(x) + m_t(x) \\ + r_t'(x)R_t^{-1}\left(z_t - \rho_{t-1}(X_t) \odot z_{t-1}(X_t) - m(x)\right) \end{aligned} \tag{II.34}$$

$$\mathrm{Var}_t(x) = \rho_{t-1}^2(x)\mathrm{Var}_{t-1}(x) + \sigma^2\left(1 - r_t'(x)R^{-1}r_t(x)\right) \tag{II.35}$$

Remarque : La notation $z_{t-1}(X_t)$ correspond au vecteur de la réponse au niveau $t-1$ aux points du plan X_t de fidélité supérieure. Cette formulation requiert alors l'imbrication des plans d'expériences, $X_s \subset \ldots \subset X_1$ (cf. Section II.2.3.).

Dans le cas du krigeage universel, le paramètre $\rho_{t-1}(x)$ peut prendre la forme d'une régression de type $\rho_{t-1}(x) = g_{t-1}(x)\beta_{\rho_{t-1}}$. On estime conjointement les coefficients de régression β_t pour $m_t(x)$ et $\beta_{\rho_{t-1}}$ pour $\rho_{t-1}(x)$. Pour cette raison, il faut redéfinir les objets $f_t(x)$, F_t, β_t que l'on notera respectivement $h_t(x)$, H_t et $\Sigma_t\nu_t$ pour respecter les notations introduites dans [Le Gratiet, 2013]. On a :

$$h_t'(x) = \begin{cases} f_1'(x) & \text{si } t = 1 \\ [g_{t-1}'(x)\mu_t(x) \quad f_t'(x)] & \text{si } t > 1 \end{cases} \tag{II.36}$$

$$H_t = \begin{cases} F_1 & \text{si } t = 1 \\ \left[G_{t-1} \odot \left(z_{t-1}(X_t)\mathbf{1}_{q_{t-1}}'\right) \quad F_t\right] & \text{si } t > 1 \end{cases} \tag{II.37}$$

La moyenne et la variance de krigeage se calculent alors :

$$\mu_t(x) = h_t'(x)\Sigma_t\nu_t + r_t'(x)R_t^{-1}\left(z_t - H_t\Sigma_t\nu_t\right) \tag{II.38}$$

$$\begin{aligned} \mathrm{Var}_t(x) = \sigma^2_{\rho_{t-1}}(x)\mathrm{Var}_{t-1}(x) + \frac{\left(z_t - H_t\Sigma_t\nu_t\right)'\left(z_t - H_t\Sigma_t\nu_t\right)}{n_t - p_t - q_{t-1}}\left(1 - r_t'(x)R_t^{-1}r_t(x)\right) \\ + \left(h_t'(x) - r_t'(x)R_t^{-1}H_t\right)\Sigma_t\left(h_t'(x) - r_t'(x)R_t^{-1}H_t\right)' \end{aligned} \tag{II.39}$$

avec

$$\Sigma_t = \left(H_t'\frac{R_t^{-1}}{\sigma^2}H_t\right)^{-1}$$

$$\nu_t = H_t'\frac{R_t^{-1}}{\sigma^2}z_t$$

$$\sigma_{\rho_{t-1}}(x) = g_{t-1}'(x)\left([\Sigma_t]_{q_1,\ldots,t-1\times 1,\ldots,q_{t-1}} + [\Sigma_t\nu_t]_{1,\ldots,q_{t-1}}[\Sigma_t\nu_t]'_{1,\ldots,q_{t-1}}\right)g_{t-1}(x)$$

Les paramètres du modèle de co-krigeage récursif sont obtenus par maximisation de la vraisemblance avec une légère modification par rapport à l'équation (II.20) :

$$-l(z_1,\ldots,z_n;\theta) = (n-p-q)\log(\sigma^2) + \log\left(\det(R)\right) \tag{II.40}$$

où p est la taille de β, q est la taille de β_ρ. σ^2 déterminé de la façon suivante :

$$\sigma^2 = \frac{\left(z - H\Sigma\nu\right)'R^{-1}\left(z - H\Sigma\nu\right)}{n-p-q} \tag{II.41}$$

Remarque : La relation entre chaque niveau de code est facilement visible dans le cas du krigeage ordinaire, ce qui rend la formulation récursive plus explicite à comprendre que dans les travaux de [Kennedy and O'Hagan, 2000] et permet bien de visualiser la contribution de chacun des niveaux de code pour la prédiction.

Remarque 2 : Similairement au krigeage universel, le modèle de co-krigeage interpole les points d'observation de fidélité maximale et la variance est nulle en ces points, bien que la lecture en soit moins évidente avec les formules énoncées ci-dessus. La figure II.4 illustre ce comportement.

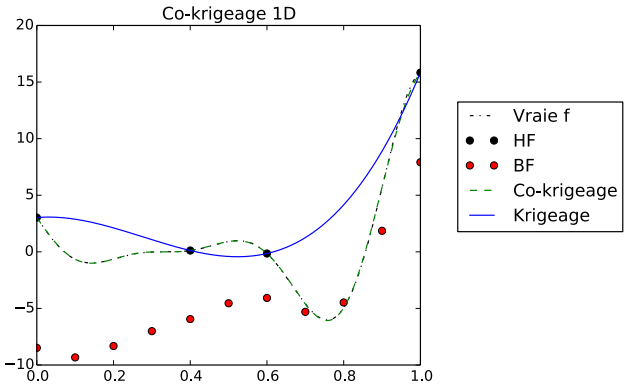


FIGURE II.4 – Allure d'un cas d'école de co-krigeage construit avec 4 points Haute Fidélité en noir et 10 points Basse Fidélité en rouge, construits par une transformation simple de la fonction. Le modèle de co-krigeage en vert interpole les points noirs et approche quasi-exactement la fonction, là où le krigeage ordinaire en bleu ne parvient pas à cerner son comportement.

II.1.5. Validation du modèle

II.1.5.a. Techniques de validation

Une fois le modèle construit, celui-ci doit faire l'objet d'une validation afin de savoir si ses performances en prédiction sont satisfaisantes. Il est relativement courant que le modèle construit ait de mauvaises performances ; ceci peut être dû à un plan d'expériences mal choisi, des hypothèses de départ inadaptées ou encore à une maximisation de la vraisemblance inachevée.

Pour s'affranchir de tels problèmes, il existe plusieurs approches. La pratique usuelle consiste à séparer le plan d'expériences initial en deux lots : un plan d'apprentissage qui sert à construire le modèle, puis un plan de validation qui permet d'étudier les erreurs de modélisation.

Il faut définir un critère pour mesurer la qualité d'un modèle. Dans ce rapport, nous utilisons le $RMSE$ défini par :

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(z_i - \hat{z}_i)^2}{n}} \tag{II.42}$$

où z, et $\hat{z}$ et désignent respectivement les valeurs de la fonction et la prédiction du modèle sur le plan de validation.

Ce critère donne un ordre de grandeur de l'erreur de prédiction. Il existe également une version normalisée appelée $NRMSE$ qui donne une erreur standard indépendante de l'ordre de grandeur de y. Nous utilisons également le Q^2 comme critère de qualité. Il est défini par la formule suivante :

$$Q^2 = 1 - \frac{\sum_{i=1}^{n} (z_i - \hat{z}_i)^2}{\sum_{i=1}^{n} (z_i - \overline{z})^2} \tag{II.43}$$

où $\overline{z}$ désigne la moyenne de z.

Ce critère permet d'évaluer la qualité de la prédiction par rapport au plus simple des modèles : la moyenne. Plus le Q^2 est proche de 1, plus la prédiction est bonne. Si le Q^2 est inférieur à 0, cela signifie que le modèle est de moins bonne qualité qu'une simple moyenne.

Si les coûts d'évaluation de la fonction sont très élevés, il peut être difficile de séparer les données en un plan d'apprentissage et un plan de validation. On utilise alors la validation croisée (ou *k-fold cross-validation*) qui consiste à diviser les données en k lots puis à construire le modèle avec $k-1$ lots, le dernier servant de base de validation. Cette opération est répétée k fois pour que tous les lots aient servi une fois de base de validation. La méthode *Leave-One-Out* reprend ce principe en fixant $k = n$ où n est le nombre d'observations disponibles ; les lots contiennent alors un unique point.

La validation croisée permet d'utiliser l'information de tous les points tout en obtenant une erreur de prédiction pour chacun d'eux. On maximise ainsi le "rendement" des données à disposition. En revanche, il faut construire k sous-modèles, ce qui est assez coûteux en temps si la dimension du problème est importante. [Dubrule, 1983] propose une méthode approximative qui consiste à conserver les paramètres θ du modèle pour chaque sous-modèle puis à modifier les matrices R et F en supprimant uniquement les lignes et colonnes des points servant à la base de validation. Les paramètres β et σ^2 peuvent être recalculés de façon simple à partir des matrices réduites dans les formules (II.21) et (II.22). Il n'est pas nécessaire de recalculer R^{-1}, il suffit d'appliquer les formules d'inversion par blocs pour obtenir simplement l'inverse de la matrice réduite.

II.1.5.b. Application à la multifidélité

La multifidélité apporte un problème supplémentaire pour valider le modèle. Si les données sont disponibles en quantité suffisante, il suffit de construire un plan d'apprentissage multifidélité et un plan de validation Haute Fidélité. Dans le cas contraire, il est possible d'utiliser la validation croisée ; les formules définies par [Dubrule, 1983] sont adaptées dans [Le Gratiet, 2013].

Attention toutefois, sauf erreur de notre part, il subsiste des erreurs dans les formules énoncées. corrections apportées sont écrites en rouge dans les formules, en conservant les mêmes notations.

Notons ξ les indices supprimés lors de la procédure.
— $A_{[\xi,\xi]}$ est la sous-matrice contenant les lignes et colonnes ξ de A
— $A_{[-\xi,-\xi]}$ est la sous-matrice à laquelle on a supprimé les lignes et colonnes ξ
— $A_{[-\xi]}$ est la sous-matrice à laquelle on a supprimé les lignes ξ

— $A_{[-\xi,xi]}$ est la sous-matrice à laquelle on a supprimé les lignes ξ et conservé uniquement les colonnes ξ
— $a_{[\xi]}$ est le sous-vecteur contenant les éléments ξ de a
— $a_{[-\xi]}$ est le vecteur auquel on a ôté les éléments ξ

Supposons que l'on supprime les données X_{test} contenant les indices ξ_s. On obtient l'erreur estimée de prédiction ϵ_{Z_s,ξ_s} au niveau s par la formule :

$$\left(\epsilon_{Z_s,\xi_s} - \rho_{s-1}(X_{test}) \odot \epsilon_{Z_{s-1},\xi_{s-1}}\right) \left[R_s^{-1}\right]_{[\xi_s,\xi_s]} = \left[R_s^{-1}\left(z_s - H_s\lambda_{s,-\xi_s}\right)\right]_{[\xi_s]} \tag{II.44}$$

avec

$$H_s = \begin{bmatrix} G_{s-1} \odot \left(z_{s-1}(X_s)1'_{q_{s-1}}\right) & F_s \end{bmatrix}$$
$$\lambda_{s,-\xi_s} = \left([H_s]'_{[-\xi_s]} K_s [H_s]'_{[-\xi_s]}\right)^{-1} [H_s]'_{[-\xi_s]} K_s z_s(X_s \backslash X_{test})$$
$$\rho_{s-1}(X_{test}) = g'_{s-1}(X_{test})[\lambda_{s,-\xi_s}]_{1,\dots,q_{s-1}}$$
$$K_s = \left[R_s^{-1}\right]_{[-\xi_s,-\xi_s]} - \left[R_s^{-1}\right]_{[-\xi_s,\xi_s]} \left(\left[R_s^{-1}\right]_{[\xi_s,\xi_s]}\right)^{-1} \left[R_s^{-1}\right]_{[\xi_s,-\xi_s]}$$

et la variance estimée de prédiction $\sigma^2_{Z_s,\xi_s}$ par la formule :

$$\sigma^2_{Z_s,\xi_s} = \sigma^2_{\rho_{s-1},-\xi_s}(X_{test}) \odot \sigma^2_{Z_{s-1},\xi_{s-1}} + \sigma^2_{s,-\xi_s} \operatorname{diag}\left(\left(\left[R_s^{-1}\right]_{[\xi_s,-\xi_s]}\right)^{-1}\right) + \mathcal{V}_s \tag{II.45}$$

avec

$$\sigma^2_{s,-\xi_s} = \frac{\left(z_s(X_s\backslash X_{test}) - [H_s]_{[-\xi_s]}\lambda_{s,-\xi_s}\right)' K_s \left(z_s(X_s\backslash X_{test}) - [H_s]_{[-\xi_s]}\lambda_{s,-\xi_s}\right)}{n_s - p_s - q_{s-1} - n_{train}}$$
$$\sigma^2_{\rho_{s-1},-\xi_s}(X_{test}) = g'_{s-1}(X_{test})\left(\Sigma_{\rho_{s-1},-\xi_s} + [\lambda_{s,-\xi_s}]_{1,\dots,q_{s-1}} [\lambda_{s,-\xi_s}]'_{1,\dots,q_{s-1}}\right) g_{s-1}(X_{test})$$
$$\Sigma_{\rho_{s-1},-\xi_s} = \left[\left([H_s]'_{[-\xi_s]} \frac{K_s}{\sigma^2_{s,-\xi_s}} [H_s]_{[-\xi_s]}\right)^{-1}\right]_{[1,\dots,q_{s-1},1,\dots,q_{s-1}]}$$
$$\mathcal{V}_s = \mathcal{U}'\left([H_s]'_{[-\xi_s]} \frac{K_s}{\sigma^2_{s,-\xi_s}} [H_s]_{[-\xi_s]}\right)^{-1} \mathcal{U}$$
$$\mathcal{U} = \left(\left[R_s^{-1}\right]_{[\xi_s,\xi_s]}\right)^{-1} \left[R_s^{-1}H_s\right]_{[\xi_s]}$$

NB : L'erreur survient suite à la confusion entre matrice de covariance et matrice de corrélation. La définition de K_s ci-dessus devrait intégrer le terme σ^2 ; les corrections sont apportées uniquement aux endroits où l'erreur se répercute.

