Optimisation auto-adaptative en environnement d'analyse multidisciplinaire via les modèles de krigeage combinés à la méthode PLS

Mohamed Amine Bouhlel

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Institut Supérieur de l'Aéronautique et de l'Espace (ISAE)*

Soutenue le *26/01/2016* **par :**

Mohamed Amine BOUHLEL

Optimisation auto-adaptative en environnement d'analyse multidisciplinaire via les modèles de krigeage combinés à la méthode PLS

JURY

Marcel MONGEAU	**Professeur ENAC**	**Président du jury**
Joseph MORLIER	**Professeur ISAE**	**Directeur de thèse**
Nathalie BARTOLI	**Ingénieur de recherche ONERA**	**Co-directrice de thèse**
Nicolas GAYTON	**Maitre de conférence (HDR) IFMA**	**Rapporteur**
Rodolphe LE RICHE	**Directeur de recherche CNRS**	**Rapporteur**
Joaquim R. R. A. MARTINS	**Professeur Université du Michigan**	**Examinateur**
Abdelkader OTSMANE	**Ingénieur Snecma**	**Invité**
Fabrice GAMBOA	**Professeur Université IMT**	**Invité**

École doctorale et spécialité :

AA : Aéronautique, Astronautique, Mathématiques Appliquées

Unité de Recherche :

Office National d'Étude et de Recherches Aérospatiales

Directeur(s) de Thèse :

Joseph MORLIER

Rapporteurs :

Nicolas GAYTON et *Rodolphe LE RICHE*

Table des matières

Remerciements

Cette thèse n'est pas le résultat d'un travail solitaire et je voudrais remercier ici toutes les personnes qui y ont participé. Mes premiers remerciements iront à Nathalie Bartoli qui a encadré ce travail. Son investissement et son implication ont contribué en grande partie à la réussite de ce projet. Elle a toujours été disponible et patiente, elle m'a soutenu durant tout le projet et j'apprécie particulièrement son respect des délais (parfois serrés) lors des différentes corrections. J'ai aussi apprécié ses qualités humaines, notamment l'écoute et la compréhension. Cette thèse a été une excellente expérience pour moi, et cela en grande partie grâce à la qualité de son encadrement et à nos discussions fructueuses.

Mes sincères remerciements à Joseph Morlier qui m'a dirigé tout au long de ces trois ans de thèse. Je le remercie pour son accueil chaleureux à chaque fois que j'ai sollicité son aide, ses conseils divers, ses encouragements durant ces trois ans ainsi que son soutien sur les plans scientifique et humain. Une grande part de la réussite de cette thèse lui revient également.

Je remercie également mon encadrant industriel Abdelkader Otsmane d'avoir facilité mon intégration à Snecma. Les nombreuses discussions que nous avons eues sont pour beaucoup dans le résultat final de ce travail. Ce fut un réel plaisir d'échanger avec lui durant les réunions, en particulier, sur l'interprétation physique des résultats.

J'adresse ma reconnaissance à Marcel Mongeau de m'avoir fait l'honneur de présider mon jury de thèse et d'apporter des corrections à ce manuscrit.

Je remercie Nicolas Gayton et Rodolphe Le Riche d'avoir accepté de relire cette thèse et d'en être rapporteurs. Je les remercie pour leur lecture attentive et pour leurs remarques précieuses.

Mes remerciements s'adressent également à Joaquim R. R. A. Martins et Fabrice Gamboa d'avoir accepté d'examiner mes travaux de thèse et pour leurs questions très enrichissantes au cours de la soutenance.

Dans le cadre de cette thèse, j'ai côtoyé de nombreuses personnes du DCPS qui ont contribué à rendre ces trois années très sympathiques et que je tiens à les remercier notamment pour leur bonne humeur. Je pense ainsi à Peter S., à qui je souhaite beaucoup de courage pour la suite de sa thèse, Thierry L., pour ses nombreux conseils, Rémi L., notre expert informatique, Claire D., pour son efficacité dans les démarches administratives, Sébastien A., notre chef d'unité, Jérôme M., Luis B., Thomas C., Thomas D., Sylvain D., Maude D., Antoine J., Claire S., Franck M. et Armand O.. Je remercie également Rémi V. et Rémy P. qui ont réalisé un sérieux travail pendant leur stage sur les méthodes

développées au cours de cette thèse.
Je souhaite également rendre hommage à l'ensemble de l'équipe méthodes de Snecma Villaroche. Je voudrais ainsi remercier Virgile M., pour ses nombreux services rendus et ses conseils, Jean C., pour ses explications du fonctionnement du cas test final et mon ami Javier L..
J'adresse mes remerciements à mes amis de Toulouse, Ramzi (mon guide et mon soutien), Asma (bent bladi, la meilleure qui cuisine les tajines) et Ahmed (ne t'inquiète pas, je vais te recruter dès que je monte mon équipe de foot) et je leur souhaite un bon avenir après leur thèse. Je remercie également Karim papi (le Cardinal me manque déjà), Zakaria (pour les soirées ligue des champions) et Najib (pour ses avis philosophiques).
Un grand merci à mes amis de Nice avec qui je partage toujours des bons moments malgré la distance qui nous sépare. Je pense particulièrement à Moucha (qui a toujours répondu présent à chaque fois que j'ai sollicité son aide), Anis (pour tous les moments de fraternité passés ensemble et les nombreux services qui m'a rendu), Marouane (pour les soirées à Pinocchio), Adel (et sa petite princesse Salma), Deb (pour son couscous irrésistible) et Bolbol (pour les pauses-café à la bibliothèque de Saint Jean).
Ces dernières lignes s'adressent à toute ma famille et mes amis qui vivent en Tunisie, à qui je dédie cette thèse. Merci à mon père et à ma mère pour tout ce qu'ils m'ont offert, et pour m'avoir encouragé à suivre ma voie. Merci à mon grand frère Mohamed-Taha et mon frère Saif. Merci également à ma sœur Rim pour son soutien et ses encouragements. Aussi, un grand merci à ma tante Chadlia qui est vraiment une "deuxième maman" et qui m'a hébergé pendant une bonne période durant mon séjour à Paris. Je remercie également mes cousins de Paris Émir et Nour pour les bons moments passés en jouant au foot et aux jeux de société. J'ai également une pensée pour Bchayra, Mostapha, Sadok, Abdelaziz, Cherifa, Olfa, Khadija (que son âme repose en paix), Rkaya, Rawda, Sami, Lalla, Sidi, Ahmed, Aymen, Yosri, Abdelsalem, Taoufik, Abed, Libi, Majdi, Ramzi et les frères Krifa. Enfin, une pensée à ma fiancé Raja qui m'a beaucoup soutenu, qui était patiente avec moi et à qui je dédie cette thèse également.

Toulouse, Janvier 2016 M.A. B.

Abstract

Aerospace turbomachinery consists of a plurality of blades. Their main function is to transfer energy between the air and the rotor. The bladed disks of the compressor are particularly important because they must satisfy both the requirements of aerodynamic performance and mechanical resistance. Mechanical and aerodynamic optimization of blades consists in searching for a set of parameterized aerodynamic shape that ensures the best compromise solution between a set of constraints.

This PhD introduces a surrogate-based optimization method well adapted to high-dimensional problems. This kind of high-dimensional problem is very similar to the Snecma's problems. Our main contributions can be divided into two parts : Kriging models development and enhancement of an existing optimization method to handle high-dimensional problems under a large number of constraints.

Concerning Kriging models, we propose a new formulation of covariance kernel which is able to reduce the number of hyper-parameters in order to accelerate the construction of the metamodel. One of the known limitations of Kriging models is about the estimation of its hyper-parameters. This estimation becomes more and more difficult when the number of dimension increases. In particular, the initial design of experiments (for surrogate modelling construction) requires an important number of points and therefore the inversion of the covariance matrix becomes time consuming.

Our approach consists in reducing the number of parameters to estimate using the Partial Least Squares regression method (PLS). This method provides information about the linear relationship between input and output variables. This information is integrated into the Kriging model kernel while maintaining the symmetry and the positivity properties of the kernels. Thanks to this approach, the construction of these new models called KPLS is very fast because of the low number of new parameters to estimate. When the covariance kernel used is of an exponential type, the KPLS method can be used to initialize parameters of classical Kriging models, to accelerate the convergence of the estimation of parameters. The final method, called KPLS+K, allows to improve the accuracy of the model for multimodal functions.

The second main contribution of this PhD is to develop a global optimization method to tackle high-dimensional problems under a large number of constraint functions thanks to KPLS or KPLS+K method. Indeed, we extended the self adaptive optimization method called "Efficient Global Optimization, EGO" for high-dimensional problems under constraints. Several enriching criteria have been

tested. This method allows to estimate known global optima on academic problems up to 50 input variables.
The proposed method is tested on two industrial cases, the first one, "MOPTA", from the automotive industry (with 124 input variables and 68 constraint functions) and the second one is a turbine blade from Snecma company (with 50 input variables and 31 constraint functions). The results show the effectiveness of the method to handle industrial problems. We also highlight some important limitations.

Key words : Optimization, Kriging, PLS, Turbomachinery blades.

Résumé

Les turbomachines aéronautiques sont composées de plusieurs roues aubagées dont la fonction est de transférer l'énergie de l'air au rotor. Les roues aubagées des modules compresseur et turbine sont des pièces particulièrement sensibles car elles doivent répondre à des impératifs de performance aérodynamique, de tenue mécanique, de tenue thermique et de performance acoustique. L'optimisation aéro-méca-acoustique ou aéro-thermo-mécanique des aubages consiste à chercher, pour un ensemble de formes aérodynamiques paramétrées (par plusieurs dizaines de variables), celle assurant le meilleur compromis entre la performance aérodynamique du moteur et la satisfaction de plusieurs dizaines de contraintes souvent contradictoires.

Cette thèse introduit une méthode d'optimisation basée sur les métamodèles et adaptée à la grande dimension pour répondre à la problématique industrielle des aubages. Les contributions de cette thèse portent sur deux aspects : le développement de modèles de krigeage, et l'adaptation d'une stratégie d'optimisation pour la gestion du grand nombre de variables et de contraintes.

La première partie de ce travail traite des modèles de krigeage. Nous avons proposé une nouvelle formulation du noyau de covariance permettant de réduire le nombre de paramètres du modèle afin d'accélérer sa construction. Une des limitations connues du modèle de krigeage concerne l'estimation de ses paramètres. Cette estimation devient de plus en plus difficile lorsque nous augmentons la dimension du phénomène à approcher. En particulier, la base de données nécessite davantage de points et par conséquent la matrice de covariance du modèle du krigeage est de plus en plus coûteuse à inverser.

Notre approche consiste à réduire le nombre de paramètres à estimer en utilisant la méthode de régression des moindres carrés partiels (PLS pour Partial Least Squares). Cette méthode de réduction dimensionnelle fournit des informations sur la relation linéaire entre les variables d'entrée et la variable de sortie. Ces informations ont été intégrées dans les noyaux du modèle de krigeage tout en conservant les propriétés de symétrie et de positivité des noyaux. Grâce à cette approche, la construction de ces nouveaux modèles appelés KPLS est très rapide étant donné le faible nombre de paramètres nécessaires à estimer. La validation de ces modèles KPLS sur des cas test académiques ou industriels a démontré leur qualité de prédiction équivalente voire même meilleure que celle des modèles de krigeage classiques. Dans le cas de noyaux de covariance de type exponentiel, la méthode KPLS peut être utilisée pour initialiser les paramètres du krigeage classique, afin d'accélérer la convergence de l'estimation des paramètres du modèle. La méthode résultante, notée KPLS+K, a

permis d'améliorer la qualité des modèles dans le cas de fonctions fortement multimodales.
La deuxième contribution de la thèse a consisté à développer une stratégie d'optimisation globale sous contraintes pour la grande dimension, en s'appuyant sur les modèles KPLS ou les modèles KPLS+K. En effet, nous avons étendu la méthode d'optimisation auto-adaptative connue dans la littérature sous le nom "Efficient Global Optimisation, EGO" pour gérer les problèmes d'optimisation sous contraintes en grande dimension. Différents critères d'enrichissement adaptatifs ont pu être explorés. Cette stratégie a permis de retrouver l'optimum global sur des problèmes académiques jusqu'à la dimension 50.
La méthode proposée a été confrontée à deux types de problèmes industriels, le cas test MOPTA issu de l'industrie automobile (124 variables d'entrée et 68 fonctions contraintes) et le cas test Snecma des aubes de turbomachines (50 variables d'entrée et 31 fonctions contraintes). Les résultats ont permis de montrer la validité de la démarche ainsi que les limites de la méthode pour une application dans un cadre industriel.

Mots clefs : Optimisation, Krigeage, PLS, Aubes de turbomachines.

Symboles et notations

Les matrices sont en majuscules et les vecteurs sont en minuscules et ils sont tous écrits en gras, contrairement aux scalaires, sauf indication contraire.

symboles	signification
i.e.	c'est à dire
s.c.	sous les contraintes
Cf.	se reporter à
$a \bmod b$	l'opérateur modulo pour calculer le reste de la division euclidienne $\frac{a}{b}$
ln	le logarithme népérien
$\mathbb{R}$	l'ensemble des réels
$\mathbb{N}$	l'ensemble des entiers
$\mathbb{N}^*$	l'ensemble des entiers non nuls
d	dimension du problème de modélisation
h	nombre de composantes principales retenues
B	un hypercube dans $\mathbb{R}^d$
$\mathbb{E}$	Espérance de probabilité
$\sim$	distribuer selon une loi
$\approx$	à peu près égal
$\mathcal{N}$	une loi normale
X	une variable aléatoire
$\mathbf{x}$	un vecteur colonne de dimension d
$\mathbf{x}^t$	transposé de $\mathbf{x}$
$\mathbf{x}_i$	la $i^{ème}$ coordonnée de $\mathbf{x}$
$\mathbf{x}^{(i)}$	la $i^{ème}$ expérience (ou point)
n	le nombre d'expériences réalisées
$\mathbf{X}$	la matrice de taille $(n \times d)$ contenant l'ensemble des expériences réalisées ; la $i^{ème}$ ligne contient $\mathbf{x}^{(i)^t}$
$y(.)$	la fonction scalaire à modéliser
y	la vraie valeur en $\mathbf{x}$
$y^{(i)}$	la vraie valeur en $\mathbf{x}^{(i)}$

symboles	signification
$\mathbf{y}$	vecteur colonne contenant les sorties de $\mathbf{X}$ par $y(.)$, $\mathbf{y} = y(\mathbf{X})$
cov(.,.)	la fonction de covariance
cor(.,.)	la fonction de corrélation
$k(.,.)$	le noyau de covariance
$\odot$	produit matriciel élément par élément
$\mathbf{1}$	le vecteur $[1 \cdots 1]^t$ de dimension d
δ_{qp}	symbole de Kronecker, $\delta_{qp} = 1$ si $p = q$ et 0 sinon
$\mathcal{O}()$	grand O ; soient f et g définies sur $\mathbb{R}$, on a $f(x) = \mathcal{O}\big(g(x)\big)$ lorsqu'il existe deux constantes N et C telles que $\forall x > N$, $\lvert f(x)\rvert \leq C\lvert g(x)\rvert$
$\lvert\lvert \cdot \rvert\rvert_2$	la norme euclidienne

I Introduction

Cadre de la thèse

L'École Doctorale Aéronautique et Astronautique (EDAA) de l'Institut Supérieur de l'Aéronautique et de l'Espace (ISAE) a accueilli ma thèse en convention CIFRE entre la Snecma, l'ONERA et l'ISAE. Elle est dirigée par le professeur Joseph Morlier de l'ISAE-SUPAERO et de l'institut Clément Ader (ICA) -FRE CNRS 3687-. Le travail de recherche est effectué au sein de l'Office Nationale d'Étude de Recherches Aérospatiale (ONERA) de Toulouse, sous la direction de Nathalie Bartoli, ingénieur de recherche de l'unité Systèmes Aéronautiques (SAE) du Département Conception et évaluation des Performances de Systèmes (DCPS). Au cours des trois ans de thèse, j'ai passé un an et demi en environnement industriel, à Snecma Villaroche au sein du département Méthodes et Outils de Développement dans l'unité lien CAO-calcul et pré/post-traitement mécanique et optimisation, sous la direction de l'adjoint métier Abdelkader Otsmane.

Dans ce rapport, certains détails d'ordre industriel (paramètres, ...), notamment au niveau du cas test décrit dans la section VI.2., ont été omis pour des raisons de confidentialité.

Contexte scientifique et industriel de la thèse

Contexte scientifique

La conception optimale de systèmes complexes (avions, fusées, satellites, turbomachines, etc...) est un sujet d'actualité de première importance tant d'un point de vue recherche que d'un point de vue industriel. Trouver le meilleur compromis entre plusieurs disciplines sachant qu'il faut répondre à diverses exigences de qualité, de sécurité et de coûts, est un enjeux majeur notamment dans le domaine aérospatial. La réussite de tels projets requiert la maîtrise de chacune des disciplines impliquées avec une facilité d'échanges entre les équipes des différentes disciplines. Ce

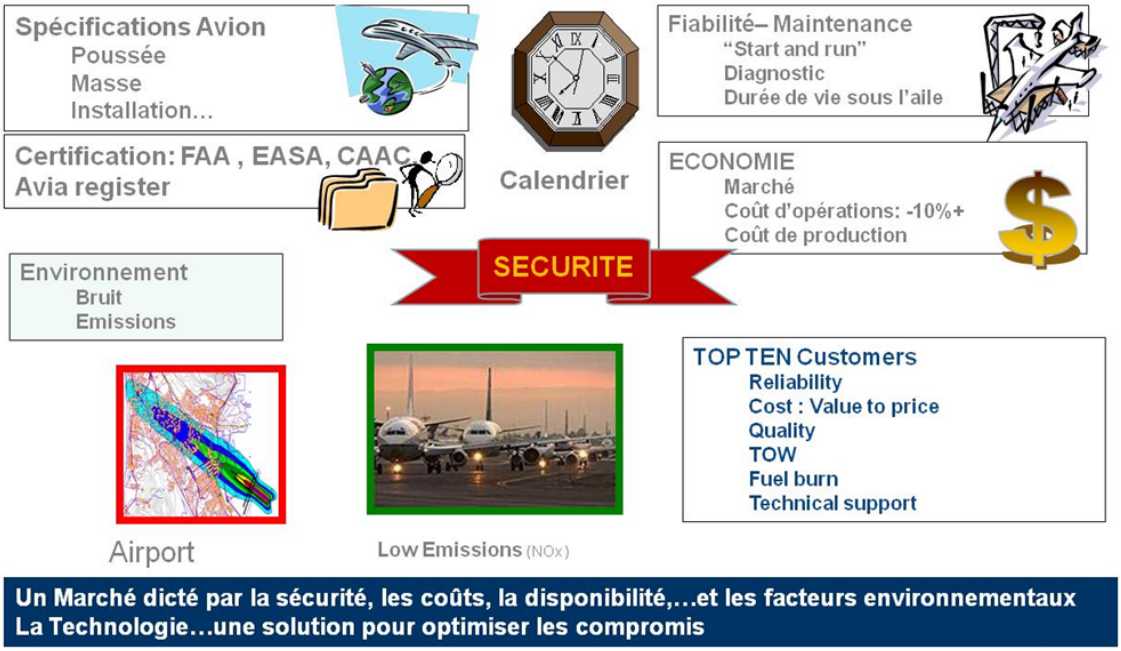


FIGURE I.1 – Les enjeux et la contrainte de la conception des avions.

processus connu par la communauté scientifique, sous l'appellation MDAO (pour "*Multidisciplinary Design Analysis and Optimization*" en anglais) englobe à la fois les idées de la MDA (pour "*MultiDisciplinary Analysis*"), qui réfère au processus où des analyses des disciplines multiples sont amenées à un état cohérent, et la MDO (pour "*MultiDisciplinary Optimization*") qui représente le processus où plusieurs disciplines sont simultanément optimisées par rapport à un ou plusieurs objectifs. Le but est donc de retrouver une conception meilleure que la référence d'origine. Nous envoyons les lecteurs intéressés à consulter les références [Sobieszczanski-Sobieski and Haftka, 1997, Keane and Nair, 2005, Martins and Lambe, 2013] qui définissent la MDO de manière générale dans le domaine de l'aéronautique.

Dans cette thèse, nous nous intéressons à l'aspect optimisation qui est l'une des étapes essentielles dans le problème de conception. L'optimisation permet de rechercher la meilleure configuration possible, pour obtenir une solution qui maximise une performance tout en satisfaisant les exigences des différentes disciplines en jeu. Le domaine de l'optimisation est un domaine très ancien en mathématiques, et ce, même dans le domaine de l'aéronautique. À titre d'exemple, [Goddard, 1920] a cherché à identifier la vélocité de la tuyère d'une roquette requise pour maximiser l'altitude obtenue ; pour ce faire, il se base sur des approximations du comportement de la roquette sur différents intervalles, tout en s'appuyant sur des essais. Avec le développement des moyens informatiques, une variété de méthodes d'optimisation sophistiquées a été développée.

D'une manière globale, les concepteurs ne cherchent pas vraiment à retrouver la meilleure conception qui puisse exister, mais ils cherchent plutôt à l'améliorer par rapport à une conception de référence et à retrouver un certain équilibre entre robustesse, sécurité etc..., car il est impossible de

considérer tous les facteurs impliqués dans les modèles physiques utilisés. En avant projet, l'optimisation permet aussi d'explorer des concepts novateurs. C'est pourquoi, les concepteurs interviennent et jugent les résultats issus d'une optimisation pour mesurer l'impact des facteurs non considérés par la modélisation dans le design final. De plus, plusieurs façons de faire sont souvent possibles pour la résolution des problèmes de conception, et chaque méthode a ses avantages et ses inconvénients.

Contexte industriel

Depuis quelques décennies, le progrès et le perfectionnement des outils mathématiques ont permis de mettre au point des simulateurs numériques de plus en plus représentatifs de la physique qui sont aujourd'hui employés dans les domaines de la recherche et de l'industrie. La communauté scientifique a très vite manifesté une très forte attirance pour ces développements scientifiques, ce qui a suscité en la matière un soutien actif des agences et des organismes qui soutiennent financièrement la recherche scientifique, comme le prouve le rapport de la Fondation National pour la Science (NSF pour "*National Science Foundation*" selon l'appellation anglo-saxonne) en 2006 [on Simulation-Based Engineering Science, 2006].

Les simulations numériques se sont bien répandues dans le domaine de l'aéronautique, en particulier, le domaine des turboréacteurs des avions. Un turboréacteur comporte principalement trois modules successifs essentiels pour son bon fonctionnement (Cf. figure I.2 et figure du LEAP I.3) :

— Le compresseur : l'air est admis dans les différents étages (un couplet roue fixe et roue mobile) de compresseur. Sa pression et sa température augmentent. Le compresseur comprend notamment la soufflante qui assure environ 80% de la poussée.

— La chambre de combustion : à pression constante, la température du fluide augmente grâce à la combustion du carburant.

— La turbine : elle a pour rôle de récupérer l'énergie des gaz de combustion afin d'entraîner les différents étages du compresseur ainsi que la soufflante via l'arbre de transmission.

Les lecteurs intéressés aux principes de la propulsion à réaction peuvent trouver des explications simples sur ce fonctionnement dans [Thevenin, 2004].

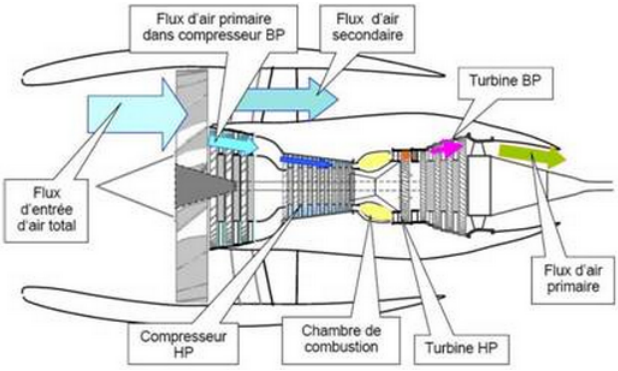


FIGURE I.2 – Vue en coupe d'un turboréacteur.

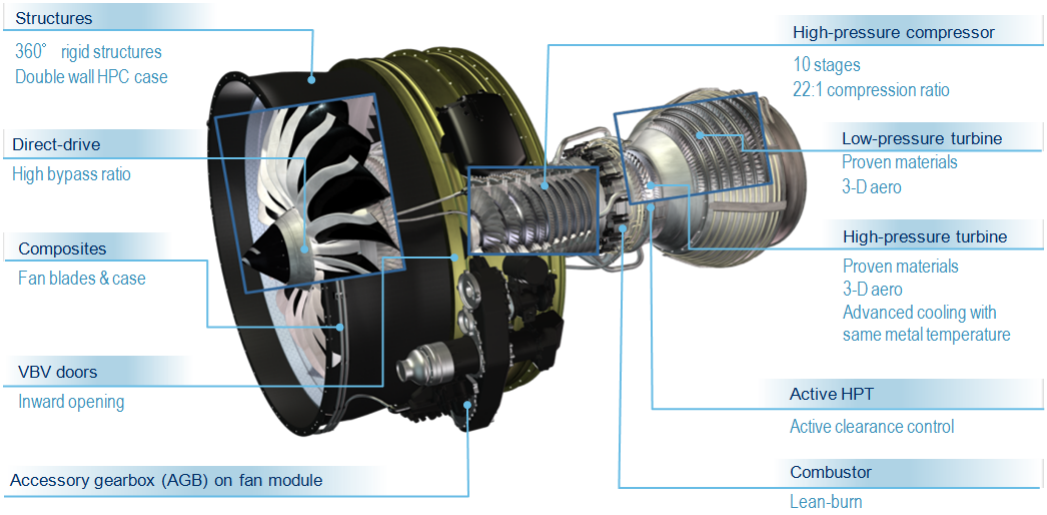


FIGURE I.3 – Turboréacteur de nouvelle génération LEAP.

Le compresseur et la turbine sont composés d'un ensemble de roues aubagées, certaines sont statiques "*stator*" et certaines sont tournantes "*rotor*". Ces roues sont particulièrement sensibles car elles doivent répondre à des impératifs de performance aérodynamique et de tenue mécanique et thermique.

Une attention particulière est portée à la conception des aubages, appelés aussi par abus de langage aubes. En effet, l'aube est l'une des pièces maîtresse permettant d'assurer le bon rendement du moteur aux différentes phases de fonctionnement : décollage, montée, croisière et atterrissage. Pour la concevoir, il faut déterminer ses dimensions caractéristiques en différentes sections le long de sa hauteur, comme le montre la figure I.4.

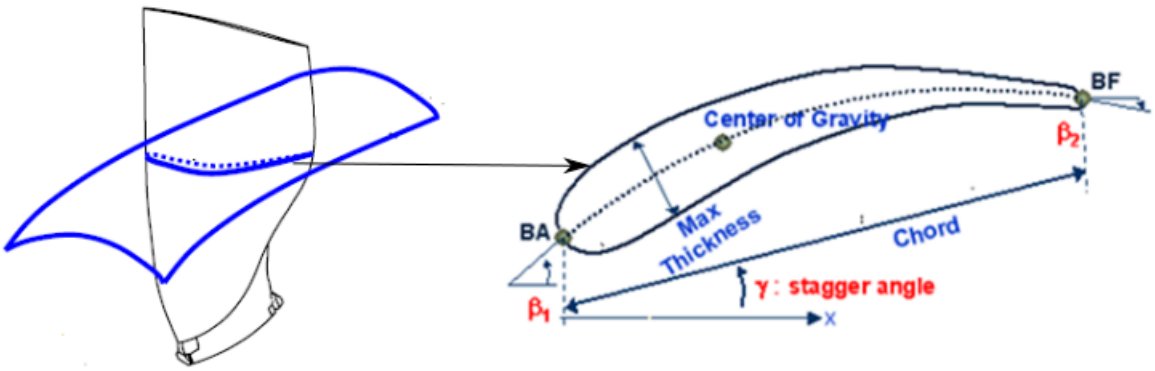


FIGURE I.4 – Une section d'un aubage définie par un jeu de paramètres géométrique.

Généralement, un certain nombre de sections sont considérées. Pour chaque section, entre deux et cinq paramètres sont envisagés. Ces paramètres sont souvent appelés : variables de conception ou variables d'entrée. Dans le cadre d'une optimisation multi-étages, plusieurs roues sont optimisées

en même temps. Le nombre d'étages de turbine peut varier de 1 à 7. La figure I.5 montre l'apport, en termes de gain de rendement, de l'optimisation multi-métiers et multi-étages des modules de turbine.

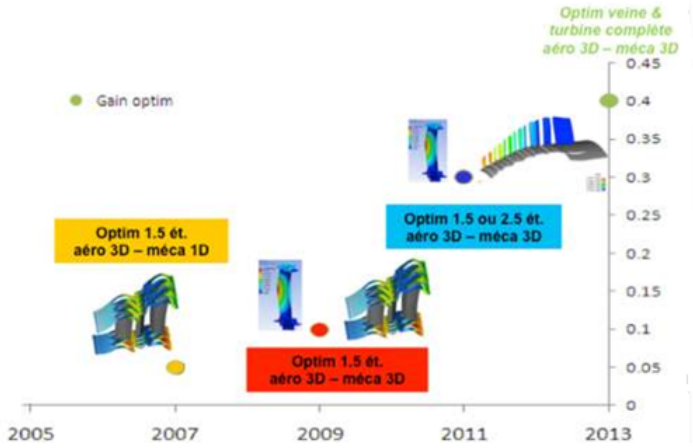


FIGURE I.5 – Évolution du gain obtenu du rendement de la turbine au cours des années à Snecma. Ce gain est obtenu grâce au perfectionnement des codes de calcul et à l'ajout du nombre d'étages considérés.

Au final, nous nous retrouvons avec une conception dépassant la centaine de variables. La modélisation et les calculs aéro-mécaniques à Snecma se font via plusieurs logiciels "maisons" et éditeurs (dont des codes de simulation numérique), dans le cadre d'une chaîne de calcul automatisée sous la plateforme *OPTIMUS*, développée par la société Noesis [Noesis, 2015]. Un exemple de chaînage aéro-mécanique est donné par la figure I.6.

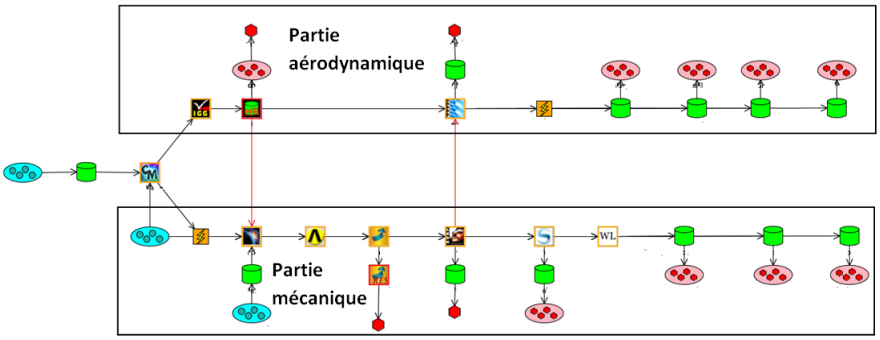


FIGURE I.6 – Exemple de chaîne de dimensionnement aéro-mécanique sous la plateforme *OPTIMUS*. La branche du dessous correspond à la partie mécanique et la branche du dessus correspond à la partie aérodynamique. Les noms des boîtes ont été omis pour des raisons de confidentialité.

Le temps CPU requis, via une chaîne de calcul, pour un calcul aéro-mécanique est souvent coûteux. À titre illustratif, sur un problème type de conception d'aubages de turbomachines, les coûts de calcul des simulations aérodynamiques et mécaniques, peuvent représenter 1 à 4 heures de calcul

pour une géométrie donnée (construction du maillage, soumission et restitution des calculs, etc...). À titre d'exemple, la durée d'un calcul aérodynamique est de l'ordre de 3 heures sur 40 CPU. Dans le cadre d'une optimisation multidisciplinaire d'un ensemble d'aubages (i.e. trouver la meilleure configuration possible de l'aubage en respectant un ensemble de contraintes géométriques, mécaniques, aérodynamiques, etc...), plusieurs difficultés sont rencontrées : outre le grand nombre de variables et le coût de calcul unitaire significatif, un grand nombre de contraintes, allant jusqu'à 300, et un manque d'information sur le gradient des sorties rendent le problème d'optimisation très difficile à résoudre. Pour cela, on adopte des stratégies d'évaluation bien choisies en s'appuyant sur des outils d'approximation tels que les surfaces de réponse, appelés aussi métamodèles ou modèles de substitution. Ce sont des modèles d'approximation des simulateurs numériques mais dont l'évaluation est immédiate. L'optimisation peut ainsi être réalisée directement sur ces surfaces de réponse d'une manière itérative suivant différentes stratégies d'enrichissement (Cf. sections IV.1.1.b. et IV.1.1.c. pour des exemples de critères d'enrichissement).

Formalisme mathématique et objectif de la thèse

Formulation mathématique du problème industriel

Le problème d'optimisation industriel décrit ci-dessus peut être formalisé mathématiquement de la manière suivante :

$$\text{Trouver } \mathbf{x}^* \text{ tel que } \begin{cases} \mathbf{x}^* = \min\limits_{\mathbf{x}} y(\mathbf{x}) \\ \text{s.c.} \\ g_i(\mathbf{x}) \leq 0, \;\; i = 1, \ldots, m \\ \mathbf{x} \in B, \end{cases} \tag{I.1}$$

avec y et $\left(g_i\right)_{i=1,\ldots,m}$ des fonctions définies sur un hypercube $B \subset \mathbb{R}^d$. La fonction y représente un objectif à optimiser (par exemple une performance aérodynamique à maximiser, etc...) et les fonctions $\left(g_i\right)_{i=1,\ldots,m}$ représentent des contraintes métiers à respecter (par exemple les contraintes géométriques, les contraintes mécaniques, etc...). Dans toute la suite, ces fonctions sont considérées comme des fonctions coûteuses à évaluer et aucune information sur les gradients associés n'est disponible (les simulateurs de calcul utilisés à Snecma ne fournissent pas ce type d'information). Comme le nombre d'évaluations disponibles des fonctions est réduit, nous sommes amenés à remplacer ces vraies fonctions par des métamodèles. Pour ce faire, on suppose que l'on dispose de n points formant la matrice $\mathbf{X} = \left[\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\right]^t$ ($\mathbf{x}^{(i)}$ vecteur colonne, pour $i = 1, \ldots, n$), appelé plan d'expériences ou points d'apprentissage [Franco, 2008], et que y et $\left(g_i\right)_{i=1,\ldots,m}$ sont connus seulement

en ces points :

$$\left\{\begin{array}{lcll} \mathbf{y} & = & \left[y^{(1)},\ldots,y^{(n)}\right]^t & = & y(\mathbf{X}) \\ \mathbf{g}_1 & = & \left[g_1^{(1)},\ldots,g_1^{(n)}\right]^t & = & g_1(\mathbf{X}) \\ & & \vdots & & \\ \mathbf{g}_m & = & \left[g_m^{(1)},\ldots,g_m^{(n)}\right]^t & = & g_m(\mathbf{X}). \end{array}\right. \tag{I.2}$$

Remarque 1 *Dans toute la suite, les vraies sorties $y, g_1, \ldots, g_m$ sont évaluées simultanément.*

Ensuite, on choisit un type de métamodèle (régression quadratique, réseau de neurones, krigeage,...) ; des exemples de métamodèles sont décrits par la suite dans la section II.1.. Enfin, on estime les paramètres relatifs à ce métamodèle. Ceci nous permettra de disposer d'une approximation pour chacune des fonctions.

Le choix du métamodèle est primordial car l'évolution de l'optimisation dépend fortement de la qualité du métamodèle. Suivant les critères d'enrichissement utilisés lors de l'optimisation, des nouveaux points sont ajoutés dans le plan d'expériences pour avoir une meilleure approximation du métamodèle dans les zones prometteuses, i.e. les zones susceptibles de contenir les optima, ou dans les zones non explorées. Donc, les métamodèles évoluent en fonction des informations apportées (i.e. les nouveaux points d'enrichissement calculés) dans le plan d'expériences au cours de l'optimisation.

Trouver la meilleure configuration possible d'un aubage répondant aux demandes et aux objectifs physiques n'est pas une tâche facile, à cause principalement du grand nombre de variables de conception à manipuler et des nombreuses contraintes aéro-mécaniques et géométriques à satisfaire et qui, de plus, sont souvent antagonistes. En outre, il faut respecter les délais de conception qui se traduisent par un budget d'évaluation de points, autrement dit, le nombre d'appels au vrai simulateur est généralement très limité.

Remarque 2 *Le problème relatif aux grandes dimensions est appelé la malédiction de la dimension ("curse of dimensionality" selon l'appellation anglo-saxonne) et c'est un terme inventé par [Bellman, 1961]. Il traduit le fait que le coût de calcul et la représentation d'une approximation pour une précision donnée dépendent exponentiellement de la dimension du problème considéré. La malédiction de la dimension rend très difficile l'approximation de la fonction traitée. Elle vient du fait que le volume est une fonction exponentielle de la dimension. Par exemple, en dimension 1 et avec 100 points, nous pouvons échantillonner un segment de taille 1 avec des espaces entre les points de* 10^{-2}*. Dans un espace de dimension 100, il faut* 10^{200} *points pour échantillonner un cube de largeur 1 avec des espaces entre les points de* 10^{-2}*. En dimension d quelconque, il en faut* 10^{2d} *points. De plus, les calculs classiques pour mesurer les distances (la métrique euclidienne) sont peu fiables lorsque le nombre de dimensions est grand. En effet, la métrique euclidienne a tendance à concentrer les distances autour d'une valeur moyenne.*

Objectif et contribution de la thèse

Pour s'intégrer dans la problématique actuelle des ingénieurs Snecma, les enjeux de cette thèse concernent les différentes phases du processus d'optimisation avec contraintes en grande dimension, à savoir la construction d'un métamodèle fiable en grande dimension et le processus d'enrichissement associé. Celui-ci s'est déroulé en trois phases principales :

1. Une étude bibliographique sur les plans d'expériences, les métamodèles et les critères d'enrichissement pour l'optimisation.
2. Le choix et le développement d'un métamodèle adapté aux problèmes de grande dimension en langage Python, ainsi que le processus d'optimisation associé. Nous avons choisi le langage Python car il est largement utilisé au sein de Snecma, et aussi, il permet d'interagir avec la plateforme *OPTIMUS* grâce à une interface codée en Python, appelée *SDK Python*.
3. L'étude du comportement du métamodèle et du processus d'optimisation sur des cas tests académiques et des cas tests industriels fournis par la Snecma ou à disposition sur le web par la communauté (comme le cas de l'automobile MOPTA [Jones, 2008]).

Au cours de ces trois années de thèse, une attention particulière a été portée sur les difficultés rencontrées par le modèle de krigeage (Cf. section III.2.) en grande dimension. Dans notre cas, nous considérons qu'une fonction est de grande dimension si elle dépend de plus d'une dizaine de variables d'entrée. L'une des difficultés principales du modèle de krigeage en grande dimension réside dans l'estimation de ses propres paramètres, appelés hyper-paramètres. Pour ce faire, nous devons maximiser une fonction de vraisemblance, donnée dans la section III.2.4.a., qui est difficile et très coûteuse à optimiser, ce qui rend le modèle de krigeage inutilisable pour certains types de problèmes. En particulier, pour les problèmes de grande dimension traités dans cette thèse, plusieurs contraintes sont à prendre en compte dans le processus d'optimisation et un métamodèle est nécessaire pour chacune de ces contraintes, ce qui multiplie le nombre de métamodèles à construire.

Pour résoudre ce problème, nous avons développé une nouvelle méthodologie en combinant le modèle de krigeage et la méthode de réduction dimensionnelle PLS (pour "*Partial Least Squares*" en anglais, Cf. sectionIII.3.). Ce nouveau modèle, désormais noté KPLS (pour "*Kriging combined Partial Least Squares*" en anglais, Cf. section III.4.), a permis d'avoir un gain très important au niveau du temps de calcul lors de l'estimation de ses hyper-paramètres. Pour ce faire, les modèles de krigeage sont basés sur un objet clef, appelé noyau, qui dépend d'un certain nombre de paramètres, et par l'intermédiaire de quelques opérations élémentaires sur les noyaux, nous avons introduit les informations apportées par la PLS dans le noyau du modèle de krigeage. Ceci a permis de réduire significativement le nombre d'hyper-paramètres à estimer et c'est pourquoi, la construction du modèle de krigeage est accélérée. Il faut noter que cette accélération n'est pas réalisée au détriment de la qualité du modèle. En effet, des tests préliminaires donnés dans la section III.5. ont démontré que le modèle KPLS permet d'obtenir une précision quasi-similaire au modèle de krigeage classique, et voir même meilleure dans certains cas.

Ce nouveau modèle KPLS a été ensuite intégré dans une stratégie d'optimisation auto-adaptative de type SuperEGO [Sasena, 2002] pour les problèmes de grande dimension (le terme EGO de Su-

perEGO fait référence à l'algorithme d'optimisation "*Efficient Global Optimization*" décrit par [Jones et al., 1998]) ; cette stratégie sera appelée SEGOKPLS pour *"Super Efficient Global Optimization using Kriging combined Partial Least Squares*".

De plus, nous avons proposé une nouvelle approche pour estimer les hyper-paramètres des modèles de krigeage pour la grande dimension lorsque le noyau utilisé est de type exponentiel ; cette approche est réalisée en se basant sur l'estimation des hyper-paramètres du modèle KPLS. Cette nouvelle approche nous permet d'améliorer la qualité des modèles KPLS au détriment d'une légère hausse du temps de calcul par rapport au temps de calcul requis pour la construction du modèle KPLS.

Organisation de la thèse

Nous commençons, à partir du chapitre suivant, par une description succincte des métamodèles les plus utilisés dans la littérature. Suit ensuite une discussion pour expliquer les choix qui ont été faits au cours de la thèse et qui seront expliqués dans les chapitres suivants.

Le chapitre III est consacré à la théorie de la construction des modèles KPLS. Dans ce chapitre, nous commençons par présenter les travaux de stage menés en collaboration avec un étudiant des Mines de Saint Étienne sur les plans d'expériences de type hypercube latin [Vauclin, 2014]. Nous continuons ensuite avec des rappels sur la théorie du modèle de krigeage universel dans la section III.2.. Puis, nous proposons une démonstration détaillée sur la validité des noyaux KPLS, noyaux classiques combinés avec les informations extraites par la PLS, en donnant des exemples concrets de noyaux classiques de la littérature. Enfin, des comparaisons entre le modèle de krigeage ordinaire et le modèle KPLS sur des cas tests académiques et industriels sont réalisées, pour illustrer l'efficacité du modèle KPLS.

La théorie sur l'optimisation auto-adaptative via les métamodèles est introduite dans la chapitre IV. Nous allons nous concentrer sur la méthode SuperEGO et notamment sur le critère d'enrichissement utilisé lors de ces travaux. Nous présenterons également quelques résultats d'optimisation académiques connus de la littérature afin de valider l'approche et l'algorithme développé.

Ensuite, nous présentons une nouvelle technique de construction des modèles de krigeage de noyaux exponentiels à partir des modèles KPLS dans le chapitre V. Une étude expérimentale sur une fonction académique fortement multimodale est réalisée pour valider le modèle.

Après avoir validé nos méthodes sur des cas académiques, nous les avons testées dans le dernier chapitre sur deux problèmes industriels : un cas test automobile MOPTA et un cas test turbine basse pression 2.5 étages fourni par la Snecma. Ces deux applications permettent de valider la méthode développée dans cette thèse.

Enfin, les perspectives de ce travail sont présentées dans la dernière partie du document.

II Généralités sur l'optimisation via les métamodèles et motivation de la stratégie suivie

Objectifs du chapitre

- Présenter un état de l'art succinct sur les métamodèles,
- Discuter sur les choix concernant la stratégie d'optimisation suivie.

Ce chapitre a pour vocation de décrire quelques techniques de métamodélisation et d'optimisation assistée par métamodèles existant dans la littérature, et, de donner aussi les principales motivations de nos choix pour la suite de ce manuscrit. La liste des techniques présentées dans la suite n'est pas exhaustive, mais on s'est restreint à décrire les méthodes les plus utilisées dans le domaine de l'optimisation assistée par métamodèles pour les fonctions coûteuses. Nous avons pris soin de présenter séparément les techniques de métamodélisation et les méthodes d'optimisation assistée par métamodèles pour plus de simplicité pour le lecteur. Un guide pratique et une synthèse des techniques de métamodélisation et d'optimisation assistée par métamodèles est disponible dans [Forrester et al., 2008, Jones, 2001]. Le chapitre se découpe en trois sections : la première et la deuxième sections nous donnent, respectivement, un aperçu sur les méthodes de métamodélisation et les méthodes d'optimisation via les métamodèles existantes dans la littérature. Dans la dernière section, nous discutons des choix suivis au cours de ces travaux de thèse.

II.1. Exemples de métamodèles dans la littérature

Comme il a été indiqué dans le chapitre précédent, un métamodèle permet de remplacer une fonction numérique (issue des simulateurs numériques) $y : \mathbf{x} \in B \subset \mathbb{R}^d \mapsto y(\mathbf{x})$, par une autre fonction mathématique $\hat{y} : \mathbf{x} \in B \subset \mathbb{R}^d \mapsto \hat{y}(\mathbf{x})$ de manière à imiter le plus possible y. La construction des métamodèles se déroule principalement en deux étapes :

1. La construction d'un plan d'expériences $(\mathbf{X}, \mathbf{y})$ pour explorer le domaine de définition du problème. Parmi les plans à remplissage d'espace les plus utilisés dans la littérature, nous trouvons les plans de type hypercube latin (Cf. section III.1.).

2. La calibration du métamodèle par l'estimation de ses paramètres. Généralement, les métamodèles dépendent d'un certain nombre de paramètres que nous devons estimer. Cette estimation peut s'avérer délicate pour plusieurs raisons (par exemple, le cas du modèle de krigeage est traité en détails dans la section III.2.4.).

Il existe différents types de métamodèles dans la littérature, faisant appel à certaines hypothèses adaptées aux types de données. Dans le cas de dépendances simples entre les données, des modèles linéaires ou polynomiaux peuvent être utilisés, alors que dans le cas des interactions plus complexes entre les données, des modèles plus élaborés sont requis, tels que les réseaux de neurones, les fonctions à base radiale, les splines ou encore la régression par processus gaussiens (également appelée krigeage). Nous commençons par donner une description du modèle de régression linéaire simple dans le paragraphe suivant.

II.1.1. Modèle de régression linéaire

La régression linéaire est la méthode la plus simple pour approcher la vraie fonction y. Elle consiste à supposer qu'il existe une dépendance linéaire entre les variables d'entrée $\mathbf{x}_1, \ldots, \mathbf{x}_d$ et la variable de sortie y. Le modèle est construit ainsi de la manière suivante :

$$y = [1, \mathbf{x}_1, \ldots, \mathbf{x}_d]\beta + \epsilon, \tag{II.1}$$

où $\beta = [\beta_0, \ldots, \beta_d]^t$ est un vecteur contenant les paramètres du modèle et ϵ correspond au résidu du modèle. La variable aléatoire ϵ est supposée souvent comme une variable normale, centrée et de variance σ_ϵ^2. L'estimation des paramètres β et σ_ϵ^2 est obtenue soit en maximisant la fonction de vraisemblance, et ceci grâce à l'hypothèse des erreurs gaussiennes, soit en minimisant la somme des carrés des écarts entre les vraies observations et les prédictions du modèle (méthode des moindres carrés). Ces deux méthodes conduisent à la même estimation et généralement la méthode des moindres carrés est la plus employée pour ce type de modèle. Le critère des moindres carrés s'écrit comme :

$$\min_{\beta} (\mathbf{y} - \mathbf{X}\beta)^t (\mathbf{y} - \mathbf{X}\beta). \tag{II.2}$$

En dérivant par rapport à β, on obtient une estimation de β :

$$\beta = (\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t\mathbf{y}. \tag{II.3}$$

L'avantage d'un tel modèle est qu'il est très simple à construire et à interpréter. Cependant, il ne permet pas de construire des modèles d'approximation très complexes et il est adapté seulement aux problèmes linéaires. À noter qu'il est possible de transformer la base de fonctions (ici, nous avons considéré la base polynomiale $(1, \mathbf{x}_1, \ldots, \mathbf{x}_d)$) pour ne pas se restreindre à des interactions strictement linéaires. Les transformations usuelles sont les fonctions $\mathbf{x}_i \rightarrow \mathbf{x}_i^k$, $i = 1, \ldots, d$, $k = 0, \ldots, n-1$, permettant la construction de surfaces polynomiales (mais à utiliser avec précaution pour les ordres supérieurs à cause des fortes variations qui peuvent apparaître), ou encore les fonctions log(.), $\sqrt{(.)}$, etc... Il est également possible d'étudier les interactions entre les variables d'entrée en ajoutant les termes correspondants à la fonction de base. Il existe d'autres généralisations de la régression linéaire qui permettent d'observer des interactions complexes. Elles consistent par exemple à effectuer des régressions polynomiales par morceaux (splines) ou encore à supposer que la sortie est une combinaison linéaire de fonctions de base. Le choix des fonctions de base est très important ; leurs propriétés conduiront à des analyses très différentes.

Remarque 3 *Le modèle de krigeage (Cf. section III.2.), qui est le modèle étudié au cours de cette thèse, peut être vu comme une régression par processus gaussien, qui reprend les concepts décrits ci-dessus en employant des fonctions particulières associées à la corrélation spatiale entre les données d'entrée.*

II.1.2. Réseaux de neurones

Les réseaux de neurones ont été inventés par Mc Culloch (neurophysiologiste) et Pitt (logicien) à la fin des années 1950 [Mcculloch et al., 1959] et ils ont proposé les premières notions de neurone formel (Cf. figure II.1). Ils ont été rendus populaires par [Hopfield, 1982]. Ils sont utilisés aujourd'hui dans diverses applications comme le tri automatique du courrier, la robotique, etc... Les lecteurs intéressés par les réseaux de neurones et ses variantes sont invités à consulter les références [Haykin, 1998, McCulloch and Pitts, 1988].

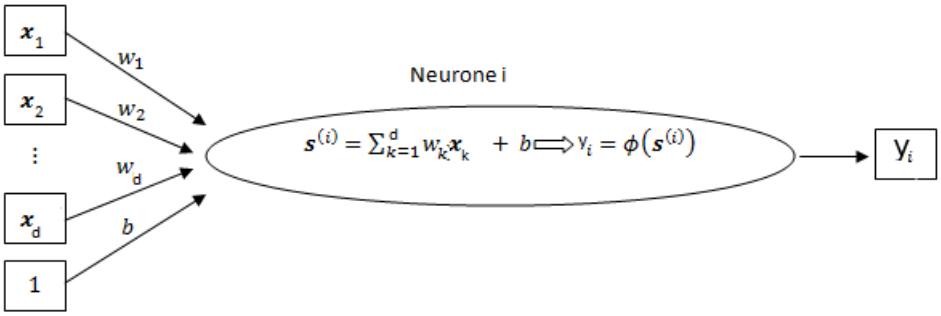


FIGURE II.1 – Éléments qui constituent un neurone formel.

Les réseaux de neurones offrent l'avantage de construire des approximations simples et dérivables. Un réseau de neurones est constitué par un ou plusieurs neurones formels, ceci forme un graphe plus ou moins complexe. Un neurone artificiel, donné par la figure II.1, est constitué :

— Des variables d'entrée qui proviennent soit directement des observations (pour $i = 1, \dots, n$, $(\mathbf{x}_1^{(i)}, \dots, \mathbf{x}_d^{(i)})$), soit d'autres neurones.
— D'un biais qui est toujours égal à 1. Il permet d'ajouter de la flexibilité lors de l'apprentissage.
— Des poids b, w_i $(i = 1, \dots, d)$ qui affectent l'influence de chaque variable d'entrée sur la sortie du neurone.
— Du cœur ou du noyau du neurone qui intègre une fonction d'activation ϕ appliquée à $s = \sum_{k=1}^{d} w_k \mathbf{x}_k + b$.
— De la sortie y qui peut être la sortie finale ou une sortie vers d'autres neurones.

L'évaluation de la fonction d'activation ϕ, qui est intégrée dans le noyau d'un neurone artificiel, se fait en deux étapes : tout d'abord nous calculons la combinaison linéaire des variables d'entrée $s = \sum_{k=1}^{d} w_k \mathbf{x}_k + b$ qui correspond à la sommation des informations reçues, ensuite la fonction d'activation ϕ est appliquée sur s pour indiquer l'état du neurone et transférer le résultat soit au neurone suivant, soit à la sortie finale. Les paramètres $(w_1, \dots, w_d, b)$ sont estimés pendant ce qui est appelée, la phase d'apprentissage.

Les différents types de neurones se différencient suivant la nature de la fonction d'activation ϕ. Les principaux types de fonctions d'activation existant dans la littérature sont :

— *Linéaire* : $\phi =$ la fonction identité.
— *Sigmoïde* : $\phi = \frac{1}{1+e^x}$.
— *Seuil* : $\phi = \mathbf{1}_{[0,+\infty[}(x)$, avec $\mathbf{1}_{[0,+\infty[}(.)$ la fonction indicatrice sur $[0,+\infty[$.
— *Radiale* : $\phi = \sqrt{\frac{1}{2\pi}} e^{-\frac{x^2}{2}}$.

Quelque soit la fonction d'activation, les réseaux de neurones ne sont pas interpolants.

II.1.3. Modèle de fonctions à base radiale

Le modèle d'approximation de fonctions à base radiale, appelé aussi RBF (pour "*Radial Basis Function models*") selon la terminologie anglo-saxonne généralement utilisée, est simple à mettre en œuvre [Powell, 1992]. Ce métamodèle est souvent utilisé pour les problèmes d'optimisation en grande dimension du fait de sa rapidité et de sa performance [Chen et al., 2012, Cheng et al., 2015, Regis, 2014a, Regis, 2014b, Regis and Shoemaker, 2007]. Ce métamodèle consiste, dans un cadre très général, à combiner un ensemble de fonctions de base entre elles. Une fonction de base radiale est une fonction ϕ_j symétrique autour d'un centre $\mathbf{c}^{(j)}$, i.e. $\phi_j(\mathbf{x}) = \phi\left(\left|\left|\mathbf{x} - \mathbf{c}^{(j)}\right|\right|_2\right)$, où $||.||_2$ est la norme euclidienne usuelle. De manière à avoir un modèle interpolant, les points d'apprentissage sont

souvent choisis comme centres des fonctions de base. L'expression du modèle RBF est donnée par :

$$\hat{y}(\mathbf{x}) = \Phi(\mathbf{x})\beta + p(\mathbf{x}) = \sum_{i=1}^{n} \beta_i \phi_i(\mathbf{x}) + p(\mathbf{x}), \tag{II.4}$$

avec $\beta = \left[\beta_1, \ldots, \beta_n\right]^t$, $\Phi(\mathbf{x}) = \left[\phi_1(\mathbf{x}), \ldots, \phi_n(\mathbf{x})\right]$ et $p(\mathbf{x})$ une tendance linéaire sur les d variables. Il existe plusieurs types de fonctions radiales dans la littérature [Powell, 1992] comme les fonctions $\left(\text{on note } \mathbf{h} = \left|\left|\mathbf{x} - \mathbf{c}^{(j)}\right|\right|_2\right)$:

— *Cubiques* : $\mathbf{h}^3$.

— *Thin plate splines* : $\mathbf{h}^2 \ln(\mathbf{h})$.

— *Multiquadratiques* : $\sqrt{1+\epsilon\mathbf{h}}$, avec un paramètre $\epsilon \in \mathbb{R}$.

— *Gaussiennes* : $e^{-(\epsilon\mathbf{h})^2}$, avec un paramètre $\epsilon \in \mathbb{R}$.

II.1.4. Modèle de krigeage

Un autre métamodèle interpolant et qui intéresse de plus en plus la communauté scientifique ces dernières années est le modèle de krigeage. Le krigeage est une méthode qui a été développée par [Matheron, 1963] en géostatistique et repris dans tous les domaines grâce à ses qualités de modélisation, entre autre dans l'aéronautique. Tous les métamodèles décrits précédemment reposent sur une hypothèse fondamentale, qui consiste à supposer que la vraie sortie y se réécrit comme $y = \hat{y} + \epsilon$, avec $\hat{y}$ une approximation déterministe de la sortie et ϵ une variable aléatoire normale, indépendante et identiquement distribuée $\left(\epsilon \sim \mathcal{N}\left(0, \sigma_\epsilon^2\right)\right)$. Cependant, l'idée principale derrière la construction du modèle de krigeage est de supposer que l'erreur ϵ n'est pas indépendante, et qu'elle dépend systématiquement de la position des points, c'est à dire de $\mathbf{x}$. Pour construire le modèle, on fait l'hypothèse que la sortie $y(\mathbf{x})$ étudiée est la réalisation d'un processus gaussien. La sortie est donc modélisée comme un processus gaussien d'espérance $\mu(\mathbf{x})$ et de structure de covariance $k(\mathbf{x}, \mathbf{x}')$, où $\mathbf{x}' \in B$, (nous donnons plus de détails sur une telle construction dans la section III.2.) avec B un hypercube de $\mathbb{R}^d$. La structure de corrélation spatiale aura une importance majeure dans les prédictions 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. De plus, le modèle de krigeage fournit une estimation de l'erreur de prédiction avec une simple formule mathématique (elle sera donnée par l'équation (III.14)). Cette estimation joue un rôle important si le modèle de krigeage est utilisé pour faire de l'optimisation (Cf. section IV.1.1.).

La liste des métamodèles donnés dans ce chapitre est loin d'être exhaustive, d'autres métamodèles existent dans la littérature comme MARS (pour "*Multivariate Adaptive regression splines*") [Hastie et al., 2001, Friedman, 1991], SVM (pour "*Support Vector Machines*") [Bennett and Campbell, 2000, Cristianini and Shawe-Taylor, 2000], ...

II.2. Optimisation itérative basée sur les modèles d'approximation

Généralités

Le domaine d'optimisation, dans un cadre très général, ne fournit pas systématiquement une méthode qui fonctionne pour tous les types de problème. Chaque méthode a ses avantages et ses inconvénients suivant plusieurs critères sur les fonctions à optimiser, comme par exemple la multimodalité des fonctions traitées ou le nombre de dimensions ou encore le coût d'une évaluation. Si nous considérons les fonctions coûteuses, ce qui est le cas dans notre contexte, les méthodes d'optimisation classiques sont inefficaces telles que les méthodes basées sur :

— Le calcul différentiel et la recherche de la direction de descente du gradient lorsque les simulateurs ne fournissent aucune information sur ce dernier. En effet, un trop grand nombre d'appels aux codes de calcul est nécessaire pour approcher les gradients.

— Une recherche aléatoire (méthodes évolutionnaires, recuit simulé, etc…) appliquée directement aux codes de calcul. En effet, un nombre rédhibitoire d'appels aux codes de calcul est nécessaire pour explorer tout le domaine et pour converger.

Durant les dernières décennies, les algorithmes d'optimisation utilisant les métamodèles deviennent progressivement populaires pour les fonctions coûteuses. Les métamodèles jouent le rôle de "*guide*" vers les zones prometteuses, susceptibles de contenir des solutions intéressantes pour le problème d'optimisation traité. L'approche la plus appropriée est d'enrichir le métamodèle avec des nouveaux points d'une manière itérative suivant des critères d'enrichissement bien choisis, et de mettre à jour le métamodèle à chaque itération. Cette mise à jour itérative permet d'améliorer la qualité du métamodèle dans les zones contenant les nouveaux points. Par conséquent, nous obtenons des métamodèles de bonne qualité dans les zones prometteuses.

Il convient aussi de bien connaître les propriétés du métamodèle utilisé afin de choisir le critère d'enrichissement le plus adapté. D. Jones présente dans son article [Jones, 2001] une taxonomie des méthodes d'optimisation suivant le métamodèle utilisé. Nous trouvons aussi deux livres intéressants dans la littérature : [Keane and Nair, 2005] et [Forrester et al., 2008] sur l'optimisation en se basant sur les métamodèles.

L'optimisation basée sur les métamodèles se déroule généralement, d'après le diagramme donné par la figure II.2, selon les étapes suivantes :

1. Construction d'un plan d'expériences initial.
2. Construction des métamodèles à partir du plan d'expériences initial.
3. Optimisation d'un critère d'enrichissement bien choisi qui nous fournit un nouveau point.
4. Évaluation du nouveau point avec le vrai code de calcul.
5. Vérification de la convergence, si oui on arrête l'algorithme, sinon on rajoute le nouveau point dans le plan d'expériences et on réitère à partir de 2.

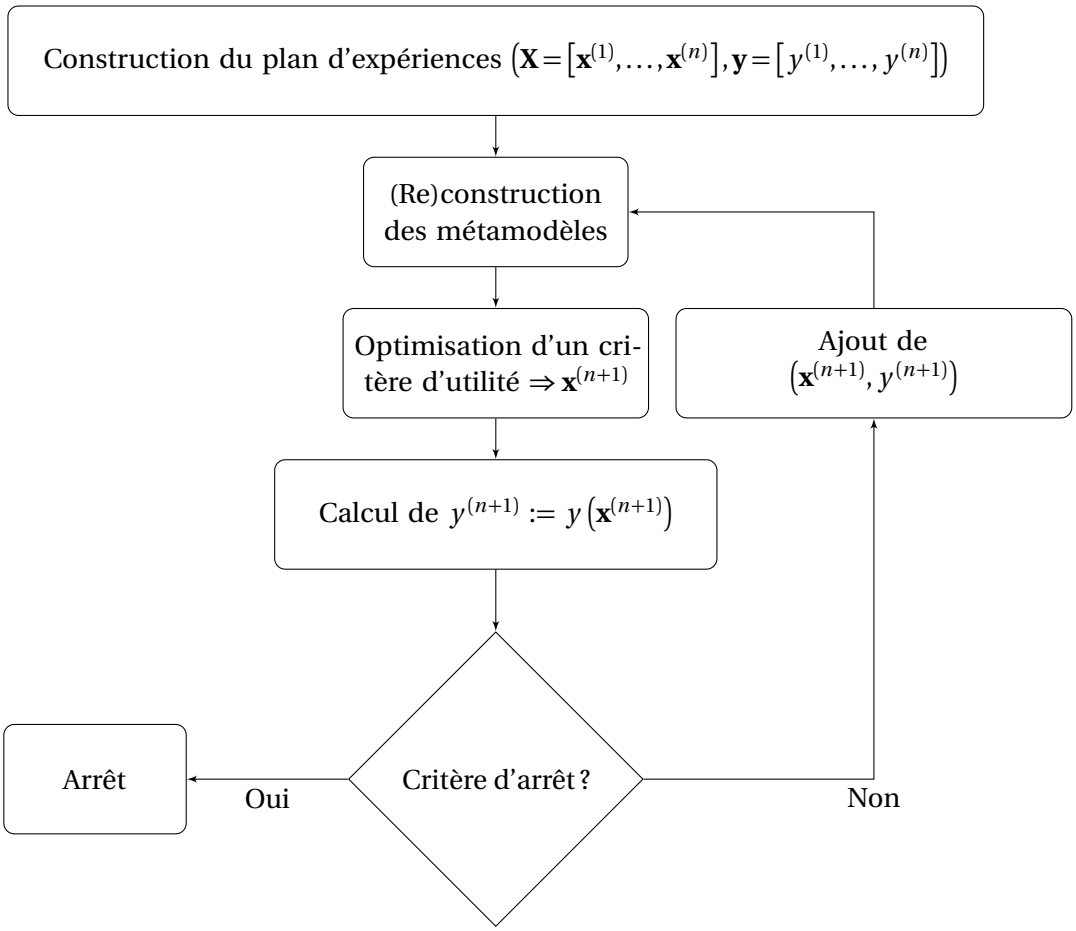


FIGURE II.2 – Les principales étapes d'une optimisation assistée par métamodèles.

L'algorithme EGO (*Efficient Global Optimization*, méthode d'optimisation sans contrainte)

La construction d'un modèle de krigeage peut avoir deux rôles différents : la prédiction ou l'optimisation. Dans le premier cas, le métamodèle permet de prédire le comportement de la fonction en des endroits non explorés tandis que dans le second, le métamodèle permet d'avoir une idée de l'endroit susceptible de contenir le minimum de la fonction. Ce sont deux stratégies complètement différentes : un métamodèle construit pour l'optimisation ne sera pas forcément bon en prédiction et réciproquement, le minimum de la fonction sera approximatif avec un métamodèle prédictif.

La connaissance de l'incertitude dans un modèle de krigeage (donnée par l'équation (III.14)) est un atout majeur. Nous pouvons définir les zones intéressantes comme un compromis entre la valeur du meilleur prédicteur et l'incertitude associée au modèle. L'algorithme EGO 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 critère d'amélioration espérée et il sera noté (EI) (pour "*Expected Improvement*", Cf. section IV.1.1.b.).
3. Enrichissement au point obtenu par la maximisation du critère d'enrichissement.
4. Reconstruction du modèle avec le point supplémentaire ; retour à l'étape 2.

Cet algorithme est décrit en détails dans la section IV.1.1. et il sera adapté aux problèmes d'optimisation avec contraintes en grande dimension (Cf. section IV.1.2.).

II.3. Discussion du choix de la stratégie suivie

Il existe une large variété de méthodes de métamodélisation dans la littérature et il est souvent difficile de choisir le métamodèle le mieux placé pour un problème donné. Dans le monde industriel par exemple, et ce, d'après notre expérience au sein de Snecma, le choix se fait généralement :

— Soit suivant les objectifs du concepteur, par exemple, les modèles de régression sont souvent utilisés pour analyser l'importance des variables sur les sorties et sélectionner les variables les plus influentes (pour une réduction dimensionnelle). Aussi, les modèles de krigeage (Cf. section III.2.) ou les modèles RBF (Cf. section II.1.3.) sont souvent utilisés dans le domaine de l'optimisation.

— Soit suivant le niveau de maîtrise et de compréhension du modèle par le concepteur. En effet, les concepteurs ne sont pas des vrais spécialistes du domaine de la métamodélisation et leurs connaissances s'enrichissent avec l'expérience, ce qui peut influencer leurs choix du métamodèle.

— Soit suivant une étape de validation. En effet, si des points tests, n'appartenant pas aux points d'apprentissage, sont disponibles, alors, le concepteur construit différents modèles à partir de différentes méthodes et choisit le modèle associé à la plus faible erreur. Dans le cas où des points supplémentaires ne sont pas disponibles, des techniques de validation existent comme la validation croisée [Refaeilzadeh et al., 2009].

Pour nos travaux, nous avons choisi d'utiliser le modèle du krigeage pour plusieurs raisons, malgré les limites que possède ce métamodèle pour l'estimation de ses paramètres en grande dimension (Cf. section III.2.4.a.). Les principales raisons de notre choix sont :

— Le krigeage est capable de fournir une estimation de sa propre erreur via une simple expression mathématique. Cette estimation est d'une grande importance, notamment lorsque nous voulons considérer les métamodèles probabilistes, c'est à dire, considérer la prédiction $\hat{y}$ comme une loi de probabilité et exploiter les propriétés probabilistes (Cf. section IV.1.1.).

— Plusieurs types de noyaux existent dans la littérature (Cf. section III.2.3.) qui permettent de couvrir différents comportements mathématiques. Par conséquent, le modèle de krigeage est capable d'approcher différents comportements physiques.

— L'intérêt grandissant de la communauté scientifique ces dernières années pour le modèle de krigeage. En effet, plusieurs travaux en optimisation ont été réalisés en utilisant le modèle de krigeage [Laurenceau, 2008, Kleijnen et al., 2012, Sasena et al., 2002, Kleijnen et al., 2010, Sakata et al., 2004].

III Théorie et validation du modèle KPLS : combiner le modèle de krigeage avec la technique PLS

Objectifs du chapitre

- Rappeler le principe de construction des plans d'expériences basé sur l'optimisation du critère de distance,
- Rappeler brièvement les théories du modèle de krigeage et de la méthode PLS,
- Présenter la théorie de la nouvelle méthode KPLS,
- Valider le modèle KPLS sur des cas tests académiques et industriels.

La méthode KPLS intègre les informations fournies par la méthode de régression PLS au cours de la construction du modèle de krigeage. L'idée est d'intégrer ces informations dans le noyau du modèle de krigeage, permettant ainsi la réduction du nombre d'hyper-paramètres à estimer. Dans ce chapitre, nous commençons par introduire les plans d'expériences de type hypercube latin dans la section III.1.. Puis, les théories du modèle de krigeage et de la régression PLS sont rappelées, respectivement dans les sections III.2. et III.3.. Nous concluons ensuite ce chapitre par la théorie et la validation du nouveau modèle KPLS respectivement dans les sections III.4. et III.5..

III.1. Plans d'expériences de type hypercube latin

Dans cette section, nous abordons les plans d'expériences de type hypercube latin qui seront utilisés par la suite, ainsi que leurs améliorations via un algorithme d'optimisation évolutionnaire. Ces améliorations ont été réalisées dans le cadre du stage de R. Vauclin [Vauclin, 2014].

III.1.1. Principe de construction

Les plans d'expériences de type hypercube latin 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 III.1 présente la différence entre un plan aléatoire pur et un plan hypercube latin.

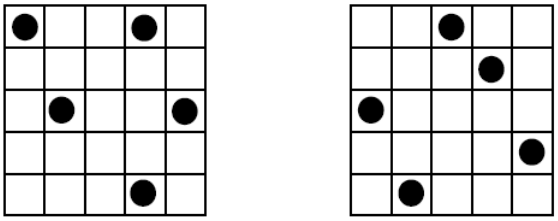


FIGURE III.1 – Différence entre un plan aléatoire pur et un plan lain hypercube pour 2 variables et 5 expériences. À gauche : plan d'expériences aléatoire où 2 lignes et 1 colonne ne sont pas représentées. À droite : plan hypercube latin où toutes les lignes et les colonnes sont représentées.

III.1.2. Optimisation des plans hypercubes latins

Il est possible qu'un plan de type hypercube latin remplisse très mal l'espace. Dans la figure III.2, il n'y a pas de point 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.

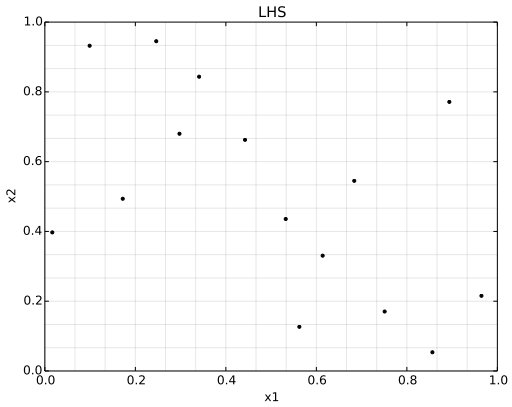


FIGURE III.2 – Exemple de plan hypercube latin pour 2 variables et 15 expériences remplissant mal l'espace.

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} (à maximiser) et le critère ϕ_p (à minimiser) défini par [Morris and Mitchell, 1995] de la façon suivante :

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

avec $p \in \mathbb{N}^*$ et d_{ij} la distance entre deux points $\mathbf{x}^{(i)}$ et $\mathbf{x}^{(j)}$. Dans toute la suite, nous choisissons $p = 10$ (ce choix est réalisé suite à plusieurs tests réalisés). Les qualités en projection sont assurées par les plans hypercubes latins.

Nous présentons ici l'algorithme ESE (*Enhanced Stochastic Evolutionary*), défini par [Jin et al., 2005] qui permet d'obtenir des plans hypercubes latins de bonne qualité. 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 de refroidissement d'un matériau pour minimiser l'énergie qu'il contient. La figure III.3 présente un tel plan en deux dimensions.

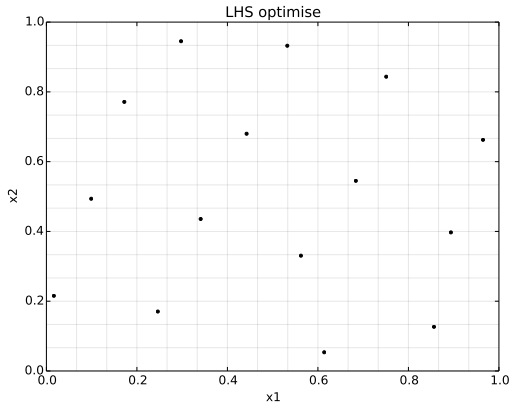


FIGURE III.3 – Plan hypercube latin optimisé avec le critère ϕ_p en dimension 2.

L'algorithme ESE est basé sur deux boucles itératives : une boucle itérative interne et une boucle itérative externe.

III.1.2.a. La boucle itérative interne

Soit M le nombre d'itérations de la boucle interne de l'algorithme ESE. À la $i^{ème}$ itération, pour $i = 1,\ldots,M$, on choisit J plans d'expériences distincts en permutant d'une manière aléatoire les éléments de la colonne i mod d du plan d'expériences $\mathbf{X}$, avec $\mathbf{X}$ un plan candidat trouvé jusqu'à la $i^{ème}$ itération dans la boucle interne. Une telle permutation est donnée par les figures III.4 et III.5 sur un exemple de 5 points en deux dimensions. Nous choisissons ensuite le meilleur plan, appelé

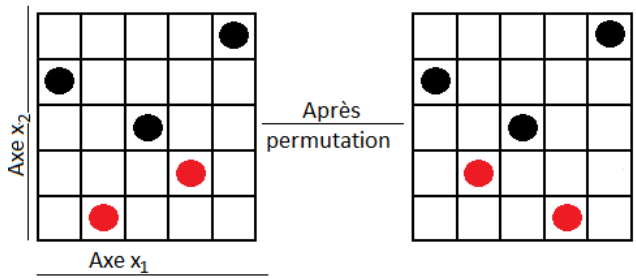


FIGURE III.4 – Exemple de permutation en deux dimensions entre deux points donnés en rouge suivant la direction x_1 ($1^{ère}$ colonne de la matrice $\mathbf{X}$).

$\mathbf{X}_{\text{try}}$, parmi les J plans calculés suivant le critère d'optimalité utilisé. Si le critère de $\mathbf{X}_{\text{try}}$ est meilleur que le critère de $\mathbf{X}$, $\mathbf{X}_{\text{try}}$ remplace donc $\mathbf{X}$, sinon $\mathbf{X}_{\text{try}}$ remplace $\mathbf{X}$ dans le cas où l'inégalité suivante est satisfaite :

$$f(\mathbf{X}_{\text{try}}) - f(\mathbf{X}) \leq T_j \text{random}(0,1) \tag{III.2}$$

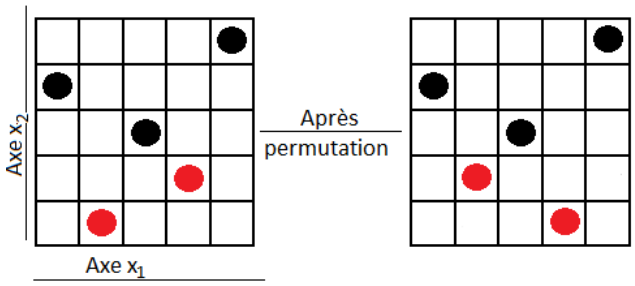


FIGURE III.5 – Exemple de permutation en deux dimensions entre deux points donnés en rouge suivant la direction x_2 ($2^{ème}$ colonne de la matrice $\mathbf{X}$).

avec f le critère d'optimalité choisi, random$(0,1)$ une fonction permettant de générer des valeurs aléatoires comprises entre 0 et 1 et $T_j > 0$ un paramètre de contrôle qui joue le rôle de la température. Par conséquent, si $f(\mathbf{X}_{\text{try}}) - f(\mathbf{X}) \geq T_j$, alors $\mathbf{X}_{\text{try}}$ n'est jamais accepté, et si $0 < f(\mathbf{X}_{\text{try}}) - f(\mathbf{X}) < T_j$ alors $\mathbf{X}_{\text{try}}$ est accepté avec une probabilité de :

$$\mathscr{P}\left(S \geq \frac{f(\mathbf{X}_{\text{try}}) - f(\mathbf{X})}{T_j}\right) = 1 - \frac{f(\mathbf{X}_{\text{try}}) - f(\mathbf{X})}{T_j}, \tag{III.3}$$

avec S une loi uniforme continue sur $[0,1]$. Cette probabilité permet parfois d'accepter un plan $\mathbf{X}_{\text{try}}$ avec un critère d'optimalité légèrement plus faible que le critère de $\mathbf{X}$.

Il est raisonnable que le nombre d'itérations de la boucle interne M soit le plus large possible, mais il faut trouver un compromis entre la durée d'exécution d'une telle procédure et la qualité du résultat final. Après la réalisation de plusieurs tests, nous avons fixé $M = 20d$ sans dépasser 100 itérations, i.e. $M = \min(20d, 100)$. De la même manière, nous avons fixé $J = 20$.

III.1.2.b. La boucle itérative externe

La boucle externe contrôle le processus d'optimisation en mettant à jour le paramètre de contrôle T_j d'une manière itérative. Au début du processus d'optimisation, le paramètre T_j est initialisé à $T_{j0} = 0.005 \times$ valeur du critère du plan d'expériences initial (comme dans [Jin et al., 2005]). Il est mis à jour suivant les deux configurations suivantes :

— *Amélioration du processus* : ceci est appliqué lorsque la boucle interne améliore le meilleur plan trouvé. En effet, le paramètre T_j est maintenu sur une faible valeur de sorte que seulement les plans de meilleure qualité ou d'une qualité légèrement plus faible sont acceptés. Plus précisément, T_j décroit si le taux d'acceptation $\frac{n_{\text{accept}}}{M}$ est plus grand qu'un certain pourcentage $\rho_1 = 10\%$ et si le taux d'amélioration $\frac{n_{\text{imp}}}{M}$ est plus petit que le taux d'acceptation, avec n_{accept} le nombre de fois où $\mathbf{X}_{\text{try}}$ est accepté et n_{imp} le nombre de fois où $\mathbf{X}_{\text{try}}$ améliore le meilleur plan trouvé $\mathbf{X}_{\text{best}}$ dans la boucle interne. Dans ce cas, le paramètre de contrôle devient $T_{j+1} = 0.8\,T_j$. D'autre part, si le taux d'acceptation est plus grand que le pourcentage ρ_1 et si le taux d'amélioration est égal au taux d'acceptation, alors T_j reste inchangé, i.e. $T_{j+1} = T_j$.

Sinon, T_j augmente avec $T_{j+1} = \frac{T_j}{0.8}$. Ce processus permet de se focaliser sur des plans avec une configuration proche de celle du plan optimal.

— *Exploration du processus* : ceci est appliqué lorsque la boucle interne n'améliore pas le meilleur plan trouvé. Dans ce cas, T_j varie suivant le taux d'acceptation. En effet, si le taux d'acceptation est plus petit que le pourcentage ρ_1, T_j augmente rapidement jusqu'à ce que le taux d'acceptation soit plus grand qu'un pourcentage $\rho_2 = 80\%$, avec $T_{j+1} = \frac{T_j}{0.7}$. Ensuite, T_j diminuera doucement jusqu'à ce que le taux d'acceptation devienne plus petit que le pourcentage ρ_1, avec $T_{j+1} = 0.9T_j$. Ce processus permet d'explorer d'autres configurations possibles de plans et d'éviter de rester coincer dans une zone locale.

Remarque 4 *Toutes les valeurs des paramètres utilisées dans cette section sont fixées en se basant sur l'article [Jin et al., 2005], où plusieurs tests ont été réalisés sur ces paramètres.*

III.1.2.c. L'algorithme Enhanced Stochastic Evolutionary

La figure III.6 décrit le processus d'optimisation complet d'un plan d'expériences avec l'algorithme ESE.

Le comportement de l'algorithme est régi par plusieurs paramètres décrits dans les sections III.1.2.a. et III.1.2.b. :

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 $\mathcal{O}\left(dn^2\right)$. 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. Les détails d'une telle réduction de complexité sont donnés dans [Jin et al., 2005].

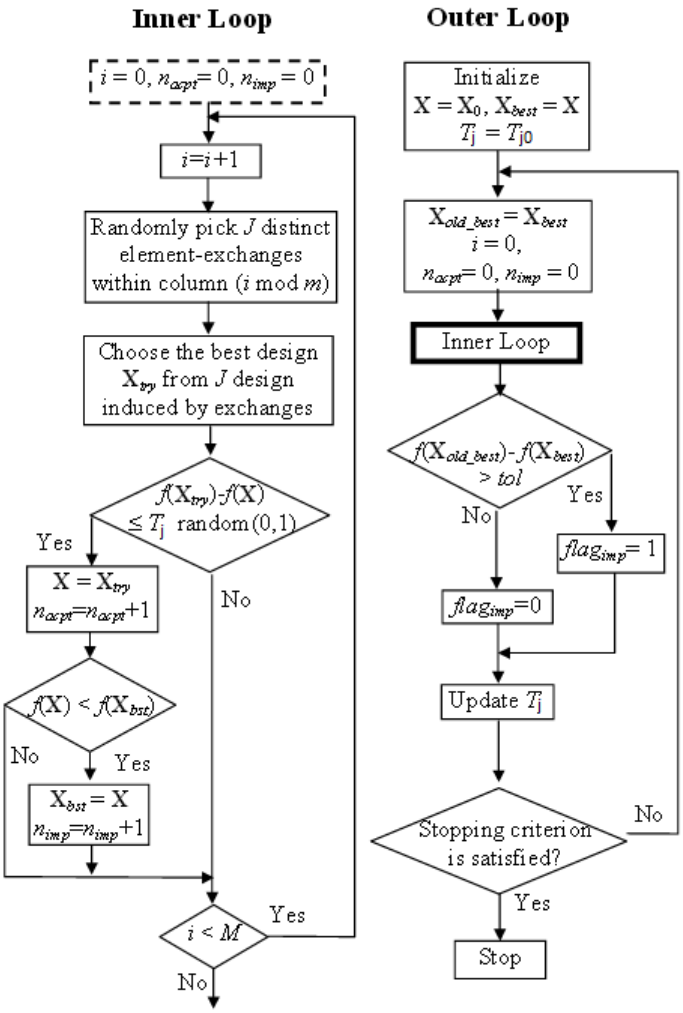


FIGURE III.6 – Fonctionnement de l'algorithme ESE (source : [Jin et al., 2005]).

III.1.3. Implémentation 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 plans hypercubes latins 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 les points du plan). Ceci est coûteux en temps et très peu performant. Le logiciel OpenMDAO (dans sa version 0.11 de juin 2014) 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.

Dans la suite, nous illustrons le comportement de l'algorithme ESE.

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 d du problème. On définit par défaut :

— $stop = \min(1.5d, 30)$;

— $M = \min(20d, 100)$;

— $J = 20$.

La figure III.7 présente un plan optimisé avec ces paramètres et la figure III.8 présente l'évolution du critère de minimisation ϕ_p ainsi que la température T de l'algorithme.

Pour avoir une idée des gains en qualité et en temps de calcul, la table III.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 à même nombre de plans construits, soit $J \times M \times stop$ plans au total avec les valeurs par défaut définies ci-dessus, pour 10 réplications à chaque fois.

Nous remarquons tout d'abord dans cette table que les temps de calcul de la recherche aléatoire explosent complètement avec la montée en dimension, ce qui n'est pas le cas pour l'algorithme 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 l'algorithme ESE sont de bien meilleure

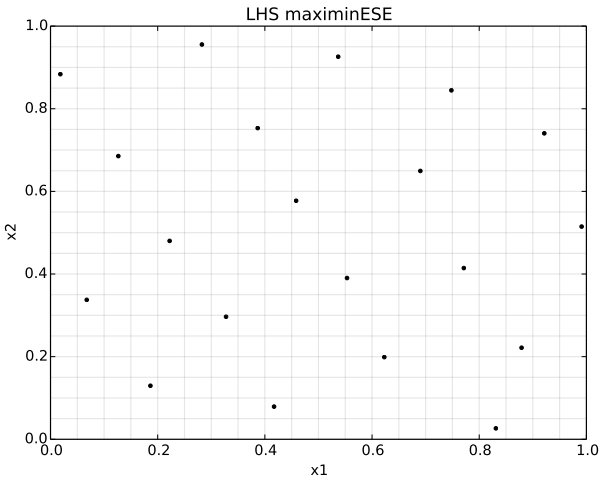


FIGURE III.7 – Plan d'expériences à 2 dimensions de 20 points construit avec l'algorithme ESE.

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.