II.2. Plans d'expériences

II.2.1. Généralités

Dans la construction d'un métamodèle, chaque observation de la fonction objectif peut représenter un temps de calcul considérable. Pour cette raison, il est important de bien choisir son plan d'expériences pour obtenir le meilleur modèle possible. Selon les hypothèses de départ, le plan pourra être construit différemment. Par exemple, lors de la construction d'un modèle de régression linéaire sans interactions, on choisira un plan en étoile, lequel permet de cerner au mieux les relations pour chacune des composantes.

Toutefois, dans la majorité des cas qui nous intéressent, il n'est pas possible d'émettre d'hypothèses aussi simplificatrices et l'enjeu du plan d'expériences revient alors à couvrir la plus grande partie du domaine de définition de la fonction. Il faut aussi que le plan ait de bonnes qualités en projection, afin d'éviter d'éventuelles réplications de données dans le cas où une variable ne serait pas influente.

Parmi les plans à remplissage d'espace, les LHD (Latin Hypercube Designs) sont les plus utilisés grâce à leur simplicité de fabrication et leurs qualités générales. Ils se construisent de la façon suivante : on découpe chaque dimension de l'espace en n sections puis chaque section doit contenir un et un seul point. En pratique, il suffit d'utiliser les permutations de $1, \ldots, n$ pour remplir chaque dimension. Une perturbation aléatoire est ajoutée à chaque point afin d'éviter l'uniformité des distances en projection, pouvant conduire à des échantillonnements malchanceux dans le cas de réponses périodiques. La figure II.5 présente une telle construction.

Par construction, les LHD ont d'excellentes qualités en projection. Toutefois, il est possible qu'un LHD remplisse très mal l'espace. Dans la figure II.5, il n'y a pas de points en haut à droite et en bas à gauche du domaine. Il est alors nécessaire d'utiliser des algorithmes pour améliorer les critères de qualité du plan.

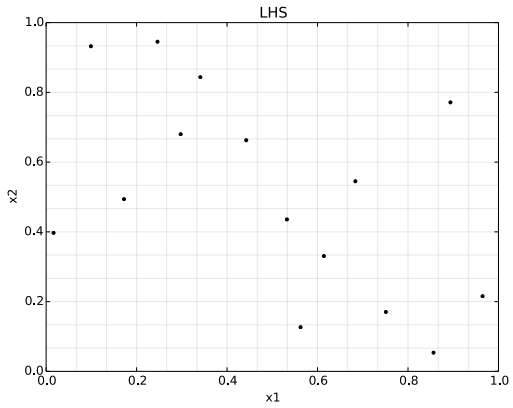


FIGURE II.5 – Exemple de LHD en deux dimensions

II.2.2. Optimisation des LHD

Il existe différents critères pour mesurer la qualité d'un plan d'expériences. Nous utiliserons deux critères différents : la distance euclidienne minimale entre deux points D_{min} et le critère ϕ_p défini par [Morris and Mitchell, 1995] de la façon suivante :

$$\phi_p = \left(\sum_{1 \leq i < j \leq n} d_{ij}^{-p} \right)^{\frac{1}{p}} \tag{II.46}$$

Les qualités en projection sont assurées par les LHD ; nous n'étudions pas ce critère par la suite.

Nous présentons ici l'algorithme ESE (*Enhanced Stochastic Evolutionary*), défini par [Jin et al., 2005] qui permet d'obtenir des LHD de qualité et qui sera utilisé dans la partie d'implémentation (partie III. C'est un algorithme d'optimisation globale qui reprend l'idée du *simulated annealing* (recuit simulé) utilisé en cristallographie, selon lequel on alterne des phases de chauffage et refroidissement d'un matériau pour minimiser l'énergie qu'il contient. La figure II.6 présente un tel plan en deux dimensions.

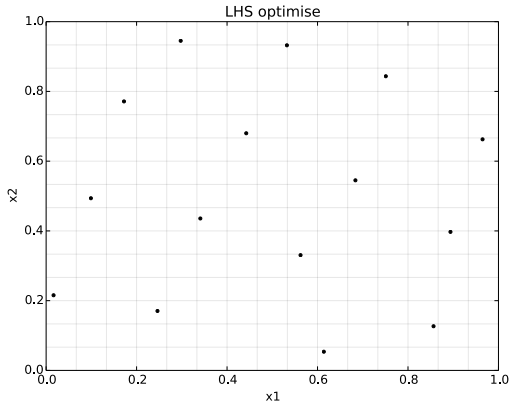


FIGURE II.6 – LHD optimisé avec le critère ϕ_p

On définit un paramètre appelé "température" qui évolue au cours de l'optimisation. L'augmentation de la température favorisera l'exploration (choix d'un plan d'expériences éloigné et potentiellement moins bon que le plan optimal actuel), tandis que sa diminution favorisera l'exploitation (recherche d'un plan d'expériences similaire au plan actuel et ayant de meilleures propriétés).

La figure II.7 décrit le processus d'optimisation d'un plan d'expériences avec l'algorithme ESE.

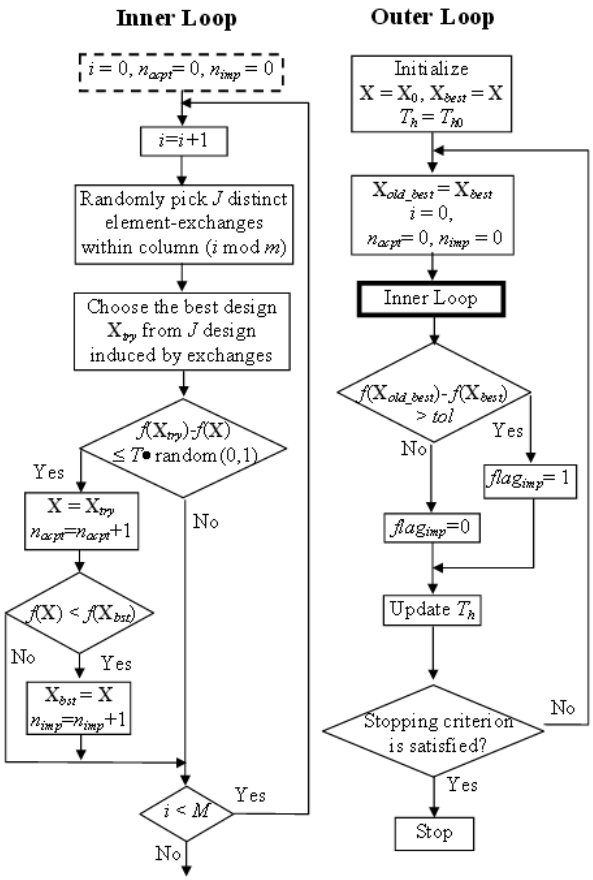


FIGURE II.7 – Fonctionnement de l'algorithme ESE (source : [Jin et al., 2005])

Le comportement de l'algorithme est régi par plusieurs paramètres :

1. J : nombre de plans créés par permutation de deux colonnes.
2. M : nombre d'itérations de la boucle interne.
3. *tol* : tolérance pour l'optimisation du critère (on acceptera un plan avec un critère moins bon jusqu'à la limite définie par *tol*).
4. *stop* : critère d'arrêt de la boucle externe. Fixé à un nombre d'itérations pour simplifier la procédure et maîtriser le temps de calcul.

Chacun de ces paramètres influence grandement le temps de calcul et la qualité du plan d'expériences en sortie.

La phase de calcul la plus lourde concerne le calcul du critère ϕ_p du plan d'expériences. Celui-ci requiert en effet de calculer la matrice de distance entre chacun des points du plan, ce qui a une complexité en temps de $\mathcal{O}(dn^2)$. En sachant que ce calcul est fait pour un nombre J de plans à chaque itération de la boucle intérieure (elle-même imbriquée dans la boucle extérieure), nous avons donc un nombre très important de plans dont il faut calculer le critère, d'où un coût élevé en grande dimension. Il est toutefois possible de calculer facilement le critère ϕ_p d'un plan après permutation

de deux colonnes, connaissant celui du plan initial, le tout avec une complexité temporelle en $\mathcal{O}(n)$. La permutation est le coeur de l'algorithme ESE ; le calcul complet du critère ϕ_p ne se fait alors qu'une seule fois, d'où une économie en temps de calcul importante.

II.2.3. Plans d'expériences imbriqués

La formulation du co-krigeage définie dans la Section II.1.4.c. requiert l'imbrication des plans d'expériences de chacun des niveaux de code, $X_s \subset \ldots \subset X_1$. Dans certains cas, il est possible que la stratégie de remplissage d'espace ne soit pas la plus adaptée pour construire un plan d'expérience multifidélité. Par exemple, si un niveau de fidélité plus bas est invalide au bord du domaine de définition tandis que le niveau de haute fidélité parvient à de bons résultats sur tout le domaine, il peut être intéressant non pas de remplir tout l'espace pour le niveau Haute Fidélité mais d'échantillonner davantage sur les bords, tandis que le niveau basse fidélité sera mieux réparti. Ceci requiert cependant des informations a priori sur le comportement de chaque niveau de fidélité. Dans la suite, nous considérerons uniquement la construction de plans à remplissage d'espace multifidélité.

De nombreux auteurs proposent des méthodes pour construire de tels plans. [Rennen et al., 2010] propose par exemple d'adapter l'algorithme ESE présenté ci-dessus en modifiant les paramètres de l'algorithme d'échange pour que les permutations aléatoires ne se fassent qu'entre points d'un même niveau ou bien par groupe pour respecter l'imbrication. Il demeure le risque qu'une mauvaise initialisation du plan multifidélité empêche d'atteindre un plan de qualité ; l'algorithme est alors lancé pour différents plans initiaux, ce qui alourdit la procédure.

Une nouvelle stratégie est introduite dans ce paragraphe (sous réserve qu'une méthode similaire n'ait déjà été proposée, la bibliographie n'a pas été approfondie dans ce sens). Elle reprend l'idée de n'effectuer les permutations de l'algorithme ESE qu'entre des points d'un même niveau de fidélité mais diffère de [Rennen et al., 2010] dans le sens où l'optimisation du plan se fait niveau après niveau, plutôt qu'une optimisation générale d'un plan multifidélité initial.

Supposons que l'on souhaite construire deux plans d'expériences imbriqués X_1, X_2 de tailles n_1 et n_2 avec $X_2 \subset X_1$ (peu importe la dimension).

La démarche est la suivante :

1. Construire le LHD Haute Fidélité de n_2 points de façon normale (avec l'algorithme ESE par exemple).
2. Découper chaque dimension de l'espace en n_1 sections égales et vérifier les sections qui sont déjà occupées par le plan Haute Fidélité.
3. Remplir chacune des $n_1 - n_2$ sections vides restantes aléatoirement par un et un unique point (pour respecter la règle de construction d'un LHD).
4. Utiliser l'algorithme ESE en n'autorisant que les permutations entre les $n_1 - n_2$ points ajoutés.

Cette stratégie permet d'obtenir un plan multifidélité qui garantit les propriétés attendues, à savoir :

— Le plan Haute Fidélité est un LHD optimisé d'excellente qualité
— Les plans de fidélité inférieure sont tous les LHD et possèdent des propriétés tout à fait satisfaisantes

Remarque : Cette méthode n'est robuste que dans le cas où les données Basse Fidélité sont $k \in \mathbb{N}$ fois plus nombreuses que les données Haute Fidélité. En effet, dans le cas contraire, le redécoupage de l'espace (2) peut poser problème : deux points Haute Fidélité peuvent apparaître dans la même section. Il est possible de traiter ce cas en réduisant la perturbation aléatoire ajoutée lors de la création du LHD initial ou bien en déplaçant légèrement les points en conflit pour qu'ils ne se chevauchent plus au sein d'une même section. Ce cas n'est pas abordé ; il est supposé que cette condition est toujours respectée.

Si la condition d'écart entre deux jeux de données successifs n'est pas respectée, il demeure possible d'employer la stratégie adoptée dans [Le Gratiet, 2013] où l'imbrication se fait par recherche des plus proches voisins. On construit le LHD Haute Fidélité optimisé de façon standard, puis pour chaque niveau de fidélité inférieur on construit un LHD (optimisé également) auquel on substitue les plus proches voisins du LHD de niveau supérieur pour garantir l'imbrication. Ceci permet d'avoir un bon plan pour le niveau de fidélité le plus élevé ; les plans des niveaux inférieurs seront légèrement moins bons mais l'effet sera moins perceptible au final.

Remarque : Cette procédure est implémentée dans le package $MuFiCoKriging$ en R mais ne fonctionne pas toujours correctement si les LHD de chaque niveau sont mal construits. Nous reprenons des exemples et effectuons la comparaison entre les deux méthodes dans la partie Implémentation (cf. III.2.2.).

II.3. Optimisation

La construction d'un modèle de krigeage peut avoir deux rôles différents : la prédiction ou l'optimisation. Dans le premier cas, le modèle permet de prédire le comportement de la fonction en des endroits non explorés tandis que dans le second, le modèle permet d'avoir une idée de l'endroit susceptible de contenir le minimum de la fonction.

Ce sont deux stratégies complètement différentes : un modèle construit pour l'optimisation ne sera pas forcément bon en prédiction et réciproquement, le minimum de la fonction sera approximatif dans un modèle prédictif.

Dans le processus d'optimisation d'un métamodèle, il convient de bien connaître ses propriétés afin de choisir l'algorithme le plus adapté. L'idée générale consiste à construire un plan d'expériences de taille réduite pour construire un modèle initial puis à l'enrichir dans les zones susceptibles de contenir le minimum de la fonction, mais il reste à définir ces zones de l'espace. [Jones, 2001] fournit

un inventaire détaillé pour différents types de métamodèles. Dans cette partie, nous nous intéressons à l'optimisation pour un modèle de krigeage, et en particulier l'algorithme EGO.

II.3.1. Algorithme EGO dans le cas déterministe

La connaissance de l'incertitude dans un modèle de krigeage est un atout majeur. On peut définir les zones intéressantes comme un compromis entre la valeur du meilleur prédicteur et l'incertitude associée au modèle. L'algorithme EGO (*Efficient Global Optimization*, cf. [Jones et al., 1998]) combine ces deux informations pour minimiser la fonction objectif.

Il se déroule selon les étapes suivantes :

1. Construction du modèle de krigeage à partir d'un plan d'expériences initial
2. Maximisation d'un critère d'enrichissement appelé *Expected Improvement* (EI). Voir (II.47)
3. Enrichissement au point obtenu par la maximisation
4. Reconstruction du modèle avec le point supplémentaire ; retour à l'étape 2.

L'EI se calcule selon la formule suivante :

$$EI(x) = (y_{min} - \mu(x))\Phi\left(\frac{y_{min} - \mu(x)}{\sigma(x)}\right) + \sigma(x)\phi\left(\frac{y_{min} - \mu(x)}{\sigma(x)}\right) \tag{II.47}$$

où $\phi(.)$ et $\Phi(.)$ sont respectivement la densité et la fonction de répartition de la loi normale.

L'EI permet de faire un compromis entre l'exploration du domaine dans les zones où le modèle a une variance élevée et l'exploitation dans les zones où les prédictions du modèle sont les plus petites. Dans le cas du krigeage avec données déterministes (non bruitées), le critère est nul aux points d'observation ; la convergence est assurée.

La figure II.3.1. montre le comportement de l'algorithme dans un cas compliqué d'optimisation en une dimension. Le modèle initial est choisi volontairement avec peu de points (ici 3). L'algorithme se trompe au début et enrichit au voisinage d'un minimum local. Toutefois, il parvient à changer de bassin d'attraction et à trouver la solution du problème à la huitième itération.

Remarque : De nombreux auteurs modifient l'EI selon le problème. [Sasena, 2002] fournit un panel de critères et détaille lesquels sont les mieux adaptés pour une situation donnée (notamment en grande dimension, cf. Section II.3.3.).

Remarque 2 : Il est possible d'enrichir le modèle avec plusieurs points simultanément. [Ginsbourger et al., 2009] démontre d'ailleurs que l'enrichissement point par point est une méthode sub-optimale. Attention cependant, la résolution du problème consistant à déterminer les points qui maximisent l'EI est relativement complexe. [Viana et al., 2013] définit l'algorithme MSEGO (Multiple Surrogate EGO) qui consiste plus simplement à construire différents modèles réduits (différentes hypothèses de krigeage, réseaux de neurones utilisant des RBF, splines, etc.). Cette démarche comporte toutefois le risque d'échantillonner dans les mêmes régions si les prédictions sont de qualité.

II.3.2. Cas de données bruitées

Nous avons vu précédemment que l'algorithme EGO n'est adapté que pour des problèmes déterministes. En effet, si on suppose que les données sont bruitées, alors la convergence n'est plus systématique ; l'EI n'est plus nul aux points observés et il devient possible d'enrichir plusieurs fois au même endroit (ou dans un voisinage très proche) lorsque le bruit a une part importante dans la variance de krigeage (si le bruit est très faible, le comportement de l'algorithme sera perturbé uniquement pour un critère de convergence très restrictif). Ceci nuit à la fois aux performances de l'algorithme mais également à la construction du modèle de krigeage. En effet, deux points trop proches détériorent le conditionnement de la matrice de corrélation et peuvent rendre impossible la décomposition de Cholesky (cf. Section III.1.1.).

Pour cette raison, il convient de redéfinir une stratégie d'optimisation. [J. Forrester et al., 2006] définit le krigeage "ré-interpolant", qui consiste à reconstruire un modèle de krigeage en supposant cette fois que la variance est nulle aux points d'observation. Ainsi l'EI devient artificiellement nul et l'algorithme peut converger (avec la possibilité que la limite ne soit pas la solution optimale). Cette stratégie peut être assez lourde en temps de calcul et plus délicate à manipuler.

Une autre stratégie consiste à modifier le critère d'enrichissement en lui ajoutant une pénalité

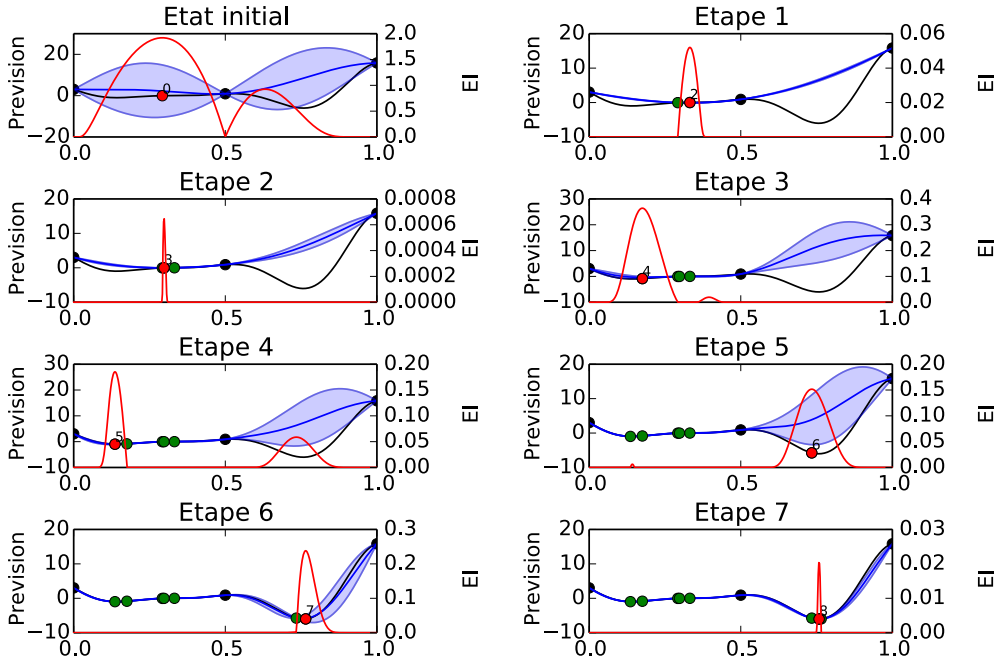


FIGURE II.8 – Déroulement de l'algorithme EGO pour une fonction 1D bimodale. En noir, la fonction objectif, en bleu, le modèle avec intervalles de confiance à 95% et en rouge la valeur de l'EI associée.

fonction du bruit, ce qui évite d'enrichir plusieurs fois dans un voisinage trop proche.

II.3.3. Cas de la grande dimension

Le critère EI a globalement de bonnes qualités pour de petites dimensions ; tant que $d < 15$, l'algorithme EGO est assez performant et trouve la solution du problème en un nombre d'itérations raisonnable. Toutefois, plus la dimension augmente, plus les zones mal connues par le modèle sont nombreuses. En conséquence, l'enrichissement se fait de plus en plus en faveur de l'exploration du domaine plutôt que de l'intensification autour d'un optimum local. Le nombre d'itérations devient très important, ce qui en termes de budget n'est plus satisfaisant.

Pour améliorer les performances de l'algorithme, on peut pénaliser l'EI de façon à forcer l'intensification dans les zones où la fonction est susceptible de contenir son minimum. [Sasena, 2002] propose simplement d'utiliser le critère appelé $wb2$ défini par

$$wb2(x) = EI(x) - \mu(x) \tag{II.48}$$

où μ est la prédiction du modèle de krigeage.

Cette construction, en dépit de sa simplicité, améliore substantiellement le comportement de l'algorithme dans la plupart des cas.

II.3.4. Optimisation sous contraintes

Dans une problématique industrielle, de nombreuses contraintes viennent complexifier l'optimisation de la fonction. On distingue deux types de contraintes : les contraintes égalité et les contraintes inégalité. Pour réaliser l'optimisation avec un modèle de krigeage, il existe différentes stratégies.

Si les contraintes sont très simples à modéliser, il peut parfois suffire de pénaliser la fonction dans les zones de violation des contraintes. Ainsi on se ramène à une optimisation sans contraintes. La définition de la pénalisation peut cependant s'avérer délicate : l'ajout d'une constante conduit à des discontinuités qui nuisent aux performances des algorithmes d'optimisation, l'ajout d'un effet quadratique peut poser problème au voisinage de la zone de violation, etc.

Une autre approche proposée par [Da Veiga and Marrel, 2012] consiste à reconstruire un modèle qui modifie la définition du processus gaussien du modèle initial pour prendre en compte intrinsèquement les contraintes (telles que les bornes de la fonction, sa monotonie ou encore sa convexité) dans la modélisation.

Si en revanche, les contraintes sont plus délicates à calculer, ce qui sera fréquemment le cas dans l'industrie, de telles stratégies ne sont plus applicables ; il faut se diriger vers d'autres méthodes. La plus simple consiste à reprendre EGO et utliser un algorithme d'optimisation qui prend en compte les contraintes (COBYLA par exemple). [Sasena et al., 2002] redéfinit un critère d'enrichissement

qui décrit la faisabilité d'un point, ce qui est particulièrement utile dans le cas où les zones de respect des contraintes sont petites ou disjointes.

II.3.5. Optimisation multifidélité

Reprenons maintenant le problème d'optimisation sous un angle multifidélité. De nouvelles questions apparaissent : à quel niveau de fidélité enrichir, comment définir les contraintes selon les niveaux de fidélité, etc. Pour faire au plus simple, il suffit de considérer que l'enrichissement se fait au plus haut niveau de fidélité en reprenant l'algorithme EGO classique. Toutefois, ceci n'est pas forcément optimal. En effet, la formulation récursive du modèle permet de connaître facilement le poids de chaque niveau de fidélité dans le calcul de la moyenne et de la variance de krigeage.

[Le Gratiet, 2013] propose un critère qui utilise les informations propres à chaque niveau pour déterminer où il est intéressant d'enrichir le plan initial. Celui-ci utilise les résultats issus de la validation croisée (voir Section II.1.5.b.) avec le calcul de l'erreur et la variance associée) ainsi qu'un partitionnage de l'espace par construction de cellules de Voronoï dans le cas d'un enrichissement avec plusieurs points simultanément.

Ce critère est plutôt orienté vers la construction d'un modèle prédictif plutôt que vers la recherche d'un minimum ; il cherche à réduire au mieux l'erreur de prédiction.

La construction d'un critère d'enrichissement multifidélité dans le but d'optimiser la fonction objectif n'a pas pu être abordée au cours du stage. Toutefois, les travaux existants fournissent une base solide pour envisager un critère exploitant au mieux les nombreuses informations issues du co-krigeage avec formulation récursive.

Partie III

Implémentation et étude comportementale du modèle

L'objectif majeur du stage était d'élaborer un code Python pour construire des modèles multi-fidélité adaptés à la problématique d'optimisation multi-disciplinaire. Le co-krigeage présenté dans la partie précédente a été choisi comme solution à ce problème.