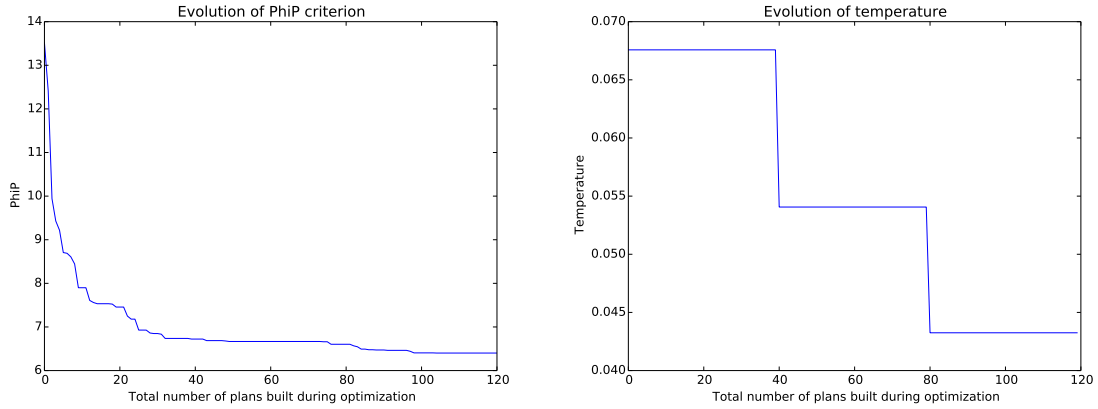
qualité, à la fois en termes de distance minimale et de ϕ_p. La limitation imposée aux paramètres en grande dimension a un léger impact ; la convergence n'est pas tout à fait achevée.

La construction des plans d'expériences via l'algorithme ESE est rapide et efficace. Dans la suite des travaux, tous les plans d'expériences sont obtenus via cet algorithme.

III.2. Le modèle de krigeage

La théorie du modèle de krigeage a été développée par le minier sud-africain Daniel Gerhardus Krige [Krige, 1951]. Elle a été formalisée pour la prospection minière par [Matheron, 1963]. À la fin des années 1980, cette technique a été adaptée et utilisée pour la prédiction des expériences déterministes de par l'essor des moyens informatiques [Sacks et al., 1989a, Sacks et al., 1989b, Welch et al., 1992].

L'hypothèse fondamentale du modèle de krigeage est de supposer que la sortie y est une réalisation



(a) (b)

FIGURE III.8 – Déroulement de l'algorithme ESE pour construire le plan représenté par la figure III.7 avec 3 itérations de la boucle externe et 40 itérations de la boucle interne. III.8a : Évolution du critère ϕ_p. III.8b : Évolution de la température T.

d'un processus gaussien Y, qui s'écrit :

$$\forall \mathbf{x} \in B,\ Y(\mathbf{x}) = \mu(\mathbf{x}) + Z(\mathbf{x}), \tag{III.4}$$

avec $\mu(\mathbf{x})$ la moyenne du modèle au point $\mathbf{x}$ et $Z(\mathbf{x})$ un processus gaussien centré de noyau connu (les paramètres du noyau sont supposés connus lors de la formulation du modèle de krigeage).

On peut distinguer trois types de modèles de krigeage suivant la nature de μ :

1. Le krigeage simple où μ est connue.
2. Le krigeage ordinaire où μ est une constante inconnue.
3. Le krigeage universel où $\mu(\mathbf{x}) = \sum_{i=1}^{a} \beta_i f_i(\mathbf{x})$ est inconnue, avec $\left(f_i(\mathbf{x})\right)_{i=1,\dots,a}$ des fonctions de base polynomiales et $\left(\beta_i\right)_{i=1,\dots,a}$ les coefficients correspondants.

Dans la suite, nous présentons les formules du modèle de krigeage universel avec une déduction pour le modèle de krigeage ordinaire.

III.2.1. Principe du fonctionnement du modèle de krigeage

L'idée principale derrière la construction du modèle de krigeage pour les fonctions déterministes est de considérer les erreurs données par $Z(\mathbf{x})$ dans l'équation (III.4) comme non-indépendantes, contrairement à la majorité des modèles existants dans la littérature. L'indépendance de l'erreur dans un cadre déterministe n'est pas vraiment rationnelle d'un point de vue théorique. En effet, n'importe quelle erreur donnée par le code de calcul est due à une erreur de modélisation qui dépend

de $\mathbf{x}$ et non à une erreur de mesure ou à une erreur causée par un bruit (la répétition d'une même expérience plusieurs fois donne toujours le même résultat). De plus, si la sortie y est continue, ce qui est le cas ici, alors l'erreur est continue car elle est donnée par $(Y-\mu)(\mathbf{x})$. Ceci implique que si $\mathbf{x}'$ et $\mathbf{x}''$ sont proches alors forcément $Z(\mathbf{x}')$ et $Z(\mathbf{x}'')$ le sont aussi. Ce principe est illustré dans la figure III.9.

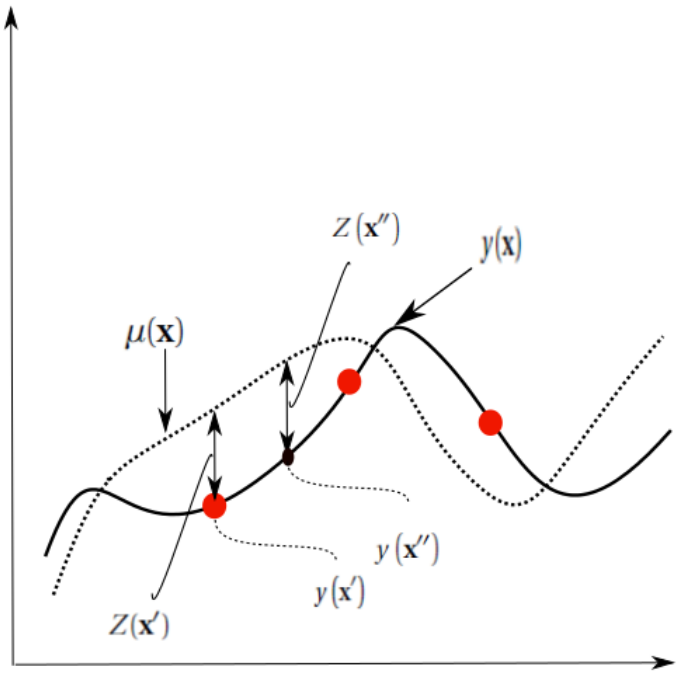


FIGURE III.9 – Principe de fonctionnement du modèle de krigeage. La régression du modèle de krigeage est donnée par la courbe en pointillée, la vraie sortie $y(\mathbf{x})$ est donnée par la courbe en trait plein et le plan d'expériences est donné par les points rouges.

III.2.2. Formulation du modèle de krigeage

Pour formuler l'expression du modèle de krigeage, nous commençons tout d'abord par donner quelques définitions et notations. Nous définissons le vecteur de fonctions de base polynomiales $\mathbf{f}$, les coefficients β associés et la matrice $\mathbf{F}$ par :

$$\mathbf{f}_{\mathbf{x}} = \left[f_1(\mathbf{x}), \ldots, f_a(\mathbf{x})\right]^t, \quad \beta = \left[\beta_1, \ldots, \beta_a\right]^t, \quad \mathbf{F} = \begin{bmatrix} \mathbf{f}^t_{\mathbf{x}^{(1)}} \\ \vdots \\ \mathbf{f}^t_{\mathbf{x}^{(n)}} \end{bmatrix}. \tag{III.5}$$

Nous notons aussi le processus gaussien :

$$\mathbf{z} = \left[Z\left(\mathbf{x}^{(1)}\right), \ldots, Z\left(\mathbf{x}^{(n)}\right)\right]^t, \tag{III.6}$$

où $Z(\mathbf{x}) \sim \mathcal{N}(0, \sigma_z^2)$.

Nous réécrivons l'équation (III.4) comme :

$$\mathbf{y} = \mathbf{F}\beta + \mathbf{z}. \tag{III.7}$$

Nous notons la fonction de covariance par $k(\mathbf{x}, \mathbf{x}') = \text{cov}(Y(\mathbf{x}), Y(\mathbf{x}'))$ (par abus de langage, nous la notons parfois k) qui est définie sur $B \times B$ et qui est appelée noyau de covariance. Nous définissons la matrice de covariance associée :

$$k(\mathbf{X}, \mathbf{X}) = \text{cov}(Y(\mathbf{X}), Y(\mathbf{X})) = \sigma_z^2 R(\mathbf{X}, \mathbf{X}) = \sigma_z^2 \begin{bmatrix} R\left(\mathbf{x}^{(1)}, \mathbf{x}^{(1)}\right) & \cdots & R\left(\mathbf{x}^{(1)}, \mathbf{x}^{(n)}\right) \\ \vdots & \ddots & \vdots \\ R\left(\mathbf{x}^{(n)}, \mathbf{x}^{(1)}\right) & \cdots & R\left(\mathbf{x}^{(n)}, \mathbf{x}^{(n)}\right) \end{bmatrix}, \tag{III.8}$$

avec la matrice de corrélation $R(\mathbf{X}, \mathbf{X}) = \text{cor}(Y(\mathbf{X}), Y(\mathbf{X}))$ où $R(\mathbf{X}, \mathbf{X})_{ij} = R\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right)$ pour tout $i, j = 1, \ldots, n$, qui est la corrélation entre le $i^{ème}$ et le $j^{ème}$ élément de $\mathbf{X}$. Pour des raisons de lisibilité, nous notons $R(\mathbf{X}, \mathbf{X})$ par la matrice $\mathbf{R}$. La fonction de corrélation R dépend généralement d'un ensemble de paramètres que nous supposons connus pour le moment. L'estimation de ces paramètres est décrite dans la section III.2.4..

Enfin, nous définissons le vecteur de corrélation $\mathbf{r}_\mathbf{x}$ entre un point $\mathbf{x}$ quelconque et les n points du plan d'expériences $\mathbf{X}$ par :

$$\mathbf{r}_\mathbf{x} = \left[R\left(\mathbf{x}^{(1)}, \mathbf{x}\right), \ldots, R\left(\mathbf{x}^{(n)}, \mathbf{x}\right)\right]^t. \tag{III.9}$$

Pour calculer le meilleur prédicteur linéaire sans biais (BLUP, *Best Linear Unbiased Predictor* pour l'appellation anglo-saxonne), nous avons besoin de mesurer l'erreur quadratique de la prédiction et de fixer une contrainte pour garantir une représentation sans biais de notre modèle.

La prédiction linéaire s'écrit sous la forme $\mathbf{c}_\mathbf{x}^t \mathbf{y}$ qui est une combinaison linéaire de $\mathbf{y}$ par les coefficients du vecteur colonne $\mathbf{c}_\mathbf{x}$. Donc, d'après l'équation (III.7) on a :

$$\mathbb{E}\left[\mathbf{c}_\mathbf{x}^t \mathbf{y}\right] = \mathbf{c}_\mathbf{x}^t \mathbf{F}\beta, \tag{III.10}$$

car $\mathbb{E}[Z(\mathbf{x})] = 0$. D'autre part, avec l'équation (III.4) on a :

$$\mathbb{E}[Y(\mathbf{x})] = \mathbb{E}\left[\mathbf{f}_\mathbf{x}^t \beta + Z(\mathbf{x})\right] = \mathbf{f}_\mathbf{x}^t \beta. \tag{III.11}$$

Par identification, on obtient :

$$\mathbf{F}^t \mathbf{c}_\mathbf{x} = \mathbf{f}_\mathbf{x}. \tag{III.12}$$

Nous minimisons maintenant l'erreur quadratique de la prédiction $\mathbb{E}\left[\left(\mathbf{c}_{\mathbf{x}}^t\mathbf{y} - Y(\mathbf{x})\right)^2\right]$ en prenant en compte la contrainte donnée par l'équation (III.12). Les expressions de la prédiction et de la variance du modèle de krigeage sont données par :

$$\hat{y}(\mathbf{x}) = \mathbf{f}_{\mathbf{x}}^t\hat{\beta} + \mathbf{r}_{\mathbf{x}}^t\mathbf{R}^{-1}(\mathbf{y} - \mathbf{F}\hat{\beta}), \tag{III.13}$$

$$s^2(\mathbf{x}) = \hat{\sigma}_z^2\left[1 + \mathbf{r}_{\mathbf{x}}^t\mathbf{R}^{-1}\mathbf{r}_{\mathbf{x}}\right]. \tag{III.14}$$

Pour plus de détails sur les calculs, nous renvoyons le lecteur vers les travaux de [Schonlau, 1998].

Les équations (III.13) et (III.14) nous donnent l'expression du modèle de krigeage universel ($a > 1$). La formule de prédiction du modèle de krigeage ordinaire [Forrester et al., 2008], qui considère la partie déterministe de l'équation (III.4) comme une constante μ inconnue (donc $a = 1$), est donnée par :

$$\hat{y}(\mathbf{x}) = \hat{\mu} + \mathbf{r}_{\mathbf{x}}^t\mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu}), \tag{III.15}$$

avec $\mathbf{1} = [1 \cdots 1]^t \in \mathbb{R}^n$. L'estimation de la variation du modèle donnée par l'équation (III.14) reste inchangée pour le modèle de krigeage ordinaire.

Les formules présentées ci-dessus dénotent les propriétés particulières du modèle de krigeage. On peut remarquer que le meilleur prédicteur interpole les données et que la variance de prédiction est nulle en ces points (Cf. figure III.10). De plus, la variance de prédiction ne dépend pas de la valeur des observations.

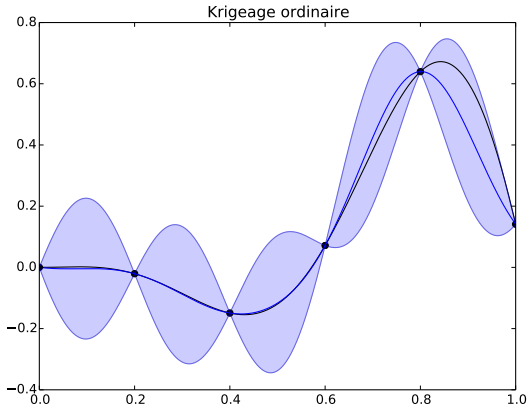


FIGURE III.10 – Allure classique d'un modèle de krigeage. Le modèle interpole les données et la variance est nulle en ces points.

III.2.3. Noyaux de covariance

Il existe dans la littérature plusieurs noyaux de covariance appartenant à deux types de noyaux : les noyaux stationnaires (noyaux invariants par translation) et les noyaux non-stationnaires. Dans ce manuscrit, nous nous intéressons seulement aux noyaux stationnaires séparables (Cf. annexe I.1. pour la définition d'un noyau séparable).

Le noyau de covariance permet de mesurer le degré de ressemblance entre deux points dans le domaine de conception, et donc de mettre en cohérence deux types d'informations distinctes. Une condition nécessaire et suffisante pour que k soit un noyau mathématiquement valide est qu'il soit symétrique semi-défini positif, autrement dit :

— **symétrie** : $\forall \mathbf{x}, \mathbf{x}' \in B, k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$,

— **positivité** : $\forall n_0 \in \mathbb{N}^*, \forall \lambda_1, \ldots, \lambda_{n_0} \in \mathbb{R}, \forall \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n_0)} \in B, \sum_{i=1}^{n_0} \sum_{j=1}^{n_0} \lambda_i \lambda_j k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \geq 0.$

Généralement, cette dernière condition est très difficile à démontrer et il est tout à fait naturel de faire appel aux noyaux connus. Cependant, de nouveaux noyaux peuvent être construits à partir de l'existant par l'intermédiaire d'opérations classiques sur les noyaux.

III.2.3.a. Validation du noyau de covariance

Le choix du noyau est primordial pour la modélisation des phénomènes physiques et la détermination de ses propriétés (continuité, dérivabilité ...). Malheureusement, il n'existe pas jusqu'à présent une méthode permettant de choisir le noyau le mieux adapté au problème traité et qui fonctionne dans tous les cas. Cependant, les techniques classiques de validation de modèle, comme la validation croisée [Refaeilzadeh et al., 2009] ou le calcul d'erreur via des points de validation ..., peuvent être employées pour guider notre choix. En effet, 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, nous utilisons alors la validation croisée ou *u-fold cross-validation* qui consiste à diviser les données en u lots puis à construire le modèle avec $u-1$ lots, le dernier servant de base de validation. Cette opération est répétée u fois pour que tous les lots aient servi une fois de base de validation. La méthode *Leave-One-Out* reprend ce principe en fixant $u = n$ où n est le nombre d'observations disponibles ; les lots contiennent alors un unique point.

Remarque 5 *La validation croisée est souvent appelée k-fold cross-validation dans la littérature, mais nous avons délibérément changé k par u pour ne pas confondre avec les noyaux notés k.*

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 u 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 $\mathbf{R}$ et $\mathbf{F}$ en supprimant uniquement les lignes et les colonnes des points servant à la base de validation. Les

paramètres β et σ_z^2 peuvent être recalculés de façon simple à partir des matrices réduites dans les formules (III.23) et (III.24). Il n'est pas nécessaire de recalculer $\mathbf{R}^{-1}$, il suffit d'appliquer les formules d'inversion par blocs pour obtenir simplement l'inverse de la matrice réduite.

III.2.3.b. Exemples de noyaux classiques

Les noyaux stationnaires les plus utilisés dans la littérature, comme l'exponentielle généralisée ou la gaussienne, ..., sont donnés dans la table III.2. Il faut noter que le nombre d'hyper-paramètres à estimer est généralement supérieur ou égal à la dimension d. Pour plus d'informations sur les propriétés de tels noyaux, nous invitons le lecteur à consulter l'ouvrage [Rasmussen and Williams, 2006].

TABLE III.2 – Exemples de noyaux stationnaires. La fonction de covariance est exprimée en fonction du $i^{ème}$ élément $\mathbf{m}_i = \left|\mathbf{x}_i - \mathbf{x'}_i\right|$ avec $\theta_i \geq 0$ et $p_i \in [0,2]$, $\forall i = 1,\ldots,d$.

fonction de covariance	expression	hyper-paramètres θ	nombre d'hyper-paramètres
Exponentielle généralisée	$\sigma^2 \prod_{i=1}^{d} \exp(-\theta_i \mathbf{m}_i^{p_i})$	$(\theta_1,\ldots,\theta_d,p_1,\ldots,p_d)$	$2d$
Gaussienne	$\sigma^2 \prod_{i=1}^{d} \exp(-\theta_i \mathbf{m}_i^2)$	$(\theta_1,\ldots,\theta_d)$	d
Matern$\frac{5}{2}$	$\sigma^2 \prod_{i=1}^{d} (1+\sqrt{5}\theta_i \mathbf{m}_i + \frac{5}{3}\theta_i^2 \mathbf{m}_i^2)\exp(-\sqrt{5}\theta_i \mathbf{m}_i)$	$(\theta_1,\ldots,\theta_d)$	d
Matern$\frac{3}{2}$	$\sigma^2 \prod_{i=1}^{d} (1+\sqrt{3}\theta_i \mathbf{m}_i)\exp(-\sqrt{3}\theta_i \mathbf{m}_i)$	$(\theta_1,\ldots,\theta_d)$	d

III.2.3.c. Création de nouveaux noyaux à partir des noyaux classiques

Nous allons voir dans cette section quelques opérations algébriques élémentaires permettant de créer de nouveaux noyaux tout en conservant les propriétés de positivité et de symétrie. Pour les démonstrations de ces résultats, nous renvoyons le lecteur au chapitre 4 de [Rasmussen and Williams, 2006].

1. La somme de deux noyaux k_1 et k_2 est un noyau :

$$k : \begin{cases} B \times B & \longrightarrow & \mathbb{R} \\ (\mathbf{x},\mathbf{x'}) & \longmapsto & k_1(\mathbf{x},\mathbf{x'}) + k_2(\mathbf{x},\mathbf{x'}) \end{cases} \tag{III.16}$$

2. Le produit de deux noyaux k_1 et k_2 est un noyau :

$$k : \begin{cases} B \times B & \longrightarrow & \mathbb{R} \\ (\mathbf{x},\mathbf{x'}) & \longmapsto & k_1(\mathbf{x},\mathbf{x'}) \times k_2(\mathbf{x},\mathbf{x'}) \end{cases} \tag{III.17}$$

ou encore

$$k : \begin{cases} B^2 \times B'^2 & \longrightarrow & \mathbb{R} \\ ((\mathbf{x},\mathbf{x}'),(\mathbf{u},\mathbf{u}')) & \longmapsto & k_1(\mathbf{x},\mathbf{x}') \times k_2(\mathbf{u},\mathbf{u}') \end{cases} \tag{III.18}$$

On peut généraliser ces formules pour $p \in \mathbb{N}^*$ noyaux.

3. Le produit d'un noyau k_1 par une fonction $h(\mathbf{x})$ est un noyau donné par :

$$k : \begin{cases} B \times B & \longrightarrow & \mathbb{R} \\ (\mathbf{x},\mathbf{x}') & \longmapsto & h(\mathbf{x})k_1(\mathbf{x},\mathbf{x}')h(\mathbf{x}') \end{cases} \tag{III.19}$$

Il existe bien évidemment d'autres opérations sur les noyaux, mais nous allons nous intéresser en particulier à la construction donnée par l'équation (III.17). Nous allons voir dans la section III.4. comment une telle opération va nous permettre de construire un nouveau noyau pour traiter les problèmes de grande dimension.

III.2.4. Estimation des hyper-paramètres

III.2.4.a. Calcul de la fonction de vraisemblance

Le modèle de krigeage fait intervenir un lot de paramètres inconnus que nous avons supposé connus lors de la construction du modèle : θ, σ_z^2 et β. À notre connaissance, deux méthodes existent dans la littérature pour estimer ces hyper-paramètres : la méthode du maximum de vraisemblance et la méthode de validation croisée [Bachoc, 2013]. Dans ce manuscrit, nous utilisons la première méthode qui consiste à maximiser la fonction de vraisemblance :

$$L\left(y^{(1)},\ldots,y^{(n)}\right) = \prod_{i=1}^{n} \phi\left(y^{(i)}\right), \tag{III.20}$$

où ϕ est la fonction de densité de probabilité du processus Y. Dans le cas du modèle de krigeage universel, nous avons (pour plus de détails, Cf. [Sasena, 2002, Schonlau, 1998, Liem et al., 2015]) :

$$L\left(\mathbf{y};\theta,\sigma_z^2,\beta\right) = \frac{1}{\sqrt{2\pi\sigma_z^2}^n \sqrt{\det(\mathbf{R})}} \exp\left(-\frac{1}{2\sigma_z^2}\left(\mathbf{y}-\mathbf{F}\beta\right)^t \mathbf{R}^{-1}\left(\mathbf{y}-\mathbf{F}\beta\right)\right). \tag{III.21}$$

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

$$-l(\mathbf{y};\theta,\sigma_z^2,\beta) = \left(\frac{n}{2}\log(\sigma_z^2) + \frac{n}{2}\log(\det(\mathbf{R}))\right)\left(-\frac{1}{2\sigma_z^2}(\mathbf{y}-\mathbf{F}\beta)^t\mathbf{R}^{-1}(\mathbf{y}-\mathbf{F}\beta)\right). \tag{III.22}$$

Il est alors possible de déterminer les paramètres $\hat{\beta}$ et $\hat{\sigma}_z^2$ de façon analytique. En dérivant la formule par rapport à β, on obtient :

$$\hat{\beta} = \left(\mathbf{F}^t \mathbf{R}^{-1} \mathbf{F}\right)^{-1} \mathbf{F}^t \mathbf{R}^{-1} \mathbf{y}. \tag{III.23}$$

Ce qui permet d'obtenir $\hat{\sigma}_z^2$ de la même façon :

$$\hat{\sigma}_z^2 = \frac{(\mathbf{y} - \mathbf{F}\hat{\beta})^t \mathbf{R}^{-1} (\mathbf{y} - \mathbf{F}\hat{\beta})}{n}. \tag{III.24}$$

De la même manière, on peut estimer μ et σ_z^2 pour le modèle de krigeage ordinaire :

$$\hat{\mu} = (\mathbf{1}^t \mathbf{R}^{-1} \mathbf{1})^{-1} (\mathbf{1}^t \mathbf{R}^{-1} \mathbf{y}), \tag{III.25}$$

$$\hat{\sigma}_z^2 = \frac{(\mathbf{y} - \hat{\mu}\mathbf{1})^t \mathbf{R}^{-1} (\mathbf{y} - \hat{\mu}\mathbf{1})}{n}. \tag{III.26}$$

En incluant les équations (III.23) et (III.24) dans (III.22), on obtient alors :

$$-l(\mathbf{y}; \theta) = n \log\left(\hat{\sigma}_z^2\right) + \log(\det(\mathbf{R})) \tag{III.27}$$

Il reste à minimiser l'expression (III.27) pour estimer les paramètres $\theta = (\theta_1, \ldots, \theta_d)$, car on rappelle que σ_z et $\mathbf{R}$ dépendent que de θ. Il n'y a pas de solution analytique à ce problème ; il faut utiliser un optimiseur. On note dans la formule (III.27) qu'il faut inverser la matrice de corrélation $\mathbf{R}$ et calculer son déterminant, ce qui a une complexité algorithmique en $\mathcal{O}\left(n^3\right)$, si n est le nombre de points observés. Cette étape est la plus longue dans la construction du modèle. Quelques travaux ont été proposés pour accélérer cette estimation comme dans [Sakata et al., 2004] où une nouvelle formule pour estimer les hyper-paramètres a été proposée. La méthode d'optimisation utilisée doit être définie judicieusement.

III.2.4.b. Optimisation de la fonction de vraisemblance

La fonction de vraisemblance définie dans la formule (III.27) est relativement difficile à minimiser. Elle est fréquemment multimodale. 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 modèle de krigeage. Une alternative simple consiste à supposer que la fonction objectif est isotrope et alors $\theta_1 = \cdots = \theta_d$, d'où une optimisation à une seule dimension.

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] programmé 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* utilise l'algorithme COBYLA (Constrained Optimization BY Linear Approximation) qui est d'ordre 0 (i.e. une méthode qui ne requiert pas le calcul du gradient), développé par Powell [Powell, 1994]. Il s'agit d'un algorithme d'optimisation séquentiel basé sur une région de confiance, dans laquelle une approximation linéaire de la fonction objectif est construite pour l'optimiser à chaque itération. David J.J. Toal, dans [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).

Dans ce rapport, nous utilisons la toolbox *Scikit-Learn* via l'algorithme COBYLA.

III.2.5. Récapitulatif du modèle de krigeage

Dans cette section, nous rappelons les principales étapes de la construction du modèle de krigeage (Cf. figure III.11) :

1. L'utilisateur fournit le plan d'expériences initial $\left(\mathbf{X}, \mathbf{y}\right)$ et le type du noyau k.

2. On suppose que les hyper-paramètres $\left(\theta, \beta, \sigma_z^2\right)$ du modèle de krigeage sont connus.

3. Sous les hypothèses du modèle de krigeage, on donne l'expression de la fonction de log-vraisemblance donnée par (III.22).

4. Sous l'hypothèse que les hyper-paramètres θ sont connus, on estime β et σ_z^2 qui sont donnés par les équations (III.23) et (III.24).

5. Après avoir injecté $\hat{\beta}$ and $\hat{\sigma}^2$ dans l'équation (III.22), on estime les hyper-paramètres θ en optimisant l'expression (III.27) via l'algorithme COBYLA ou d'autres algorithmes comme les algorithmes génétiques.

6. Enfin, on obtient l'expression de la prédiction du modèle de krigeage donnée par l'équation (III.13) et l'expression de l'erreur associée donnée par l'équation (III.14).

Dans la suite de ce chapitre, nous allons détailler notre approche pour accélérer l'optimisation des hyper-paramètres θ. Pour ce faire, nous allons réduire leur nombre via la méthode PLS. Pour davantage de clarté, nous allons donc commencer par décrire la méthode PLS, avant de développer la nouvelle approche KPLS.

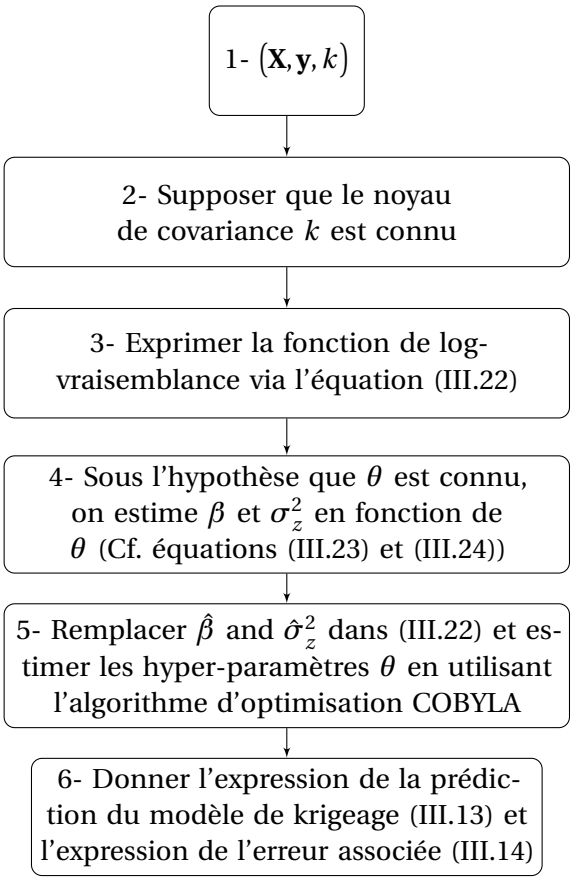


FIGURE III.11 – Les principales étapes nécessaires pour la construction du modèle de krigeage universel.

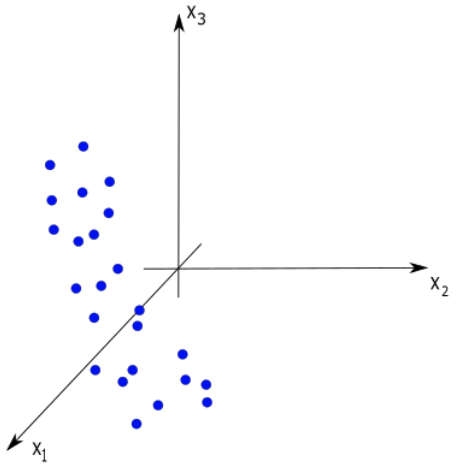


FIGURE III.12 – Données dans l'espace d'origine.

III.3. La régression PLS (Partial Least Squares regression)

La régression PLS a été inventée en 1983 par Herman Wold. Cette technique associe l'algorithme NIPALS (pour *Non linear Iterative PArtial Least Squares*), qui a été développé par [Wold, 1966] pour l'analyse en composantes principales, et l'approche PLS, qui a été proposée par le même auteur dans [Wold, 1975] pour l'estimation des modèles d'équations structurelles sur les variables latentes. De Jong propose une méthode alternative, SIMPLS (pour *straightforward implementation of a Statistically Inspired Modification of the PLS method*), décrite dans [De Jong, 1993] pour l'estimation des composantes de la régression. Pour plus de détails sur ces méthodes, le lecteur pourra se référer à l'ouvrage de [Tenenhaus, 1998].

La régression PLS consiste à projeter l'espace des données, généralement très grand, sur un espace latent de dimension réduite. Ce dernier est utilisé pour mieux expliquer la relation entre les entrées et les sorties, ce qui est difficile à détecter directement à partir de l'espace de départ (dû au grand nombre de variables). En pratique, le nombre de dimensions utilisé dans la construction des variables latentes (la base de l'espace latent) ne dépasse pas quatre ou cinq dimensions. La construction de l'espace latent est très proche de celle utilisée dans l'ACP (pour *Analyse en Composantes Principales*). Pour cette raison, nous commençons par présenter l'ACP [Tenenhaus, 1998].

III.3.1. Description de l'analyse en composantes principales

Pour décrire la construction itérative, on note désormais la matrice $\mathbf{X}$ par la matrice $\mathbf{X}^{(0)}$ et $\mathbf{y}$ par $\mathbf{y}^{(0)}$.

On commence par une standardisation des données en soustrayant la moyenne des données et en les normalisant. Les figure III.12, III.13 et III.14 permettent de visualiser ces étapes sur un exemple simple en dimension 3.

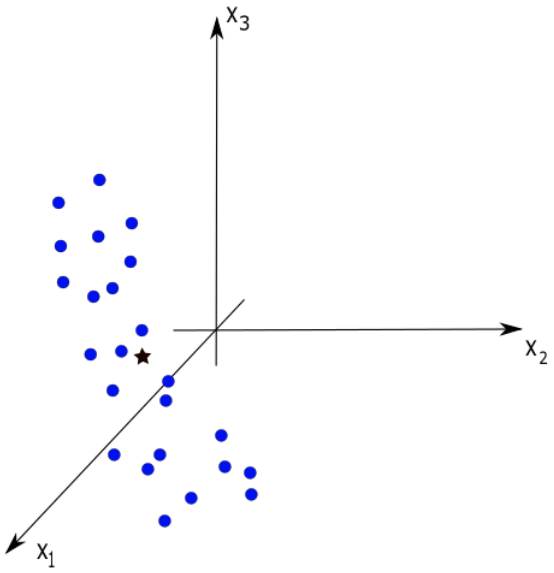


FIGURE III.13 – Calcul de la moyenne des données (point désigné par une étoile).

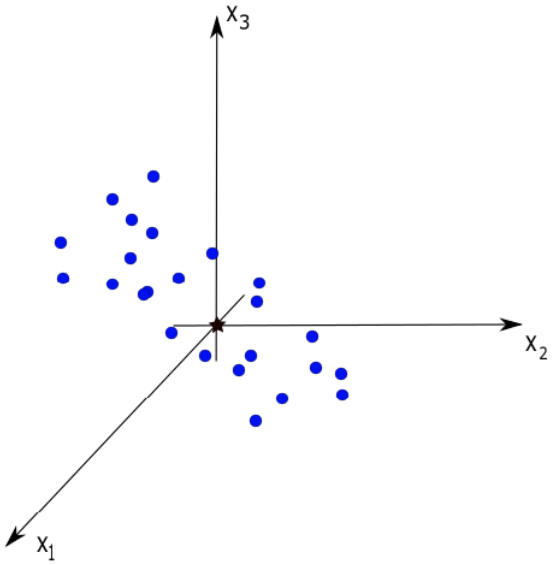


FIGURE III.14 – Centrer et normaliser les données.

Les conséquences immédiates de cette transformation sont :

— Le déplacement du repère des données.

— La considération des données sur la même échelle (même ordre de grandeur pour les données).

Une première droite passant par l'origine (la moyenne des points) est construite en approchant au mieux les points $\mathbf{X}^{(0)}$. Cette droite explique au mieux les observations avec un petit résidu. Ce résidu correspond à la distance entre les observations et la droite. Mathématiquement, le vecteur directeur de cette droite, noté $\mathbf{w}^{(1)}$ et de taille $d \times 1$ (Cf. figure III.15), suit la direction qui maximise la variance des projections des observations sur cette même droite, autrement dit :

$$\mathbf{w}^{(1)} = \begin{cases} \underset{\mathbf{w} \in \mathbb{R}^d}{\operatorname{argmax}} \ \mathbf{w}^t \mathbf{X}^{(0)^t} \mathbf{X}^{(0)} \mathbf{w} \\ \quad \texttt{s.c.} \ \mathbf{w}^t \mathbf{w} = 1. \end{cases} \tag{III.28}$$

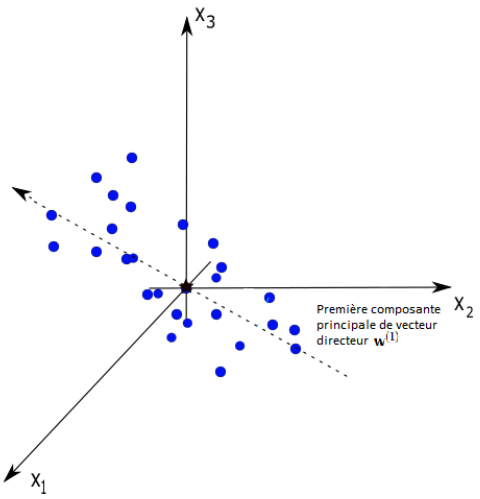


FIGURE III.15 – Construction de la première composante principale (ACP).

Ensuite, nous projetons les observations sur cette droite et nous calculons les distances entre l'origine et ces projections qui forment un premier vecteur $\mathbf{t}_1$ de taille $n \times 1$. Il faut noter que les distances algébriques sont considérées, i.e. qu'elles peuvent être positives ou négatives. Ainsi, nous obtenons la première composante principale, appelée aussi variable latente, qui se décompose en deux parties :

— une droite de vecteur directeur $\mathbf{w}^{(1)}$ de dimension $d \times 1$ passant par l'origine,

— un vecteur $\mathbf{t}_1$ de taille $n \times 1$ contenant les projections des observations sur cette droite.

La droite passant par l'origine de vecteur directeur $\mathbf{w}^{(1)}$ définit une nouvelle coordonnée dans l'espace de dimension d qui sera notée t_1. Par abus de langage, la coordonnée t_1 sera appelée "première composante principale" et représentative de la droite passant par l'origine et de vecteur directeur $\mathbf{w}^{(1)}$ (Cf. figure III.16).

Pour une meilleure compréhension géométrique de ce procédé, nous considérons le triangle (Cf. figure III.17) formé par l'origine, une observation $\mathbf{x}^{(i)}$ et l'axe t_1. Soit ρ l'angle formé par la droite

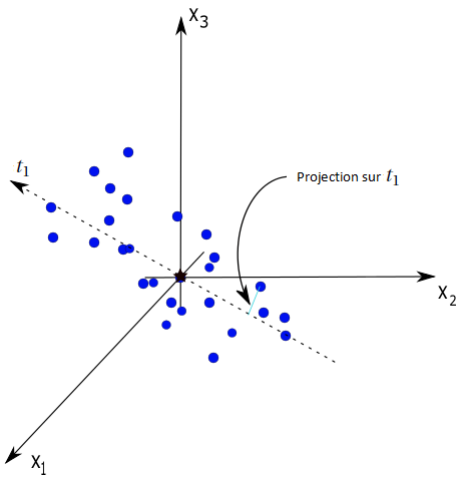


FIGURE III.16 – Notation de la première composante principale (ACP).

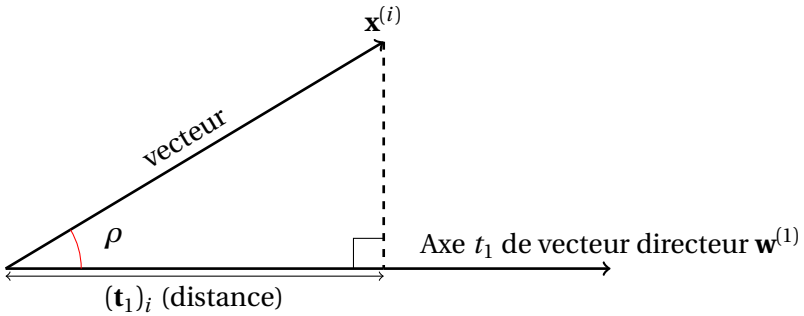


FIGURE III.17 – Triangle formé par l'origine, $\mathbf{x}^{(i)}$ et $(\mathbf{t}_1)_i$.

passant par l'origine et l'observation $\mathbf{x}^{(i)}$ et l'axe t_1. Nous avons, d'une part,

$$\cos(\rho) = \frac{\text{coté adjacent}}{\text{hypoténuse}} = \frac{(\mathbf{t}_1)_i}{||\mathbf{x}^{(i)}||_2},$$

où $(\mathbf{t}_1)_i$ est la distance entre l'origine et la projection de $\mathbf{x}^{(i)}$ sur t_1, et d'autre part,

$$\cos(\rho) = \frac{\mathbf{x}^{(i)^t}\mathbf{w}^{(1)}}{||\mathbf{x}^{(i)}||_2||\mathbf{w}^{(1)}||_2} \text{ (produit scalaire de deux vecteurs),}$$

avec $||\cdot||_2$ la norme euclidienne. Nous obtenons donc $(\mathbf{t}_1)_i = \mathbf{x}^{(i)^t}\mathbf{w}^{(1)}$ car $||\mathbf{w}^{(1)}||_2 = 1$ par définition. $(\mathbf{t}_1)_i$ est donc une combinaison linéaire de l'observation $\mathbf{x}^{(i)}$, i.e.,

$$(\mathbf{t}_1)_i = \mathbf{x}_1^{(i)}\mathbf{w}_1^{(1)} + \ldots + \mathbf{x}_d^{(i)}\mathbf{w}_d^{(1)}.$$

De la même manière, nous procédons pour les autres observations pour obtenir $\mathbf{t}_1 = \mathbf{X}^{(0)}\mathbf{w}^{(1)}$, avec $\mathbf{X}^{(0)}$ de taille $n \times d$ et $\mathbf{w}^{(1)}$ de taille $d \times 1$. Par conséquent, le fait de trouver la solution de l'équation (III.28), revient à maximiser la variance de $\mathbf{t}_1^t\mathbf{t}_1$.

Comme mentionné plus haut, notre but est d'expliquer au mieux les observations, mais en même temps, de minimiser la distance entre les observations et la droite passant par l'origine et de vecteur directeur $\mathbf{w}^{(1)}$. Par conséquent, une première régression ordinaire de $\mathbf{x}$ sur la première composante principale est construite :

$$\begin{aligned} \mathbf{X}^{(0)} &= \mathbf{X} \\ \mathbf{X}^{(0)} &= \mathbf{t}_1\mathbf{p}^{(1)} + \mathbf{X}^{(1)}, \end{aligned} \tag{III.29}$$

avec $\mathbf{p}^{(1)}$ le vecteur $(1 \times d)$ de poids attribués au vecteur $\mathbf{t}_1$ $(n \times 1)$ et $\mathbf{X}^{(1)}$ la matrice résidu $(n \times d)$. Cette régression (III.29) est écrite pour les n points de la base de données, nous avons donc un système matriciel.

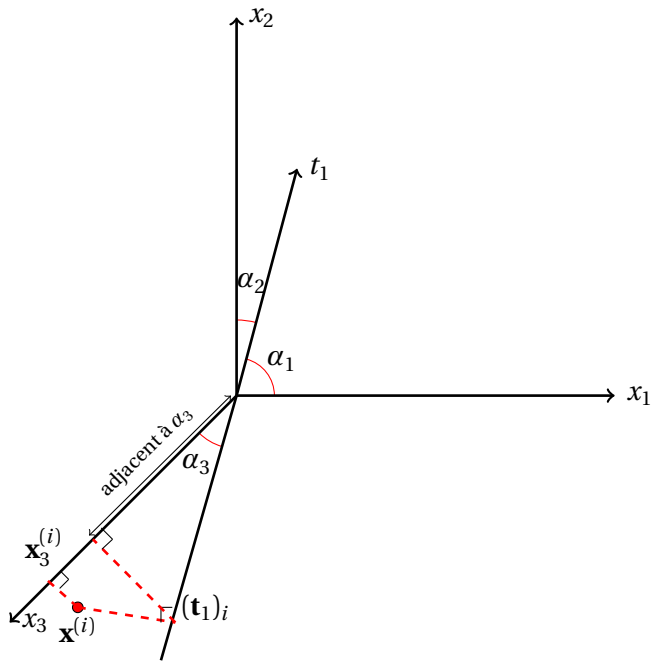


FIGURE III.18 – Explication géométrique des coefficients de pondération du vecteur $\mathbf{p}^{(1)}$ de la régression linéaire de $\mathbf{X}^{(0)}$ sur $\mathbf{t}_1$.