La formulation du co-krigeage multifidélité de façon récursive est récente. Elle permet la résolution d'un problème multifidélité complexe tout en présentant de nombreux avantages en termes de rapidité de calcul, clarté des résultats et simplicité à mettre en oeuvre. Toutefois, elle est disponible uniquement dans le package *MuFiCoKriging* ([Le Gratiet et al., 2012]).

Le projet se déroulant en environnement Python, deux choix ont été possibles :

— Wrapper le package R dans Python, au moyen d'outils comme Rpy, Rpy2

— Développer un outil en Python en se basant sur les Toolbox de krigeage existantes

L'encapsulation de code est un procédé assez rigide. Bien qu'il permette d'obtenir rapidement des résultats, il alourdit l'installation de l'outil et rend difficile la maintenance du code par les dépendances multiples qu'il induit. De plus, cela rend complexe d'éventuelles modifications du coeur de l'algorithme pour enrichir la modélisation avec des cas non disponibles en R. Par ailleurs, le langage Python est en plein essor au sein de la communauté scientifique et propose de plus en plus d'outils en rapport avec notre sujet. Pour ces raisons, il a été choisi de redévelopper le co-krigeage avec formulation récursive en Python, en vue de l'intégrer par la suite dans la plateforme OpenMDAO. La toolbox Scikit-Learn ([Pedregosa et al., 2011]), permettant de faire du krigeage universel, a offert une base de programmation pour implémenter la suite du travail.

Le co-krigeage a été implémenté pour permettre également de faire du krigeage universel à un unique niveau de fidélité, ceci pour faciliter les comparaisons et visualiser l'impact des différents enrichissements proposés dans ce rapport.

Dans cette partie, nous présentons les méthodes pratiques d'implémentation du co-krigeage puis nous effectuons les tests sur les différentes techniques implémentées pour valider leur robustesse et leur stabilité.

III.1. Considérations mathématiques

III.1.1. Transformations algébriques

Nous avons vu dans la Section II.1.3. que l'optimisation des paramètres de krigeage requiert le calcul du déterminant et l'inversion de la matrice de covariance. Ces deux opérations sont des problèmes informatiques de complexité temporelle $\mathcal{O}(n^3)$, ce qui est assez lourd puisqu'elles doivent être refaites à chaque itération lors de la maximisation de la vraisemblance. Heureusement, les propriétés de la matrice de covariance permettent d'en réduire la complexité.

Nous avons vu que la fonction de covariance est symétrique, définie positive. Par conséquent, la matrice R est symétrique définie positive. il est possible d'utiliser la décomposition de Cholesky pour obtenir la forme suivante :

$$R = CC'$$

où C est une matrice triangulaire inférieure. Alors le déterminant se calcule très simplement :

$$\det(R) = \det(CC') = \det(C)^2 = \prod_{i=1}^{n} C_{ii}^2$$

À noter que la précision numérique en Python est relativement mauvaise (de l'ordre de 10^{-16}). Les termes C_{ii} peuvent prendre des valeurs très faibles si les corrélations des données en entrée sont très fortes, et alors le produit des C_{ii}^2 peut donner des résultats erronés. Pour cette raison, nous utilisons la formule équivalente :

$$\log\left(\det(R)\right) = n \log\left(\prod_{i=1}^{n} C_{ii}^{\frac{2}{n}}\right)$$

Pour accélérer encore les calculs, il est possible de s'affranchir de l'inversion de R. En effet, on ne cherche jamais à obtenir R^{-1} de façon explicite mais seulement à résoudre des systèmes de la forme $RX = b$. Ainsi, on peut opérer les calculs suivants :

$$\beta = \left(F'R^{-1}F\right)^{-1} F'R^{-1}z$$

$$\beta = \left(F_c'F_c\right)^{-1} F_c'z_c \text{ avec } \begin{cases} CF_c &= F \\ Cz_c &= z \end{cases}$$

On utilise généralement la décomposition QR pour résoudre le problème des moindres carrés : $F_c = QU$ avec Q orthogonale et U triangulaire.
Alors :

$$\beta = \left((QU)^{-1}QU\right)^{-1} (QU)'z_c$$

$$\beta = \left(U'U\right)^{-1} U'Q'z_c$$

$$\beta = U^{-1}Q'z_c$$

D'où

$$U\beta = Q'z_c$$

On simplifie également le calcul de σ^2 :

$$\sigma^2 = (z - F\beta)'R^{-1}(z - F\beta)$$
$$\sigma^2 = (z_c - F_c\beta)'(z_c - F_c\beta)$$

La moyenne et la variance de krigeage peuvent subir les mêmes transformations :

$$\mu(x) = f'(x)\beta + r'(x)R^{-1}(z - F\beta)$$
$$\mu(x) = f'(x)\beta + r'_c(x)(z_c - F_c\beta)$$

$$\begin{aligned}
\sigma^2(x) &= \sigma^2\Big(1 - r'(x)R^{-1}r(x) \\
&\qquad + \left(f'(x) - r'(x)R^{-1}F\right)\left(F'R^{-1}F\right)^{-1}\left(f'(x) - r'(x)R^{-1}F\right)'\Big) \\
\sigma^2(x) &= \sigma^2\Big(1 - r'_c(x)r_c(x) + \left(f'_c(x) - r'_c(x)F_c\right)\left(F'_cF_c\right)^{-1}\left(f'_c(x) - r'_c(x)F_c\right)'\Big) \\
\sigma^2(x) &= \sigma^2\Big(1 - r'_c(x)r_c(x) + \left(f'_c(x) - r'_c(x)F_c\right)\left(U'U\right)^{-1}\left(f'_c(x) - r'_c(x)F_c\right)'\Big) \\
\sigma^2(x) &= \sigma^2\Big(1 - r'_c(x)r_c(x) + \Gamma\Gamma'\Big) \quad \text{avec } \Gamma = (f'_c(x) - r'_c(x)F_c)U^{-1}
\end{aligned}$$

La décomposition de Cholesky a une complexité temporelle en $\mathcal{O}\Big(d^3 + \frac{nd^2}{2}\Big)$, la décomposition QR a une complexité temporelle en $\mathcal{O}(nd^2)$, la résolution d'un système triangulaire de n équations se fait en n opérations. par conséquent, le gain en temps de calcul est substantiel. De plus, cette méthode apporte un gain en stabilité par rapport à l'inversion standard.

En dépit de ces améliorations, il est possible, lorsque deux observations sont très proches, que la matrice R soit trop mal conditionnée, ce qui rend impossible la décomposition de Cholesky. Ce problème peut facilement être résolu en choisissant un plan d'expériences optimisé (cf. Section II.2.), alors dans ce cas les points seront suffisamment éloignés les uns des autres. Mais ce problème pourra réintervenir dans le cadre de l'optimisation séquentielle. Nous traitons de ce cas dans la section II.3.2..

Remarque : Les notations utilisées pour les calculs ci-dessus correspondent au krigeage universel à un unique niveau de fidélité ; l'adaptation à la multifidélité n'est qu'une question d'écriture.

III.1.2. Optimisation de la vraisemblance

III.1.2.a. Traitement de l'optimisation des paramètres

Les paramètres θ du modèle de krigeage doivent être déterminés par un algorithme d'optimisation globale. Ceux-ci peuvent prendre des valeurs très différentes selon la nature de la corrélation des données en entrée. Pour des données fortement corrélées, nous avons $\theta \to 0$, tandis que pour des données peu corrélées, $\theta \to +\infty$. En particulier, la recherche de petites valeurs peut être compliquée pour un optimiseur dont la portée initiale est fixée pour pouvoir chercher dans tout le domaine $[0, +\infty[$.

Pour cette raison, nous appliquons la transformation $\log_{10}$ ce qui facilite la recherche de θ dans tout le domaine. Les structures de corrélation sont alors énoncées en fonction de 10^{θ}. La même opération est réalisée dans le cas où on cherche à estimer le bruit ; celui-ci est en effet généralement très faible. On ajoute alors à la diagonale de la matrice de corrélation le terme 10^{λ}.

Remarque : Cette transformation présente en plus l'avantage de résoudre le problème d'une optimisation bornée par 0 dans le cas où l'algorithme ne respecte pas toujours la contrainte de borne (voir ci-dessous).

Remarque 2 : Il est également possible de normaliser les données en entrée pour (éventuellement) faciliter l'optimisation.

III.1.2.b. Choix de l'optimiseur

Pour maximiser la vraisemblance des paramètres, il convient d'utiliser un algorithme d'optimisation globale. Celui-ci doit pouvoir prendre en compte les bornes du domaine de définition de θ (des valeurs trop petites ou trop grandes détériorent le conditionnement de la matrice de covariance, d'où la nécessité d'imposer des limites). D'une implémentation à l'autre, le choix de cet algorithme varie. Le package DiceKriging [Roustant et al., 2012] en *R* utilise au choix l'algorithme L-BFGS-B (une méthode Quasi-Newton dans un domaine borné avec approximation de la matrice Hessienne) ou bien un algorithme génétique avec approximation du gradient de la fonction. La Toolbox Matlab *DACE* [Lophaven et al., 2002] utilise un algorithme de recherche directe de type *Pattern Search*. La Toolbox Python *Scikit-Learn* [Pedregosa et al., 2011] utilise l'algorithme COBYLA (Constrained Optimization BY Linear Approximation). [Toal et al., 2011] développe un algorithme hybride spécifiquement adapté à la résolution du problème, combinant un essaim de particules (*Particle Swarm*) avec l'algorithme d'optimisation locale SQP (Sequential Quadratic Programming).

Il est relativement difficile de déterminer lequel de ces algorithmes est le plus adapté, d'autant que d'un langage à l'autre et d'une implémentation à l'autre, les performances de calcul varient. L'algorithme *L-BFGS-B* de DiceKriging est généralement plus rapide que l'algorithme génétique, COBYLA a parfois des difficultés à respecter les contraintes de bornes du domaine ce qui peut poser problème dans le cas de données mal conditionnées.

En rapport avec les résultats présentés dans la Section II.1.3.b., j'ai choisi d'implémenter l'algorithme L-BFGS-B avec calcul analytique du gradient si cela est possible. Le code disponible permet également d'utiliser les algorithmes COBYLA et CMA-ES, à savoir que ce dernier a présenté des résultats médiocres par rapport aux deux autres en petite dimension ; il serait peut-être adapté en plus grande dimension.

Comme prévu, une accélération du calcul a été observée lors du choix de L-BFGS-B par rapport à COBYLA, grâce au calcul analytique du gradient. La méthode est stable dans le cas d'un noyau gaussien mais peut échouer si un noyau de type *Matérn* est sélectionné et que l'échantillonnage initial présente trop peu de points.

III.2. Construction des plans d'expériences

Les plans d'expériences font partie intégrante de la construction d'un modèle réduit. Pour obtenir un modèle de qualité, il faut nécessairement que le plan ait de bonnes propriétés en remplissage d'espace, en projection, etc.

Les Toolbox Python permettant la génération de LHD optimisés sont peu nombreuses et globalement inefficaces pour produire des plans d'expériences en un temps raisonnable. À titre d'exemple, le package PyDOE construit un nombre défini de plans et conserve celui qui possède le meilleur critère, les critères disponibles étant *maximin*(maximiser la distance minimale) et *corr* (minimiser la corrélation entre points du plan). Ceci est coûteux en temps et très peu performant. Le logiciel OpenMDAO utilise quant à lui un algorithme à stratégie évolutionnaire standard en optimisant le critère ϕ_p ce qui permet d'obtenir des plans de bonne qualité mais en un temps important.

Pour cette raison, un nouveau code Python a été créé pour construire des plans d'expériences selon l'algorithme ESE défini par [Jin et al., 2005] et décrit dans la Section II.2.2.. Il permet de construire des plans de qualité et s'adapte bien à la problématique des plans imbriqués avec la méthode proposée dans la Section II.2.3..

Dans cette section, nous illustrons le comportement de l'algorithme ESE et nous comparons les plans multifidélité avec l'approche décrite dans la Section II.2.3. et ceux décrits dans [Le Gratiet, 2013].

III.2.1. Comportement de l'algorithme ESE

L'algorithme ESE fait intervenir trois paramètres majeurs dans son déroulement : le critère d'arrêt *stop*, le nombre d'itérations de la boucle interne M et le nombre J de plans construits à chaque itération de la boucle interne. Dans notre cas, le critère d'arrêt correspond à un nombre d'itérations de la boucle externe. Il serait également possible de demander à l'algorithme de s'arrêter quand il ne parvient plus à améliorer la solution après un certain nombre d'itérations.

Pour permettre à l'algorithme d'obtenir un résultat satisfaisant quand la dimension et le nombre

de points du plan augmentent, les paramètres sont définis en fonction de la dimension du problème. On définit par défaut :

— $stop = \min(1.5d, 30)$
— $M = \min(20d, 100)$
— $J = 20$

La figure III.1 présente un plan optimisé avec ces paramètres ; la figure III.2 présente l'évolution du critère de minimisation ϕ_p ainsi que la température T de l'algorithme.

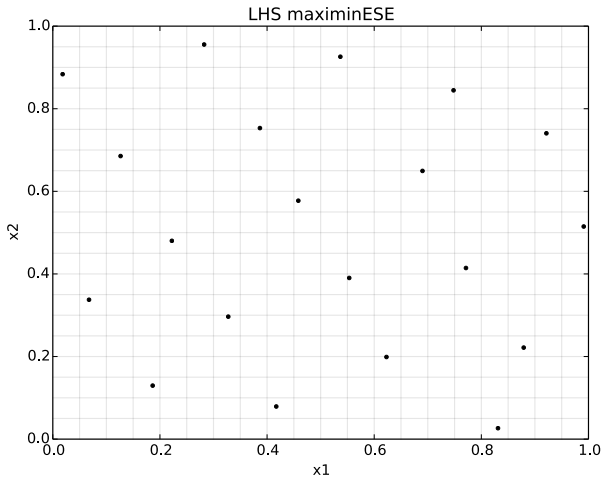
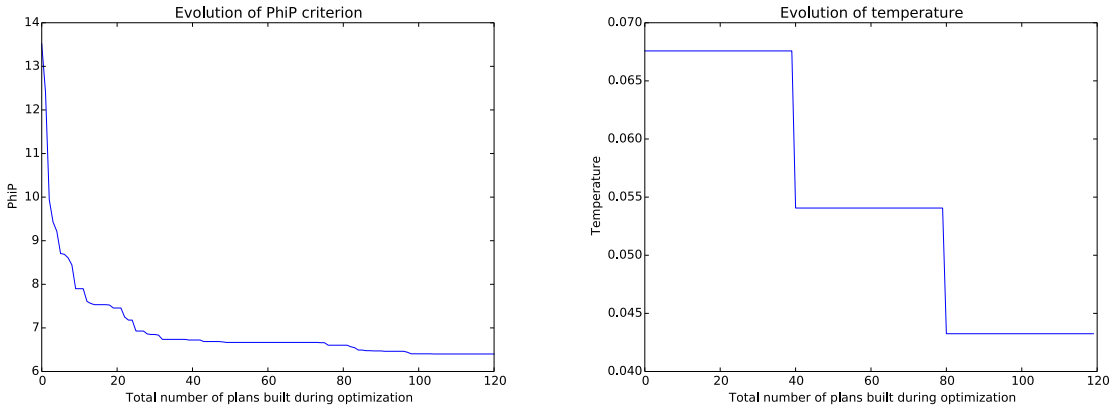


FIGURE III.1 – Plan d'expériences 2D de 20 points construit avec l'algorithme ESE



(a) (b)

FIGURE III.2 – Déroulement de l'algorithme ESE pour construire le plan III.1 avec 3 itérations de la boucle externe et 40 itérations de la boucle interne.(a) Évolution du critère ϕ_p. (b) Évolution de la température T.

Taille		ESE			pyDOE		
d	n	Temps (s)	D_{min}	ϕ_p	Temps (s)	D_{min}	ϕ_p
2	20	0.81	0.190	6.352	0.49	0.144	7.951
2	40	0.83	0.127	10.16	0.61	0.068	15.51
5	50	5.76	0.505	3.033	7.71	0.319	3.987
5	100	5.98	0.393	3.955	12.1	0.235	5.441
10	100	13.9	0.854	2.076	46.4	0.583	2.444
20	200	38.2	1.371	1.570	324.1	1.016	1.669
50	500	87.0	2.270	1.151	2051	1.962	1.172

TABLE III.1 – Comparaison des critères de qualité et des temps de calcul pour des plans construits avec l'algorithme ESE et avec une recherche aléatoire maximimisant le critère D_{min}, avec 10 réplications.

Pour avoir une idée des gains en qualité et en temps de calcul, le tableau III.2.1. compare le temps de calcul, la distance minimale D_{min} ainsi que le ϕ_p pour différents plans construits avec l'algorithme ESE et l'algorithme standard disponible dans pyDOE qui est une recherche aléatoire au terme de laquelle on conserve le plan maximisant D_{min}. La comparaison se fait pour un nombre total de plans construits égal, soit $J \times M \times stop$ avec les valeurs par défaut définies ci-dessus, pour 10 réplications à chaque fois.

On remarque tout d'abord dans ce tableau que les temps de calcul de la recherche aléatoire explosent complètement avec la montée en dimension, ce qui n'est pas le cas pour ESE. Ceci s'explique grâce à la rapidité de calcul du critère ϕ_p, notamment grâce aux raccourcis suggérés dans [Jin et al., 2005]. En plus de cela, les plans générés par ESE sont de bien meilleure qualité, à la fois en termes de distance minimale et de ϕ_p. La limitation imposée aux paramètres en grande dimension a un léger impact ; la convergence n'est pas tout à fait achevée.

III.2.2. Comparaison des plans multifidélité

La construction de plans multifidélité requiert une imbrication des plus hauts niveaux dans les plus bas. Deux approches ont été suggérées dans la Section II.2.3. pour construire cette imbrication : la recherche des plus proches voisins ainsi que le remplissage du LHD avec optimisation niveau par niveau par l'algorithme ESE.

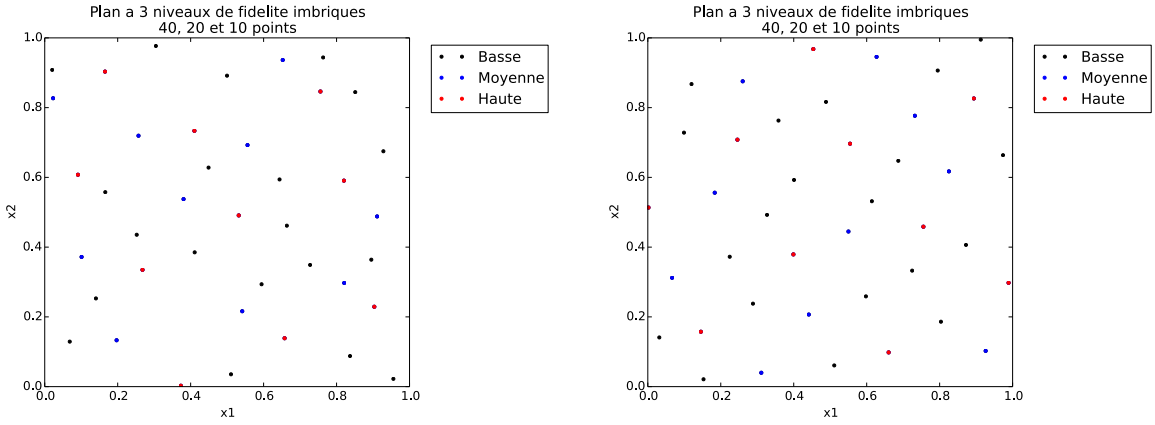
Nous comparons ici les deux approches pour différents plans d'expériences à trois niveaux en étudiant les critères de qualité des plans de chaque niveau et les temps de calcul requis pour chacun. La figure III.3 donne un aperçu de plans 2D construits avec les deux méthodes.

Paramètres		Méthode des plus proches voisins		
d	$n_1 - n_2 - n_3$	Temps (s)	D_{min} (1-2-3)	ϕ_p (1-2-3)
2	20 - 10 - 5	2.573	0.119 - 0.196 - 0.381	9.121 - 5.254 - 2.744
2	40 - 20 - 10	2.580	0.083 - 0.120 - 0.270	13.18 - 9.091 - 4.197
5	100 - 50 - 25	16.56	0.251 - 0.368 - 0.603	4.590 - 3.458 - 2.346
10	200 - 100 - 50	41.13	0.555 - 0.633 - 0.974	2.635 - 2.204 - 1.718
20	400 - 200 - 100	115.7	0.984 - 1.055 - 1.494	1.853 - 1.594 - 1.340
50	1000 - 500 - 250	299.34	2.053 - 2.105 - 2.359	1.327 - 1.152 - 0.995

TABLE III.2 – Critères de qualité et des temps de calcul pour des plans à trois niveaux de fidélité construits avec l'algorithme ESE par recherche des plus proches voisins, avec 10 réplications.

Paramètres		Méthode de remplissage des LHD		
d	$n_1 - n_2 - n_3$	Temps (s)	D_{min} (1-2-3)	ϕ_p (1-2-3)
2	20 - 10 - 5	2.595	0.144 - 0.205 - 0.367	7.678 - 5.255 - 2.873
2	40 - 20 - 10	2.603	0.103 - 0.138 - 0.277	11.46 - 7.696 - 4.112
5	100 - 50 - 25	16.48	0.395 - 0.453 - 0.584	3.987 - 3.129 - 2.346
10	200 - 100 - 50	41.24	0.763 - 0.844 - 0.969	2.495 - 2.083 - 1.721
20	400 - 200 - 100	115.9	1.216 - 1.352 - 1.488	1.830 - 1.570 - 1.339
50	1000 - 500 - 250	302.6	2.134 - 2.255 - 2.362	1.325 - 1.150 - 0.995

TABLE III.3 – Critères de qualité et des temps de calcul pour des plans à trois niveaux de fidélité construits avec l'algorithme ESE par remplissage et optimisation niveau par niveau, avec 10 réplications.



(a) (b)

FIGURE III.3 – Plan d'expériences à 3 niveaux imbriqués de 40, 20 et 10 points construit par la méthode des plus proches voisins (a) ou par remplissage niveau par niveau (b)

La méthode des plus proches voisins avait été choisie dans [Le Gratiet, 2013] pour sa rapidité. Nous constatons ici en comparant les tableaux III.2.2. et III.2.2. que le remplissage des plans niveau par niveau est aussi rapide, tout en étant plus performant. Les indicateurs sont meilleurs dans tous les cas, en particulier en petite dimension où ils s'approchent même la qualité d'un plan classique (voir tableau III.2.1.).

Nous rappelons cependant que la procédure n'est réalisable que s'il y a au moins k fois plus de points Basse Fidélité que de points Haute Fidélité (k entier).

III.3. Le co-krigeage sur des cas analytiques

Une fois les plans d'expériences construits, il est possible de construire et d'éprouver la robustesse des modèles. Cette section présente quelques cas tests pour vérifier la cohérence du comportement de l'estimation des paramètres du modèle de co-krigeage implémenté, de la validation croisée ainsi que d'une optimisation par l'algorithme EGO.

Remarque : En dépit de l'appellation "Co-krigeage avec formulation récursive", aucune fonction n'est définie récursivement dans l'implémentation.

III.3.1. Construction du modèle et estimation des paramètres

III.3.1.a. Paramètres de co-krigeage

Reprenons le cas d'école multifidélité défini dans [Forrester et al., 2007] en une dimension par un niveau haute fidélité

$$y_e(x) = (6x - 2)^2 \sin(12x - 4)$$

et un niveau basse fidélité

$$y_c(x) = \frac{1}{2} y_e(x) + 10(x - \frac{1}{2}) - 5$$

On utilise un noyau gaussien, une tendance de krigeage linéaire et une valeur constante pour le paramètre d'échelle ρ. La figure III.4 présente les résultats en utilisant le krigeage universel en bleu, le co-krigeage universel en vert. Le tableau III.3.1.a. présente les coefficients estimés pour le modèle de co-krigeage. Comme attendu, le modèle de co-krigeage est quasiment confondu avec la fonction objectif ; il fonctionne très bien puisque la différence entre les deux niveaux de codes a la forme d'une régression linéaire très simple à modéliser. Les coefficients β_ρ et β sont parfaitement estimés, tandis que σ^2 quasi-nulle au niveau 2 reflète une telle simplicité.

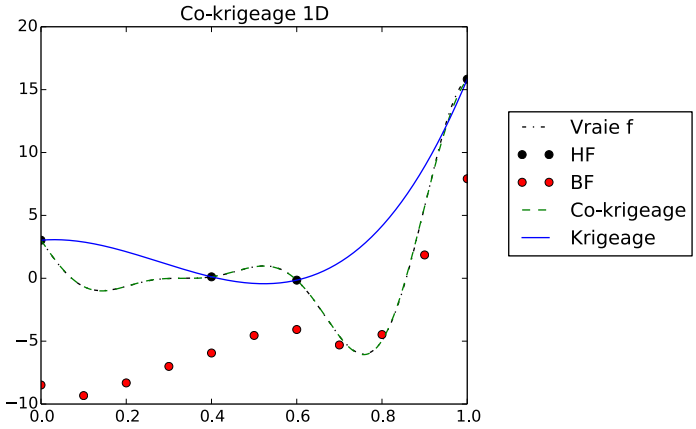


FIGURE III.4 – Comparaison entre un modèle de krigeage avec 4 points haute fidélité (HF) et un modèle de co-krigeage qui contient 10 points basse fidélité (BF) supplémentaires. La vraie fonction (en noir) est confondue avec le modèle de co-krigeage (en vert).