Les d coefficients donnés par le vecteur $\mathbf{p}^{(1)}$ correspondent respectivement au cosinus de l'angle formé par les axes de l'espace $\mathbf{x}$ avec la droite passant par l'origine et de vecteur directeur $\mathbf{w}^{(1)}$. Pour une meilleure compréhension géométrique, nous reprenons l'exemple à 3 dimensions traité ci-dessus, nous avons pour la $i^{ème}$ observation :

$$\mathbf{x}^{(i)} = \left[\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \mathbf{x}_3^{(i)}\right] = (\mathbf{t}_1)_i \left[\mathbf{p}_1^{(1)}, \mathbf{p}_2^{(1)}, \mathbf{p}_3^{(1)}\right] + \left[\mathbf{X}_{i1}^{(1)}, \mathbf{X}_{i2}^{(1)}, \mathbf{X}_{i3}^{(1)}\right].$$

En considérant seulement l'égalité de la troisième composante, on obtient :

$$\mathbf{x}_3^{(i)} = (\mathbf{t}_1)_i \mathbf{p}_3^{(1)} + \mathbf{X}_{i3}^{(1)}. \tag{III.30}$$

D'après la figure III.18, nous avons alors :

$$\cos(\alpha_3) = \frac{\text{adjacent à } \alpha_3}{\text{hypoténuse}} = \frac{\text{adjacent à } \alpha_3}{(\mathbf{t}_1)_i},$$

et donc :

$$\mathbf{x}_3^{(i)} = \text{adjacent à } \alpha_3 + \mathbf{X}_{i3}^{(1)}. \tag{III.31}$$

À partir des équations (III.30) et (III.31), on en déduit donc que $\mathbf{p}_3^{(1)} = \cos(\alpha_3)$. De la même manière, nous retrouvons $\mathbf{p}_1^{(1)} = \cos(\alpha_1)$ et $\mathbf{p}_2^{(1)} = \cos(\alpha_2)$.

On désigne désormais par h le nombre de composantes principales à construire, t_l $(l = 1, \ldots, h)$ la $l^{ième}$ composante principale et par $\mathbf{t}_l$ le vecteur contenant les valeurs de la projection de $\left(\mathbf{x}^{(i)}\right)_{i=1,\ldots,n}$ sur la $l^{ième}$ composante principale.

Une deuxième droite passant aussi par l'origine et perpendiculaire à la première composante principale est construite en approchant le résidu issu de la première régression ordinaire (toujours par la méthode de minimisation des moindres carrés). Une deuxième régression ordinaire de $\mathbf{X}^{(1)}$ sur $\mathbf{t}_2$ est construite :

$$\mathbf{X}^{(1)} = \mathbf{t}_2\mathbf{p}^{(2)} + \mathbf{X}^{(2)}. \tag{III.32}$$

Cette droite sert à tirer de l'information des points qui ont été mal approchés par la première droite, on obtient alors en combinant les équations (III.29) et (III.32) :

$$\mathbf{X}^{(0)} = \mathbf{t}_1\mathbf{p}^{(1)} + \mathbf{t}_2\mathbf{p}^{(2)} + \mathbf{X}^{(2)}. \tag{III.33}$$

La deuxième droite est la deuxième composante principale illustrée sur la figure III.19.

Ensuite, on réitère ce procédé jusqu'au nombre maximal de composantes principales voulues (généralement autour de 4 dimensions au maximum).

L'ACP analyse les données sans se préoccuper du résultat $\mathbf{y}$ de ses données d'entrée. Cette analyse peut être intéressante pour faire du clustering en grande dimension (apprentissage non supervisé).

III.3.2. Description de la méthode PLS

Nous avons constaté que l'ACP analyse les données sans donner d'importance aux résultats (valeurs de la sortie) des données d'entrée. Ce dernier point est la principale différence entre l'ACP et la PLS. La construction des composantes principales dans la régression PLS suit le même enchaînement que celui utilisé lors de l'ACP, à la différence que cette fois nous allons tenir compte de la régression ordinaire de la sortie sur la composante principale. Autrement dit, les composantes principales sont construites pour simultanément, et aussi bien que possible, décrire les composantes du vecteur $\mathbf{x}_j$

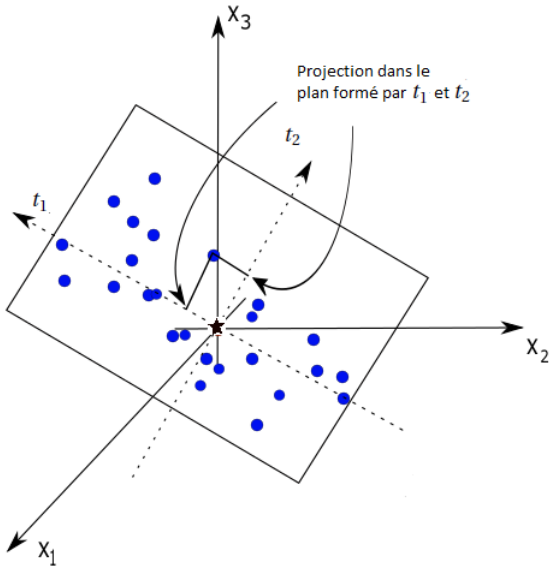


FIGURE III.19 – Construction de la deuxième composante principale (ACP).

($j = 1, \ldots, d$) et expliquer la sortie y. Notons que la priorité porte sur l'explication de la sortie puisque le nombre de composantes principales est fixé en fonction de la qualité de la prédiction de la sortie (par une validation croisée par exemple).

Remarque 6 *La démarche, qui est présentée dans la suite, décrit l'idée principale sur la construction de la PLS via l'algorithme NIPALS. Plusieurs variantes de la PLS existent dans la littérature, d'une part, au niveau de la dimension de l'espace des sorties comme la PLS1 dans le cas où il y a une seule sortie et la PLS2 dans le cas où il y a plusieurs sorties, et d'autre part, au niveau du critère à maximiser qui peut être différent d'une formulation à l'autre.*

III.3.2.a. Principe général de la PLS

Une normalisation des données est effectuée comme pour l'ACP dans la section III.3.1. (Cf. figure III.20).

Si cela n'est pas fait, les calculs peuvent être dominés (biaisés) par les variables qui ont de grandes valeurs et de fortes variations.

Une première droite passant par l'origine de l'espace des entrées est construite en approchant au mieux à la fois les entrées et la sortie $\mathbf{y}$ (Cf. figure III.21).

$$\begin{cases} \mathbf{X}^{(0)} & = & \mathbf{t}_1 \mathbf{p}^{(1)} + \mathbf{X}^{(1)} \\ \mathbf{y} = \mathbf{y}^{(0)} & = & c_1 \mathbf{t}_1 + \mathbf{y}^{(1)}, \end{cases} \tag{III.34}$$

où $c_1 \in \mathbb{R}$ est le coefficient de régression ordinaire de $\mathbf{y}^{(0)}$ sur $\mathbf{t}_1$ et $\mathbf{y}^{(1)} \in \mathbb{R}^n$ est le vecteur des résidus.

Une fois la première composante principale trouvée, nous construisons une deuxième composante

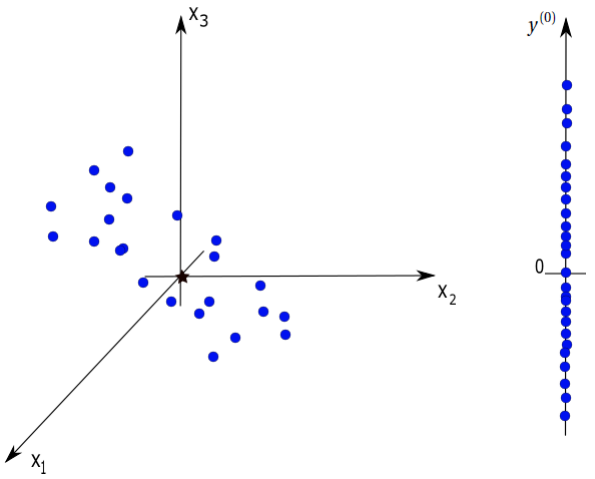


FIGURE III.20 – Centrer et normaliser les données.

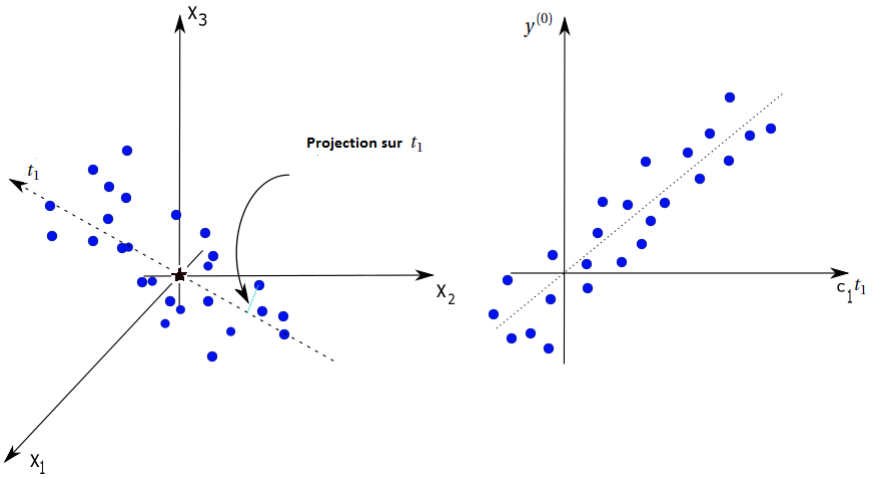


FIGURE III.21 – Première composante principale approchant simultanément au mieux la distribution des entrées et la sortie (PLS).

principale orthogonale à la première, en approchant au mieux, d'une part, le résidu $\mathbf{X}^{(1)}$ issu de la régression ordinaire de $\mathbf{X}^{(0)}$ sur $\mathbf{t}_1$, et d'autre part, le résidu $\mathbf{y}^{(1)}$ issu de la régression ordinaire de $\mathbf{y}^{(0)}$ sur $\mathbf{t}^{(1)}$ (Cf. figure III.22).

$$\begin{cases} \mathbf{X}^{(1)} &= \mathbf{t}_2\mathbf{p}^{(2)} + \mathbf{X}^{(2)} \\ \mathbf{y}^{(1)} &= c_2\mathbf{t}_2 + \mathbf{y}^{(2)}. \end{cases} \tag{III.35}$$

La construction de la deuxième composante principale nous donne une meilleure estimation de la

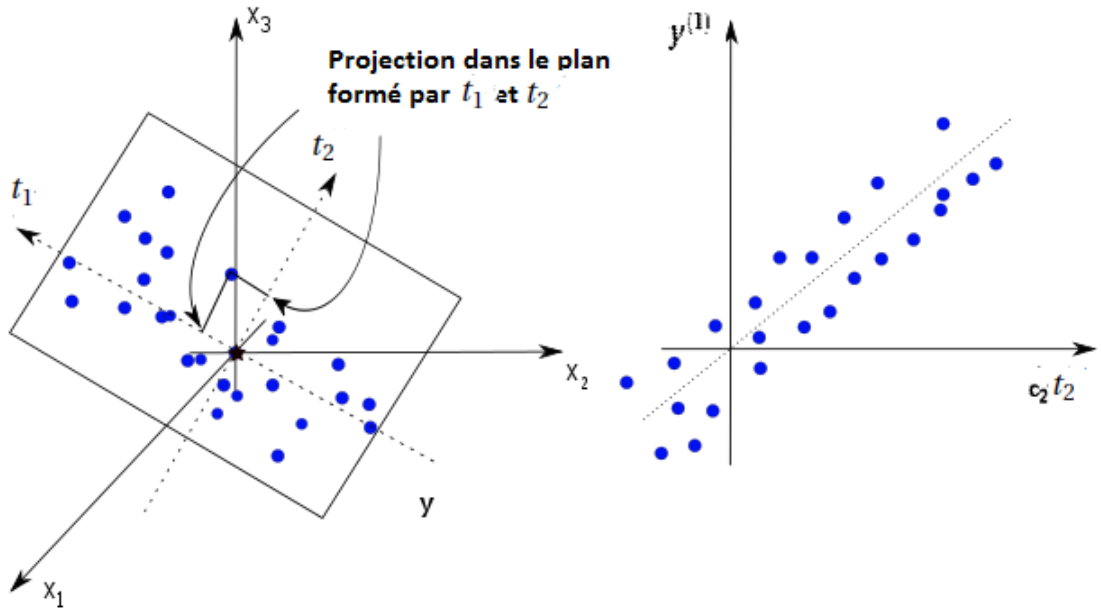


FIGURE III.22 – Deuxième composante principale orthogonale à la première composante principale (PLS).

prédiction (Cf. figure III.23).

$$\begin{cases} \mathbf{X} &= \mathbf{t}_1\mathbf{p}^{(1)} + \mathbf{t}_2\mathbf{p}^{(2)} + \mathbf{X}^{(2)} \\ \mathbf{y} &= c_1\mathbf{t}_1 + c_2\mathbf{t}_2 + \mathbf{y}^{(2)} \end{cases} . \tag{III.36}$$

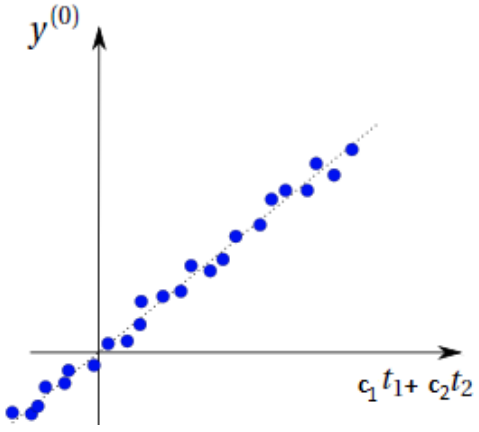


FIGURE III.23 – Estimation du résultat en fonction de deux composantes principales (PLS).

On réitère ce procédé jusqu'au nombre de composantes principales voulues.

III.3.2.b. Formulation mathématique de la PLS

La construction de la première composante principale est donnée par :

$$\mathbf{t}_1 = \mathbf{X}^{(0)}\mathbf{w}^{(1)} = \begin{bmatrix} \left(\mathbf{x}^{(0)}\right)_1^{(1)}\mathbf{w}_1^{(1)} + \cdots + \left(\mathbf{x}^{(0)}\right)_d^{(1)}\mathbf{w}_d^{(1)} \\ \vdots \\ \left(\mathbf{x}^{(0)}\right)_1^{(n)}\mathbf{w}_1^{(1)} + \cdots + \left(\mathbf{x}^{(0)}\right)_d^{(n)}\mathbf{w}_d^{(1)} \end{bmatrix}, \tag{III.37}$$

où $\left(\mathbf{x}^{(0)}\right)_j^{(i)} = \left(\mathbf{X}^{(0)}\right)_{ij}$, $\forall i = 1,\ldots,n$ et $j = 1,\ldots,d$ et

$$\mathbf{w}^{(1)} = \begin{cases} \underset{\mathbf{w}\in\mathbb{R}^d}{\operatorname{argmax}}\ \mathbf{w}^t\mathbf{X}^{(0)\,t}\mathbf{y}^{(0)}\mathbf{y}^{(0)\,t}\mathbf{X}^{(0)}\mathbf{w} \\ \texttt{s.c.}\ \mathbf{w}^t\mathbf{w} = 1 \end{cases}, \tag{III.38}$$

La solution de l'équation (III.38) correspond au vecteur propre de la matrice $\mathbf{M} = \mathbf{X}^{(0)\,t}\mathbf{y}^{(0)}\mathbf{y}^{(0)\,t}\mathbf{X}^{(0)}$ associé à la plus grande valeur propre. On peut résoudre l'équation (III.38) par la méthode de la puissance itérée [Lanczos, 1950] qui nous permet d'estimer la plus grande valeur propre de $\mathbf{M}$ très rapidement.

On obtient ensuite $\mathbf{p}^{(1)}$ par une régression ordinaire de $\mathbf{X}^{(0)}$ sur $\mathbf{t}_1$:

$$\mathbf{X}^{(0)} = \mathbf{t}_1\mathbf{p}^{(1)} + \mathbf{X}^{(1)}. \tag{III.39}$$

En parallèle, une régression ordinaire de $\mathbf{y}^{(0)}$ sur $\mathbf{t}_1$ est donnée par :

$$\mathbf{y}^{(0)} = c_1\mathbf{t}_1 + \mathbf{y}^{(1)}. \tag{III.40}$$

Si le pouvoir explicatif de cette régression est faible, on cherche à construire une deuxième composante, qui est une combinaison linéaire des vecteurs colonnes de $\mathbf{X}^{(1)}$, non corrélée à $\mathbf{t}_1$ (orthogonale à $\mathbf{t}_1$) et expliquant le résidu $\mathbf{y}^{(1)}$:

$$\mathbf{t}_2 = \mathbf{X}^{(1)}\mathbf{w}^{(2)} = \begin{bmatrix} \left(\mathbf{x}^{(1)}\right)_1^{(1)}\mathbf{w}_1^{(2)} + \cdots + \left(\mathbf{x}^{(1)}\right)_d^{(1)}\mathbf{w}_d^{(2)} \\ \vdots \\ \left(\mathbf{x}^{(1)}\right)_1^{(n)}\mathbf{w}_1^{(2)} + \cdots + \left(\mathbf{x}^{(1)}\right)_d^{(n)}\mathbf{w}_d^{(2)} \end{bmatrix}, \tag{III.41}$$

où

$$\mathbf{w}^{(2)} = \begin{cases} \underset{\mathbf{w}\in\mathbb{R}^d}{\operatorname{argmax}}\ \mathbf{w}^t\mathbf{X}^{(1)\,t}\mathbf{y}^{(1)}\mathbf{y}^{(1)\,t}\mathbf{X}^{(1)}\mathbf{w} \\ \texttt{s.c.}\ \mathbf{w}^t\mathbf{w} = 1. \end{cases} \tag{III.42}$$

On obtient donc :

$$\begin{cases} \mathbf{X}^{(1)} &= \mathbf{t}_2\mathbf{p}^{(2)} + \mathbf{X}^{(2)} \\ \mathbf{y}^{(1)} &= c_2\mathbf{t}_2 + \mathbf{y}^{(2)}. \end{cases} \tag{III.43}$$

On réitère cette procédure jusqu'à h composantes.

Les composantes principales retenues représentent une sélection de nouvelles coordonnées obtenues par une rotation du système de coordonnées $(x_1, \ldots, x_d)$. Autrement dit, pour $l = 1, \ldots, h$, $\mathbf{t}_l$ peut se réécrire comme (pour plus de détails, Cf. [Alberto and González, 2012, Tenenhaus, 1998]) :

$$\mathbf{t}_l = \mathbf{X}^{(l-1)}\mathbf{w}^{(l)} = \mathbf{X}\mathbf{w}_*^{(l)}. \tag{III.44}$$

La matrice $\mathbf{W}_* = \left[\mathbf{w}_*^{(1)}, \ldots, .\mathbf{w}_*^{(h)}\right]$ est obtenue par la relation suivante [Manne, 1987] :

$$\mathbf{W}_* = \mathbf{W}\left(\mathbf{P}^t\mathbf{W}\right)^{-1}, \tag{III.45}$$

avec $\mathbf{W} = \left[\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(h)}\right]$ et $\mathbf{P} = \left[\mathbf{p}^{(1)^t}, \ldots, \mathbf{p}^{(h)^t}\right]$.

Le vecteur $\mathbf{w}^{(l)}$ correspond à la $l^{ème}$ direction principale dans l'espace de $\mathbf{X}$ tout en maximisant la covariance de $\mathbf{X}^{(l-1)^t}\mathbf{y}^{(l-1)}\mathbf{y}^{(l-1)^t}\mathbf{X}^{(l-1)}$. Si $h = d$, $\mathbf{W}_* = \left[\mathbf{w}_*^{(1)}, \ldots, .\mathbf{w}_*^{(d)}\right]$ est la matrice de rotation du système de coordonnées de $(\mathbf{x}_1, \ldots, \mathbf{x}_d)$ vers le nouveau système de coordonnées $(t_1, \ldots, t_d)$ suivant respectivement les directions principales orthonormées $\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(d)}$.

La PLS et l'ACP permettent de réduire le nombre de dimensions de l'espace d'origine en faisant une sélection implicite des variables d'origine. Cette sélection est réalisée par une attribution d'un poids de pondération à chaque variable. De plus, la PLS nous fournit un modèle réduit linéaire dans l'espace réduit et également des informations intéressantes sur la relation entre les variables d'entrée et la variable de sortie, à travers l'équation (III.44). Pour cette raison, nous avons décidé d'exploiter et d'adapter ces informations dans la construction de modèles "hybrides" basés sur le modèle de krigeage.

III.4. La méthode KPLS

Nous avons vu dans la section III.2.4. que l'estimation des paramètres d'un modèle de krigeage peut être coûteuse en temps de calcul, notamment si la dimension du problème en entrée est importante, car le problème d'optimisation de la fonction de vraisemblance se fait dans un espace de dimension d. Nous avons également vu dans la section précédente III.3. comment la PLS a permis d'extraire des informations intéressantes sur la relation entre les variables d'entrée $\mathbf{X}^{(l-1)}$ et les variables de sortie $\mathbf{y}^{(l-1)}$, pour tout $l = 1, \ldots, h$.

Nous présentons dans cette partie une technique d'accélération du calcul combinant la PLS et le

modèle de krigeage.

III.4.1. Transformation dans le cas isotrope par une application linéaire

Commençons tout d'abord par donner la définition d'une application linéaire.

Définition Soit F une application de $B \subset \mathbb{R}^d$ dans $B' \subset \mathbb{R}^d$. L'application F est dite linéaire si :

— $\forall \mathbf{x}, \mathbf{x}' \in B, F(\mathbf{x}+\mathbf{x}') = F(\mathbf{x}) + F(\mathbf{x}')$;
— $\forall \mathbf{x} \in B,\ \forall \lambda \in \mathbb{R},\ F(\lambda \mathbf{x}) = \lambda F(\mathbf{x})$.

Pour accélérer l'estimation des hyper-paramètres θ d'une fonction de covariance k, il est naturel de réduire le nombre de paramètres à estimer, idéalement se contenter d'un seul paramètre en rendant le phénomène isotrope. Cependant, la majorité des cas physiques ne le sont pas et les variables de $\mathbf{x}$ n'ont pas toutes la même influence sur y. Si nous disposons de quelques informations sur la relation entre les variables d'entrée et la variable de sortie, nous pouvons modifier l'espace des variables par une application linéaire pour tenter de se rapprocher du cas isotrope. Dans ces travaux, nous proposons l'application linéaire suivante :

$$\begin{array}{rccl} F: & B & \longrightarrow & B' \\ & \mathbf{x} & \longmapsto & [\alpha_1 \mathbf{x}_1, \ldots, \alpha_d \mathbf{x}_d], \end{array} \tag{III.46}$$

avec $\alpha_1, \ldots, \alpha_d \in \mathbb{R}$ bien choisis.

La composition d'un noyau isotrope k avec l'application linéaire F (noyau isotrope où toutes les directions ont la même influence, autrement dit, $\theta = \theta_1 = \cdots = \theta_d$) sera notée $k_F(\mathbf{x}, \mathbf{x}') = k(F(\mathbf{x}), F(\mathbf{x}'))$ pour tout $\mathbf{x}, \mathbf{x}' \in B$ et k défini sur B' dans $\mathbb{R}$. Bien évidemment, le choix des paramètres $\alpha_1, \ldots, \alpha_d \in \mathbb{R}$ est primordial et doit représenter les vraies relations entre les entrées et la sortie. En effet, une variable $\mathbf{x}_i$ $(i = 1, \ldots, d)$ très influente sur y serait représentée par un poids α_i qui est fort (c'est à dire une grande valeur de $|\alpha_i|$). De ce fait, la distance suivant la $i^{ème}$ direction (la distance entre $\mathbf{x}_i$ et $\mathbf{x}'_i$) serait donnée par $|\alpha_i||\mathbf{x}_i - \mathbf{x}'_i|$ au lieu de $|\mathbf{x}_i - \mathbf{x}'_i|$. Nous augmentons donc la variation du modèle suivant la $i^{ème}$ direction et inversement pour un poids α_i faible.

III.4.2. Noyau du modèle KPLS

Les hyper-paramètres $\theta_1, \ldots, \theta_d$ (Cf. section III.2.3.b.) peuvent être interprétés comme la mesure du niveau d'activité de chaque variable $\mathbf{x}_1, \ldots, \mathbf{x}_d$ sur la sortie y. L'idée ici est d'intégrer les informations extraites par la méthode KPLS dans le modèle de krigeage, au niveau de son noyau, pour accélérer sa construction en réduisant le nombre d'hyper-paramètres, θ, à estimer. Le vecteur $\mathbf{w}_*^{(1)}$, défini par l'équation (III.45), est utilisé pour construire la première composante principale $\mathbf{t}_1 = \mathbf{X}\mathbf{w}_*^{(1)}$ où la covariance entre $\mathbf{t}_1$ et $\mathbf{y}$ est maximale. Les coefficients $\mathbf{w}_{*1}^{(1)}, \ldots, \mathbf{w}_{*d}^{(1)}$ peuvent être considérés comme une mesure de l'importance de $\mathbf{x}_1, \ldots, \mathbf{x}_d$ sur la construction de t_1, où la covariance de ce dernier est

maximale avec la sortie y. Ainsi, nous considérons l'application linéaire suivante :

$$\begin{array}{rccl} F_1: & B & \longrightarrow & B' \\ & \mathbf{x} & \longmapsto & \left[\mathbf{w}_{*1}^{(1)}\mathbf{x}_1, \ldots, \mathbf{w}_{*d}^{(1)}\mathbf{x}_d\right], \end{array} \tag{III.47}$$

ce qui nous permet de définir un nouveau noyau, appelé noyau KPLS_1 (l'indice 1 désigne la première itération dans la construction des composantes principales de la méthode PLS), donné par :

$$k\left(F_1(\mathbf{x}), F_1\left(\mathbf{x}'\right)\right) = k_{F_1}\left(\mathbf{x}, \mathbf{x}'\right);\ \mathbf{x}, \mathbf{x}' \in B, \tag{III.48}$$

où k est un noyau de covariance isotrope.

Les coefficients du vecteur $\mathbf{w}_*^{(1)}$ nous donnent une information sur la dépendance de $\mathbf{X}^{(0)}$ et $\mathbf{y}^{(0)}$ par l'intermédiaire de la première composante principale. Ces informations sont généralement insuffisantes pour expliquer la relation entre les variables d'entrée et la sortie. Pour cela, elles sont complétées par d'autres informations données par les coefficients des vecteurs $\mathbf{w}_*^{(l)}$, pour $l = 2, \ldots, h$. Pour synthétiser toutes ces informations dans un seul noyau, on commence par définir les noyaux KPLS_l par :

$$k\left(F_l(\mathbf{x}), F_l\left(\mathbf{x}'\right)\right) = k_{F_l}\left(\mathbf{x}, \mathbf{x}'\right);\ \mathbf{x}, \mathbf{x}' \in B. \tag{III.49}$$

Ensuite, on construit un nouveau noyau à partir du produit des h noyaux KPLS_l, appelé noyau $\text{KPLS}_{1:h}$, donné par (en se basant sur la propriété donnée par (III.17)) :

$$k_{1:h}(\mathbf{x}, \mathbf{x}') = \prod_{l=1}^{h} k_{F_l}\left(\mathbf{x}, \mathbf{x}'\right);\ \mathbf{x}, \mathbf{x}' \in B. \tag{III.50}$$

Un tel noyau nous permet de réduire les d paramètres θ à seulement h paramètres qui ne dépassent pas généralement 4, quelque soit le nombre de dimensions de notre problème initial (avec $d \geq h$).

III.4.3. Construction du modèle de krigeage *versus* le modèle KPLS

La figure III.24 nous montre comment les informations extraites par la méthode PLS sont injectées dans le modèle de krigeage.

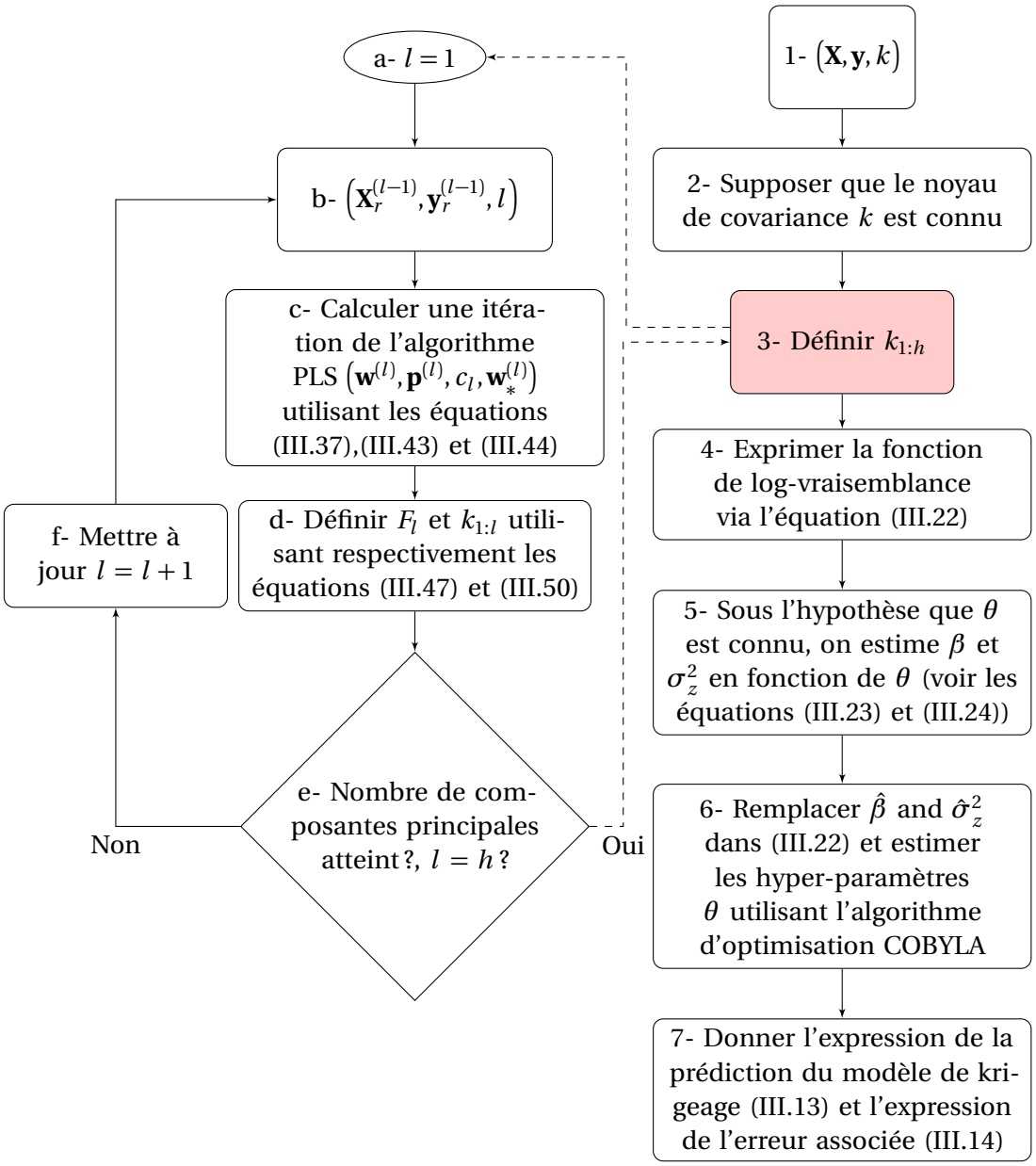


FIGURE III.24 – Les principales étapes lors de la construction du modèle KPLS.
La seule différence avec le modèle de krigeage classique est donnée à l'étape 3 (la boîte rouge).

Toutes les étapes de construction du modèle KPLS sont similaires aux étapes de construction du modèle de krigeage classique, i.e. nous avons les mêmes définitions et équations. En comparant les figures III.11 et III.24, la seule différence réside au niveau de la troisième étape de la figure III.24, qui est représentée par une boîte rouge. Cette étape utilise l'algorithme PLS pour définir le nouveau noyau $k_{1:h}$ donné par l'équation (III.50) :

a. On initialise l'algorithme PLS avec $l = 1$.
b. Si $l \neq 1$ alors on calcule les résidus $\mathbf{X}_r^{(l-1)}$ et $\mathbf{y}_r^{(l-1)}$.
c. On calcule le vecteur $\mathbf{w}_*^{(l)}$ correspondant à la $l^{ème}$ itération PLS.
d. On définit le noyau $k_{1:l}$ utilisant l'expression (III.50), des exemples sont fournis à la section III.4.4..
e. On vérifie si le nombre de composantes principales est atteint. Si oui ($l = h$), on retourne à l'étape 3, sinon ($l \neq h$)
f. On passe à l'itération suivante en considérant $l = l + 1$.

III.4.4. Exemples de noyaux KPLS

Dans cette section, nous reprenons les noyaux donnés dans la table III.2 et nous les adaptons aux noyaux KPLS (Cf. table III.3).

TABLE III.3 – Exemples de noyaux de covariance de type KPLS. La fonction de covariance est exprimée en fonction du $i^{ème}$ élément $\mathbf{m}_i^{(l)} = \left|\mathbf{w}_{*i}^{(l)}(\mathbf{x}_i - \mathbf{x'}_i)\right|$ avec $\theta_l \geq 0$ et $p_l \in [0,2]$, $\forall l = 1,\ldots,h$.

fonction de covariance	expression	hyper-paramètres θ	nombre d'hyper-paramètres
Exponentielle généralisée	$\sigma^2 \prod_{l=1}^{h}\prod_{i=1}^{d} \exp\left(-\theta_l \left(\mathbf{m}_i^{(l)}\right)^{p_l}\right)$	$(\theta_1,\ldots,\theta_h,p_1,\ldots,p_h)$	$2h << 2d$
Gaussienne	$\sigma^2 \prod_{l=1}^{h}\prod_{i=1}^{d} \exp\left(-\theta_l \left(\mathbf{m}_i^{(l)}\right)^2\right)$	$(\theta_1,\ldots,\theta_h)$	$h << d$
Matern $\frac{5}{2}$	$\sigma^2 \prod_{l=1}^{h}\prod_{i=1}^{d} \left(1+\sqrt{5}\theta_l \mathbf{m}_i^{(l)} + \frac{5}{3}\theta_l^2 \left(\mathbf{m}_i^{(l)}\right)^2\right) \exp\left(-\sqrt{5}\theta_l \mathbf{m}_i^{(l)}\right)$	$(\theta_1,\ldots,\theta_h)$	$h << d$
Matern $\frac{3}{2}$	$\sigma^2 \prod_{l=1}^{h}\prod_{i=1}^{d} \left(1+\sqrt{3}\theta_l \mathbf{m}_i^{(l)}\right) \exp\left(-\sqrt{3}\theta_l \mathbf{m}_i^{(l)}\right)$	$(\theta_1,\ldots,\theta_h)$	$h << d$

III.5. Étude numérique sur le modèle KPLS

La réduction du nombre d'hyper-paramètres du modèle de krigeage, en s'appuyant sur la technique PLS, permet d'accélérer la construction du modèle. Une telle accélération provient de l'optimisation de la fonction de vraisemblance dans un espace réduit à h dimensions, au lieu de l'espace de départ à d dimensions. Cependant, le modèle KPLS doit être de bonne qualité afin d'être utilisable. Pour

vérifier sa performance, nous allons le comparer avec quelques métamodèles de la littérature sur des cas académiques et industriels.
Nous avons choisi les modèles de krigeage ordinaire et de RBF comme candidats pour cette comparaison, et ceci pour différentes raisons :

— Une comparaison avec le modèle de krigeage nous permettra de mesurer le gain, au niveau du temps de calcul (temps CPU) et de la qualité du modèle (utilisant une mesure d'erreur), apporté par le modèle KPLS.

— Le modèle RBF est souvent utilisé pour les problèmes d'optimisation en grande dimension [Chen et al., 2012, Cheng et al., 2015, Regis, 2014a, Regis, 2014b, Regis and Shoemaker, 2007].

Dans cette section, nous comparons le modèle KPLS avec le modèle de krigeage ordinaire sur des cas académiques et industriels (deux exemples Snecma), respectivement dans les sous-sections III.5.1. et III.5.2.. Nous terminons, dans la sous-section III.5.3., par une comparaison entre le modèle KPLS avec le modèle RBF sur un cas test issu du milieu de l'automobile connu pour son grand nombre de variables, qui est appelé MOPTA.

III.5.1. Tests expérimentaux sur des fonctions académiques

Dans cette section, les tests expérimentaux sont réalisés sur des fonctions académiques, qui sont :

— La fonction $y_{g_{07}}$ [Michalewicz and Schoenauer, 1996].

— La fonction de Griewank [Regis and Shoemaker, 2013a].

Pour cette dernière, nous faisons varier le nombre de dimensions.

Le comportement et les variations de ces deux fonctions permettent de couvrir les principales difficultés, généralement rencontrées dans le domaine de la métamodélisation. Tous les calculs dans ce chapitre ont été réalisés avec la machine Intel(R) Celeron(R) CPU 900 @2.20GHz desktop machine. Pour construire les métamodèles sur ces fonctions, un plan d'expériences de type hypercube latin optimisé est construit dans $[0,1]^d$, qui est transformé par la suite dans l'espace des variables par la formule

$$\mathbf{x}_i(\mathbf{x}_{i,max} - \mathbf{x}_{i,min}) + \mathbf{x}_{i,min}, \tag{III.51}$$

avec $\mathbf{x}_{i,min}$ et $\mathbf{x}_{i,max}$, respectivement, les bornes inférieure et supérieure de la variable $\mathbf{x}_i$. La dernière transformation est réalisée afin d'effectuer le calcul des sorties du plan d'expériences. Ensuite, 5000 points sont choisis aléatoirement dans le domaine de définition et un calcul d'erreur relative (en pourcentage %) est effectué par la formule :

$$ER = \frac{||\hat{\mathbf{y}} - \mathbf{y}_{test}||_2}{||\mathbf{y}_{test}||_2} 100, \tag{III.52}$$

avec $||.||_2$ la norme euclidienne, $\mathbf{y}_{test}$ le vecteur contenant les vraies sorties des points tests et $\hat{\mathbf{y}}$ le vecteur contenant les prédictions des points tests. Nous nous intéressons aussi au temps de calcul nécessaire pour construire les métamodèles (temps CPU). Ces deux critères comparatifs sont utilisés

également pour les tests industriels Snecma.

Remarque 7 *Le critère d'erreur* (III.52) *(comme tous les critères d'erreur dans la littérature) nous donne une indication sur la qualité globale du métamodèle, il se peut qu'un métamodèle possède une bonne qualité globale mais une qualité moins bonne localement. Il faut noter que, dans une stratégie d'optimisation auto-adaptative via les métamodèles, le métamodèle doit bien approcher les tendances (les principales variations) de la vraie fonction lors des premières itérations de l'optimisation. Dans ces travaux, nous nous intéressons aux métamodèles de bonne qualité dans les zones prometteuses (zones susceptibles de contenir les optima du problème) notamment pendant les dernières itérations de l'optimisation, ce qui est l'intérêt du chapitre suivant. En somme, nous considérons que l'erreur relative donnée par l'équation* (III.52) *est une métrique suffisante pour nos travaux.*

Enfin, la toolbox Python "Scikit-learn v.014" [Pedregosa et al., 2011] est utilisée pour réaliser les tests proposés. Cette toolbox permet de construire le modèle de krigeage ordinaire et aussi de construire les composantes principales de la PLS. Pour le modèle KPLS, nous combinons le modèle de krigeage ordinaire (utilisant un noyau gaussien) avec la technique PLS (en faisant varier les composantes principales de 1 à 3) tous deux implémentés dans cette toolbox.

Remarque 8 *Dans toute la suite, un modèle KPLS à h composantes principales sera noté KPLSh.*

III.5.1.a. Comparaison sur la fonction g_{07}

La fonction g_{07} est souvent utilisée dans les problèmes d'optimisation du fait de son comportement proche des cas industriels [Hock and Schittkowski, 1981, Mezura-Montes and Cetina-Domínguez, 2012, Michalewicz and Schoenauer, 1996, Regis, 2014a, Regis, 2014b]. Elle est donnée par :

$$\begin{aligned} y_{g_{07}}(\mathbf{x}) \quad = \quad & \mathbf{x}_1^2+\mathbf{x}_2^2+\mathbf{x}_1\mathbf{x}_2-14\mathbf{x}_1-16\mathbf{x}_2 \\ & +(\mathbf{x}_3-10)^2+4(\mathbf{x}_4-5)^2+(\mathbf{x}_5-3)^2 \\ & +2(\mathbf{x}_6-1)^2+5\mathbf{x}_7^2+7(\mathbf{x}_8-11)^2 \\ & +2(\mathbf{x}_9-10)^2+(\mathbf{x}_{10}-7)^2+45, \\ & \\ & -10\leq \mathbf{x}_i \leq 10, \text{ pour } i=1,...,10. \end{aligned}$$

Remarque 9 *Pour ce cas, nous avons considéré seulement la fonction objectif du problème d'optimisation appelé g_{07} et donné par l'équation* (A.13) *dans l'annexe.*

Pour cette fonction à 10 dimensions, 100 points sont utilisés pour construire le plan d'expériences. L'erreur et le temps d'exécution sont donnés par la table III.4.

TABLE III.4 – Erreur de prédiction de la fonction g_{07} en dimension 10, avec un plan d'expériences de 100 points. Pour le temps de construction du métamodèle, on note "h" pour les heures, "m" pour les minutes et "s" pour les secondes.

Métamodèles	ER (%)	temps CPU
Krigeage ordinaire	0.013	5.14 s
KPLS1	0.014	0.11 s
KPLS2	0.0015	0.43 s
KPLS3	0.0008	0.44 s

Lorsque le nombre de composantes principales est supérieur à 1, nous observons dans la table III.4 que la qualité du modèle KPLS est meilleure que celle du modèle de krigeage ordinaire. Néanmoins, le modèle KPLS1 est quasiment de même qualité que le modèle de krigeage ordinaire. Dans ce cas, un seul paramètre θ estimé de la fonction de covariance est suffisant, en comparant aux 10 paramètres estimés pour le modèle de krigeage ordinaire. Intuitivement, le modèle de krigeage devait être de meilleure qualité que les modèles KPLS dans tous les cas, car il estime les hyper-paramètres dans un espace plus grand et contenant l'espace réduit utilisé par les modèles KPLS. Malgré le fait que tous les modèles utilisent le même optimiseur pour estimer les hyper-paramètres, ces résultats montrent que les hyper-paramètres du krigeage ordinaire sont mal estimés par rapport aux modèles KPLS à cause de la multimodalité de la fonction de vraisemblance et de la dimension du problème considéré.

D'autre part, nous observons une construction très rapide des modèles KPLS par rapport au modèle de krigeage, par exemple : le modèle de krigeage est presque 12 fois plus lent que le KPLS2 (5.14 s pour le premier contre 0.43 s pour le second).

Même si la méthode PLS ne traite que les dépendances linéaires entre les variables d'entrée et de sortie, nous avons montré avec cet exemple que la méthode KPLS peut bien approcher les fonctions non linéaires, grâce aux propriétés interpolantes du modèle de krigeage. Le modèle KPLS est donc un bon candidat pour approcher cette fonction de dimension 10. Dans la section suivante, nous faisons varier la dimension pour la fonction de Griewank.

III.5.1.b. Comparaison sur la fonction Griewank dans l'intervalle [-600,600]

Nous étudions ici la fonction de Griewank en faisant varier le nombre de dimensions (2, 5, 7, 10, 20 et 60), ce qui nous permettra de vérifier la performance de notre approche sur une grande plage de variation pour la dimension d. La définition de cette fonction est donnée par :

$$\begin{aligned} y_{\text{Griewank}}(\mathbf{x}) &= \sum_{i=1}^{d} \frac{\mathbf{x}_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{\mathbf{x}_i}{\sqrt{i}}\right) + 1, \\ & -600 \leq \mathbf{x}_i \leq 600, \text{ pour } i = 1, \ldots, d. \end{aligned}$$

Cette fonction est illustrée dans la figure III.25 pour le cas à deux dimensions.

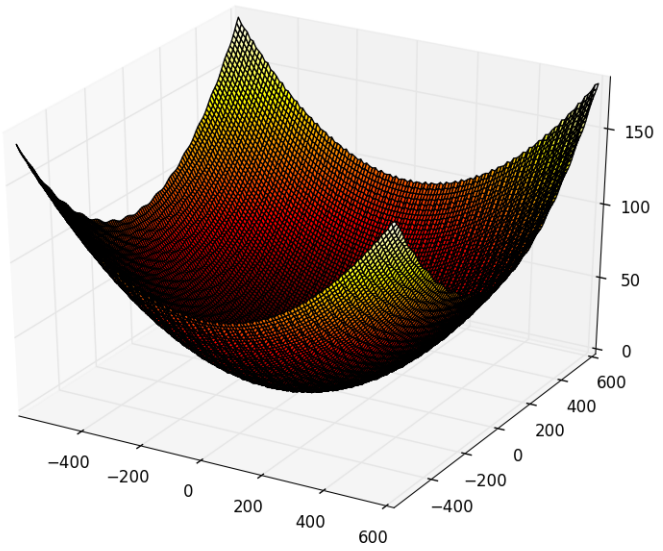


FIGURE III.25 – Fonction de Griewank en dimension 2 sur $[-600, 600]$.

Un plan d'expériences de type hypercube latin optimisé est construit avec un nombre de n points, comme mentionné dans la table III.5.

TABLE III.5 – Nombre de points utilisés dans le plan d'expériences pour la fonction de Griewank.

$d=2$	$d=5$	$d=7$	$d=10$	$d=20$	$d=60$
$n=70$	$n=100$	$n=200$	$n=300$	$n=400$	$n=800$
$\frac{n}{d}=35$	$\frac{n}{d}=20$	$\frac{n}{d}\approx 28.57$	$\frac{n}{d}=30$	$\frac{n}{d}=20$	$\frac{n}{d}\approx 13.33$

Les résultats pour $d=2$ sont donnés dans la table III.6. Si deux composantes principales sont utilisées dans ce cas, il est tout à fait logique de retrouver la même qualité du modèle et le même temps CPU que le modèle de krigeage ordinaire. En effet, il s'agit d'une simple transformation du système de coordonnées de l'espace d'origine. Cependant, l'utilisation d'une seule composante principale pour ce problème à deux dimensions dégrade la qualité de la solution.

TABLE III.6 – Erreur de prédiction de la fonction de Griewank en dimension 2, avec un plan d'expériences de 70 points.

métamodèles	ER (%)	temps CPU
Krigeage ordinaire	5.50	0.09 s
KPLS1	7.23	0.04 s
KPLS2	5.50	0.10 s

Dans les tables III.7, III.8 et III.9, nous augmentons le nombre de dimensions, respectivement, avec 5, 7 et 10 dimensions. Nous observons que l'augmentation du nombre de composantes principales n'a pas beaucoup d'impact sur la qualité du modèle KPLS, mais nous obtenons un gain important au niveau du temps CPU. Pour ces trois exemples, l'utilisation d'une seule composante principale suffit pour obtenir une bonne qualité du modèle KPLS, avec un temps de calcul divisé par 35 par rapport au modèle de krigeage ordinaire, pour 10 dimensions par exemple (21 secondes pour le krigeage ordinaire et 0.6 secondes pour le KPLS1).

TABLE III.7 – Erreur de prédiction de la fonction de Griewank en dimension 5, avec un plan d'expériences de 100 points.

métamodèles	ER (%)	temps CPU
Krigeage ordinaire	0.605	0.55 s
KPLS1	0.635	0.12 s
KPLS2	0.621	0.31 s
KPLS3	0.623	0.51 s

TABLE III.8 – Erreur de prédiction de la fonction de Griewank en dimension 7, avec un plan d'expériences de 200 points.

métamodèles	ER (%)	temps CPU
Krigeage ordinaire	0.138	3.09 s
KPLS1	0.141	0.25 s
KPLS2	0.138	0.52 s
KPLS3	0.141	0.94 s

TABLE III.9 – Erreur de prédiction de la fonction de Griewank en dimension 10, avec un plan d'expériences de 300 points.

métamodèles	ER (%)	temps CPU
Krigeage ordinaire	0.052	21 s
KPLS1	0.033	0.6 s
KPLS2	0.035	2.41 s
KPLS3	0.034	3.58 s

Pour l'exemple à 20 dimensions, donné par la table III.10, l'utilisation d'une seule composante principale n'est pas du tout suffisante. En effet, l'erreur relative ER est très mauvaise (10.15 %) lorsque nous la comparons avec les autres métamodèles. Dans ce cas, au moins deux composantes principales sont nécessaires pour obtenir une qualité meilleure que le modèle de krigeage ordinaire. De plus, le temps CPU du modèle KPLS2 (11.7 s) est bien meilleur que celui donné par krigeage ordinaire (1 mn 47 s).

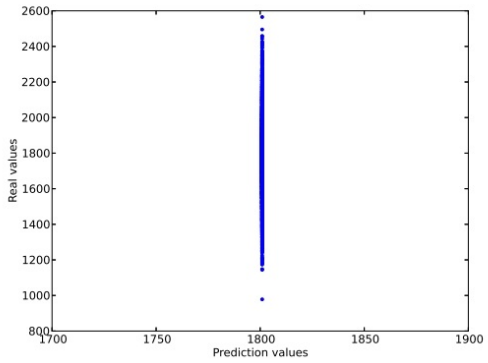
TABLE III.10 – Erreur de prédiction de la fonction de Griewank en dimension 20, avec un plan d'expériences de 400 points.

métamodèles	ER (%)	temps CPU
Krigeage ordinaire	0.35	1 mn 47 s
KPLS1	10.15	1.16 s
KPLS2	0.003	11.7 s
KPLS3	0.002	16.23 s

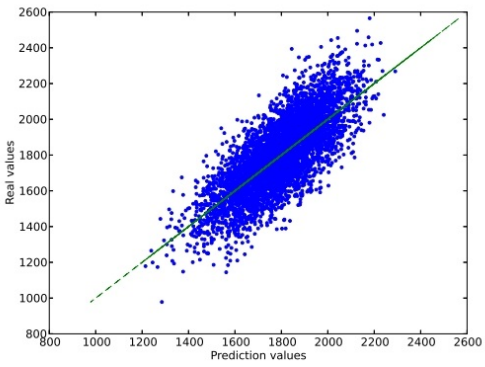
Dans la table III.11 pour 60 dimensions, le modèle KPLS est beaucoup plus rapide que le modèle de krigeage ordinaire. Par exemple, le temps CPU est réduit par un facteur de 42 pour le modèle KPLS1 et d'un facteur de plus de 17 pour le modèle KPLS3 comparé au modèle de krigeage ordinaire. Dans ce cas, le modèle de krigeage ordinaire échoue dans l'optimisation de la fonction de vraisemblance donnée par l'équation (III.27). En effet, l'expression $\mathbf{r}_{\mathbf{x}}^t \mathbf{R}^{-1}(\mathbf{y}-\mathbf{1}\hat{\mu})$ de l'équation (III.15) donne toujours 0 et la prédiction est égale à la partie déterministe $\hat{\mu}$ quelque soit le point de l'espace (hormis les points d'apprentissage), comme le montre la figure III.26a. Les figures III.26b, III.26c and III.26d montrent l'efficacité des modèles KPLS par rapport au modèle de krigeage ordinaire sur ce cas à 60 dimensions.

TABLE III.11 – Erreur de prédiction de la fonction de Griewank en dimension 60, avec un plan d'expériences de 800 points.

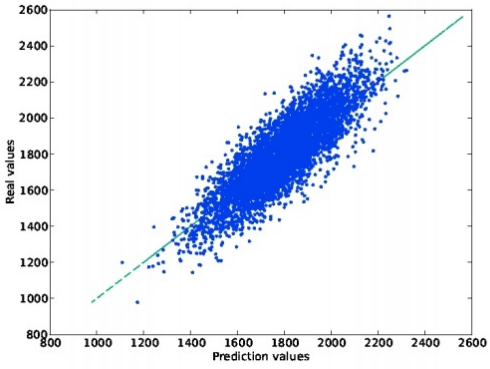
métamodèles	ER (%)	temps CPU
Krigeage ordinaire	11.47	4 mn 53 s
KPLS1	7.4	6.88 s
KPLS2	6.04	12.57 s
KPLS3	5.23	16.82 s



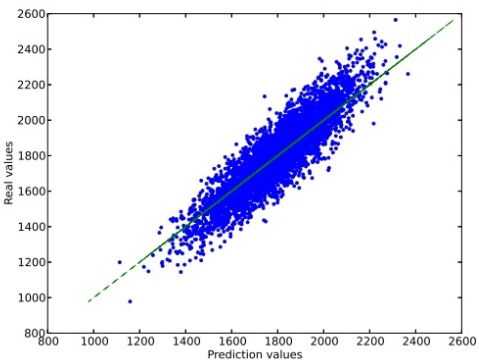
(a) Le krigeage ordinaire.



(b) KPLS1.



(c) KPLS2.



(d) KPLS3.

FIGURE III.26 – La fonction de Griewank en 60 dimensions. Les valeurs de la prédiction $\hat{\mathbf{y}}$ sont données sur l'axe des abscisses et les vraies valeurs $\mathbf{y}$ sont données sur l'axe des ordonnées. Une bonne prédiction est observée lorsque les points sont positionnés sur la diagonale, i.e. la droite d'équation $y = x$.

III.5.1.c. Conclusion des cas tests académiques

La majorité des résultats, pour les modèles KPLS, obtenus sur les cas académiques sont meilleurs que le modèle de krigeage ordinaire, notamment en grande dimension. Ce sont des résultats intéressants, en particulier pour le gain observé au niveau du temps CPU. Il faut noter aussi que nous devons utiliser les modèles KPLS avec précaution pour les problèmes de petite dimension, comme le démontre le cas de Griewank à deux dimensions. Nous avons aussi remarqué que l'augmentation du nombre de composantes principales ne signifie pas forcément une amélioration sur la qualité du modèle KPLS, comme le montrent les tests sur la fonction de Griewank en 5, 7 et 10 dimensions. Pour cela, une étape de validation sur le nombre de composantes principales nous paraît plus que nécessaire.

III.5.2. Tests expérimentaux sur des tables Snecma

Les exemples proposés dans cette section sont basés sur des résultats stockés, dans la base de données Snecma, issus d'une ancienne conception d'aube (Cf. section VI.2. pour plus de détails sur la conception). Deux tests sont traités dans cette étude :

— 1295 points dans le plan d'expériences et 500 points de validation sont disponibles pour la première table de données. Ce sont des points calculés avec un simulateur peu coûteux, ce qui explique le grand nombre de points disponibles. Ce problème, noté pbme_1, est de dimension 10 avec une seule sortie.

— 99 points dans le plan d'expériences et 52 points de validation sont disponibles pour la deuxième table de données. Ce sont des points calculés avec un simulateur plus coûteux que le premier. Ce problème, noté pbme_2, est de dimension 24 avec quatre sorties notées y_1, y_2, y_3 et y_4.

Pour réaliser notre étude, nous avons choisi de comparer le modèle KPLS avec le krigeage ordinaire implémenté sous la plateforme *OPTIMUS*, qui est une version industrialisée et optimisée. Un noyau gaussien est utilisé pour tous les modèles. Nous utilisons les deux critères "ER" et le temps CPU pour comparer les deux types de métamodèles. Les résultats de cette comparaison sont donnés dans les tables III.12 et III.13.

III.5.2.a. Comparaison sur le problème pbme_1

TABLE III.12 – Erreur de prédiction de la sortie du problème pbme_1 à 10 dimensions, avec un plan d'expériences de 1295 points.

	métamodèles	ER(%)	temps CPU
pbme_1	Krigeage	5.37	1 h 30 mn
	KPLS1	5.07	11.69 s
	KPLS2	5.02	1 mn 22 s
	KPLS3	5.34	7 mn 34 s

D'après la table III.12, la qualité du modèle KPLS est meilleure que le modèle de krigeage ordinaire dans tous les cas (de 1 à 3 composantes principales), malgré le fait que la qualité du modèle KPLS se dégrade légèrement lorsque nous passons de 2 à 3 composantes principales (de 5.02% à 5.34 %). Cependant, le gain au niveau du temps CPU est beaucoup plus marquant pour ce cas test, par comparaison, nous obtenons environ un temps CPU de 11 s pour le KPLS1 alors que le modèle de krigeage ordinaire nécessite 1 h 30 min, i.e. le temps CPU est divisé par un facteur 700 entre la construction du modèle KPLS1 et celle du modèle de krigeage ordinaire.

III.5.2.b. Comparaison sur le problème pbme_2

Pour ce problème à 4 sorties, nous avons dû réaliser 4 modèles de krigeage associés respectivement à chacune des 4 sorties y_1, y_2, y_3 et y_4. Les résultats du problème pbme_2, donnés par la table III.13,

sont quasi-similaires aux résultats des exemples traités précédemment. En effet, nous constatons un gain au niveau du temps de construction du modèle KPLS par rapport au krigeage ordinaire pour toutes les sorties. Mais, nous remarquons aussi, pour la sortie y_3, une dégradation de la qualité du modèle lorsque nous utilisons le modèle KPLS quelque soit le nombre de composantes principales.

TABLE III.13 – Erreur de prédiction de y_1, y_2, y_3 et y_4 du problème pbme_2 à 24 dimensions, avec un plan d'expériences de 99 points.

	métamodèles	y_1		y_2		y_3		y_4	
		$ER(\%)$	temps CPU	$ER(\%)$	temps CPU	$ER(\%)$	temps CPU	$ER(\%)$	temps CPU
pbme_2	Krigeage	0.082	8 s	4.45	8.4 s	8.97	8.17 s	6.27	8.12 s
	KPLS1	0.079	0.12 s	4.04	0.11 s	10.35	0.18 s	5.67	0.11 s
	KPLS2	0.079	0.43 s	4.06	0.69 s	10.33	0.42 s	5.67	0.19 s
	KPLS3	0.079	0.82 s	4.05	0.5 s	10.41	1.14 s	5.67	0.43 s

III.5.2.c. Conclusion des cas tests Snecma

Le modèle KPLS est efficace sur les deux exemples industriels de la conception d'aubes. Nous avons obtenu une erreur inférieure pour les modèles KPLS, à l'exception de la sortie y_3 qui est mieux approchée par le modèle krigeage ordinaire. Dans la section V.1.4., nous allons voir comment améliorer la qualité de la sortie y_3. Concernant le gain au niveau du temps CPU, nous confirmons les conclusions données lors des cas académiques traitées dans la section III.5.1.. Donc, le modèle KPLS est visiblement bien adapté aux problématiques Snecma.

III.5.3. Tests expérimentaux sur le cas MOPTA à 124 variables

MOPTA est un cas test industriel issu de l'industrie automobile [Jones, 2008] (General motors, Cf. figure III.27). Il s'agit d'une problématique de réduction de masse d'un véhicule sous contraintes de crash, de réduction de vibration, et de résistance mécanique. Mathématiquement, il s'agit de métamodèles de krigeage de divers codes coûteux à évaluer (1-3 jours) comportant 124 variables d'entrée normalisées, 68 contraintes elles aussi normalisées et un objectif. Un exécutable calculant les sorties du cas MOPTA est disponible à l'adresse "http ://anjos.mgi.polymtl.ca/MOPTA2008Benchmark.html".

Remarque 10 *L'information sur le type de noyau utilisé pour reproduire les sorties du problème MOPTA n'est pas disponible.*

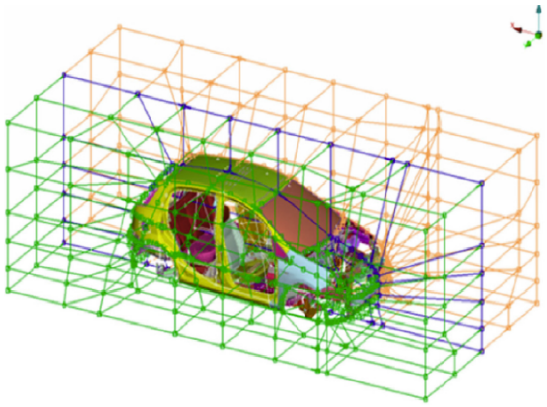


FIGURE III.27 – Cas test MOPTA issu de l'industrie automobile (General Motors).

Pour cet exemple, nous avons choisi de comparer le modèle KPLS3 (le choix de 3 composantes principales est réalisé après une validation en amont sur le nombre de composantes principales) avec un modèle RBF (Cf. section II.1.3. pour le modèle RBF), qui est souvent utilisé dans les problèmes d'optimisation de grande dimension, comme nous l'avons déjà mentionné au début de ce chapitre. Un noyau cubique avec une tendance donnée par un polynôme linéaire est utilisé pour le modèle RBF. Cependant, un noyau gaussien est utilisé pour les modèles KPLS

Remarque 11 *Ce travail a été réalisé dans le cadre d'une collaboration avec le professeur R. Régis de l'université "Saint Joseph's University" située à Philadelphie aux États-Unis. Nous avons laissé toute la liberté à R. Régis de choisir le paramétrage de son modèle RBF.*

Pour cette comparaison, 500 points dans le plan d'expériences et 100 points de validation sont utilisés. Les résultats de cette comparaison sont donnés dans la table III.14, où nous utilisons le critère d'erreur ER donné par l'équation (III.52).

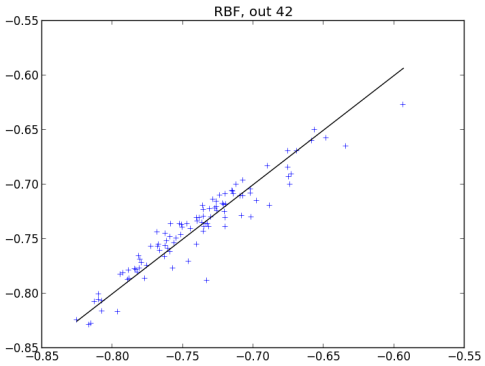
Remarque 12 *Nous n'avons pas retranscrit le temps CPU pour ce cas test, car les calculs n'étaient pas réalisés sur la même machine.*

TABLE III.14 – Calcul d'erreur relative (Cf. équation (III.52)) des 69 sorties du problème MOPTA, avec 500 points d'apprentissage et 100 points de validation.

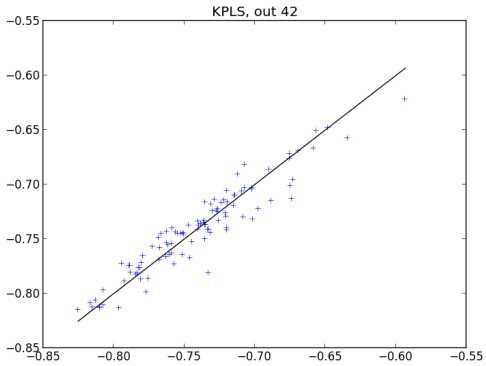
sorties	métamodèles		sorties	métamodèles		sorties	métamodèles	
	KPLS3	RBF		KPLS3	RBF		KPLS3	RBF
y_1	$2.23.10^{-6}$	$1.03.10^{-5}$	y_{24}	3.46	7.35	y_{47}	18.81	59.73
y_2	27.64	32.59	y_{25}	16.53	52.77	y_{48}	0.96	1.24
y_3	17.24	23.03	y_{26}	1.48	13.22	y_{49}	16.07	39.03
y_4	53.70	55.64	y_{27}	5.9	16.86	y_{50}	0.51	0.65
y_5	25.32	44.03	y_{28}	1.55	13.14	y_{51}	13.52	51.22
y_6	27.27	49.92	y_{29}	14.06	46.55	y_{52}	0.84	1.07
y_7	30.25	52.56	y_{30}	7.32	10.13	y_{53}	19.46	63.18
y_8	14.81	16.80	y_{31}	17.33	54.71	y_{54}	0.73	0.90
y_9	18.24	28.22	y_{32}	11.17	12.39	y_{55}	3.75	30.94
y_{10}	10.56	14.29	y_{33}	15.99	15.53	y_{56}	0.30	0.35
y_{11}	4.6	10.25	y_{34}	17.64	33.55	y_{57}	7.94	16.90
y_{12}	4.89	13.43	y_{35}	15.63	55.83	y_{58}	12.47	20.95
y_{13}	4.28	9.40	y_{36}	19.84	29.61	y_{59}	25.16	33.56
y_{14}	6.74	19.63	y_{37}	23.75	66.53	y_{60}	8.48	17.19
y_{15}	6.67	16.06	y_{38}	0.65	0.85	y_{61}	3.30	2.67
y_{16}	15.69	28.25	y_{39}	11.91	44.77	y_{62}	15.69	27.00
y_{17}	14.26	23.58	y_{40}	0.29	0.36	y_{63}	12.41	13.12
y_{18}	1.96	5.13	y_{41}	27.11	64.19	y_{64}	31.04	54.60
y_{19}	1.53	3.60	y_{42}	1.87	1.81	y_{65}	2.08	3.25
y_{20}	2.87	4.04	y_{43}	2.97	25.03	y_{66}	10.13	15.12
y_{21}	12.71	19.77	y_{44}	6.57	14.24	y_{67}	8.05	12.42
y_{22}	1.36	2.77	y_{45}	2.88	25.44	y_{68}	41.84	64.43
y_{23}	14.06	34.32	y_{46}	6.58	13.28	y_{69}	31.71	51.67

III.5.3.a. Comparaison sur le problème MOPTA

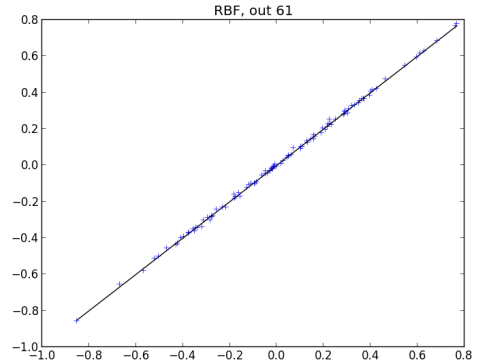
D'après les résultats donnés par la table III.14, le modèle KPLS3 est de meilleure qualité que le modèle RBF sur la totalité des sorties, à l'exception des sorties 42 et 61 avec moins de 1% d'écart. Nous avons choisi de représenter ces sorties sur la figure III.28.



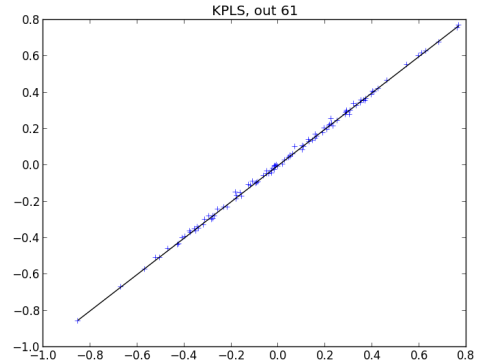
(a) Le modèle RBF pour la sortie 42.



(b) Le modèle KPLS3 pour la sortie 42.



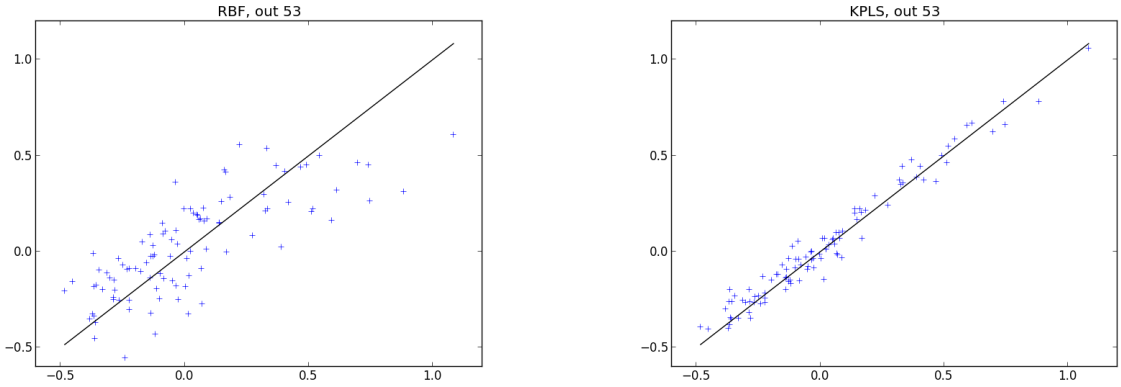
(c) Le modèle RBF pour la sortie 61.



(d) Le modèle KPLS3 pour la sortie 61.

FIGURE III.28 – Les sorties 42 et 61 du problème MOPTA. Les valeurs de la prédiction $\hat{\mathbf{y}}$ sont données sur l'axe des abscisses et les vraies valeurs $\mathbf{y}$ sont données sur l'axe des ordonnées. Une bonne prédiction est observée lorsque les points sont positionnés sur la diagonale, i.e. la droite d'équation $y = x$.

De plus, nous remarquons que le modèle KPLS3 est meilleur d'au moins de 10% (au niveau de l'erreur relative) que le modèle RBF, par exemple une erreur relative de 19.46% contre une erreur de 63.18% pour la sortie 53 (Cf. figure III.29 pour la représentation de cette sortie).



(a) Le modèle RBF pour la sortie 53.

(b) Le modèle KPLS3 pour la sortie 53.

FIGURE III.29 – La sortie 53 du problème MOPTA. Les valeurs de la prédiction $\hat{\mathbf{y}}$ sont données sur l'axe des abscisses et les vraies valeurs $\mathbf{y}$ sont données sur l'axe des ordonnées. Une bonne prédiction est observée lorsque les points sont positionnés sur la diagonale, i.e. la droite d'équation $y = x$.

III.5.3.b. Conclusion du cas test MOPTA

Comme pour la comparaison avec le modèle de krigeage ordinaire (les sections III.5.1.a. et III.5.1.b.), le modèle KPLS est plus performant que le modèle RBF sur ce cas MOPTA. Nous validons donc la fiabilité du modèle KPLS sur les problèmes de grande dimension (jusqu'à 124 variables). Nous allons maintenant vérifier la performance du modèle KPLS dans une stratégie d'optimisation afin de pouvoir répondre à la problématique industrielle.

III.6. Conclusion

Dans ce chapitre, nous avons repris les travaux de stage sur l'optimisation des plans d'expériences de type hypercube latin, avec une comparaison expérimentale entre les plans d'expériences de type hypercube latin classiques et ceux optimisés par l'algorithme ESE. Nous avons ensuite rappelé les théories des modèles de krigeage et de la méthode PLS pour décrire la méthode KPLS. Enfin, nous avons démontré d'un point de vue numérique, l'efficacité des modèles KPLS sur plusieurs cas académiques et industriels. D'autres tests sur des problèmes d'aubages au sein de Snecma ont été réalisés et ont montré la supériorité des modèles KPLS, en comparaison avec des modèles de krigeage et de RBF. En effet, outre la bonne qualité des modèles KPLS sur les cas tests, un gain considérable en temps de calcul est observé sur tous les exemples présentés.

La performance des modèles KPLS est satisfaisante pour envisager leur exploitation en contexte industriel pour de la prédiction de réponses aéro-mécaniques d'aubages de turbomachines, lorsque :

— ces réponses sont nombreuses ; nous devons construire un métamodèle par réponse,

— la dimension du problème est grande ou très grande.

IV Couplage du modèle KPLS avec l'algorithme d'optimisation SEGO

Objectifs du chapitre

- Coupler la méthode d'optimisation EGO avec les modèles KPLS,
- Valider la nouvelle stratégie d'optimisation SEGOKPLS sur des cas tests académiques.

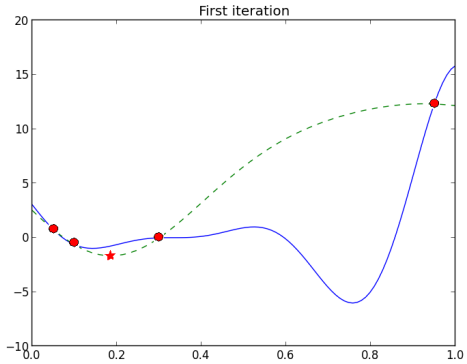
Nous avons décrit dans le chapitre précédent la construction du modèle KPLS et nous avons vérifié sa performance sur plusieurs exemples. Nous allons maintenant intégrer ce nouveau métamodèle dans l'algorithme d'optimisation EGO [Jones et al., 1998]. Ce chapitre se découpe en deux sections : nous commençons par rappeler la théorie de la méthode EGO que nous couplons aux modèles KPLS dans la section IV.1.. Ainsi, nous décrivons dans un premier temps les techniques de la méthode pour les problèmes d'optimisation sans contrainte puis avec contraintes. Nous intégrons dans un second temps les modèles KPLS à la méthode dans la section IV.1.2.. Enfin, nous appliquons la méthode d'optimisation EGO couplée aux modèles KPLS sur plusieurs exemples académiques dans la section IV.2..

IV.1. Optimisation via l'algorithme EGO sans et avec contraintes à base du modèle KPLS

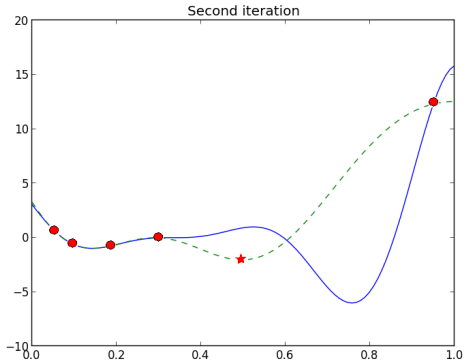
IV.1.1. Principe de fonctionnement de la méthode EGO

IV.1.1.a. Généralités

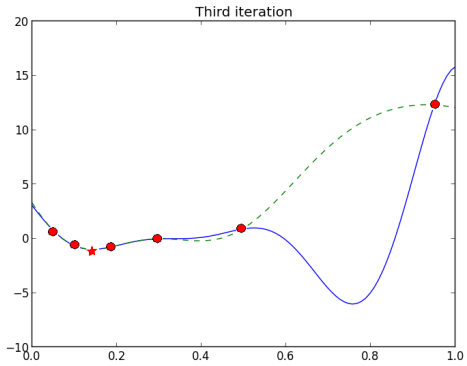
La manière la plus simple d'utiliser une surface de réponse dans une stratégie d'optimisation est d'optimiser directement la prédiction du métamodèle et d'enrichir à l'optimum. Cependant, la convergence vers un optimum global n'est pas toujours garantie, comme le montre la figure IV.1 sur un exemple mono-dimensionnel. En effet, le métamodèle approche mal la fonction dans la zone autour de l'optimum global car les points d'apprentissage sont choisis volontairement autour de l'optimum local. Par conséquent, l'enrichissement se focalise dans cette zone car elle correspond au minimum du métamodèle ; le métamodèle sera de bonne qualité autour de l'optimum local et de qualité insuffisante autour de l'optimum global.



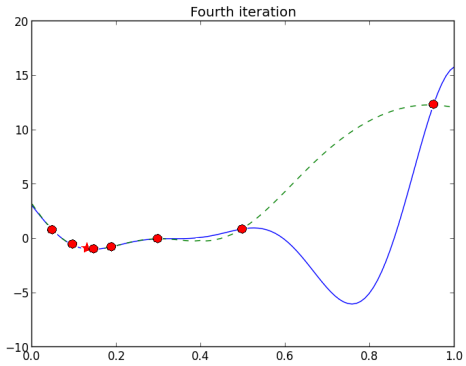
(a) Itération 1



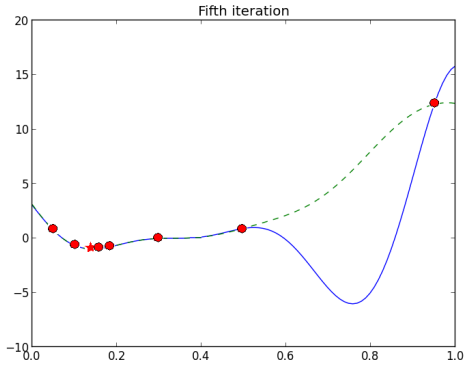
(b) Itération 2



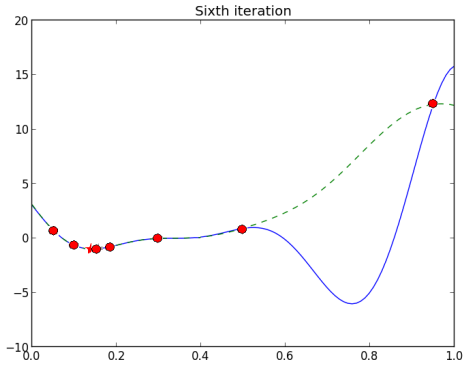
(c) Itération 3



(d) Itération 4



(e) Itération 5



(f) Itération 6

FIGURE IV.1 – Minimisation de la fonction $f(x) = (6x-2)^2 \sin(12x-4)$ directement sur le métamodèle. L'algorithme converge en 6 itérations vers un optimum local autour de 0.17. La vraie fonction f est représentée en trait plein, l'approximation de f est représentée en pointillé, les 4 points d'apprentissage sont représentés par des points rouges et le point d'enrichissement par une étoile rouge.

Pour remédier à ce problème, nous devons prendre en compte l'incertitude du métamodèle au cours de l'optimisation, et échantillonner dans les zones où la qualité du métamodèle est insuffisante afin d'explorer plus largement le domaine. Autrement dit, nous devons trouver un certain équilibre entre l'exploitation et l'exploration des points (Cf. [Forrester et al., 2008]).

Par ailleurs, l'incertitude explicite du modèle de krigeage (donnée par l'équation (III.14)) est un atout majeur, car elle permet de transformer notre problème déterministe en un problème probabiliste, et ceci en reprenant l'hypothèse que $y(\mathbf{x})$ est la réalisation d'une loi normale Y, de moyenne $\hat{y}(\mathbf{x})$ et de variance $s^2(\mathbf{x})$ données, respectivement, par les équations (III.13) et (III.14). L'algorithme EGO est un algorithme d'optimisation basé sur le modèle de krigeage (une réalisation d'une loi normale). Il est basé aussi sur un critère d'enrichissement appelé critère "*d'amélioration espérée*", noté EI pour "*Expected Improvement*" suivant l'appellation anglophone. Ce critère définit les zones intéressantes comme un compromis entre la valeur du meilleur prédicteur et la valeur de l'incertitude associée au modèle. Il permet donc d'éviter une convergence prématurée vers un optimum local. Pour chaque itération d'optimisation, nous cherchons le point $\mathbf{x} \in B$ maximisant le critère d'enrichissement EI. La formulation mathématique de ce critère est donnée dans la section suivante.

IV.1.1.b. Description du critère d'enrichissement EI

Nous définissons l'*amélioration* d'un point $\mathbf{x} \in B$ par :

$$I(\mathbf{x}) = \max\{0, f_{min} - Y(\mathbf{x})\}, \tag{IV.1}$$

où $Y(\mathbf{x}) \sim \mathcal{N}(\hat{y}(\mathbf{x}), s^2(\mathbf{x}))$ et f_{min} est la valeur minimale connue de $\mathbf{y}$. $I(\mathbf{x})$ est une variable aléatoire car $Y(\mathbf{x})$ l'est aussi et donc on peut calculer son espérance. L'*amélioration espérée* (Cf. [Forrester et al., 2008, Jones et al., 1998]) est donc donnée par :

$$EI(\mathbf{x}) = E[I(\mathbf{x})] = E[\max\{f_{min} - Y(\mathbf{x}), 0\}]. \tag{IV.2}$$

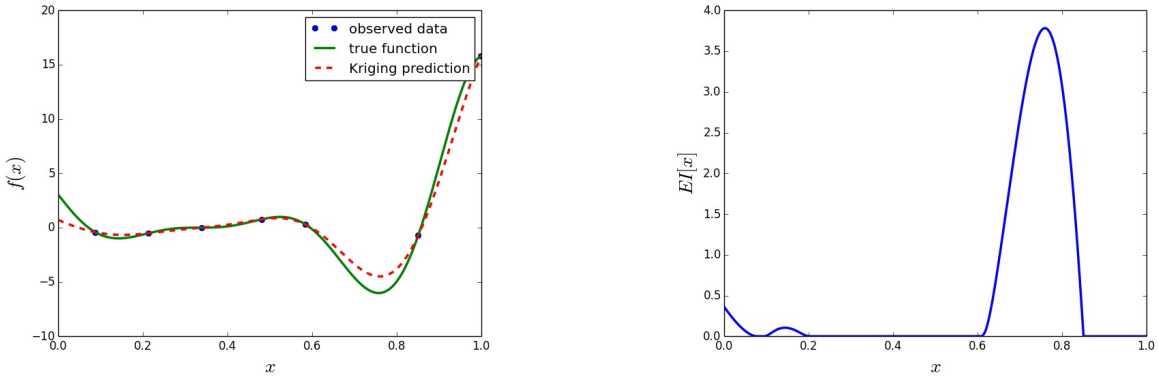
En exprimant le terme de droite de l'équation (IV.2) comme une intégrale, et en faisant une intégration par partie, nous obtenons une expression analytique du critère EI donnée par :

$$EI(\mathbf{x}) = \begin{cases} \left(f_{min} - \hat{y}(\mathbf{x})\right)\Phi\left(\frac{f_{min}-\hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) + s(\mathbf{x})\phi\left(\frac{f_{min}-\hat{y}(\mathbf{x})}{s(\mathbf{x})}\right), & \text{si } s > 0 \\ 0, & \text{si } s = 0, \end{cases} \tag{IV.3}$$

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

Le premier terme de l'équation (IV.3) est grand lorsque $\hat{y}$ est plus petit que f_{min}. Le second terme est grand lorsque nous avons une grande incertitude sur $\hat{y}$. Par conséquent, le critère EI est grand pour les régions avec une amélioration de f_{min} et/ou pour les régions avec une grande incertitude. Les points prometteurs sont donc donnés en maximisant le critère EI.

Soit la fonction définie par $f(\mathbf{x}) = (6x-2)^2 \sin(12x-4)$ pour $x \in [0,1]$ traitée précédemment. La figure IV.2a représente une approximation de la fonction f (vraie fonction en trait plein) utilisant 7 points dans le plan d'expériences initial. Le critère d'enrichissement EI de cette fonction pour la première itération est représenté dans la figure IV.2b.



(a) Approximation de la fonction f.

(b) Le critère EI.

FIGURE IV.2 – Première itération de l'algorithme EGO. À gauche : représentation de la fonction $f(x) = (6x-2)^2 \sin(12x-4),\ x \in [0,1]$. À droite : représentation du critère EI de la fonction $f(\mathbf{x})$ pour $x \in [0,1]$.

L'EI définit 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. Le critère est nul aux points d'observation et la convergence est assurée. La figure IV.3 montre le comportement de l'algorithme sur la fonction f définie ci-dessus. 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 à partir de la cinquième itération et à trouver la solution du problème à la huitième itération.