Coefficients	Co-krigeage	
	niveau 1	niveau 2
θ	18.26	0.249
β	-8.824 ; 11.134	20.00 ;-20.00
σ^2	21.8	4.2×10^{-27}
β_ρ	-	2

TABLE III.4 – Paramètres du modèle de krigeage présenté sur la figure III.4

III.3.1.b. Estimation du bruit

Nous avons vu dans la Section II.1.4.c. que le co-krigeage interpole les données de fidélité maximale. Étant donné qu'il se construit du plus bas niveau vers le plus haut, chaque niveau est amené à interpoler les données à un moment du calcul. Ceci peut être gênant si les données disponibles en entrée ne sont pas déterministes ; forcer l'interpolation peut notoirement détériorer la qualité du modèle.

L'approche multifidélité augmente le risque que certains niveaux de code aient des résultats avec une composante aléatoire (simulateur stochastique, bruit d'observation, incertitude de mesure, etc.). Pour cette raison, le co-krigeage régressant présenté dans la Section II.1.2.b. a été implémenté au sein du co-krigeage. La formulation récursive permet facilement d'adapter cette stratégie ; pour chaque niveau de code il est possible de choisir si un bruit λ doit être estimé ou non.

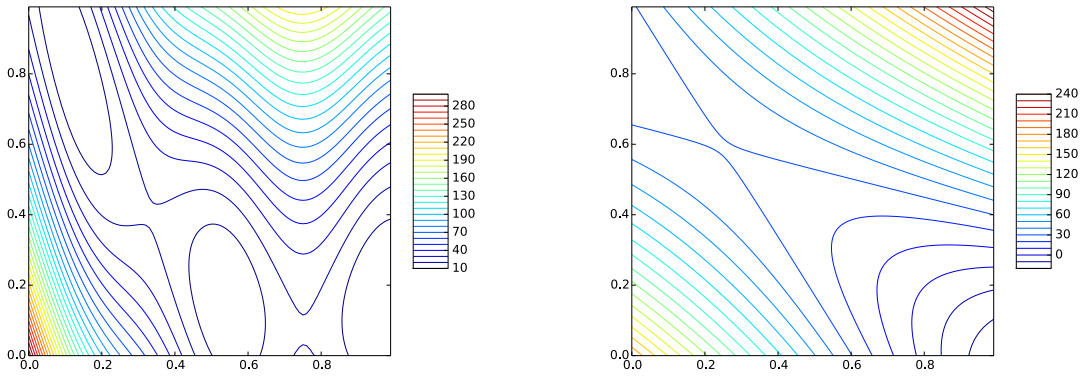
III.3.2. Comportement du co-krigeage

L'exemple décrit dans la Section III.3.1.a. ci-dessus représente le cas idéal pour construire un modèle de co-krigeage. Toutefois, le bénéfice de cette méthode n'est pas systématique. Une présentation a été faite aux membres de l'ONERA et de la SNECMA pour montrer qu'elle n'est bénéfique que dans le cas où la relation entre deux niveaux de fidélité est plus simple à modéliser que la fonction elle-même.

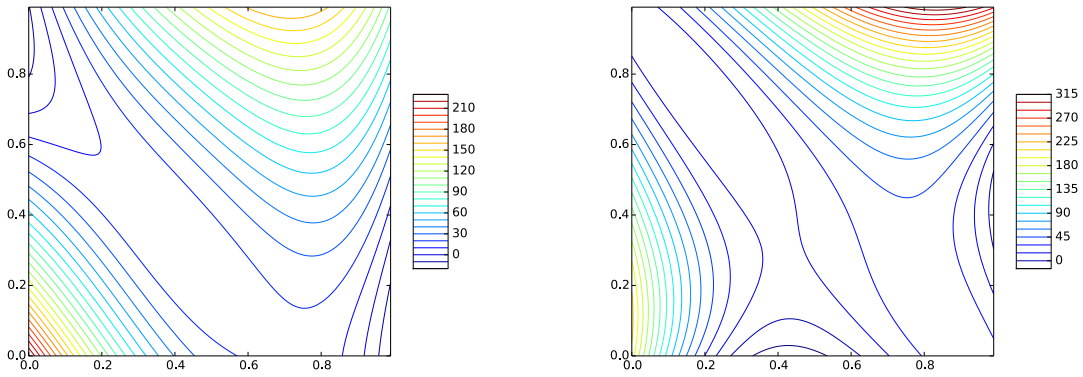
À titre d'exemple, utiliser un modèle de régression quadratique de la fonction *Branin* comme modèle Basse Fidélité n'apporte aucun bénéfice. La régression ne réduit en rien la complexité de la fonction, ce qui se ressent dans les prédictions (voir Figure III.5). L'apport de 40 points supplémentaires sur la figure III.5(d) n'apporte aucune amélioration puisque ces 40 points sont trop pauvres en information.

Supposons en revanche que la fonction Haute Fidélité requiert n_a points pour être modélisée correctement et que la relation entre les deux niveaux en requiert $n_b < n_a$. Alors en construisant X_1 (Basse Fidélité) et X_2 (Haute Fidélité) de tailles respectives n_a et $n_a - n_b$, on obtiendra un modèle quasiment aussi bon en prévision que si on utilisait un unique X de taille n_a.

La figure III.6 illustre le cas d'une fonction 2D f_e qui requiert environ 30 points pour être bien modélisée (avec un Q^2 supérieur à 0.95 en moyenne) par le krigeage ordinaire ; nous construisons une fonction "Basse Fidélité" $f_c = af_e + g$ où $a \in \mathbb{R}$, g est une fonction bien modélisée par 15 points



(a) (b)



(c) (d)

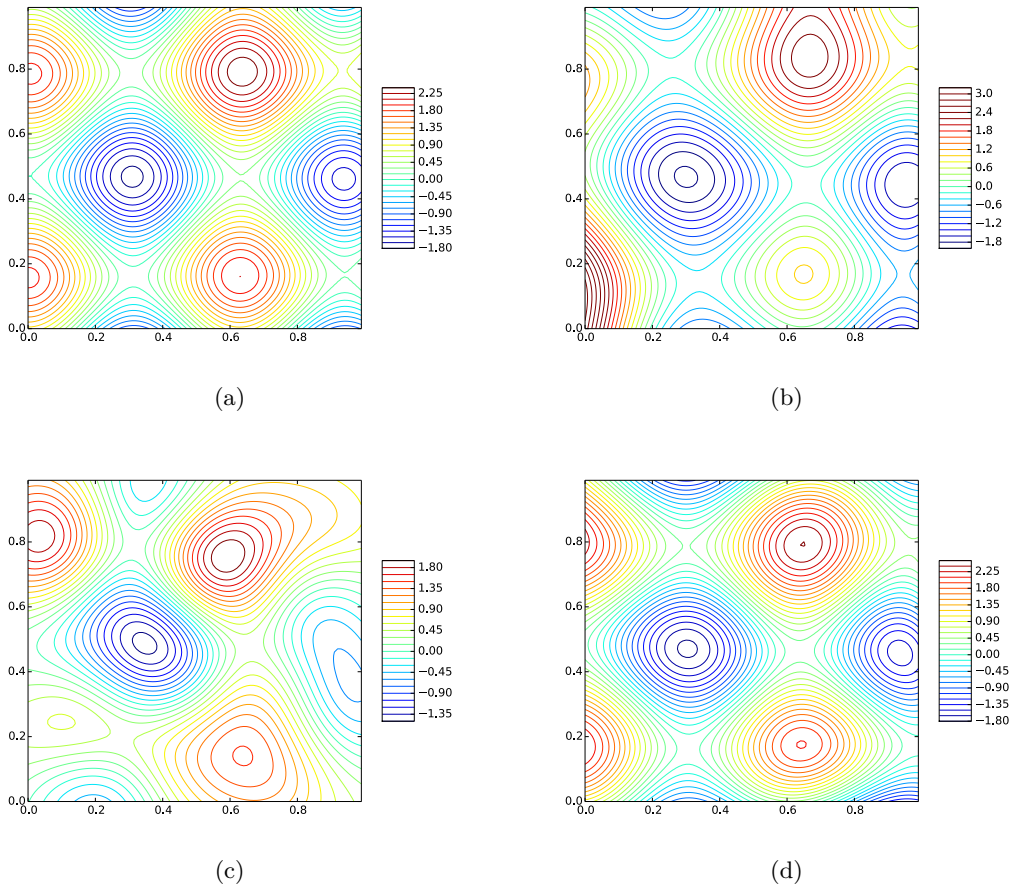
FIGURE III.5 – Allure de la fonction Branin (a), la régression polynômiale d'ordre 2 (b), un modèle de krigeage construit avec 10 points (c) et un modèle de co-krigeage construit avec 10 points et 40 points supplémentaire issus de la régression (d).

uniquement. La différence entre f_c et f_e est par conséquent plus simple à modéliser que f_e.

On peut remarquer que la fonction f_e (Figure III.6(a)) et le co-krigeage avec 15 points de f_e et 30 points de f_c (Figure III.6(d)) sont quasiment identiques. L'apport des 30 points Basse Fidélité est significatif (en comparaison avec le krigeage à 15 points sur la figure III.6(c)).

Avec ce résultat, nous montrons que le co-krigeage est efficace si le niveau Basse Fidélité permet d'appréhender au moins en partie la complexité du niveau Haute Fidélité. Dans le cas contraire, l'apport serait plus discutable.

Dans un contexte industriel, il n'est pas possible de savoir à l'avance si une telle réduction s'opère entre deux niveaux de fidélité. Cependant, cela permet d'avoir une idée des cas où il est très



(a) (b)

(c) (d)

FIGURE III.6 – Étude en prédiction de la fonction f_e représentée en (a) avec la fonction basse fidélité f_c correspondante (b), un modèle utilisant 15 appels à f_e (c) et un modèle de co-krigeage utilisant 15 appels à f_e et 30 appels à f_c (d)

probable que le co-krigeage n'apporte pas de bénéfice. Un cas concret est fourni dans la Section IV.1. où le co-krigeage est inopérant (les niveaux de fidélité étant des temps de convergence d'une méthode Monte-Carlo).

Remarque : Un exemple où le co-krigeage apporte des informations serait le cas d'un résultat Basse Fidélité acceptable dans une portion Q du domaine de définition uniquement, alors que sa version Haute Fidélité serait valable en tout point. Q serait au final mieux décrite avec l'approche multiniveau. En revanche, l'espace restant ne connaîtrait pas d'amélioration. Ce problème soulève la question de l'échantillonage des données pour différents niveaux ; dans certains cas on préférera par exemple avoir des données haute Fidélité au bord du domaine plutôt que bien réparties dans l'espace.

III.3.3. Procédure de validation croisée

La procédure de validation croisée décrite dans la Section II.1.5.b. est implémentée informatiquement. Cette procédure considère que les paramètres θ sont fixés et permet de conserver les paramètres β et σ^2 ou bien de les recalculer, au choix.

Pour s'assurer qu'elle fonctionne comme il faut, des tests sont réalisés pour calculer l'erreur et la variance obtenue par *Leave-One-Out*.

La comparaison se fait entre les deux constructions suivantes :

— "Vraie" validation croisée où le modèle est reconstruit entièrement en fixant θ (et éventuellement β et σ^2) à la valeur obtenue dans le modèle initial après avoir supprimé 1 point Haute Fidélité et son équivalent Basse Fidélité

— Validation croisée définie dans II.1.5.b. en recalculant (ou pas) les paramètres du modèles

La fonction utilisée pour effectuer la comparaison est la fonction *Branin* en deux dimensions. Le niveau Basse Fidélité correspond à la fonction soustraite d'une interaction linéaire. Le modèle de krigeage est construit avec une tendance (ou dérive) quadratique, pour repérer l'impact de la ré-estimation des paramètres.

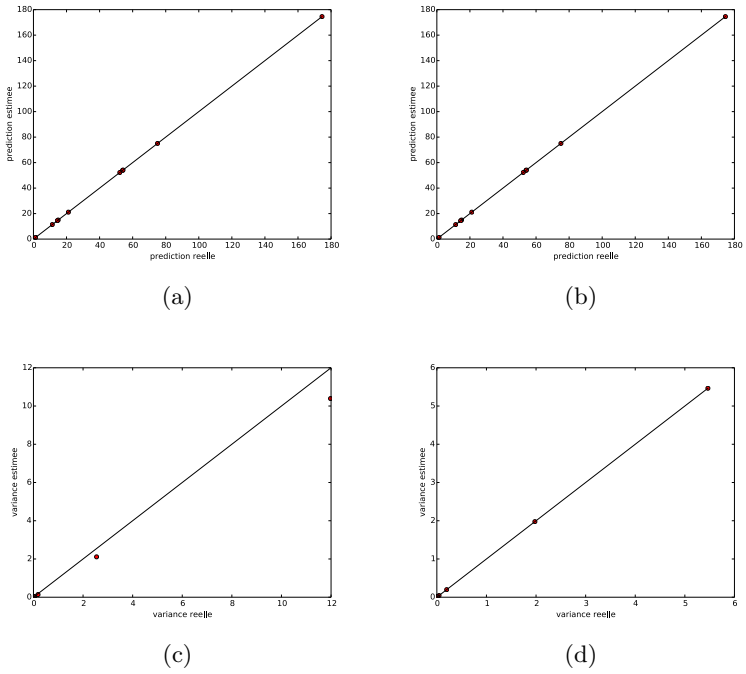
La figure III.7 présente les écarts entre les deux procédures, sans réestimation des paramètres (III.7(a) et III.7(c)) et avec réestimation des paramètres (III.7(b) et III.7(d)). La vérification de la robustesse des formules de validation croisée a permis de détecter la présence d'une erreur dans les équations décrites par [Le Gratiet, 2013]. Après correction, nous pouvons remarquer que les deux constructions sont identiques, hormis le léger écart sur la figure III.7(c) qui est dû au fait que l'on utilise $n-1$ dans le calcul de σ^2 (voir la formule (II.22)) au lieu de n.