Remarque 13 *Le critère EI peut être calculé à l'aide d'une estimation de la variance du modèle de krigeage via la technique de rééchantillonnage [Kleijnen et al., 2012], ou, en anglais, "bootstrap" qui désigne un ensemble de méthodes qui consistent à faire de l'inférence statistique sur de nouveaux échantillons tirés à partir d'un échantillon initial. Néanmoins, cette technique requiert un grand nombre de tirages pour minimiser le risque d'erreur sur l'estimation de la variance du modèle de krigeage, c'est pourquoi, elle peut être coûteuse et donc à éviter dans notre problème.*

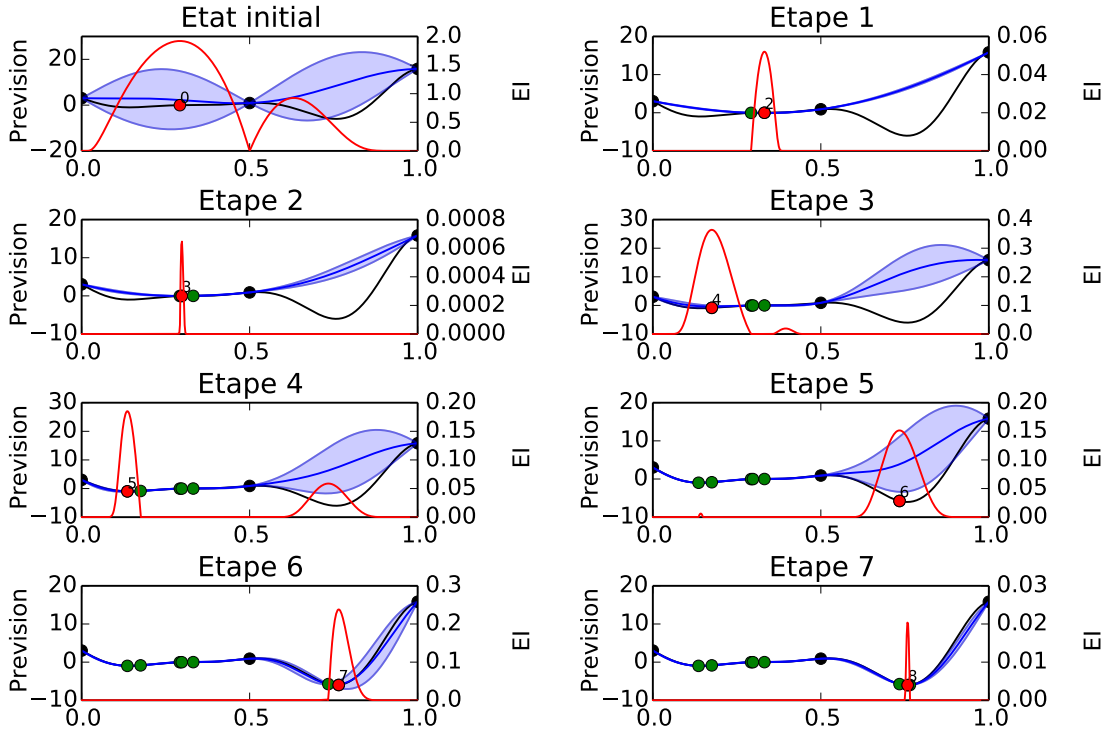


FIGURE IV.3 – Déroulement de l'algorithme EGO pour la fonction f bimodale. En noir, la fonction objectif f, en bleu, le métamodèle avec intervalles de confiance à 95% et en rouge la valeur de l'EI associée.

Remarque 14 *Il est possible d'enrichir le métamodèle avec plusieurs points simultanément. D. Ginsbourger, dans [Ginsbourger and Le Riche, 2009], démontre d'ailleurs que l'enrichissement point par point est une méthode sous-optimale. D'autre part, la résolution du problème consistant à déterminer les points qui maximisent l'E I est relativement complexe. F. Viana, dans [Viana et al., 2013], définit l'algorithme MSEGO (Multiple Surrogate EGO) qui consiste plus simplement à construire différents métamodèles (différentes hypothèses de krigeage, réseaux de neurones utilisant des RBF, splines, etc...) et maximiser l'E I pour chacun d'entre eux (l'incertitude fournit par le modèle de krigeage est utilisée dans tous les cas). Cette démarche comporte toutefois le risque d'échantillonner dans les mêmes régions. Dans ces travaux, l'enrichissement est réalisé point par point.*

Le critère EI est globalement de bonne qualité pour les petites dimensions ; tant que $d < 15$, l'algorithme EGO est assez performant et trouve la solution du problème en un nombre raisonnable d'itérations. 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. De nombreux auteurs modifient l'EI selon le problème. M. Sasena, dans [Sasena, 2002], fournit un panel de critères et détaille lesquels

sont les mieux adaptés pour une situation donnée.

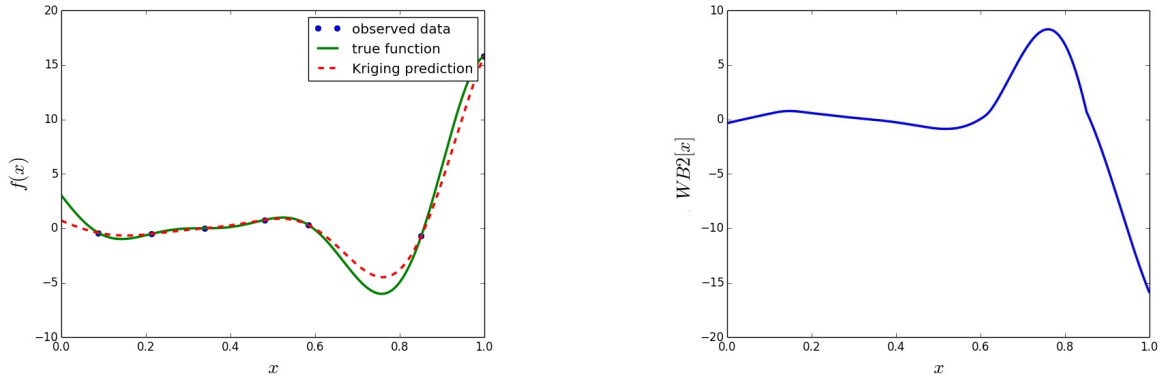
Pour améliorer les performances de l'algorithme EI en grande dimension, on peut pénaliser l'EI de façon à forcer l'intensification dans les zones où la fonction est susceptible de contenir son minimum. M. Sasena, dans [Sasena, 2002], propose simplement d'utiliser le critère appelé "locating the regional extreme" décrit ci-dessous. Nous utilisons la même abréviation $wb2$ que dans [Sasena, 2002].

IV.1.1.c. Description du critère d'enrichissement $wb2$

Le critère $wb2$ correspond essentiellement à la somme de l'opposé de l'approximation de la fonction objectif et du critère EI. Pour cela, il est légèrement plus local que le critère EI. Il est défini par :

$$wb2(\mathbf{x}) = \begin{cases} -\hat{y}(\mathbf{x}) + \left(f_{min} - \hat{y}(\mathbf{x})\right)\Phi\left(\frac{f_{min}-\hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) + s(\mathbf{x})\phi\left(\frac{f_{min}-\hat{y}(\mathbf{x})}{s(\mathbf{x})}\right), & \text{si } s > 0 \\ 0, & \text{si } s = 0. \end{cases} \tag{IV.4}$$

Le comportement du critère $wb2$ est assez similaire à celui du critère EI que nous voulons maximiser. Néanmoins, le critère $wb2$ est plus lisse que le critère EI et donc plus facile à optimiser (Cf. figures IV.2 et IV.4 pour la comparaison).



(a) La fonction f.

(b) Le critère $wb2$

FIGURE IV.4 – Première itération de l'algorithme EGO. À gauche : représentation de la fonction $f(x) = (6x-2)^2 \sin(12x-4),\ x \in [0,1]$. À droite : représentation du critère $wb2$ de la fonction $f(\mathbf{x})$ pour $x \in [0,1]$.

M. Sasena, dans [Sasena, 2002], recommande l'utilisation du critère $wb2$ (pour les problèmes de petites et de moyennes dimensions) après une comparaison avec le critère EI. En effet, il a écrit dans son rapport : *The expected improvement function returns to a value of zero at each sample point. As a consequence, the expected improvement function becomes extremely "bumpy" as sample points begin to accrue around the optimum. Such a function is difficult to optimize when solving the infill sampling criteria sub-problem. The wb2 criterion does not exhibit this spiked behaviour and is therefore more*

likely to produce a good iterate. It is apparent that the wb2 criterion is much more likely to successfully locate the optimal infill sample during the infill sampling criteria sub-problem. For these reason, the wb2 criterion may provide a more robust criterion than the expected improvement function.

L'algorithme EGO est un algorithme itératif adapté aux fonctions coûteuses sans contrainte. Dans la section suivante nous présentons l'algorithme SuperEGO, décrit par M. Sasena [Sasena, 2002], qui est une extension de l'algorithme EGO pour la prise en compte de contraintes.

IV.1.2. Description de la méthode EGO avec contraintes

Dans une problématique industrielle, de nombreuses contraintes viennent complexifier l'optimisation de la fonction. On distingue deux types de contraintes : les contraintes d'égalité et les contraintes d'inégalité. Pour le premier type de contraintes, nous pouvons transformer chacune des contraintes d'égalité en deux contraintes d'inégalité avec un écart de $\eta > 0$ très petit. En effet, une contrainte d'égalité $g(x) = 0$ peut être transformée en deux contraintes d'inégalité $g(x) < \eta$ et $g(x) > -\eta$ avec $\eta > 0$ très petit.

M. Sasena, dans [Sasena et al., 2002, Sasena, 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. Ce critère de faisabilité est utile si le plan d'expériences initial ne contient aucun point faisable, car il permet de rechercher un premier point faisable dans le domaine de conception. Pour ce faire, il quantifie la probabilité de faisabilité de chaque point du domaine en supposant que toutes les contraintes sont indépendantes et en définissant la probabilité de faisabilité $P_{\mathbf{x}}$ par :

$$P_{\mathbf{x}} = \prod_{i=1}^{m} \mathscr{P}\left(g_i(\mathbf{x}) \leq 0\right), \tag{IV.5}$$

avec $g_i \sim \mathscr{N}\left(\hat{g}_i(\mathbf{x}), s_i^2(\mathbf{x})\right)$. Nous utilisons donc les modèles de krigeage relatifs aux contraintes pour l'estimation de la probabilité de faisabilité de chacune des fonctions g_i :

$$\mathscr{P}\left(g_i(\mathbf{x}) \leq 0\right) = \Phi\left(\frac{-\hat{g}_i(\mathbf{x})}{s_i(\mathbf{x})}\right), \ \forall\, i = 1, \ldots, m, \tag{IV.6}$$

où Φ est la fonction de répartition de la loi normale centrée réduite, $\hat{g}_i$ et s_i^2 correspondent, respectivement, à la prédiction de la contrainte g_i et à la variance du modèle de krigeage associée à la même contrainte. Il suffit donc de maximiser $P_{\mathbf{x}}$ pour trouver des points susceptibles d'être faisables.

Remarque 15 *Si $\mathbf{x}^{(i)}$ pour $i \in 1, \ldots, n$ est un point faisable de la base de données, alors $P_{\mathbf{x}^{(i)}}$ est forcément égale à 1. Nous sommes donc certains que $\max_{\mathbf{x}} P_{\mathbf{x}} = 1$ et la solution contient au moins le point $\mathbf{x}^{(i)}$. C'est pourquoi, ce critère n'est plus utilisable lorsque un point faisable existe déjà dans le plan d'expériences.*

Le critère de faisabilité $P_{\mathbf{x}}$ permet de trouver un premier point initial ou d'échantillonner dans des zones où un grand nombre de contraintes sont respectées. Une fois cette étape terminée, avec un nombre d'itérations fixé par l'utilisateur suivant le budget disponible, nous poursuivons par l'algorithme EGO, mais en intégrant les prédictions données par les métamodèles des contraintes g_i, $i = 1, \ldots, m$, dans le problème d'optimisation du critère d'enrichissement (Cf. [Sasena, 2002]) :

$$\begin{cases} \max\limits_{\mathbf{x} \in B} wb2(\mathbf{x}) \\ \quad \text{s.c.} \\ \quad \hat{g}_1 < 0 \\ \quad \vdots \\ \quad \hat{g}_m < 0. \end{cases} \tag{IV.7}$$

Cet algorithme est appelé SuperEGO par [Sasena, 2002]. Notons que l'estimation $\hat{g}_i$ peut être donnée par un métamodèle quelconque dans l'équation (IV.7).

Pour les problèmes d'optimisation basés à la fois sur les métamodèles sur un processus d'enrichissement, le métamodèle est reconstruit à chaque itération de l'enrichissement. Cependant, l'optimisation des paramètres du modèle de krigeage est assez longue en grande dimension, ce qui rallonge le temps nécessaire à l'optimisation. L'utilisation du modèle KPLS défini dans la section III.4. permet de réduire le temps alloué au calcul du modèle (Cf. section III.5. pour la vérification sur quelques cas tests). L'idée ici est de remplacer le modèle de krigeage par le modèle KPLS pour les problèmes en grande dimension. La stratégie d'optimisation suivie, notée SEGOKPLS (pour "*SuperEGO using Kriging combined PLS*"), est donnée par la figure IV.5 :

1. Construction d'un plan d'expériences initial.
2. Recherche du premier point faisable si le plan d'expériences initial n'en contient pas. Pour faciliter la lisibilité de la figure IV.5 (au niveau des notations), nous supposons que le plan d'expériences initial contient déjà des points faisables et que cette étape est ignorée.
3. (Re)construction des modèles KPLS à partir du plan d'expériences.
4. Maximisation du critère $wb2$ avec contraintes donné par l'équation (IV.7) utilisant l'algorithme COBYLA.
5. Évaluation du nouveau point avec le vrai code de calcul.
6. Vérification de la convergence, si oui on arrête l'algorithme, sinon on rajoute le nouveau point dans le plan d'expériences et on réitère à partir de 3.

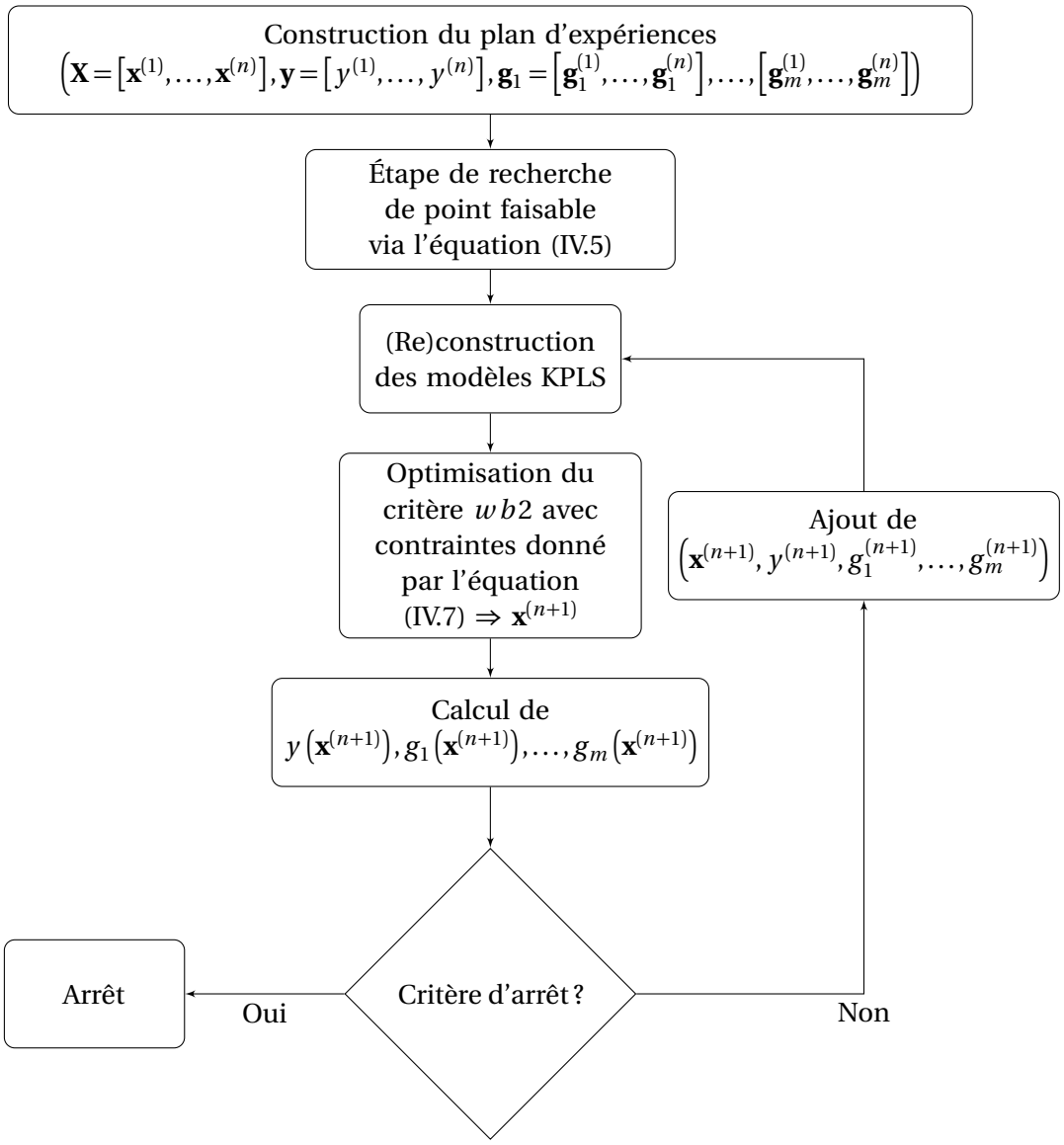


FIGURE IV.5 – Les principales étapes de la stratégie suivie SEGOKPLS.

IV.2. Tests expérimentaux sur des cas tests académiques

Dans cette section, nous allons comparer la stratégie d'optimisation, décrite par le diagramme donné dans la figure IV.5, avec plusieurs algorithmes d'optimisation, donnés par [Regis, 2014a], sur plusieurs fonctions académiques données dans l'annexe I.2.. Pour ces tests, nous avons volontairement désactivé l'étape 2 de la recherche d'un premier point faisable dans l'algorithme SEGOKPLS (étape 2 de la figure IV.5), car nous avons remarqué que l'algorithme SEGOKPLS reste performant même en ignorant cette étape pour ces cas tests.

Dans [Regis, 2014a], deux algorithmes d'optimisation développés, COBRA (pour "*Constrained Opti-*

mization By RAdial basis function interpolation") et ConstrLMSRBF (pour "*Constrained Local Metric Stochastic Radial Basis Function*"), ont été comparés avec plusieurs algorithmes concurrents de la littérature. COBRA et ConstrLMSRBF sont deux algorithmes d'optimisation d'ordre 0 et ils utilisent les modèles RBF pour guider l'optimisation vers des points prometteurs. De plus, ces algorithmes commencent par une première phase de recherche qui consiste à chercher un premier point faisable, et ils terminent par une deuxième phase qui consiste à améliorer les points faisables trouvés. Nous invitons les lecteurs à consulter l'article [Regis, 2014a] pour plus de détails sur ces algorithmes. Dans cet article, R. Regis compare ces deux méthodes avec d'autres algorithmes de la littérature :

— La méthode d'optimisation SDPEN [Liuzzi et al., 2010] (pour "*Sequential PENalty Derivative-free method for nonlinear constrained optimization*").

— La méthode d'optimisation NOMADS [Le Digabel, 2011] (pour "*Nonlinear Optimization by Mesh Adaptive Direct Search*") qui implémente la technique MADS [Audet and Denis, 2006] (pour "*Mesh Adaptive Direct Search*").

— L'algorithme GLOBAL [Sendin et al., 2009], qui est un algorithme d'optimisation avec multistart.

— L'algorithme de recherche global disponible sur "*Matlab Global Optimization toolbox*" (the Mathworks 2010) qui implémente un algorithme multistart OQNPL [Ugray et al., 2007].

Pour une meilleure lisibilité et compréhension de la comparaison SEGOKPLS avec les algorithmes décrits ci-dessus, le meilleur résultat obtenu par R. Regis est retranscrit et comparé avec le résultat obtenu par l'algorithme SEGOKPLS.

IV.2.1. Les fonctions académiques utilisées

L'algorithme SEGOKPLS est comparé aux méthodes concurrentielles sur 13 fonctions dont les optima sont donnés par la table IV.1. Les expressions de ces fonctions sont données dans l'annexe I.2.

TABLE IV.1 – Les fonctions académiques testées. d est le nombre de dimensions et m est le nombre de contraintes d'inégalité.

fonctions	d	m	domaine de définition	optimum ou meilleure valeur connue [Regis, 2014a]
WB4	4	6	$[0.125,10]\times[0.1,10]^3$	1.725
GTCD4	4	1	$[20,50]\times[1,10]\times[20,50]$ $\times[0.1,60]$	2964893.85
PVD4	4	3	$[0.1]^2\times[0,50]\times[0,240]$	5804.45
Hesse	6	6	$[0,5]\times[0,4]\times[1,5]$ $\times[0,6]\times[1,5]\times[0,10]$	-310
SR7	7	11	$[2.6,3.6]\times[0.7,0.8]\times[17,28]$ $\times[7.3,8.3]^2\times[2.9,3.9]\times[5,5.5]$	2994.42
g_{02}	10	2	$[0,10]^{10}$	-0.4
g_{03}	20	1	$[0,1]^{20}$	-0.69
g_{04}	5	6	$[78,102]\times[33,45]\times[27,45]^3$	-30655.539
g_{05}	4	5	$[0,1200]^2\times[-0.55,0.55]^2$	5126.5
g_{06}	2	2	$[13,100]\times[0,100]$	-6961.8136
g_{07}	10	8	$[-10,10]^{10}$	24.3062
g_{09}	7	4	$[-10,10]^7$	680.6301
g_{10}	8	6	$[10^2,10^4]\times[10^3,10^4]\times[10,10^3]^5$	7049.3307

IV.2.2. Le dispositif expérimental

L'algorithme SEGOKPLS est exécuté via la toolbox Python "Scikit-learn v.014" [Pedregosa et al., 2011] sur une machine Intel(R) Core(TM) i7-4500U CPU @ 1.80 Hz 2.40 GHz. Les résultats des méthodes concurrentielles sont retranscrits à partir de l'article [Regis, 2014a] (malheureusement, les plans d'expériences utilisés par R. Régis ne sont pas disponibles pour les réutiliser dans notre cas) et elles ont été exécutées via Matlab 7.11.0 sur une machine Intel(R) Core(TM) i7 CPU 860 2.8GHZ. Tous les tests sont réalisés de la manière suivante :

— Chaque test est répété 30 fois.

— Un plan d'expériences de type hypercube latin amélioré est construit pour chaque fonction contenant d points + un point faisable (sauf pour l'algorithme SEGOKPLS où tous les $d+1$ points sont non faisables).

— Le nombre d'itérations de l'optimisation est fixé à 100 itérations par test.

Pour comparer tous les algorithmes d'optimisation utilisés, la meilleure solution, la plus mauvaise solution, la médiane, la moyenne et l'écart-type des 30 tests de chaque fonction sont calculés. Pour chaque fonction testée et pour chaque critère de comparaison (la moyenne, la médiane, ...), le résultat de SEGOKPLS est comparé avec le meilleur résultat de tous les algorithmes obtenus dans [Regis, 2014a]. Les résultats de tous les cas tests pour 30 répétitions sont donnés par la table IV.2.

TABLE IV.2 – Statistiques des meilleurs points faisables des fonctions objectifs testées avec 30 répétitions pour 100 itérations d'optimisation. Les résultats entre parenthèses représentent : à gauche, le résultat de l'algorithme SEGOKPLS, à droite, le meilleur résultat obtenu parmi tous les algorithmes testés par R. Régis dans [Regis, 2014a]. Le meilleur résultat issu de la comparaison est écrit en gras. Les résultats sont en vert lorsqu'ils correspondent à la meilleure solution du problème.

fonction	SR7	WB4	GTCD4
meilleure valeur connue	2994.42	1.725	2964893.85
meilleur	(**2994.42**,**2994.42**)	(**2.22**,**2.22**)	(**1932396.19**,2966625.40)
mauvais	(**2994.42**,2994.69)	(**2.88**,3.25)	(**1999153.70**,333681396)
médiane	(**2994.42**,2994.66)	(**2.22**,2.56)	(**1997639.96**,3033080.81)
moyenne	(**2994.42**,2994.67)	(**2.35**,2.61)	(**1991268.39**,3087778.98)
écart-type	$2.69e^{-4}$	$2.08e^{-6}$	25084.61
fonction	Hesse	PVD4	g_{02}
meilleure valeur connue	-310.00*	5804	-0.40*
meilleur	(**-310.00**,**-310.00**)	(5848.66,**5807.79**)	(-0.30,**-0.33**)
mauvais	(**-310.00**,-261.82)	(**6179.71**,7669.36)	(**-0.19**,-0.16)
médiane	(**-310.00**,-297.87)	(**5959.71**,6303.98)	(**-0.23**,-0.20)
moyenne	(**-310.00**,-296.25)	(**5960.54**,6492.45)	(**-0.23**,-0.21)
écart-type	$4.90e^{-5}$	75.66	0.04
fonction	g_{03}	g_{04}	g_{05}
meilleure valeur connue	-0.69*	-30665.54*	5126.50*
meilleur	(**-0.69**,-0.64)	(**-30665.54**,**-30665.54**)	(**5126.50**,**5126.50**)
mauvais	(**0.00**,**0.00**)	(**-30665.54**,-30664.58)	(**5126.50**,5126.53)
médiane	($-2.33e^{-15}$,**-0.01**)	(**-30665.54**,-30665.15)	(**5126.50**, **5126.50**)
moyenne	(**-0.15**,-0.12)	(**-30665.54**,-30665.07)	(**5126.50**,5126.51)
écart-type	0.23	$5.94e^{-4}$	$1.11e^{-5}$
fonction	g_{06}	g_{07}	g_{09}
meilleure valeur connue	6961.81*	24.30*	680.63*
meilleur	(**-6961.81**,**-6961.81**)	(**24.30**,24.48)	(920.51,**702.04**)
mauvais	(**-6961.81**,-6677.16)	(**24.30**,1514.98)	(**1861.88**,3131.82)
médiane	(**-6961.81**,-6936.90)	(**24.30**,25.28)	(**1198.47**,1308.49)
moyenne	(**-6961.81**,-6868.93)	(**24.30**,25.35)	(**1305.15**,1413.85)
écart-type	0.02	$5.51e^{-6}$	302.2
fonction	g_{10}		
meilleure valeur connue	7049.33*		
meilleur	(8210.08,**7818.53**)		
mauvais	(**11285**,13965.60)		
médiane	(9957.84,**9494.06**)		
moyenne	(**9702.63**,10018.33)		
écart-type	1647		

IV.2.3. Comparaison et analyse des résultats

Les résultats donnés par la table IV.2 montrent que l'algorithme SEGOKPLS fournit des bons résultats lorsque nous le comparons avec les autres méthodes sur les 13 fonctions testées. Plusieurs constatations justifient cette conclusion. En effet, l'algorithme SEGOKPLS a de meilleurs résultats, en termes de meilleure solution trouvée, que les algorithmes concurrents sur toutes les fonctions testées, sauf pour les fonctions PVD4, g_{02}, g_{09} et g_{10}. En termes de plus mauvaise solution, de médiane et de moyenne, la méthode SEGOKPLS a de meilleurs résultats sur toutes les fonctions sauf deux, qui sont les fonctions g_{03} et g_{10}. De plus, la méthode SEGOKPLS trouve au moins une fois la meilleure solution connue pour les fonctions g_{03}, g_{07} et GTCD4, contrairement aux méthodes utilisées par R. Régis (l'algorithme SEGOKPLS améliore même la meilleure solution connue pour la fonction GTCD4).

Les fonctions g_{02}, g_{10} et PVD4 sont difficiles à approcher car elles sont très sensibles autour de l'optimum (Cf. les expressions (A.16), (A.4) et (A.8)). Ce comportement est très difficile à approcher avec un seul modèle (KPLS) et c'est pourquoi l'algorithme SEGOKPLS n'arrive pas à atteindre cette solution. L'utilisation d'un mélange d'experts (plusieurs modèles KPLS par exemple) basé sur la technique "*Expectation-Maximization*" pourrait peut-être résoudre ce problème (Cf. [Bettebghor et al., 2011]). L'algorithme SEGOKPLS rencontre quelques difficultés pour optimiser la fonction g_{03}, car l'optimiseur COBYLA retourne fréquemment des valeurs en dehors du domaine de définition lors de la maximisation du critère $wb2$. En effet, l'algorithme COBYLA considère les bornes du domaine comme des contraintes d'inégalité. Pour la fonction g_{03}, il suffit d'avoir une seule variable (x_i) égale à 0 pour annuler la fonction objectif, ce qui explique l'obtention d'une médiane nulle.

Globalement, l'algorithme SEGOKPLS a une meilleure médiane et une meilleure moyenne sur la plupart des fonctions, ce qui rend cet algorithme plus stable et plus robuste que les autres algorithmes concurrents. Ce comportement est très important pour la résolution de problèmes dans un contexte industriel.

IV.3. Conclusion

Dans ce chapitre, nous avons repris la théorie de la méthode EGO et nous l'avons adaptée aux problèmes de grande dimension en introduisant les modèles KPLS. Ensuite, nous avons comparé la performance de l'algorithme SEGOKPLS avec d'autres algorithmes d'optimisation de la littérature sur des problèmes académiques. Les premiers résultats obtenus sont très encourageants et valident la méthode sur les problèmes d'optimisation de dimension allant jusqu'à 20. Avant d'appliquer la méthode SEGOKPLS sur des cas industriels plus complexes, nous proposons dans le chapitre suivant, nos derniers travaux sur les métamodèles KPLS qui permettent d'améliorer le maximum de la fonction de vraisemblance.

V Améliorations de la stratégie SEGOKPLS

Objectifs du chapitre

- Démontrer comment passer d'un noyau KPLS à un noyau exponentiel classique,
- Proposer une nouvelle approche pour estimer les hyper-paramètres du modèle de krigeage pour la grande dimension,
- Mettre en place la stratégie SEGOKPLS+K.

Dans ce chapitre, nous allons proposer une amélioration du modèle KPLS. Nous commençons par une étude expérimentale sur la fonction de Griewank sur l'intervalle $[-5,5]$ dans la section V.1.1.. Cette étude permettra de relever quelques difficultés rencontrées par les modèles KPLS. Ensuite, nous proposons une nouvelle approche pour estimer les hyper-paramètres du modèle de krigeage pour la grande dimension en se basant sur l'estimation des hyper-paramètres du modèle KPLS. Nous validons ensuite cette nouvelle méthode sur la fonction de Griewank sur l'intervalle $[-5,5]$, respectivement dans les sections V.1.2. et V.1.3.. Enfin, ce nouveau modèle sera intégré dans l'algorithme SEGOKPLS comme décrit dans la section V.2..

V.1. Amélioration du modèle KPLS

Dans cette section, nous reprenons le cas test de la fonction de Griewank traité dans la section III.5.1.b.. Nous rappelons l'expression de cette fonction :

$$\begin{aligned} y_{\text{Griewank}}(\mathbf{x}) &= \sum_{i=1}^{d} \frac{\mathbf{x}_i^2}{4000} - \prod_{i=1}^{d} \cos(\frac{\mathbf{x}_i}{\sqrt{i}}) + 1, \\ & -600 \leq \mathbf{x}_i \leq 600, \text{ pour } i = 1, ..., d. \end{aligned} \tag{V.1}$$

L'intervalle [-600,600], sur lequel cette fonction a été étudiée, semble être très large pour pouvoir représenter correctement les vraies variations de la fonction. La figure III.25 montre que la fonction de Griewank est très proche d'une fonction parabolique. En effet, lorsque $\mathbf{x}_i \in [-600, 600]$, pour tout $i = 1, \ldots, d$, le terme en "cos" dans l'équation (V.1) a peu d'influence sur le résultat final (produit de plusieurs termes compris entre -1 et 1) et donc le terme parabolique "$\frac{\mathbf{x}_i^2}{4000}$" est dominant. Lorsque nous réduisons l'intervalle $[-600, 600]$ à un intervalle beaucoup moins large, comme par exemple l'intervalle $[-5, 5]$, le contraire se produit et le terme en "cos" dans l'équation (V.1) devient dominant, ce qui expliquerait les multimodalités plus prononcées de la fonction de Griewank, donnée par la figure V.1. Dans la suite, nous allons réétudier cette fonction en se restreignant à l'intervalle [-5,5], afin

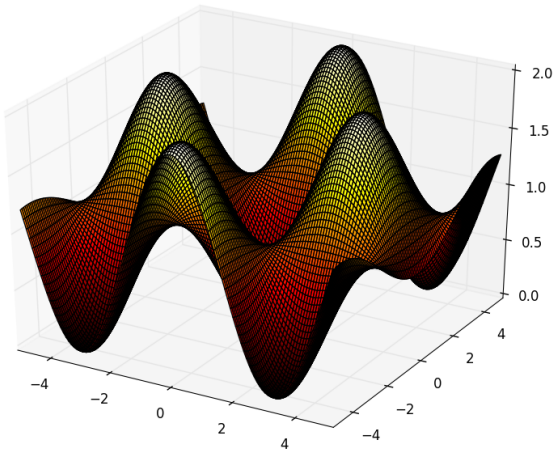


FIGURE V.1 – Fonction de Griewank en dimension 2 sur $[-5, 5]$.

de rendre l'approximation plus locale et de bien faire apparaître les multimodalités de la fonction.

V.1.1. Étude expérimentale de la fonction de Griewank sur l'intervalle $[-5, 5]$ avec le modèle KPLS

Dans cette section, nous reprenons la fonction de Griewank sur l'intervalle [-5,5]. Les tests expérimentaux sont organisés comme suit :

— 4 modèles sont utilisés : KPLS1, KPLS2, KPLS3 et le krigeage ordinaire ; nous utilisons un noyau

gaussien pour tous ces modèles.

— Deux dimensions sont traitées : $d = 20$ et $d = 60$.

— 50, 100, 200 et 300 points d'apprentissage de type hypercube latin sont utilisés pour chaque cas test.

— 5000 points de validation sont utilisés pour chaque cas test.

— 10 répétitions sont réalisées pour chaque cas test.

— L'erreur relative ER donnée par l'équation (III.52) et le temps CPU sont les critères utilisés pour comparer les modèles KPLS avec le krigeage ordinaire.

— Des boîtes à moustaches sont utilisées pour représenter l'erreur relative ER et le temps CPU afin de mieux visualiser les résultats.

— La moyenne et l'écart-type de chaque cas test sont donnés dans les tables A.1 et A.2 de l'annexe I.3..

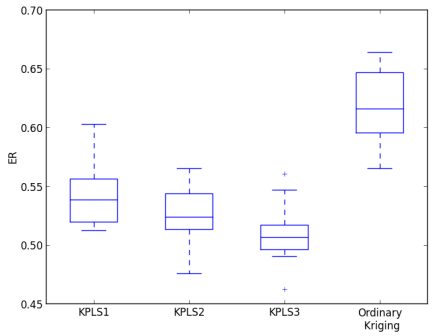
— Tous les calculs sont réalisés avec une machine Intel(R) Core(TM)2 Duo CPU E6850 @ 3.00GHz desktop machine.

Pour 20 dimensions, les modèles KPLS sont plus précis que le modèle de krigeage ordinaire lorsque nous utilisons 50 points d'apprentissage, comme illustré sur la figure V.2a. Lorsque nous augmentons le nombre de points d'apprentissage, le taux d'amélioration de la précision des modèles KPLS est plus faible que le taux d'amélioration observé par le modèle de krigeage, comme le montrent les figures V.2c, V.2e et V.2g.

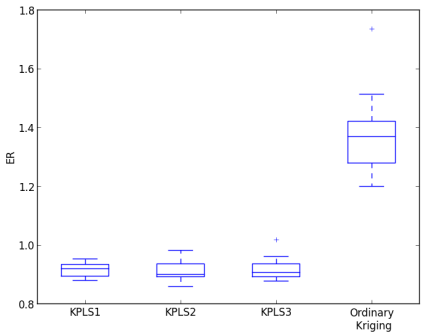
Pour 60 dimensions, nous remarquons une évolution des résultats assez proche du cas test à 20 dimensions, comme le montrent les figures V.2b, V.2d, V.2f et V.2h. En effet, le taux d'amélioration de la précision du modèle de krigeage est meilleur que les taux observés par les modèles KPLS. Cependant, 100 points d'apprentissage suffisent pour obtenir un modèle de krigeage de meilleure qualité que les modèles KPLS pour 20 dimensions, alors que 300 points d'apprentissage sont requis pour le cas test à 60 dimensions. Ceci laisse penser que l'évolution des résultats dépend du ratio $\frac{n}{d}$.

Remarque 16 *La dernière constatation n'a pas été démontrée et nécessiterait d'être vérifiée sur plus de cas tests.*

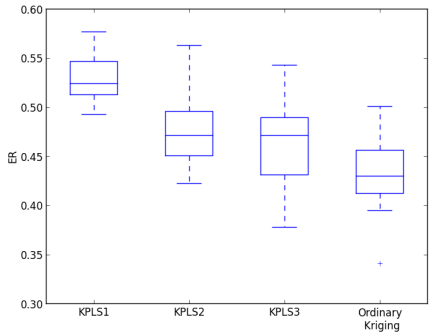
Concernant le temps CPU, un gain considérable est observé pour les dimensions 20 et 60, comme le montre la figure V.3.



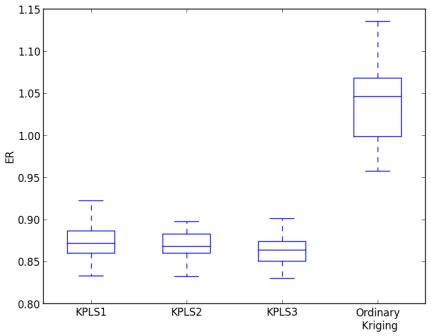
(a) $ER(\%)$ pour $20D$ et 50 points d'apprentissage.



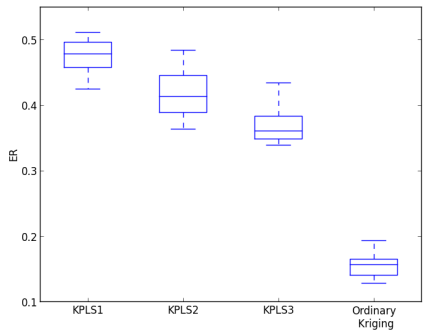
(b) $ER(\%)$ pour $60D$ et 50 points d'apprentissage.



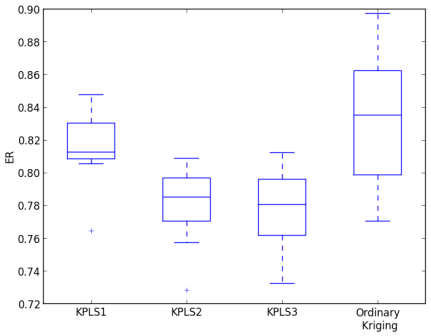
(c) $ER(\%)$ pour $20D$ et 100 points d'apprentissage.



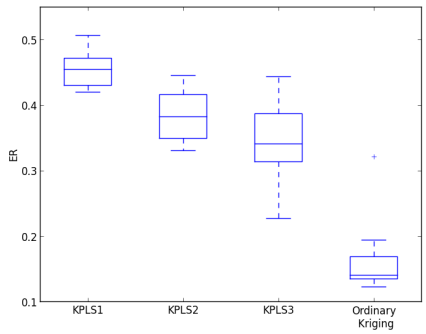
(d) $ER(\%)$ pour $60D$ et 100 points d'apprentissage.



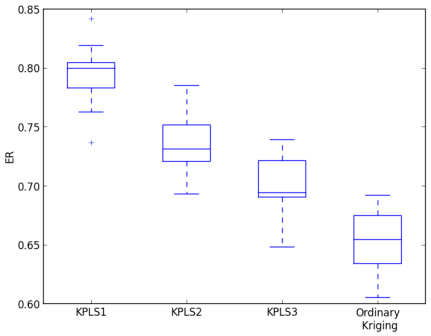
(e) $ER(\%)$ pour $20D$ et 200 points d'apprentissage.



(f) $ER(\%)$ pour $60D$ et 200 points d'apprentissage.

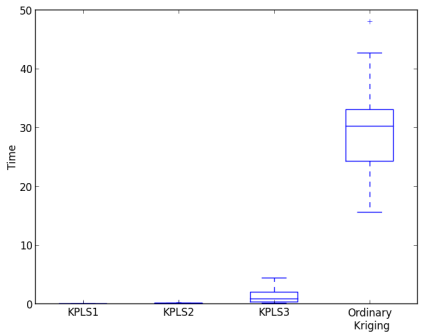


(g) $ER(\%)$ pour $20D$ et 300 points d'apprentissage.

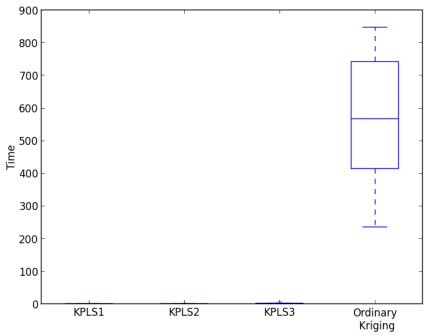


(h) $ER(\%)$ pour $60D$ et 300 points d'apprentissage.

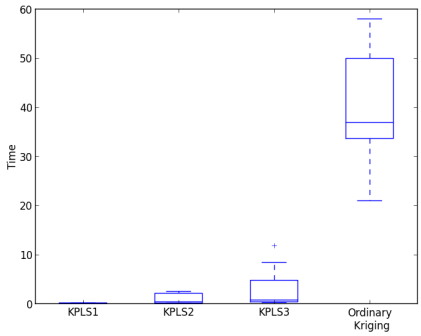
FIGURE V.2 – Boîtes à moustaches de l'erreur relative ER des modèles KPLS et du modèle de krigeage pour approcher la fonction de Griewank sur l'intervalle $[-5,5]$. 10 plans d'expériences de type hypercube latin sont utilisés. À gauche : 20 dimensions ; À droite : 60 dimensions.



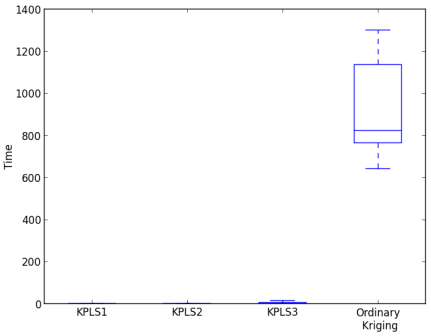
(a) Temps CPU pour $20D$ et 50 points d'apprentissage.



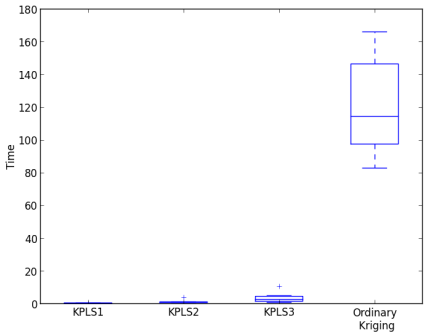
(b) Temps CPU pour $60D$ et 50 points d'apprentissage.



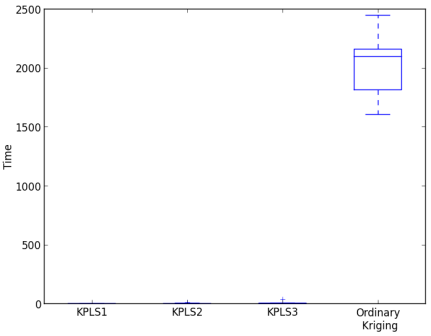
(c) Temps CPU pour $20D$ et 100 points d'apprentissage.



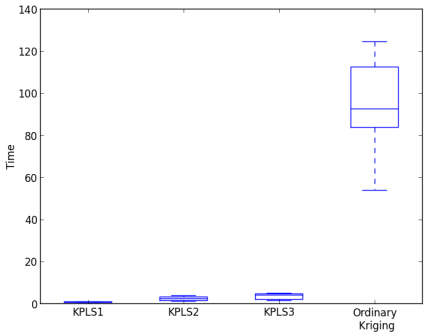
(d) Temps CPU pour $60D$ et 100 points d'apprentissage.



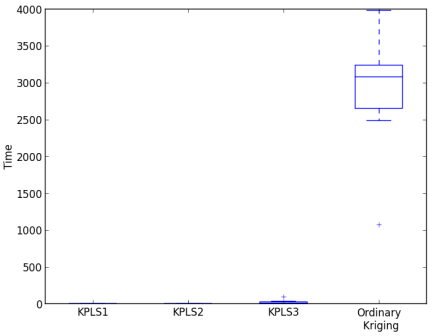
(e) Temps CPU pour $20D$ et 200 points d'apprentissage.



(f) Temps CPU pour $60D$ et 200 points d'apprentissage.



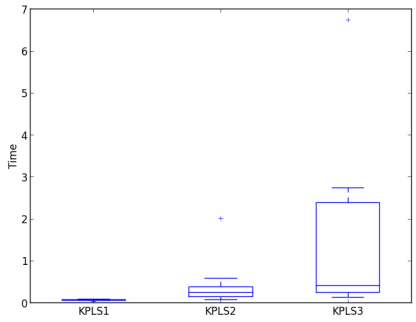
(g) Temps CPU pour $20D$ et 300 points d'apprentissage.



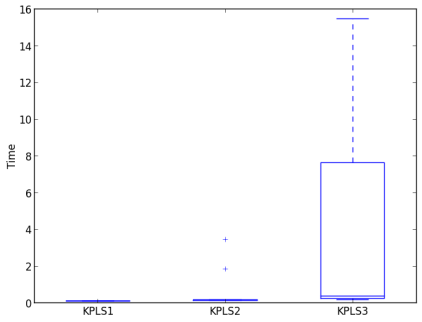
(h) Temps CPU pour $60D$ et 300 points d'apprentissage.

FIGURE V.3 – Boîtes à moustaches du temps CPU des modèles KPLS et du modèle de krigeage pour approcher la fonction de Griewank sur l'intervalle $[-5,5]$. 10 plans d'expériences de type hypercube latin sont utilisés. À gauche : 20 dimensions ; À droite : 60 dimensions.

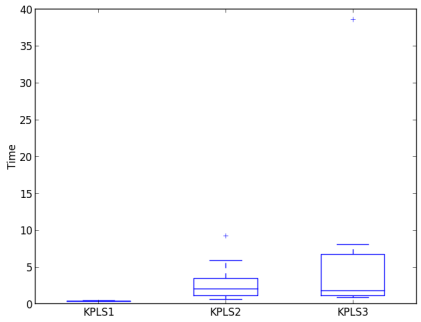
Pour le cas à 60 dimensions, le temps CPU des modèles KPLS est difficile à distinguer. Pour cela, nous avons besoin de le représenter indépendamment du temps CPU du modèle de krigeage (Cf. figure V.4).



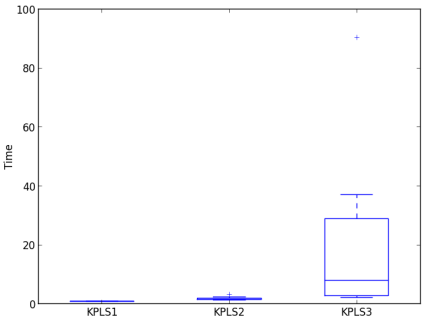
(a) Temps CPU pour 60D et 50 points d'apprentissage.



(b) Temps CPU pour 60D et 100 points d'apprentissage.



(c) Temps CPU pour 60D et 200 points d'apprentissage.



(d) Temps CPU pour 60D et 300 points d'apprentissage.

FIGURE V.4 – Boîtes à moustaches du temps CPU des modèles KPLS pour approcher la fonction de Griewank sur l'intervalle $[-5,5]$ en 60 dimensions. 10 plans d'expériences de type hypercube latin sont utilisés.

D'après la figure V.4, l'ajout de composantes principales fait augmenter progressivement les coûts de calcul. Pour cela, nous recommandons de se restreindre à des noyaux KPLS avec au plus 3 ou 4 composantes principales.

Dans la section suivante, nous proposons une amélioration de la qualité des modèles KPLS au détriment d'une légère hausse du temps CPU.

V.1.2. Transition du modèle KPLS au modèle de krigeage pour un noyau exponentiel

Dans cette section, nous démontrons que si tous les noyaux KPLS_l donnés par l'équation (III.49), pour $l = 1, \ldots, h$, sont de type exponentiel et de même nature (tous gaussiens par exemple), alors le nouveau noyau $\text{KPLS}_{1:h}$ donné par l'équation (III.50), est également de type exponentiel. À titre d'exemple, si tous les noyaux KPLS_l, pour $l = 1, \ldots, h$, sont de type gaussien, alors le nouveau noyau $\text{KPLS}_{1:h}$ l'est aussi. Pour démontrer cet exemple, nous développons :

$$\begin{aligned} k_{1:h}(\mathbf{x}, \mathbf{x}') &= \prod_{l=1}^{h} \prod_{i=1}^{d} \exp\left(-\theta_l \mathbf{w}_{*i}^{(l)^2} (\mathbf{x}_i - \mathbf{x}'_i)^2\right) \\ &= \exp\left(\sum_{i=1}^{d} \sum_{l=1}^{h} -\theta_l \mathbf{w}_{*i}^{(l)^2} (\mathbf{x}_i - \mathbf{x}'_i)^2\right) \\ &= \exp\left(\sum_{i=1}^{d} -\eta_i (\mathbf{x}_i - \mathbf{x}'_i)^2\right) \\ &= \prod_{i=1}^{d} \exp\left(-\eta_i (\mathbf{x}_i - \mathbf{x}'_i)^2\right), \end{aligned} \tag{V.2}$$

en posant pour $i = 1, \ldots, d$:

$$\eta_i = \sum_{l=1}^{h} \theta_l \mathbf{w}_{*i}^{(l)^2}. \tag{V.3}$$

De la même manière, nous pouvons le démontrer pour les autres noyaux exponentiels de la forme suivante :

$$\sigma^2 \prod_{l=1}^{h} \prod_{i=1}^{d} \exp\left(-\theta_l \left|\mathbf{w}_{*i}^{(l)} (\mathbf{x}_i - \mathbf{x}'_i)\right|^{p_l}\right) \tag{V.4}$$

où les valeurs $p_1 = \cdots = p_h$ doivent être identiques pour les h noyaux.

À l'aide de cette transition des modèles KPLS et le modèle de krigeage pour les noyaux exponentiels, nous proposons une étape supplémentaire dans la construction des modèles KPLS. Cette étape intervient directement après l'estimation des hyper-paramètres θ_l, pour $l = 1, \ldots, h$. Elle consiste à faire une optimisation locale de la fonction de vraisemblance du modèle de krigeage équivalent au modèle KPLS. Le point de départ de cette optimisation locale est calculé avec l'équation (V.3) en considérant les hyper-paramètres θ_l, pour $l = 1, \ldots, h$, solutions du modèle KPLS. Les différentes étapes de construction de ce nouveau modèle sont récapitulées dans la figure V.5.

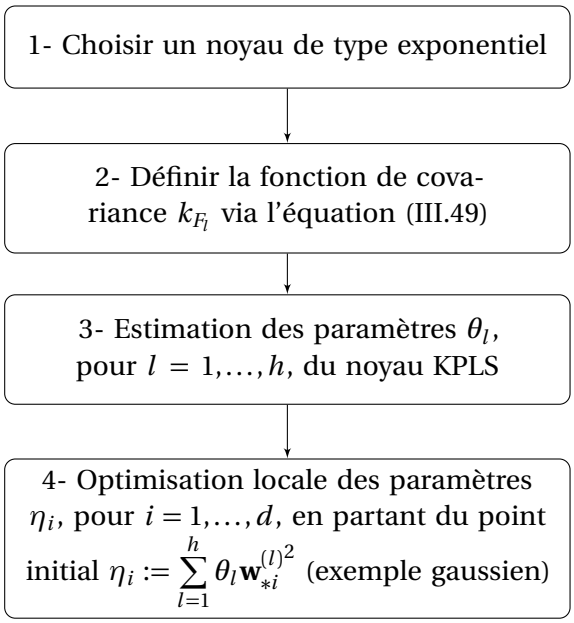


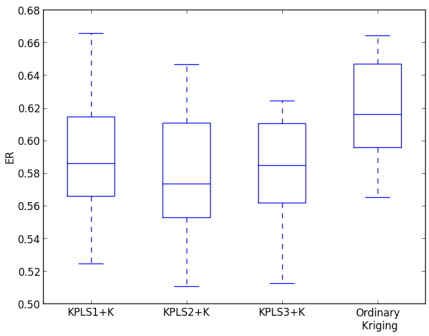
FIGURE V.5 – Les principales étapes nécessaires pour la construction du modèle de krigeage par l'intermédiaire du modèle KPLS.

Le but de cette étape est d'améliorer la solution de la fonction de vraisemblance du modèle de krigeage équivalent au modèle KPLS associé. Le fait de considérer la formule du modèle de krigeage équivalent au lieu du modèle KPLS, permet de relancer l'optimisation des hyper-paramètres directement sur chaque direction du problème d'origine. Ceci nous permet ainsi d'agrandir l'espace de recherche des hyper-paramètres et par conséquent d'améliorer la fonction de vraisemblance du modèle. Cette initialisation faite, une optimisation locale de la fonction de vraisemblance sur l'espace d'origine en partant de la solution du modèle KPLS est utilisée pour améliorer le maximum de la fonction de vraisemblance déjà obtenu. Dans toute la suite du rapport, nous utilisons l'algorithme COBYLA pour réaliser cette optimisation locale.

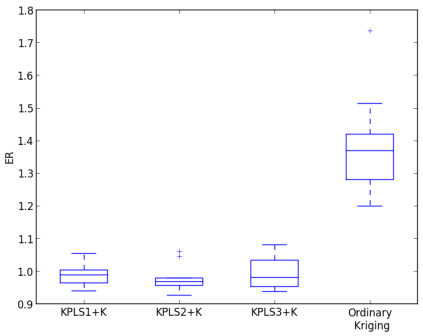
Dans la section suivante, une étude expérimentale du nouveau modèle, que nous notons 'KPLS+K', sur l'exemple de Griewank sur l'intervalle $[-5,5]$ est réalisée.

V.1.3. Étude expérimentale de la fonction de Griewank sur l'intervalle $[-5,5]$ avec le modèle KPLS+K

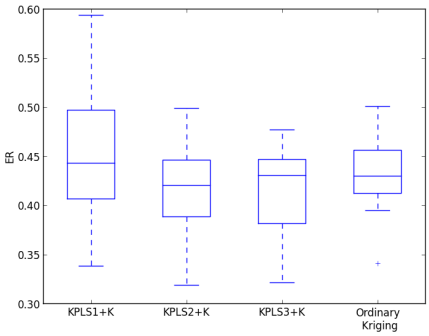
Dans cette section, nous reprenons la fonction de Griewank sur l'intervalle $[-5,5]$ en considérant les mêmes conditions que celles appliquées dans la section V.1.1., c'est à dire, les mêmes plans d'expériences, les mêmes critères de comparaisons, ... Pour distinguer le nombre de composantes principales utilisées l, $l = 1,\ldots,d$, pour la construction du modèle KPLS+K, nous le notons par KPLSl+K. La figure V.6 représente les résultats de comparaison de l'erreur relative ER des modèles KPLS+K, utilisant différents nombres de composantes principales allant de 1 à 3, avec le modèle de krigeage ordinaire.



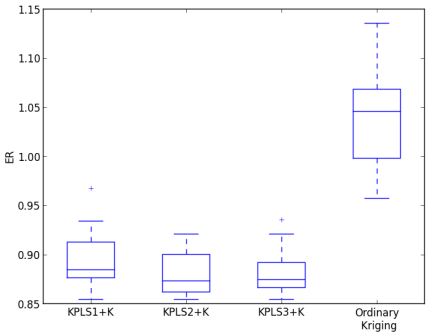
(a) $ER(\%)$ pour $20D$ et 50 points d'apprentissage.



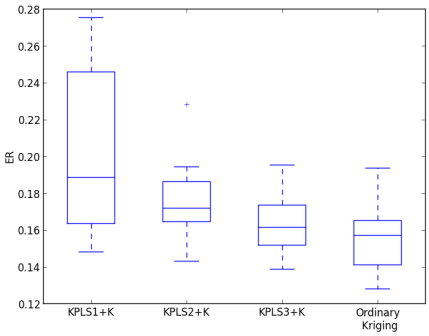
(b) $ER(\%)$ pour $60D$ et 50 points d'apprentissage.



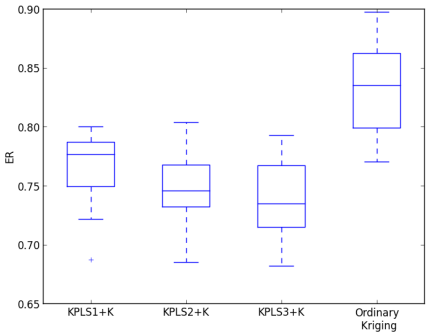
(c) $ER(\%)$ pour $20D$ et 100 points d'apprentissage.



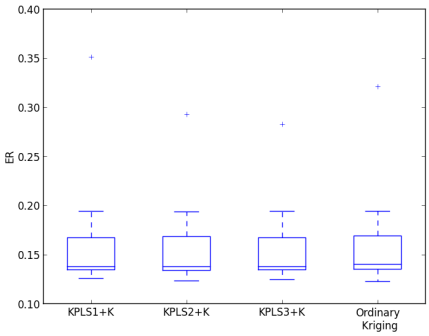
(d) $ER(\%)$ pour $60D$ et 100 points d'apprentissage.



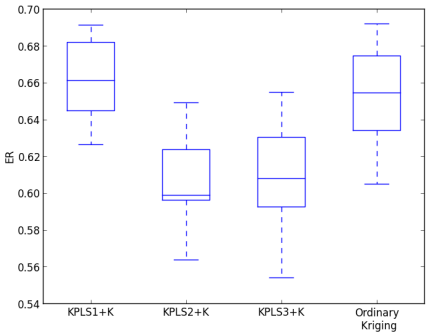
(e) $ER(\%)$ pour $20D$ et 200 points d'apprentissage.



(f) $ER(\%)$ pour $60D$ et 200 points d'apprentissage.



(g) $ER(\%)$ pour $20D$ et 300 points d'apprentissage.



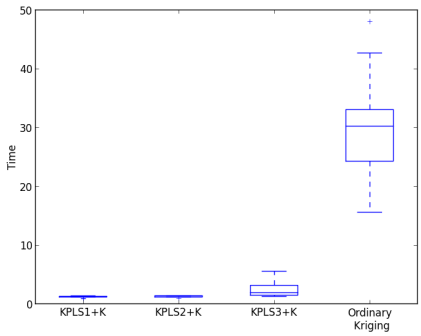
(h) $ER(\%)$ pour $60D$ et 300 points d'apprentissage.

FIGURE V.6 – Boîtes à moustaches de l'erreur relative ER des modèles KPLS+K et du modèle de krigeage pour approcher la fonction de Griewank sur l'intervalle $[-5,5]$. 10 plans d'expériences de type hypercube latin sont utilisés. À gauche : 20 dimensions ; À droite : 60 dimensions.

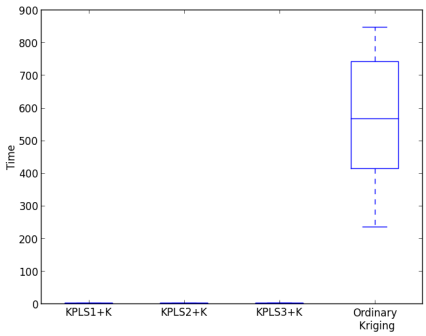
Pour 20 dimensions, la qualité des modèles KPLS+K est légèrement dégradée par rapport à la qualité des modèles KPLS lorsque nous utilisons 50 points d'apprentissage (Cf. figures V.2a et V.6a). En effet, nous obtenons une moyenne entre 0.51 % et 0.54 % pour les modèles KPLS, à comparer avec une moyenne entre 0.58 % et 0.59 % pour les modèles KPLS+K. Les détails de ces résultats sont donnés dans les tables A.1 et A.3 de l'annexe. Cependant, lorsque nous augmentons le nombre de points dans le plan d'expériences ($\geq$ 100 points), la qualité des modèles KPLS+K est proche de celle observée par le modèle de krigeage, comme le montrent les figures V.6c, V.6e et V.6g. De plus, nous avons une qualité quasi-identique entre tous les modèles pour 300 points d'apprentissage. En effet, nous avons une erreur relative moyenne autour de 0.16 % avec un faible écart-type d'environ 0.06 % pour tous les modèles (Cf. table A.3 en annexe). Par conséquent, la qualité des modèles KPLS est améliorée grâce à l'étape supplémentaire décrite dans la section V.1.2., lorsque nous utilisons un nombre de points d'apprentissage supérieur à 100.

Pour 60 dimensions, les conclusions sont assez semblables au cas test à 20 dimensions. En effet, les résultats de l'erreur relative ER des modèles KPLS sont meilleurs que ceux des modèles KPLS+K pour 50 points d'apprentissage, nous avons une moyenne d'environ 0.92 % pour les modèles KPLS contre une moyenne d'environ 0.99 %, d'après les tables A.2 et A.4 de l'annexe. Cependant, lorsque le nombre de points d'apprentissage est supérieur à 200 points, les modèles KPLS+K deviennent plus précis que les modèles KPLS (Cf. tables A.2 et A.4 de l'annexe). Contrairement aux résultats des modèles KPLS donnés par la figure V.2h, les modèles KPLS+K à 2 ou 3 composantes principales restent de meilleure qualité que le modèle de krigeage pour un plan d'expériences de 300 points, comme le montre la figure V.6h.

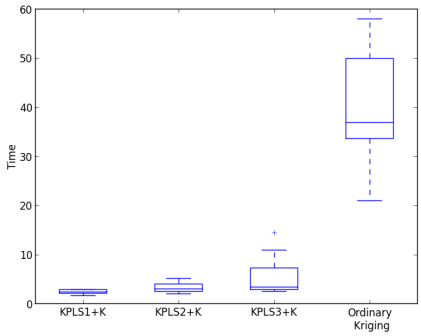
En termes de temps CPU, qui est donné par la figure V.7, les modèles KPLS+K restent tout de même plus rapides que le modèle de krigeage. Pour le premier cas à 20 dimensions, le temps CPU des modèles KPLS est plus rapide que le modèle de krigeage d'un facteur proche de 4, par exemple, 19.89 s sont requises pour construire le modèle KPLS2+K, alors que le modèle de krigeage ordinaire nécessite 94.31 s (Cf. tables A.1 et A.3). Pour le deuxième cas à 60 dimensions, un gain considérable en temps CPU pour les modèles KPLS+K est observé. À titre d'exemple, un gain d'un facteur 223 est obtenu pour le KPLS1+K et un gain d'un facteur 70 est obtenu pour le KPLS3+K, pour le cas à 300 points d'apprentissage (Cf. table A.4).



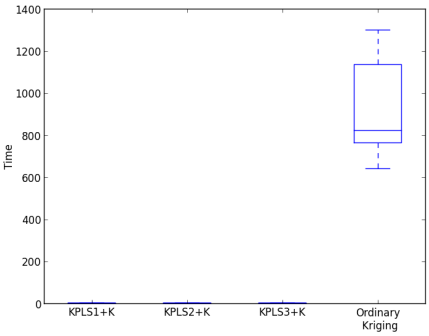
(a) Temps CPU pour $20D$ et 50 points d'apprentissage.



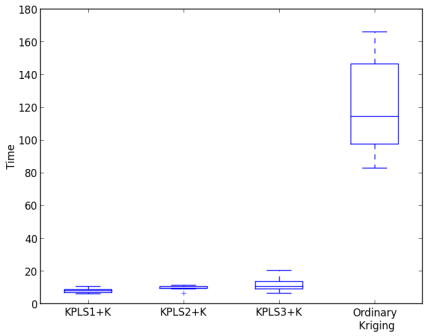
(b) Temps CPU pour $60D$ et 50 points d'apprentissage.



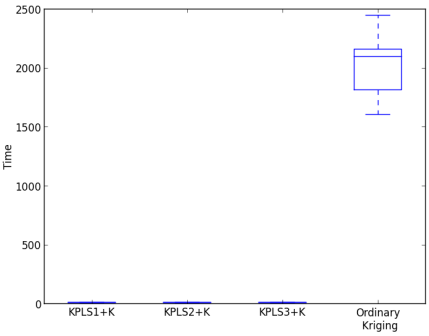
(c) Temps CPU pour $20D$ et 100 points d'apprentissage.



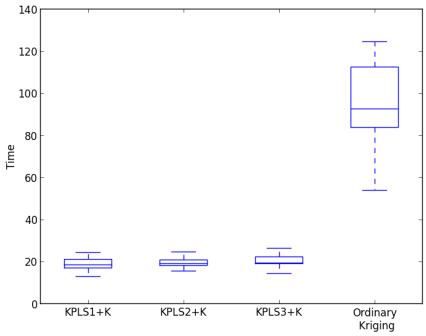
(d) Temps CPU pour $60D$ et 100 points d'apprentissage.



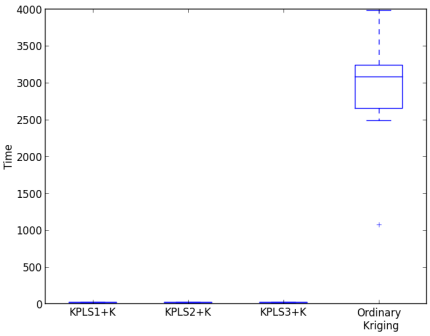
(e) Temps CPU pour $20D$ et 200 points d'apprentissage.



(f) Temps CPU pour $60D$ et 200 points d'apprentissage.



(g) Temps CPU pour $20D$ et 300 points d'apprentissage.



(h) Temps CPU pour $60D$ et 300 points d'apprentissage.

FIGURE V.7 – Boîtes à moustaches du temps CPU des modèles KPLS+K et du modèle de krigeage pour approcher la fonction de Griewank sur l'intervalle $[-5,5]$. 10 plans d'expériences de type hypercube latin sont utilisés. À gauche : 20 dimensions ; À droite : 60 dimensions.

V.1.4. Exemple d'un cas industriel

Nous reprenons maintenant le problème industriel pbme_2 décrit dans la section III.5.2.. En particulier. Nous nous intéressons à la sortie y_3, pour laquelle les modèles KPLS se sont avérés de moins bonne qualité que le modèle de krigeage (Cf. table III.13). Nous rappelons qu'il s'agit d'un problème à 24 dimensions avec 99 points d'apprentissage et 52 points de validation. D'après les résultats de la table V.1, le modèle KPLS+K permet non seulement d'améliorer la qualité des modèles KPLS, mais aussi d'avoir une qualité meilleure que celle du modèle de krigeage avec une réduction de coût de calcul d'un facteur d'environ 0.5.

TABLE V.1 – Erreur de prédiction de la sortie y_3 du problème pbme_2 à 24 dimensions, avec un plan d'expériences de 99 points.

	métamodèles	$ER(\%)$	temps CPU
pbme_2	Krigeage	8.97	8.17 s
	KPLS1	10.35	0.18 s
	KPLS2	10.33	0.42 s
	KPLS3	10.41	1.14 s
	KPLS1+K	8.77	2.15 s
	KPLS2+K	8.72	4.22 s
	KPLS3+K	8.73	4.53 s

V.2. Application du modèle KPLS+K

Dans la section précédente, nous avons proposé une nouvelle étape dans la construction des modèles KPLS. Cette étape nous a permis d'avoir une certaine stabilité dans l'évolution de la qualité du modèle en fonction du nombre de points d'apprentissage. Dans la suite, nous intégrons les modèles KPLS+K dans l'algorithme SEGOKPLS décrit dans la section IV.1.2.. En effet, il s'agit simplement de remplacer les modèles KPLS par les modèles KPLS+K dans le diagramme donné par la figure IV.5, comme le montre la figure V.8. Dans toute la suite, cet algorithme est noté SEGOKPLS+K.

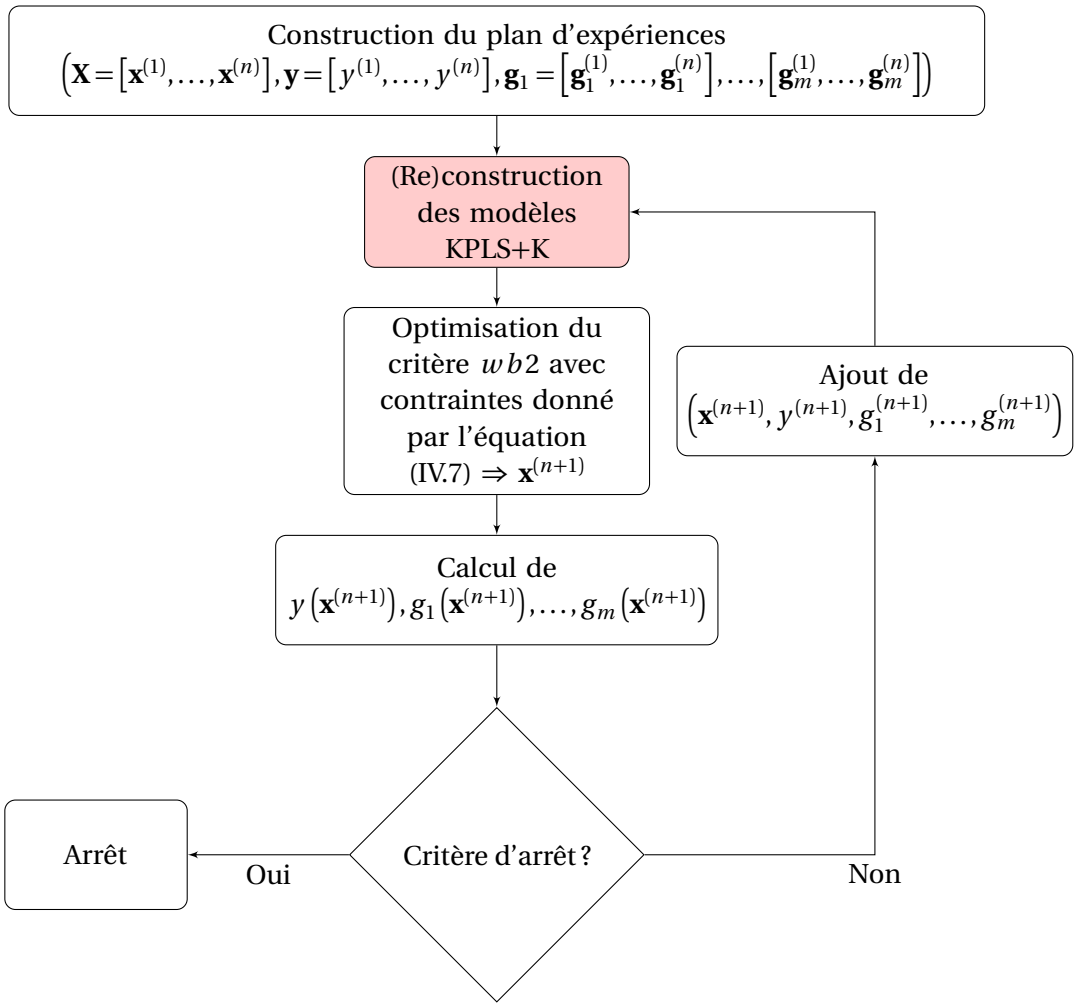


FIGURE V.8 – Les principales étapes de la stratégie suivie SEGOKPLS+K. La différence de cet algorithme par rapport à l'algorithme SEGO classique réside dans l'utilisation des modèles KPLS+K au lieu des modèles de krigeage (étape indiquée en rouge dans le diagramme).

V.3. Conclusion

Dans ce chapitre, nous avons intégré une nouvelle étape pour améliorer le maximum de la fonction de vraisemblance des modèles des modèles KPLS, en réalisant une optimisation locale sur l'espace complet des hyper-paramètres. Nous avons mené une étude expérimentale sur la fonction de Griewank sur l'intervalle [-5,5]. Cette étude montre le gain obtenu par les modèles KPLS+K en termes d'erreur relative ER notamment lorsque nous augmentons le ratio $\frac{n}{d}$ (Cf. figure V.9).

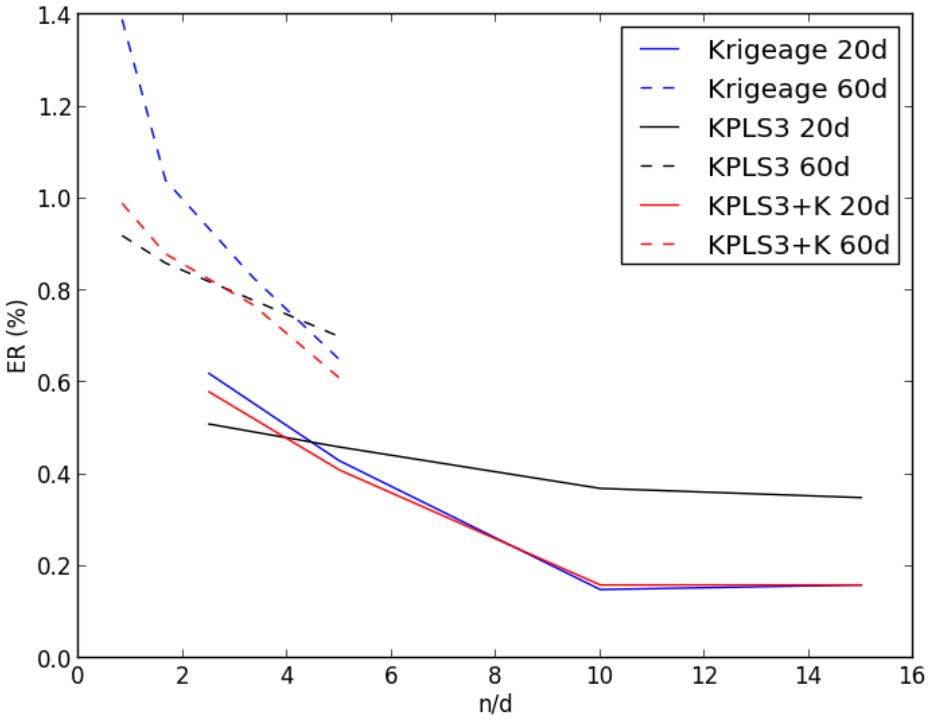


FIGURE V.9 – Évolution de l'erreur relative en fonction de $\frac{n}{d}$ de la fonction de Griwank sur l'intervalle [-5,5] en 20 et 60 dimensions. Trois modèles sont représentés : le modèle de krigeage, le modèle KPLS3 et les modèles KPLS3+K en 20 dimensions et KPLS3+K en 60 dimensions

Cependant, le temps CPU reste toujours inférieur au temps nécessaire pour un krigeage ordinaire. De plus, cette étude nous a permis de relever de nouvelles interrogations comme :

— Les raisons d'avoir un modèle KPLS de meilleure qualité que le modèle de krigeage lorsque le facteur $\frac{n}{d}$ est faible.

— L'évolution de l'erreur relative ER par rapport au facteur $\frac{n}{d}$ (Cf. figure V.9).

— Les raisons de dégrader la qualité des modèles KPLS en utilisant les modèles KPLS+K lorsque le ratio $\frac{n}{d}$ est faible, alors que le maximum de la fonction de vraisemblance est meilleur.

Pour pouvoir répondre à toutes ces questions, une étude supplémentaire sur d'autres cas tests serait à réaliser, mais le manque de temps ne permet pas de l'envisager dans le cadre de ce travail de thèse.

VI Application de la méthode d'optimisation SEGOKPLS+K sur des cas industriels

Objectifs du chapitre

- Valider l'algorithme SEGOKPLS+K sur des cas tests industriels,
- Identifier les limitations de l'implémentation de l'algorithme SEGOKPLS+K dans un contexte industriel.

Pour vérifier la performance de l'algorithme d'optimisation décrit dans la section V.2., deux cas industriels, le cas test MOPTA et le cas test Snecma turbine basse pression 2.5 étages, sont étudiés dans ce chapitre. Dans tout le chapitre, le seuil des contraintes est fixé à 10^{-5}, c'est à dire, les contraintes inférieures à 10^{-5} sont considérées comme des contraintes faisables.

VI.1. Test expérimental sur le cas automobile MOPTA

VI.1.1. Généralités

Dans cette section, nous reprenons le cas industriel issu de l'industrie automobile MOPTA décrit dans la section III.5.3.. L'objectif du problème est de réduire la masse d'un véhicule sous des contraintes mécaniques, des contraintes vibratoires et des contraintes de crash. Nous rappelons qu'il s'agit d'un problème d'optimisation à très grand nombre de dimensions (124 dimensions) et de fonctions

contraintes (68 contraintes). Il est à noter aussi que la meilleure solution connue (MSC) du problème est de 222.23 kg. Nous rappelons que les données d'entrée ainsi que les données de sortie sont déjà normées pour ce cas test.

Pour ce cas test, nous avons tout d'abord appliqué l'algorithme SEGOKPLS+K sur le problème complet, c'est à dire le problème à 124 dimensions. Mais nous n'avons pas réussi à retrouver la MSC du problème (Cf. section VI.1.2.) malgré la performance de l'algorithme SEGOKPLS au cours des premières itérations de l'optimisation. Pour cela, nous avons décidé de réduire la complexité du problème en réduisant son nombre de dimensions. Pour ce faire, nous avons fixé quelques variables d'entrée sur les valeurs de la MSC du problème, ensuite, nous avons optimisé sur les variables d'entrée restantes (Cf. section VI.1.3.). Enfin, nous augmentons progressivement le nombre de dimensions du problème pour déterminer la limite de l'algorithme SEGOKPLS+K (en termes de dimensions). Dans notre cas, nous avons traité 3 problèmes MOPTA simplifiés :

— optimisation en 12 dimensions donnée dans la section VI.1.3.a.,

— optimisation en 50 dimensions donnée dans la section VI.1.3.b.,

— optimisation en 70 dimensions donnée dans la section VI.1.3.c..

Tous les calculs sont réalisés avec une machine Intel(R) Core(TM) i7-4500U CPU @ 1.80 Hz 2.40 GHz.

VI.1.2. Résultats du cas test MOPTA complet

Pour appliquer l'algorithme SEGOKPLS+K sur le problème d'optimisation MOPTA à 124 dimensions, nous considérons un plan d'expériences initial de type hypercube latin de 125 points et 221 points d'enrichissement. Nous avons choisi un nombre de points dans le plan d'expériences initial égal à $d+1$ pour se mettre dans les mêmes conditions que R. Régis dans [Regis, 2014a]. En outre, cela nous permet de se rapprocher des conditions réelles d'une optimisation au sein de Snecma, où généralement le nombre de points dans le plan d'expériences initial ne dépasse pas $2d$.

Pour faciliter l'interprétation des résultats, nous présentons l'évolution de la moyenne de différents critères. Cette moyenne est évaluée toutes les 10 itérations. Les critères considérés sont :

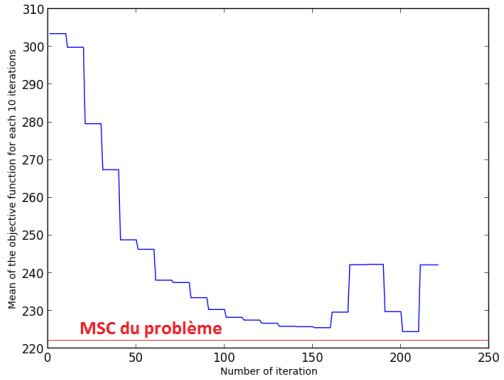
— La fonction objectif en fonction du nombre d'itérations d'enrichissement,

— Le nombre de contraintes violées en fonction du nombre d'itérations d'enrichissement,

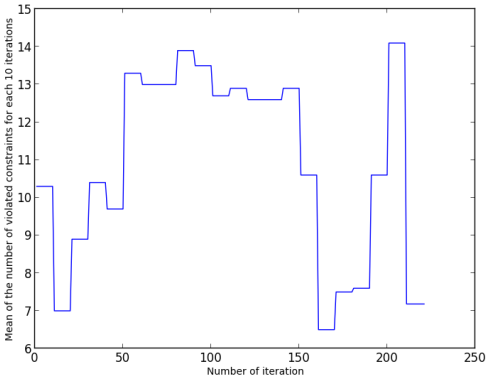
— La somme des violations des contraintes en fonction du nombre d'itérations d'enrichissement,

— La plus grande valeur des contraintes violées en fonction du nombre d'itérations d'enrichissement.

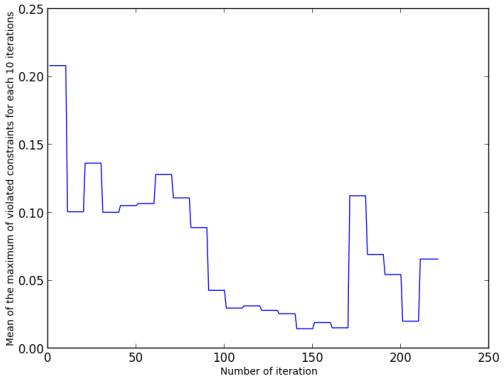
Les résultats de cette optimisation en dimension 124 sont donnés par la figure VI.1



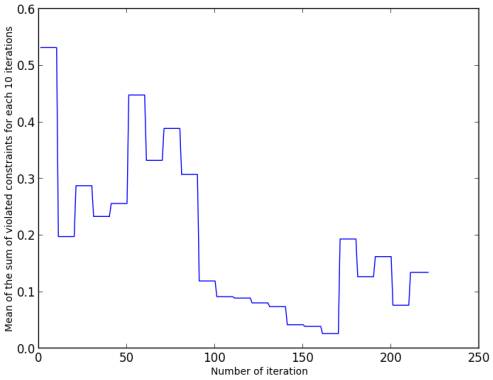
(a) Fonction objectif.



(b) Nombre de contraintes violées.



(c) Maximum des contraintes violées.



(d) Somme des violations de contraintes.

FIGURE VI.1 – Cas test MOPTA à **124 dimensions**. En haut à gauche : Évolution de l'objectif en fonction du nombre d'itérations. En haut à droite : Évolution du nombre de contraintes violées en fonction du nombre d'itérations. En bas à gauche : Évolution de la valeur maximale de contraintes violées en fonction du nombre d'itérations. En bas à droite : Évolution de la somme des violations des contraintes en fonction du nombre d'itérations.

L'évolution de l'objectif de cette optimisation est décroissante au cours des premières itérations comme le montre la figure VI.3a. Cette évolution commence à se dégrader à partir de l'itération 165. En effet, la valeur de l'objectif de l'itération 165 est égale à 236 alors que les valeurs de l'objectif des dernières itérations qui la précèdent sont égales à 225 (avec un niveau de violation des contraintes quasi-équivalent). Au niveau du respect des contraintes, nous n'avons pas une évolution similaire entre le nombre de contraintes violées et la somme des violations des contraintes, comme le montrent respectivement les figures VI.3b et VI.3d. Le nombre de points d'enrichissement utilisés pour ce cas est très faible lorsque l'on compare avec le nombre de points d'enrichissement utilisés par Régis dans ses travaux (Cf. [Regis, 2014a] où 4000 points d'enrichissement sont utilisés). À partir d'un certain nombre d'itérations, l'enrichissement commence à se focaliser dans une zone de recherche très réduite avec des points très proches, ce qui engendre un mauvais conditionnement de la matrice

de covariance du modèle de krigeage et des difficultés dans la construction du modèle. Parmi les solutions pour contourner ce type de difficulté, l'une d'elle serait d'éliminer quelques points très proches pour améliorer le conditionnement de la matrice de covariance, comme le recommande [Sasena, 2002]. Ceci n'a pas été traité au cours de ces travaux. C'est pourquoi nous nous sommes limités à 221 points d'enrichissement au total.

Toutefois, quelques points intéressants sont retrouvés et ils sont donnés par la table VI.1, où la solution de référence est donnée en bleu. De plus, nous présentons dans cette table le nombre de coordonnées trouvées proches de la solution de référence sous la forme d'un pourcentage. En effet, ce pourcentage est calculé de la manière suivante :

— points très proches, Card($|\mathbf{x}-\mathbf{x}^{\text{ref}}| \leq 0.01$)$\frac{100}{d}$%, donnés avec la couleur verte,

— points moyennement proches, Card($|\mathbf{x}-\mathbf{x}^{\text{ref}}| > 0.01$ et $|\mathbf{x}-\mathbf{x}^{\text{ref}}| \leq 0.1$)$\frac{100}{d}$%, donnés avec la couleur orange,

— points loin, Card($|\mathbf{x}-\mathbf{x}^{\text{ref}}| > 0.1$)$\frac{100}{d}$%, donnés avec la couleur rouge,

avec $\mathbf{x}$ une solution trouvée par l'algorithme SEGOKPLS+K, $\mathbf{x}^{\text{ref}}$ la solution de référence et Card le nombre de composantes du vecteur $|\mathbf{x}-\mathbf{x}^{\text{ref}}|$ satisfaisant la formule d'inégalité.

TABLE VI.1 – Valeur de la fonction objectif, nombre de contraintes violées, somme des violations des contraintes et le pourcentage du nombre de coordonnées des solutions trouvées proches de la solution de référence (≤ 0.01 ;]0.01,0.1] ;>0.1) pour le cas test MOPTA à 124 dimensions, avec un plan d'expériences initial de 125 points et 221 points d'enrichissement. La solution de référence est donnée en bleu.

Num. d'itération	Valeur de l'objectif	Nb. de contraintes violées	Somme des violations des contraintes	Pourcentage de variables avec Card($\|\mathbf{x}-\mathbf{x}^{\text{ref}}\|$) ($\leq 0.01$;]0.01,0.1] ;>0.1)
Solution référence	**222.23**	**0**	**0**	(100 ;0 ;0)
164	225.34	9	0.003	(40.32 ;22.58 ;37.10)
166	225.29	10	0.004	(41.13 ;17.74 ;41.13)
167	228.47	2	0.02	(26.62 ;31.45 ;41.93)
178	237.26	2	0.001	(23.39 ;32.26 ;44.35)

D'après la table VI.1, environ 41% des coordonnées des solutions données par les itérations 164 et 166 correspondent aux coordonnées de la MSC du problème (avec une erreur inférieure à 0.01 par rapport aux valeurs de la MSC du problème). Cependant, un pourcentage quasi-équivalent correspond à une différence supérieure à 0.1 entre les coordonnées des itérations 164 ou 166 avec

les valeurs de la MSC du problème. En parallèle, le nombre de contraintes violées est relativement grand pour ces itérations en comparant avec les itérations 167 et 178 où le nombre de contraintes violées est faible (seulement 2 contraintes violées). Ceci laisse penser que l'algorithme SEGOKPLS+K concentre sa recherche dans deux zones distinctes :

— Une première zone qui est proche de la MSC du problème où nous avons une faible violation des contraintes, mais nous avons en contrepartie un nombre relativement grand de contraintes violées (itérations 164 et 166).