Nous rappelons que cette procédure permet de construire une erreur de validation croisée de façon très rapide, mais n'est pas aussi représentative que si tous les paramètres (θ inclus) étaient réestimés.

III.4. Implémentation dans OpenMDAO

La plateforme OpenMDAO est en pleine évolution. Elle a subi une mise à jour conséquente cet été, restructurant l'intégralité des méthodes de construction de modèles réduits. Ceci a rendu plus délicate l'implémentation du code.

Rémi Lafage (ONERA) avait déjà construit un plugin pour intégrer le code fourni par Mohamed Bouhlel (SNECMA-ISAE) sur la plateforme. Étant donné que la structure des fonctions est très similaire, celui-ci s'est proposé de finaliser l'intégration du co-krigeage multifidélité après avoir pris connaissance du code et de son architecture.



(a) (b)

(c) (d)

FIGURE III.7 – Vérification de l'exactitude des calculs sur la prédiction et la variance de prédiction sans recalculer les paramètres ((a) et (c)) et en recalculant les paramètres ((b) et (d)) avec la procédure de validation croisée. En noir, la valeur attendue, calculée en reconstruisant le modèle. En rouge, les valeurs obtenues par la procédure.

Partie IV

Résultats

Le modèle de co-krigeage multi-fidélité avec formulation récursive a été implémenté en Python avec différents enrichissements. L'enjeu du stage a consisté ensuite à l'éprouver sur des cas réels pour montrer l'intérêt (ou non) de l'approche multifidélité pour construire des modèles réduits.

Trois cas sont présentés dans cette partie : dans un premier temps, nous étudions le comportement du modèle sur des données issues d'un simulateur de type Monte-Carlo fournies par Y. Richet, chercheur à l'Institut de Radioprotection et de Sécurité Nucléaire (IRSN) et professeur à l'EMSE. Nous faisons ensuite l'étude sur un jeu de données banalisé construit par la SNECMA, puis sur un jeu de données construit par l'ONERA.

IV.1. Comparaison à budget équivalent : données IRSN

Le problème présenté dans cette section consiste à évaluer la criticité nucléaire k_{eff} d'une centrale en fonction de deux paramètres, à savoir la densité de combustible et la densité de l'eau entre les tubes. Les données utilisées sont issues du simulateur MORET construit avec une technique de type Monte-Carlo. On dispose d'une grille de 75×75 points et pour chacun des points, l'historique de calcul est disponible pour 200 pas de temps différents (ce qui représente donc 200 niveaux de précision d'évaluation de la fonction et potentiellement 200 niveaux de fidélité).

Ce benchmark a initialement été conçu pour définir un critère d'enrichissement sensible au bruit d'estimation. Il s'agit en effet d'un simulateur stochastique où les données sont associées à une incertitude quantifiée. Il est intéressant de l'étudier selon une approche multifidélité : on dispose de nombreux niveaux et surtout, il est facile de comparer le krigeage et le co-krigeage pour un budget initial équivalent.

La figure IV.1 présente l'allure de la surface de réponse au niveau de fidélité maximal. On se base sur un budget égal à 20 points Haute Fidélité pour construire les différents modèles. Les plans d'expériences utilisés sont des LHD imbriqués, avec la règle des plus proches voisins pour pouvoir utiliser les données de la grille.

On peut remarquer sur les graphiques IV.2(a) et IV.2(b) que le modèle de co-krigeage est meilleur en moyenne que le modèle avec 20 points Haute Fidélité mais demeure moins bon que la prédiction avec 40 points Basse Fidélité.

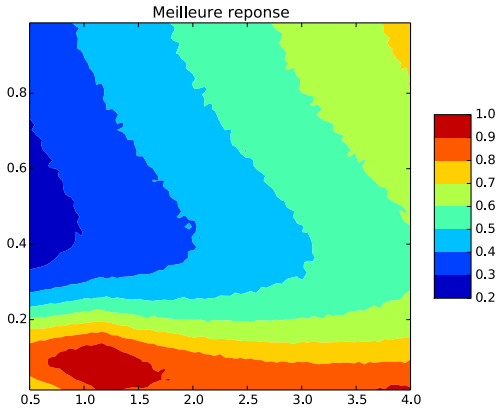
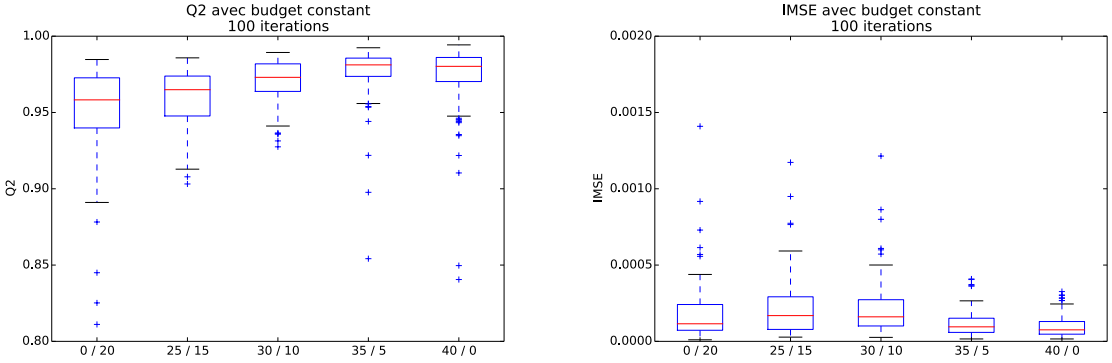


FIGURE IV.1 – Surface de réponse de k_{eff} construite au niveau de fidélité maximal.



(a) (b)

FIGURE IV.2 – Comparaison du Q^2 (a) et de l'$IMSE$ (b) entre modèles de krigeage avec différentes répartitions Basse/Haute fidélité pour un même budget total.

On peut en tirer deux conclusions :

— Le choix de la fidélité maximale n'est en fait pas pertinent. Bien que ces données soient plus précisément estimées, il est en fait intéressant de plus remplir l'espace, quitte à perdre légèrement sur la précision des données recueillies.

— Comme présenté dans la Section III.3., le co-krigeage fonctionne au mieux quand la différence entre les deux niveaux de fidélité est relativement simple. Dans le cas des données IRSN, cette différence peut être assimilée à un bruit, d'où la relative inefficacité de la méthode.

Dans ce cas de figure, il peut être plus intéressant de considérer un modèle de krigeage "régressant" avec différents niveaux de bruit comme présenté dans la Section II.1.4.a. puisque c'est bien de cela qu'il s'agit.

IV.2. Bilans

Le co-krigeage est une technique qui enrichit les modèles classiques par l'apport de plusieurs réponses corrélées pour modéliser la fonction objectif. Il peut par conséquent s'adapter à la problématique où différents niveaux de fidélité sont disponibles en entrées. La reformulation de façon récursive permet de construire le modèle pour un temps de calcul et un espace mémoire réduits. Cependant, le co-krigeage n'a un intérêt que si l'apport d'un niveau de fidélité inférieure permet d'assimiler en partie la complexité de la fonction objectif. Dans le cas présenté dans la Section IV.1., nous illustrons un cas où cette condition n'est pas vérifiée puisque deux niveaux différents diffèrent essentiellement par un bruit.

Nous montrons en revanche dans les cas industriels fournis par la SNECMA que l'apport d'un niveau Basse Fidélité supplémentaire améliore généralement de façon importante les qualités en prédiction du modèle, à la fois en terme d'erreur et de variance.

Attention cependant aux hypothèses de départ dans la construction du co-krigeage. La structure de covariance, la tendance de krigeage ainsi que les enrichissements proposés font varier les prédictions et l'optimisation de façon assez sensible. Le co-krigeage multifidélité est délicat à manipuler et chaque hypothèse peut fortement se répercuter sur chaque niveau puis sur la globalité. Pour cette raison, il est important de faire une analyse a priori sur le comportement des données récoltées. Une étude en optimisation devra être précédée par la construction et la validation de plusieurs modèles (par une procédure de validation croisée ou par comparaison des erreurs sur un plan test).

À noter que l'algorithme MSEGO, qui consiste à enrichir le plan d'expériences en utilisant plusieurs modèles de départ, se prête tout particulièrement à la situation que nous rencontrons ici. Il propose une approche simple et plus robuste que le cas où l'on effectuerait toute la chaîne d'optimisation en partant d'un unique modèle, au risque qu'il ne soit pas le meilleur possible. Ceci requiert davantage d'appels à la fonction objectif, lesquels peuvent être effectués en parallèle si le contexte le permet.

Conclusion

L'optimisation multidisciplinaire est une technique qui nécessite le couplage d'informations issues de milieux potentiellement très différents. Dans ce rapport, nous nous focalisons sur la problématique multifidélité : nous réalisons le couplage de plusieurs jeux de données aux précisions différentes en vue de construire un modèle réduit pour approximer au mieux une la fonction objectif.

Parmi les différents types de modèles réduits, le co-krigeage est sélectionné et implémenté en Python en utilisant les dernières avancées de la recherche pour améliorer la rapidité et la précision des calculs. Divers enrichissements sont proposés et testés pour traiter le cas de données bruitées ou encore le cas de la grande dimension. La formulation du co-krigeage utilisée requiert l'imbrication des plans d'expériences de chaque niveau de fidélité. Une nouvelle méthode simple de remplissage niveau par niveau est développée et permet de construire sous certaines hypothèses et à moindres frais des plans d'expériences imbriqués de qualité.

Nous illustrons ensuite le fonctionnement du co-krigeage sur des cas analytiques simples et présentons dans quelles situations il peut apporter un gain majeur ou bien n'apporter aucune information pour prédire la fonction objectif. Le constat est relativement simple : pour que le co-krigeage soit efficace, il faut que la relation entre deux niveaux de fidélité soit plus simple à modéliser que le niveau de fidélité supérieur.

Partant de ce constat, le modèle est éprouvé sur des cas industriels réels, à savoir :

— Un benchmark simplifié de données déterminant le coefficient de criticité nucléaire d'une centrale à l'aide un simulateur utilisant les méthodes Monte Carlo. Dans ce cas, la multifidélité est représentée par les niveaux de convergence Monte Carlo. Nous montrons que le co-krigeage n'est pas plus efficace qu'une méthode de krigeage classique avec estimation d'un bruit hétéroscédastique.

— Deux problèmes industriels à 5 dimensions fournis par la SNECMA et l'ONERA où deux niveaux de fidélité sont disponibles. Nous montrons que le co-krigeage apporte un bénéfice quasi-systématique et assez important. En réduisant à la fois l'erreur et la variance associée, l'approche multifidélité permet d'améliorer la précision et de réduire le nombre d'itérations nécessaires lors du processus d'optimisation. Nous montrons également que le choix des hypothèses de départ a un impact significatif sur le modèle : en changeant son noyau, le type de tendance (ou dérive) ou encore ses propriétés interpolantes, les résultats sont assez variables.

Expériences et compétences acquises

Dans le cadre de mon stage de fin d'études, j'ai eu l'occasion d'utiliser pleinement les compétences que j'ai acquises pendant les cours de l'option Mathématiques Appliquées. Initialement, je souhaitais limiter la dimension recherche de mon stage pour mettre à profit concrètement mes acquis. J'ai immédiatement compris que les problématiques réelles des industriels faisaient intervenir des techniques que je ne maîtrisais pas encore et qu'il serait nécessaire de m'ouvrir au monde de la recherche.

J'ai appris à établir une bibliographie complète et à exploiter au mieux les techniques existantes, dans une démarche que l'on pourrait qualifier d'éclectique (par extrapolation du sens réel de ce mot). Par dessus tout, j'ai pris goût à cette recherche et à ce perfectionnement pour accélérer et améliorer la construction de mes modèles. La liberté d'action dont j'ai disposé m'a permis de faire des choix et d'explorer les domaines qui me semblaient prometteurs, avec certes quelques ratés et un certain nombre de jours perdus mais qui m'ont conforté dans ma démarche.

Un autre point fort de mon stage a été l'apprentissage du langage Python. J'avais déjà eu l'occasion d'utiliser ce langage en deuxième année à l'École des Mines mais dans un contexte complètement différent, ce qui m'a permis de le redécouvrir. Parce qu'il s'agit d'un langage interprété, il faut maîtriser les techniques basiques de programmation mais également aller beaucoup plus loin si l'on veut obtenir des performances satisfaisantes. Par exemple, il ne faut pas hésiter à repenser le problème mathématique et transformer les formules en amont pour qu'elles s'adaptent aux méthodes précompilées disponibles dans les bibliothèques Python. Par ailleurs, la communauté grandissante et le développement de l'OpenSource combinés à un langage multi-plateforme me donnent le profond sentiment que cette compétence me sera très utile dans le futur.

L'aspect humain a eu une dimension importante dans mon stage. En travaillant dans un département majoritairement réservé à des problématiques différentes des miennes, j'ai ressenti au début une forme d'isolement dans mon travail. Ceci s'est rapidement estompé, notamment grâce à une ambiance de travail à la fois détendue et sérieuse à l'ISAE, ainsi qu'au suivi à distance avec l'ONERA et la SNECMA, lesquels ont suscité beaucoup d'intérêt pour l'avancement de mes travaux. Le ralentissement au cours de la période estivale s'est malgré tout fortement ressenti.

J'ai côtoyé de nombreux étudiants en thèse, ce qui m'a donné un aperçu postif envers une formation que je n'envisageais pas initialement mais qui aujourd'hui fait partie de mes réflexions sur mon futur emploi. Le stage que j'effectuerai à l'Université de Floride en Octobre m'éclairera probablement davantage sur cette question.

En résumé, ce stage a été très formateur pour moi. Il m'a permis de prendre goût à la recherche et a correspondu à mes attentes en termes de technicité, d'autonomie et d'orientation. Un regret, s'il en est, est sa durée légèrement trop courte qui ne m'a pas permis d'aboutir dans la démarche que je m'étais fixé initialement. J'ai cependant la conviction que les personnes qui m'ont encadré ont aujourd'hui une vision claire de mon travail, ce dont je suis pleinement satisfait.

Références

Da Veiga, S. and Marrel, A. (2012). Gaussian process modeling with inequality constraints. In *Annales de la faculté des sciences de Toulouse Mathématiques*, volume 21, pages 529–555. Université Paul Sabatier, Toulouse. 34

Dubrule, O. (1983). Two methods with different objectives : splines and kriging. *Journal of the International Association for Mathematical Geology*, 15(2) :245–257. 25

Forrester, A. I., Sóbester, A., and Keane, A. J. (2007). Multi-fidelity optimization via surrogate modelling. *Proceedings of the royal society A : mathematical, physical and engineering science*, 463(2088) :3251–3269. 45

Ginsbourger, D., Le Riche, R., et al. (2009). Towards gp-based optimization with finite time horizon. 32

Gray, J., Moore, K. T., and Naylor, B. A. (2010). Openmdao : an open source framework for multidisciplinary analysis and optimization. In *AIAA/ISSMO Multidisciplinary Analysis Optimization Conference Proceedings*. 10

Griewank, A. et al. (1989). On automatic differentiation. *Mathematical Programming : recent developments and applications*, 6 :83–107. 18

J. Forrester, A. I., Keane, A. J., and Bressloff, N. W. (2006). Design and analysis of "noisy" computer experiments. *AIAA journal*, 44(10) :2331–2339. 16, 33

Jin, R., Chen, W., and Sudjianto, A. (2005). An efficient algorithm for constructing optimal design of computer experiments. *Journal of Statistical Planning and Inference*, 134(1) :268–287. 28, 29, 40, 42

Jones, D. R. (2001). A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, 21(4) :345–383. 31

Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4) :455–492. 32

Kennedy, M. C. and O'Hagan, A. (2000). Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1) :1–13. 21, 22, 23

Le Gratiet, L. (2013). *Multi-fidelity Gaussian process regression for computer experiments*. PhD thesis, Université Paris-Diderot-Paris VII. 2, 22, 23, 25, 31, 35, 40, 44, 49

Le Gratiet, L., VII-CEA, A. P., DAM, D., Le Gratiet, M. L., and DiceKriging, D. (2012). Package 'muficokriging'. 36

Liu, W. (2003). *Development of gradient-enhanced kriging approximations for multidisciplinary design optimization*. PhD thesis. 15

Lophaven, S. N., Nielsen, H. B., and Søndergaard, J. (2002). Dace-a matlab kriging toolbox, version 2.0. Technical report. 39

Matheron, G. (1962). *Traité de géostatistique appliquée. 1 (1962)*, volume 1. Editions Technip. 13

Morris, M. D. and Mitchell, T. J. (1995). Exploratory designs for computational experiments. *Journal of statistical planning and inference*, 43(3) :381–402. 28

Patterson, H. D. and Thompson, R. (1971). Recovery of inter-block information when block sizes are unequal. *Biometrika*, 58(3) :545–554. 17

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn : Machine learning in python. *The Journal of Machine Learning Research*, 12 :2825–2830. 36, 39

Qian, P. Z. and Wu, C. J. (2008). Bayesian hierarchical modeling for integrating low-accuracy and high-accuracy experiments. *Technometrics*, 50(2) :192–204. 22

Rennen, G., Husslage, B., Van Dam, E. R., and Den Hertog, D. (2010). Nested maximin latin hypercube designs. *Structural and Multidisciplinary Optimization*, 41(3) :371–395. 30

Roustant, O., Ginsbourger, D., Deville, Y., et al. (2012). DiceKriging, DiceOptim : Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization. 39

Sasena, M. J. (2002). *Flexibility and efficiency enhancements for constrained global design optimization with kriging approximations*. PhD thesis, University of Michigan. 32, 34

Sasena, M. J., Papalambros, P., and Goovaerts, P. (2002). Exploration of metamodeling sampling criteria for constrained global optimization. *Engineering optimization*, 34(3) :263–278. 34

Tikhonov, A. (1963). Solution of incorrectly formulated problems and the regularization method. In *Soviet Math. Dokl.*, volume 5, pages 1035–1038. 16

Toal, D. J., Bressloff, N., Keane, A., and Holden, C. (2011). The development of a hybridized particle swarm for kriging hyperparameter tuning. *Engineering optimization*, 43(6) :675–699. 18, 39

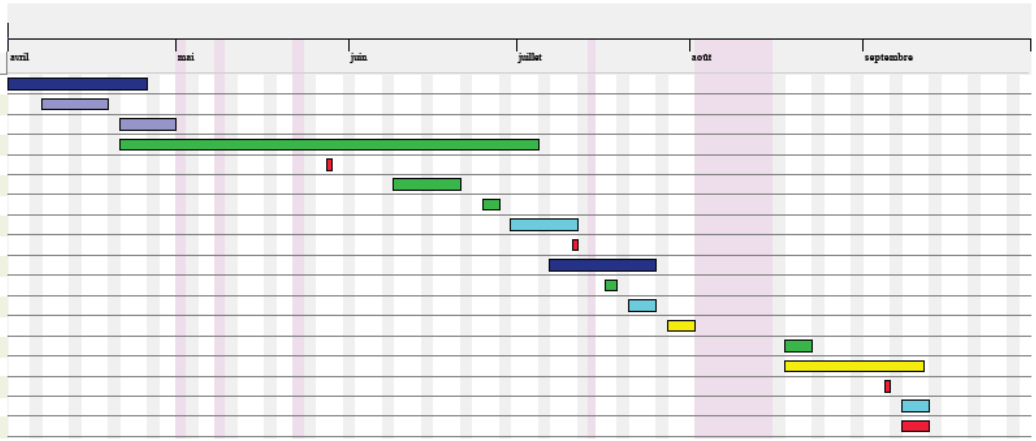
Toal, D. J., Forrester, A. I., Bressloff, N. W., Keane, A. J., and Holden, C. (2009). An adjoint for likelihood maximization. *Proceedings of the Royal Society A : Mathematical, Physical and Engineering Science*, page rspa20090096. 18

Vauclin, M., Vieira, S., Vachaud, G., and Nielsen, D. (1983). The use of cokriging with limited field soil observations. *Soil Science Society of America Journal*, 47(2) :175–184. 21

Viana, F. A., Haftka, R. T., and Watson, L. T. (2013). Efficient global optimization algorithm assisted by multiple surrogate techniques. *Journal of Global Optimization*, 56(2) :669–689. 32

Annexe A - Planning

Nom	Date de début	Date de fin
Étude bibliographique sur les modèles multifidélité	01/04/14	25/04/14
Étude et essais sur les codes Matlab de co-krigeage	07/04/14	18/04/14
Apprentissage du langage Python et tests sur le KPLS	21/04/14	30/04/14
Développement du co-krigeage multifidélité	21/04/14	04/07/14
Présentation de l'étude comportementale du co-krigeage	28/05/14	28/05/14
Développement de la procédure de validation croisée	09/06/14	20/06/14
Développement de l'algorithme ESE	25/06/14	27/06/14
Étude des données ONERA et SNECMA	30/06/14	11/07/14
Présentation des résultats su le co-krigeage	11/07/14	11/07/14
Étude bibliographique sur les méthodes d'optimisation	07/07/14	25/07/14
Développement de l'algorithme EGO	17/07/14	18/07/14
Étude et adaptation des codes SNECMA pour SuperEGO	21/07/14	25/07/14
Apprentissage de LATEX	28/07/14	01/08/14
Implémentation de l'optimiseur L-BFGS-B	18/08/14	22/08/14
Rédaction du rapport	18/08/14	11/09/14
Livraison des codes finaux	05/09/14	05/09/14
Réalisation de la chaîne d'optimisation ONERA	08/09/14	12/09/14
Présentations finales	08/09/14	12/09/14



Annexe B - Construction de noyaux de covariance

Formulation de noyaux

Un essai a été développé pour construire de nouveaux noyaux en se basant sur la classe de noyaux *Matérn*.

Plutôt que d'effectuer un produit tensoriel classique, on construit un équivalent des noyaux $Matérn\frac{3}{2}$ et $Matérn\frac{5}{2}$ de la façon suivante :

$$k(x, \tilde{x}) = \sigma^2 \left(1 + \sqrt{3}X\right) e^{-\sqrt{3}X}$$
$$k(x, \tilde{x}) = \sigma^2 \left(1 + \sqrt{5}X + \frac{5}{3}X^2\right) e^{-\sqrt{5}X}$$

avec

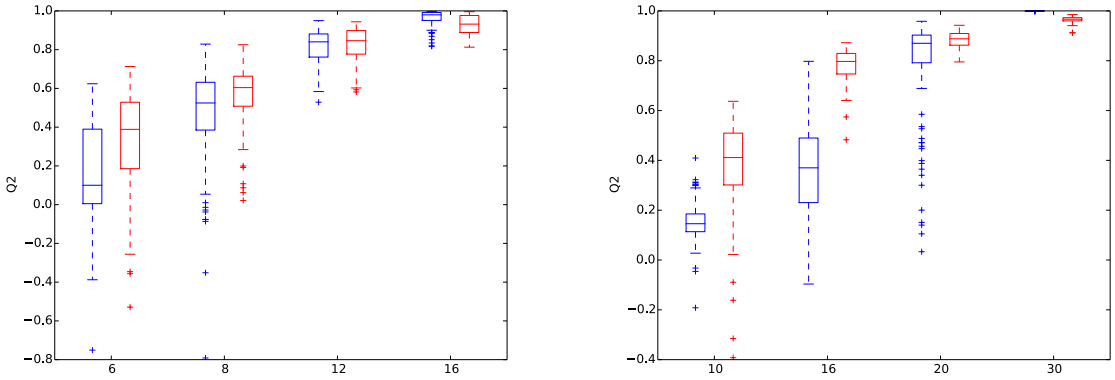
$$X = \sqrt{\sum_{i=1}^{d} 10^{\theta_i} |x_i - \tilde{x}_i|^2}$$

ce qui correspond à une sorte de norme euclidienne pondérée par les paramètres de portée.

Ces noyaux sont définis positifs comme composées d'une fonction à valeurs dans $\mathbb{R}^+$ avec les noyaux *Matérn* à une dimension correspondants, eux-mêmes définis positifs.

Les résultats obtenus en prédiction sur quelques cas simples sont meilleurs que ceux obtenus avec les noyaux construits par produits tensoriels lorsque l'on dispose de peu de points en entrée. Si le nombre de points est plus important en revanche, les prédictions s'équilibrent en faveur des noyaux construits par produit tensoriel.

Les figures ci-après donnent un exemple pour la fonction Branin en faisant varier le nombre de points d'observation disponibles.



(a) (b)

FIGURE IV.3 – Comparaison du Q^2 obtenu pour les fonctions Branin (a) et cosin2 (b) en utilisant un noyau $Matérn\frac{5}{2}$ classique (en bleu) et en utilisant le noyau présenté ci-dessus (en rouge).