— Une deuxième zone qui est un peu plus loin de la MSC du problème où nous avons un faible nombre de contraintes violées (itérations 167 et 178).

D'autre part, nous présentons dans la figure VI.2 une cartographie des contraintes de l'optimisation réalisée. Il s'agit d'un moyen de visualiser avec des couleurs la faisabilité des points de notre base de données avant et après enrichissement :

— contraintes faisables en vert ($\leq 10^{-5}$),

— contraintes non faisables avec une faible violation en rouge ($]10^{-5}, 0.01]$),

— contraintes non faisables avec une forte violation en noir (> 0.01).

D'après cette cartographie, nous remarquons une amélioration des contraintes au niveau des points d'enrichissement par rapport aux points du plan d'expériences initial. Certaines contraintes n'ont jamais été respectées par les points du plan d'expériences initial, comme les contraintes 1 et 9 (notées par C1 et C2 dans la figure VI.2), mais elles ont été améliorées et respectées plusieurs fois par les points d'enrichissement. Nous remarquons aussi que les contraintes avec une faible violation (rouge) sont plus présentes que les contraintes avec une forte violation (noir) au niveau des points d'enrichissement. **En effet, si l'algorithme SEGOKPLS+K ne trouve pas de solution faisable, il cherche malgré tout à trouver un compromis entre toutes les contraintes et à minimiser le maximum de contraintes violées. C'est une observation intéressante car, dans un contexte industriel où aucune solution faisable n'est obtenue, les solutions généralement choisies sont caractérisées par une légère violation sur une ou plusieurs contraintes. Compte tenu du faible nombre d'évaluations totales de la vraie fonction (346 points pour un problème d'optimisation à 124 dimensions), l'algorithme SEGOKPLS parait efficace et aboutit à des résultats très prometteurs**.

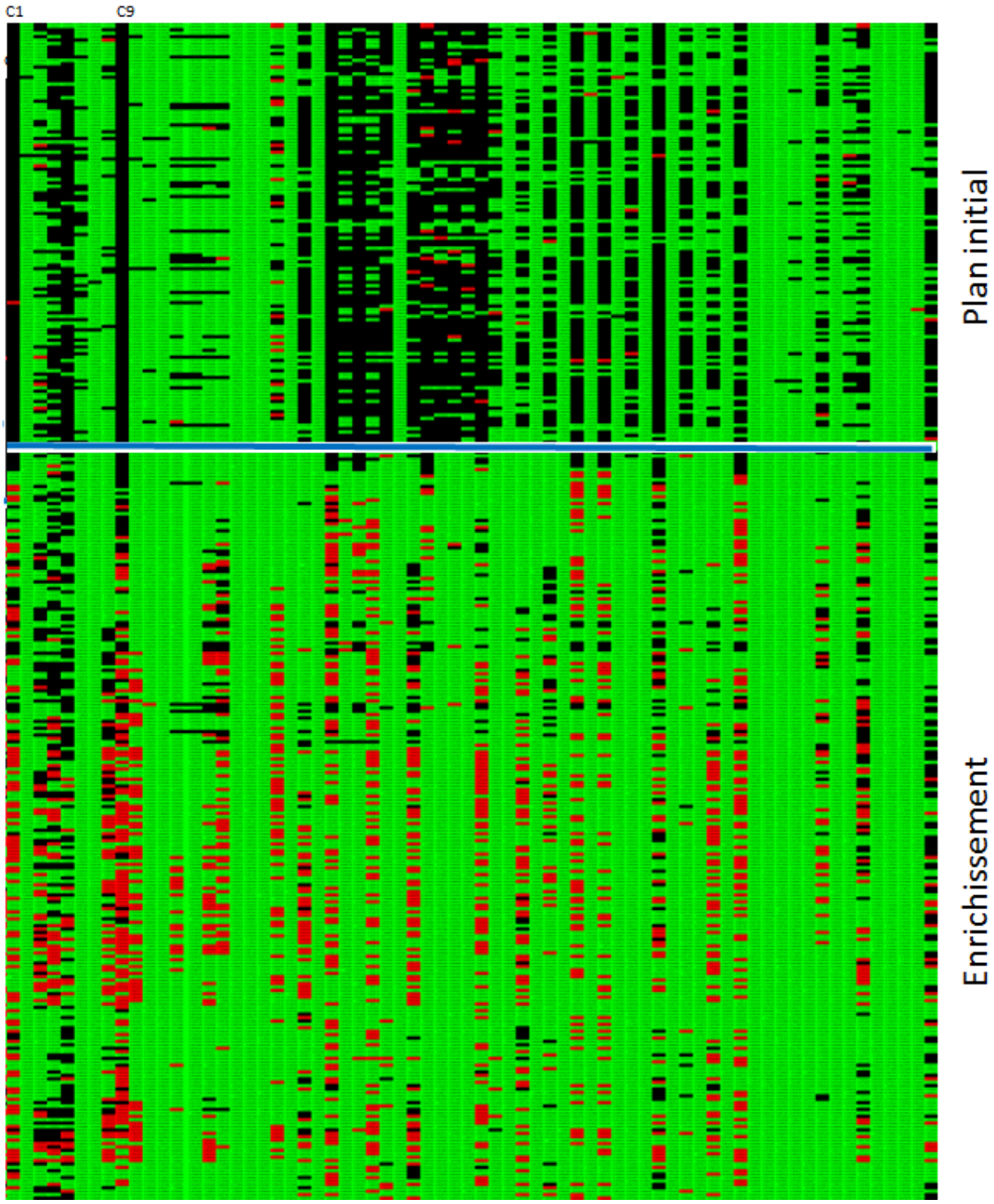


FIGURE VI.2 – Cartographie du cas test MOPTA à **124 dimensions**. Le plan d'expériences initial est au dessus du trait bleu. Les points d'enrichissement sont au dessous du trait bleu. Les contraintes sont données en abscisse et les itérations d'optimisation sont données en ordonnées. Les contraintes faisables sont vertes ($\leq 10^{-5}$), les contraintes non faisables avec une faible violation sont rouges ($]10^{-5}, 0.01]$) et les contraintes non faisables avec une forte violation sont noires (> 0.01).

VI.1.3. Résultats du cas test MOPTA réduit

Dans cette section, trois différentes dimensions sont traitées sur le cas test MOPTA : 12, 50 et 70 dimensions. Pour les deux premiers cas, pour lesquels nous avons réussi à retrouver la MSC du problème, 5 répétitions ont été réalisées avec respectivement 39 et 63 points d'enrichissement. Pour le dernier cas à 70 dimensions, pour lequel nous n'avons pas retrouvé la MSC du problème, 150 itérations ont été réalisées sans répétition.

Remarque 17 *Le choix des variables à optimiser est réalisé d'une manière aléatoire.*

VI.1.3.a. Cas test MOPTA à 12 dimensions

Pour le problème d'optimisation du cas MOPTA à 12 dimensions, nous avons utilisé un plan d'expériences de type hypercube latin avec 13 points. 39 itérations suffisent pour retrouver la MSC du problème. 5 répétitions ont été réalisées pour ce cas test et nous calculons l'évolution de la moyenne de nos différents critères :

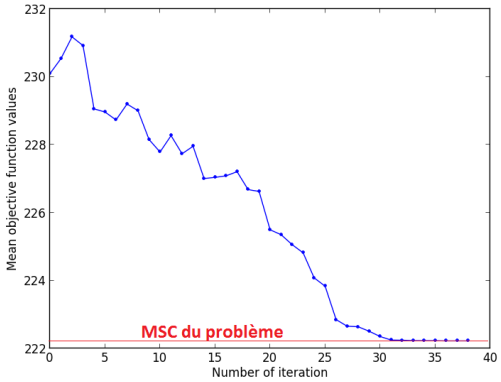
— La fonction objectif en fonction du nombre d'itérations d'enrichissement,

— Le nombre de contraintes violées en fonction du nombre d'itérations d'enrichissement,

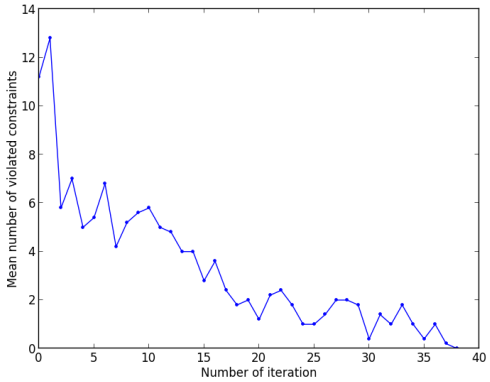
— La somme des violations des contraintes en fonction du nombre d'itérations d'enrichissement,

— La plus grande valeur des contraintes violées en fonction du nombre d'itérations d'enrichissement.

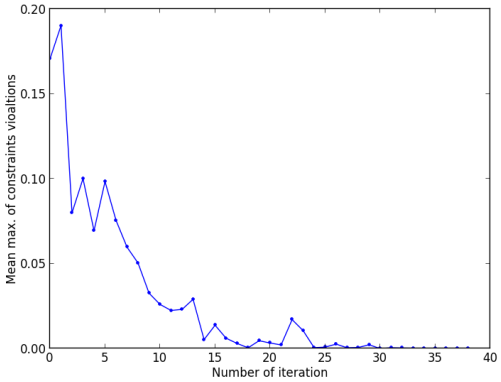
Les résultats de ce problème sont donnés par la figure VI.3.



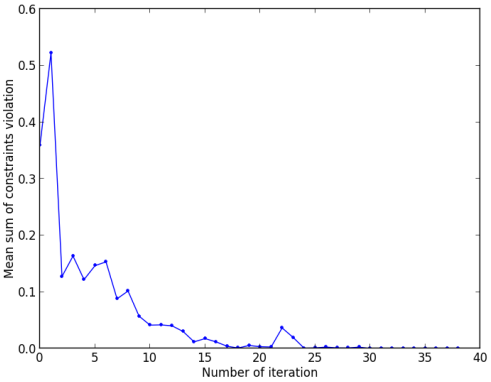
(a) Fonction objectif.



(b) Nombre de contraintes violées.



(c) Maximum des contraintes violées.



(d) Somme des violations de contraintes.

FIGURE VI.3 – Cas test MOPTA à **12 dimensions**. En haut à gauche : Évolution de la moyenne de l'objectif en fonction du nombre d'itérations. En haut à droite : Évolution de la moyenne du nombre de contraintes violées en fonction du nombre d'itérations. En bas à gauche : Évolution de la moyenne de la valeur maximale de contraintes violées en fonction du nombre d'itérations. En bas à droite : Évolution de la moyenne de la somme des violations des contraintes en fonction du nombre d'itérations.

La convergence de ce problème est très rapide. La moyenne du nombre de contraintes violées devient inférieure à 4 contraintes (sur 68 contraintes) à partir de la $15^{ème}$ itération. Parallèlement, la moyenne de la somme des violations des contraintes décroit assez rapidement en passant de 0.52 à la deuxième itération à 0.08 à la $8^{ème}$ itération. L'algorithme SEGOKPLS+K est très performant pour ce cas test où 51 points au total suffisent pour retrouver la MSC du problème (les points d'apprentissage avec les points d'enrichissement). **Les coordonnées des 5 solutions retrouvées correspondent exactement aux coordonnées de la MSC du problème pour ce cas test à 12 dimensions**.

VI.1.3.b. Cas test MOPTA à 50 dimensions

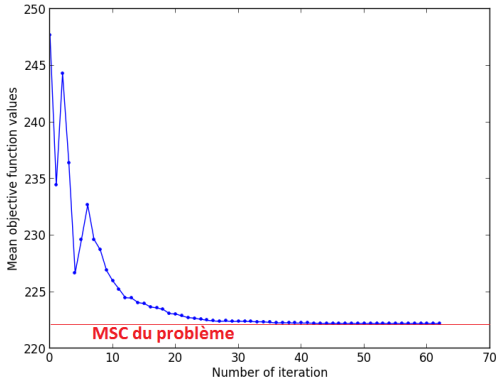
Dans cette section, nous augmentons le nombre de dimensions de 12 à 50 dimensions. Pour ce cas, 51 points dans le plan d'expériences initial, qui est de type hypercube latin, et 63 points d'enrichissement sont utilisés. Comme dans le cas MOPTA à 12 dimensions, 5 répétitions sont réalisées et donc nous représentons l'évolution de la moyenne des mêmes critères que dans la section précédente sur la figure VI.4, qui sont :

— La fonction objectif en fonction du nombre d'itérations d'enrichissement,

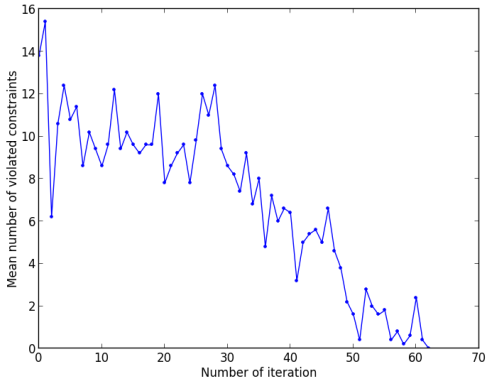
— Le nombre de contraintes violées en fonction du nombre d'itérations d'enrichissement,

— La somme des violations des contraintes en fonction du nombre d'itérations d'enrichissement,

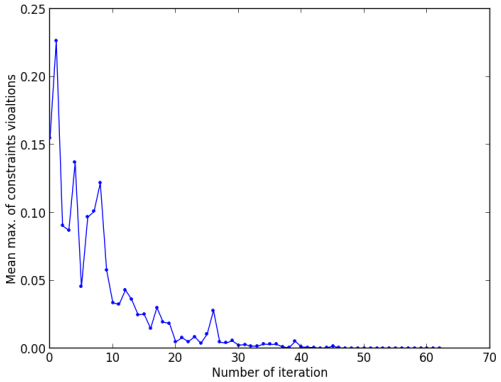
— La plus grande valeur des contraintes violées en fonction du nombre d'itérations d'enrichissement.



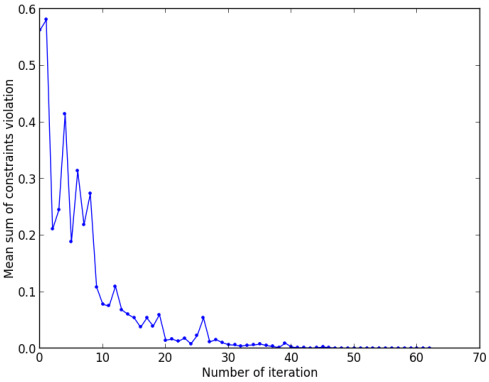
(a) Fonction objectif.



(b) Nombre de contraintes violées.



(c) Maximum des contraintes violées.



(d) Somme des violations de contraintes.

FIGURE VI.4 – Cas test MOPTA à **50 dimensions**. En haut à gauche : Évolution de la moyenne de l'objectif en fonction du nombre d'itérations. En haut à droite : Évolution de la moyenne du nombre de contraintes violées en fonction du nombre d'itérations. En bas à gauche : Évolution de la moyenne de la valeur maximale de contraintes violées en fonction du nombre d'itérations. En bas à droite : Évolution de la moyenne de la somme des violations des contraintes en fonction du nombre d'itérations.

D'après la figure VI.4a, la moyenne de la fonction objectif diminue d'une manière monotone à partir de la $8^{ème}$ itération jusqu'à atteindre la MSC du problème. Au niveau des contraintes, la somme des violations des contraintes diminue assez rapidement comme le montre la figure VI.4d, alors que le nombre de contraintes violées commence à diminuer d'une manière significative à partir de la $30^{ème}$ itération (Cf. figure VI.4b). Pour ce cas, 63 itérations d'enrichissement suffisent pour retrouver la MSC du problème. **Comme dans le cas test MOPTA à 12 dimensions, les coordonnées des 5 solutions retrouvées en 50 dimensions correspondent exactement aux coordonnées de la MSC du problème**.

VI.1.3.c. Cas test MOPTA à 70 dimensions

Nous augmentons maintenant le nombre de dimensions du problème MOPTA à 70 dimensions. 71 points dans le plan d'expériences initial de type hypercube latin et 150 points d'enrichissement sont utilisés. 1 seule réalisation est effectuée pour ce cas test.

De la même manière que dans les cas précédents, nous présentons la moyenne (prise sur 10 itérations d'optimisation) de l'évolution des critères suivants :

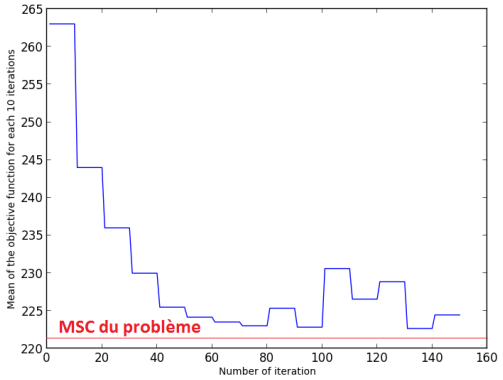
— La fonction objectif en fonction du nombre d'itérations d'enrichissement,

— Le nombre de contraintes violées en fonction du nombre d'itérations d'enrichissement,

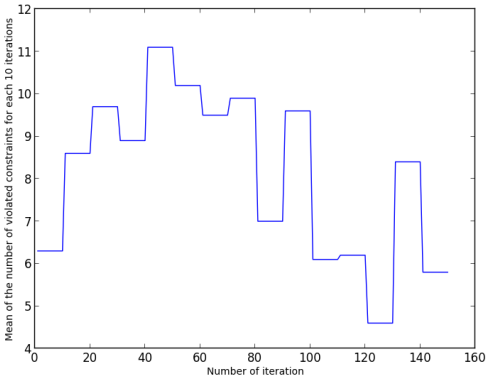
— La somme des violations des contraintes en fonction du nombre d'itérations d'enrichissement,

— La plus grande valeur des contraintes violées en fonction du nombre d'itérations d'enrichissement.

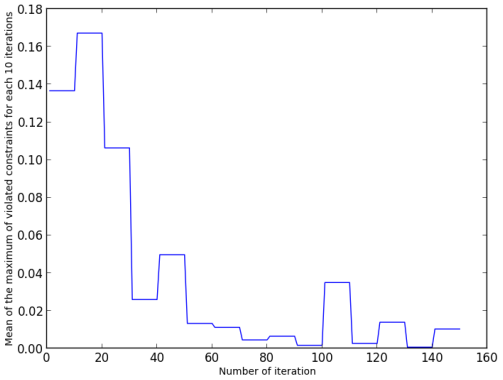
Les résultats de cette optimisation sont donnés par la figure VI.5.



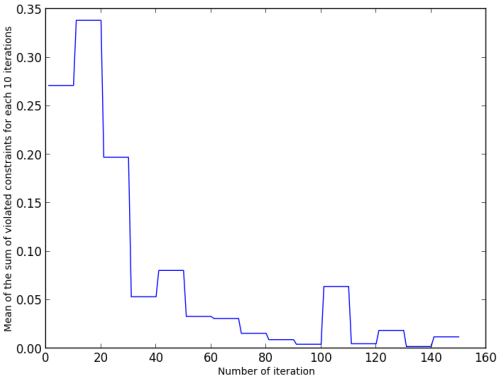
(a) Fonction objectif.



(b) Nombre de contraintes violées.



(c) Maximum des contraintes violées.



(d) Somme des violations de contraintes.

FIGURE VI.5 – Cas test MOPTA à **70 dimensions**. En haut à gauche : Évolution de l'objectif en fonction du nombre d'itérations. En haut à droite : Évolution du nombre de contraintes violées en fonction du nombre d'itérations. En bas à gauche : Évolution de la valeur maximale de contraintes violées en fonction du nombre d'itérations. En bas à droite : Évolution de la somme des violations des contraintes en fonction du nombre d'itérations.

D'après la figure VI.5a, l'algorithme SEGOKPLS+K minimise la fonction objectif d'une manière quasi-monotone, mais des oscillations commencent à apparaître à partir de la $89^{ème}$ itération. Ce phénomène est déjà observé dans le cas MOPTA à 124 dimensions dans la section VI.1.2.. Néanmoins, une solution faisable est obtenue à la $89^{ème}$ itération avec une valeur de l'objectif égale à 233.6. L'évolution du nombre de contraintes violées n'est pas stable comme le montre la figure VI.5b, contrairement à la somme des violations des contraintes où les oscillations sont beaucoup moins visibles (Cf. figure VI.5d).

Au final, deux solutions faisables sont obtenues pour ce cas test (itérations 89 et 118 avec respectivement 233.6 et 235.61 comme valeurs de la fonction objectif). De plus, plusieurs points très proches de la MSC du problème sont obtenus mais avec une légère violation des contraintes. Comme dans le

cas test MOPTA à 124 dimensions, nous présentons les meilleures solutions trouvées par l'algorithme SEGOKPLS+K dans la table VI.2, avec la MSC du problème donnée en bleu.

TABLE VI.2 – Valeur de la fonction objectif, nombre de contraintes violées, somme des violations des contraintes et le pourcentage du nombre de coordonnées des solutions trouvées proches de la solution de référence (≤ 0.01 ;$]0.01,0.1]$;>0.1) pour le cas test MOPTA à 70 dimensions, avec un plan d'expériences initial de 71 points et 150 points d'enrichissement. La solution de référence est donnée en bleu.

Num. d'itération	Valeur de l'objectif	Nb. de contraintes violées	Somme des violations des contraintes	Pourcentage de variables avec Card($\lvert\mathbf{x}-\mathbf{x}^{ref}\rvert$) ($\leq 0.01$;$]0.01,0.1]$;>0.1)
Solution référence	**222.23**	**0**	**0**	(100 ;0 ;0)
89	233.60	0	0	(21.43 ;44.29 ;34.28)
117	222.73	5	0.0003	(50 ;32.86 ;17.14)
118	235.61	0	0	(10 ;50 ;40)
124	222.69	2	5.0310^{-05}	(50 ;31.43 ;18.57)
125	222.69	5	0.0006	(50 ;31.43 ;18.57)
134	222.67	6	0.0006	(57.14 ;28.57 ;14.29)
140	222.65	8	0.0009	(52.86 ;31.43 ;15.71)
144	222.65	4	0.0004	(50 ;34.29 ;15.71)

D'après cette table, nous remarquons quelques similitudes avec le cas test MOPTA à 124 dimensions. En effet, nous avons deux zones de recherche distinctes dans lesquelles l'algorithme SEGOKPLS+K effectue principalement ses recherches :

— Une première zone contenant les deux points faisables (itérations 89 et 118) qui est relativement loin de la MSC du problème.

— Une deuxième zone relative aux itérations 117, 124, 125, 134, 140 et 144, où 50% des coordonnées trouvées sont assez proches de la MSC du problème (erreur inférieure à 0.01). Pour ces itérations, on recense une moyenne d'environ 16% du nombre de coordonnées qui ont une erreur supérieure à 0.1 par rapport à la MSC du problème, avec une somme des violations des contraintes proche du seuil fixé à 10^{-5}.

De la même manière que le cas test MOPTA à 124 dimensions, la figure VI.6 représente une cartographie des contraintes du problème d'optimisation réalisé. Nous rappelons les codes couleurs utilisés dans cette cartographie :

— contraintes faisables en vert ($\leq 10^{-5}$),

— contraintes non faisables avec une faible violation en rouge ($]10^{-5}, 0.01]$),

— contraintes non faisables avec une forte violation en noir (> 0.01).

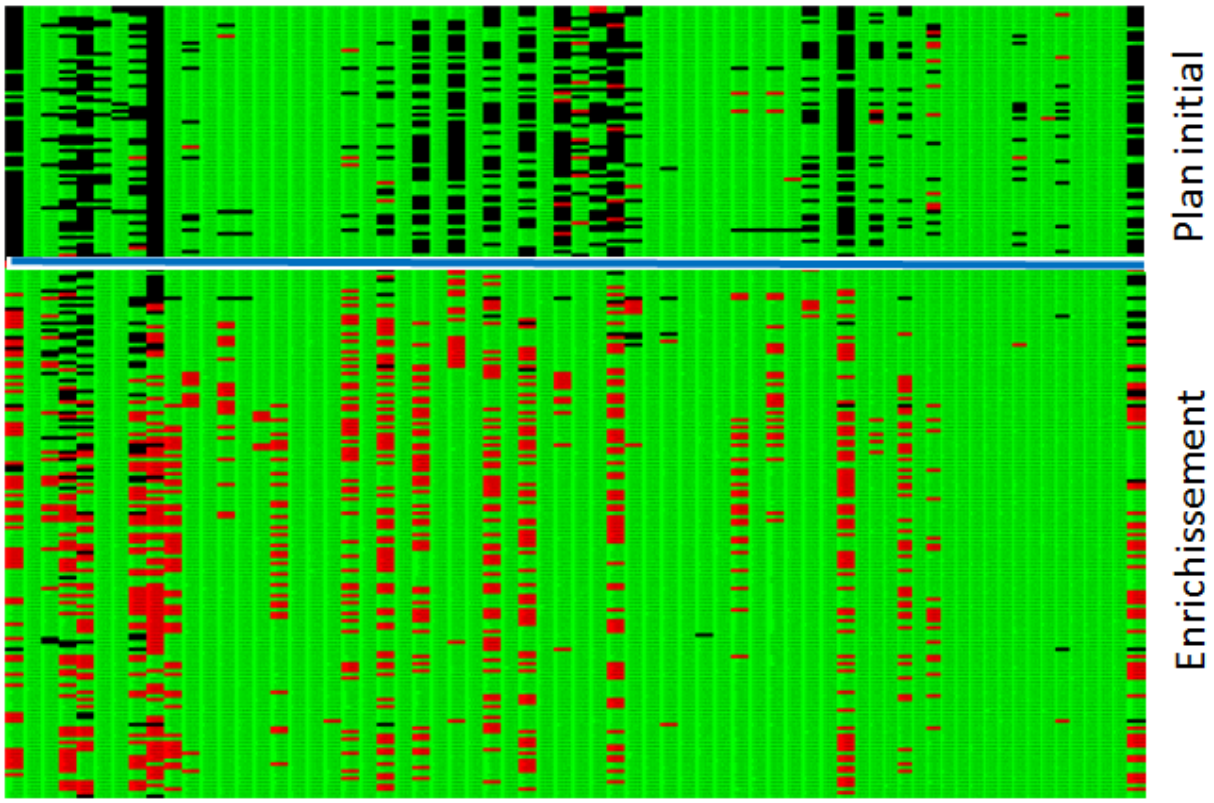


FIGURE VI.6 – Cartographie du cas test MOPTA à **70 dimensions**. Le plan d'expériences initial est au dessus du trait bleu. Les points d'enrichissement sont au dessous du trait bleu. Les contraintes sont données en abscisse et les itérations d'optimisation sont données en ordonnées. Les contraintes faisables sont vertes ($\leq 10^{-5}$), les contraintes non faisables avec une faible violation sont rouges ($]10^{-5}, 0.01]$) et les contraintes non faisables avec une forte violation sont noires (> 0.01).

Comme dans le cas test MOPTA à 124 dimensions, nous remarquons une nette amélioration de la faisabilité des points d'enrichissement par rapport aux points du plan d'expériences initial. Au niveau des points d'enrichissement, le nombre de contraintes avec une forte violation (en noir avec un taux de violation supérieur à 0.01) est faible par rapport aux contraintes avec une légère violation (en rouge avec un taux de violation dans l'intervalle $]10^{-5}, 0.01]$).

VI.1.4. Conclusion du cas test MOPTA

Les résultats obtenus sur le cas test MOPTA sont très concluants jusqu'à la dimension 50. En effet, une convergence assez rapide vers la MSC du problème est observée avec 39 et 63 itérations respectivement pour les cas tests à 12 et 50 dimensions, ce qui est très satisfaisant pour ce type de problème. D'autre part, les résultats des cas tests MOPTA à 70 et 124 dimensions sont très encourageants. Pour ces derniers cas, nous considérons que l'algorithme SEGOKPLS est efficace. En effet, l'algorithme SEGOKPLS parvient à trouver des solutions proches de la MSC avec un taux de violation très faible et avec un faible nombre d'appels aux vrais codes de calcul (plan d'expériences initial + points d'enrichissement). Ce type de performance est généralement recherché et demandé par les industriels car il est souvent très difficile de trouver une solution faisable avec peu de calcul pour leurs problèmes d'optimisation. En outre, un compromis entre les différentes contraintes du problème d'optimisation est souvent recherché, sans forcément respecter toutes les contraintes. Toutefois, plus d'efforts sur les problèmes de conditionnement de la matrice de covariance du modèle de krigeage seraient nécessaires, mais le temps a manqué pour approfondir ces travaux.

VI.2. Test expérimental sur le cas turbine basse pression 2.5 étages

VI.2.1. Descriptif du cas test

Dans cette section, nous appliquons la méthode SEGOKPLS+K sur un cas industriel fourni par la Snecma. Ne disposant pas d'un cas d'étude aéro-mécanique et d'une chaîne associée opérationnelle au moment des tests industriels, nous avons fait le choix d'une optimisation aérodynamique sur la Turbine Basse Pression (TuBP) 2.5 étages. La chaîne de calcul associée est opérationnelle et nous disposons d'un résultat d'optimisation de référence. L'objectif d'une optimisation aérodynamique TuBP 2.5 étages D2 / R2 / D3 / R3 / D4 (Cf. figure VI.7) est d'améliorer le rendement de la TuBP tout en respectant les exigences métiers et en modifiant la géométrie de cinq grilles à la fois.

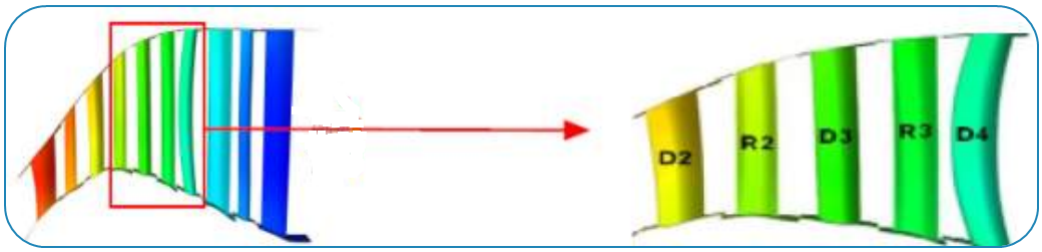


FIGURE VI.7 – Vue en coupe de plusieurs étages de roues. Un étage comporte une roue mobile "R" et une roue fixe "D".

La chaîne de calcul associée à ce problème industriel, appelée Chaîne d'Optimisation Aérodynamique basée sur le Logiciel elsA (COALA), est donnée par la figure VI.8.

Remarque 18 *Pour des raisons de confidentialité, nous ne donnons pas tous les détails de la modéli-*

FIGURE VI.8 – Vue générale de la chaîne COALA sous la plateforme *OPTIMUS*.

sation de l'aubage notamment les informations relatives à la paramétrisation (discrétisation radiale et par profil de l'aubage), les détails des réponses physiques et géométriques analysées et les valeurs numériques associées.

Cette chaîne intègre le processus automatisé de calcul aérodynamique, allant de la génération de la géométrie jusqu'au post-traitement aérodynamique, en passant par la modélisation et la simulation $3D$ aérodynamique des aubages des 5 grilles. Les principales composantes de cette chaîne sont :

— Les variables d'entrée pouvant être optimisées (cadres 1 dans la figure VI.8).

— Une application "maison" permettant la génération de la géométrie de l'aubage (format NURBS) selon les valeurs fixées des variables géométriques (cadre 2 dans la figure VI.8).

— Une application permettant la génération au format CGNS du maillage de l'aubage (cadre 3 dans la figure VI.8).

— Une application de soumission et de rapatriement des calculs aérodynamiques : il s'agit d'un serveur de calcul permettant de réaliser les simulations sur 80 cœurs pour notre problème aérodynamique (cadre 4 dans la figure VI.8).

— Les variables de sortie qui forment l'objectif et les contraintes aérodynamiques (cadres 5 dans la figure VI.8).

La première étape du processus de modélisation est le choix de la paramétrisation. Les profils des aubes de turbine sont paramétrés, à différentes hauteurs, avec des fonctions de type spline afin d'assurer une certaine régularité du profil de l'aubage. Pour des raisons de confidentialité, nous

ne donnons pas le nombre de hauteurs sur l'aubage utilisé et nous ne précisions pas les points de contrôle des splines.

Pour notre problème, nous avons au total 99 variables de conception qui se déclinent comme suit :

— Paramètres d'angle β et de calage (Cf. figure VI.9).

— Points de contrôle des splines sur les courbes extrados et intrados des aubages.

— Paramètres géométriques permettant l'ajout de matière localement en certaines zones spécifiques de l'aubage, par exemple la zone de jonction entre l'aube et le moyeu, afin d'améliorer des critères aérodynamiques et/ou mécaniques.

— Paramètres de compensation en tête de pale.

— Paramètres de pression statique en aval de la turbine.

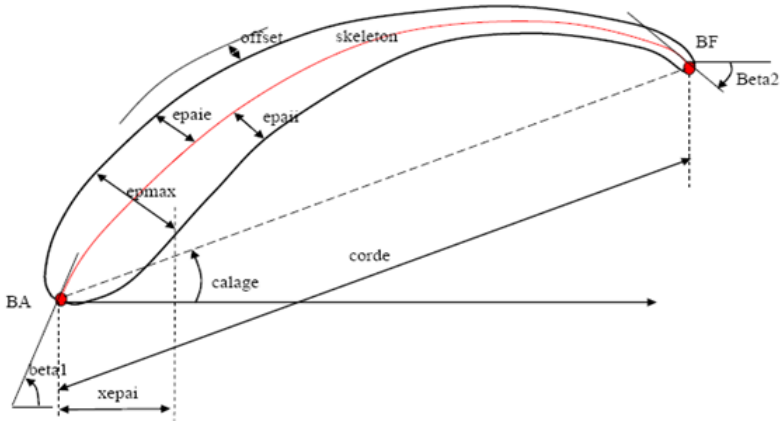


FIGURE VI.9 – Principaux paramètres géométriques d'une coupe d'aube notamment l'angle β et l'angle de calage γ.

Ensuite, nous définissons l'objectif et les contraintes. L'objectif de ce problème d'optimisation est le rendement global de la turbine qui est à maximiser. Le nombre total de sorties de la chaîne d'optimisation 2.5 étages est extrêmement élevé, on en recense environ 400 ce qui rend quasi-impossible l'obtention d'un point faisable en considérant toutes ces sorties comme des contraintes. La chaîne d'optimisation COALA fournit les sorties suivantes :

— Sorties de contrôle de la géométrie.

— Sorties de contrôle de la charge des aubages.

— Sorties de contrôle de la triangulation. Elle consiste à déterminer l'orientation optimale de la vitesse du fluide par rapport à l'aubage en entrée et en sortie d'une grille en vue de maximiser l'efficacité aérodynamique de l'aubage. Elle est essentiellement optimisée via les angles : β_1, β_2 et le calage.

— Sorties de contrôle de l'aérodynamique et thermodynamique étage et turbine complète.

— Sorties de contrôle de la qualité du maillage.

— Sorties de contrôle de la répartition des Mach isentropiques. C'est la vitesse qu'aurait le fluide suite à une transformation sans perte (exemple : sans frottement, sans dissipation turbulente),

correspondant en fait à une transformation adiabatique et réversible du fluide. Déterminé sur des profils à différentes hauteurs d'aubage, le Mach isentropique permet notamment d'identifier les risques de décollement du fluide sur la paroi de l'aubage ainsi que les risques de dépassement de la vitesse du son, tous les deux préjudiciables à l'efficacité aérodynamique de l'aubage. Un exemple de répartition du mach isentropique d'un aubage est donné par la figure VI.10.

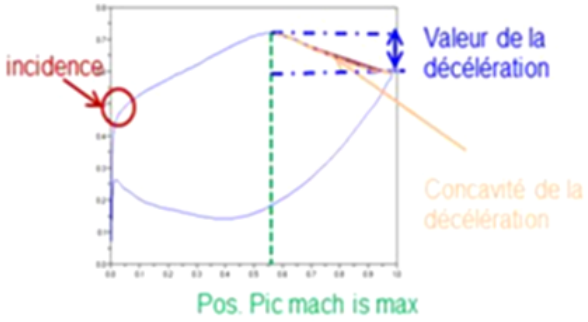


FIGURE VI.10 – Un exemple de répartition du Mach isentropique d'une coupe d'aube.

Le temps de calcul nécessaire pour la chaîne COALA est d'environ 1 heure et 30 minutes sur 80 CPU. Le temps de soumission des calculs est très variable selon le degré de sollicitation du serveur de calcul par les ingénieurs des bureaux d'études. La queue d'attente pour le lancement d'un calcul peut varier de 5 minutes à 10 heures suivant la charge des serveurs de calcul.

La méthodologie actuellement mise en place pour résoudre ce problème très contraint consiste à complexifier progressivement le problème d'optimisation (en termes de contraintes) jusqu'à converger vers une ou plusieurs solutions respectant les contraintes et améliorant l'objectif. **Les contraintes sont rarement respectées en même temps, si bien que la solution retenue est le fruit d'un savant compromis entre des contraintes critiques et d'autres qui le sont moins, plaçant ainsi l'expert métier au centre de la réflexion**.

En résumé, le problème d'optimisation proposé est constitué de :

— 99 variables d'entrée,

— 1 objectif à maximiser qui est le rendement aérodynamique de la turbine,

— 31 contraintes géométriques et aérodynamiques.

De plus, nous avons une solution de référence avec laquelle nous pouvons nous comparer. En effet, cette solution de référence a été obtenue après environ 300 itérations d'optimisation d'un algorithme d'optimisation très utilisé à Snecma. Outre ces 300 itérations d'optimisation, quelques itérations supplémentaires ont été réalisées manuellement, i.e. le concepteur a réalisé quelques évaluations sur des petites modifications de la solution trouvée en se basant sur ses connaissances métier.

En raison des difficultés rencontrées par l'algorithme SEGOKPLS+K en très grande dimension (Cf. la section VI.1.), nous avons décidé de réduire la complexité du problème industriel TuBP à 50 dimensions. Pour ce faire, nous avons fixé 49 variables d'entrée sur les valeurs données par l'optimum de référence. Pour réaliser cette optimisation, nous avons considéré un plan d'expériences

initial de type hypercube latin calculé via la plateforme *OPTIMUS* avec 50 points. 80 itérations d'optimisation ont été réalisées pour ce cas test. Les calculs algorithmiques ont tourné sur une machine Intel(R) Xeon(R) CPU W3550 @3.07GHz desktop machine. En revanche, les simulations aérodynamiques sont réalisées sur un serveur de calcul déporté (calcul parallélisé sur 80 CPU).

VI.2.2. Résultats

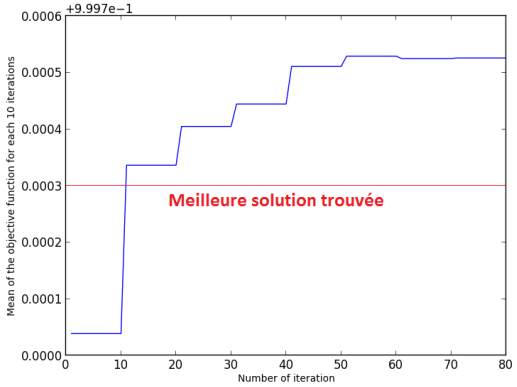
Remarque 19 *Pour des raisons de confidentialité, nous avons divisé tous les résultats du rendement aérodynamique de la turbine par la valeur de la solution de référence fournie par la Snecma.*

Nous rappelons que les contraintes ont été normalisées après l'optimisation pour pouvoir comparer le niveau de violation des contraintes entre elles.

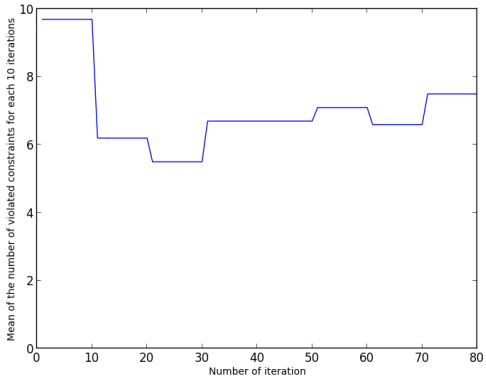
De la même manière que dans les cas tests MOPTA, nous présentons la moyenne (prise sur 10 itérations d'optimisation) de l'évolution des critères suivants :

- La fonction objectif en fonction du nombre d'itérations d'enrichissement,
- Le nombre de contraintes violées en fonction du nombre d'itérations d'enrichissement,
- La somme des violations des contraintes en fonction du nombre d'itérations d'enrichissement,
- La plus grande valeur des contraintes violées en fonction du nombre d'itérations d'enrichissement.

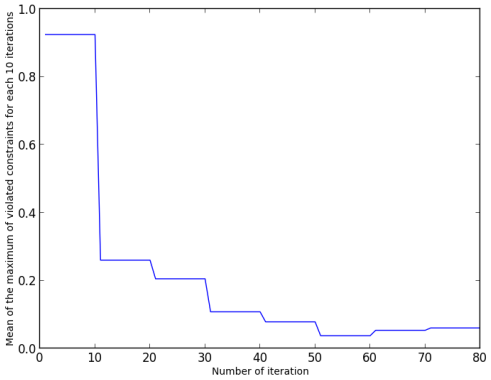
Les résultats de l'optimisation du cas industriel TuBP 2.5 étages sont donnés par la figure VI.11.



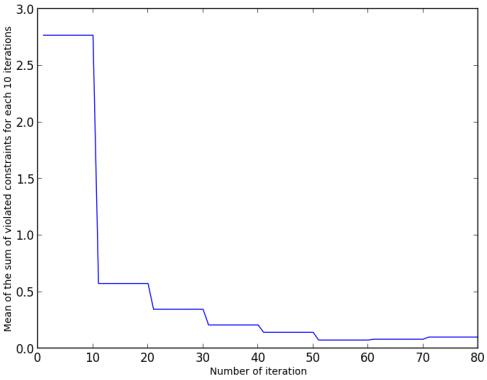
(a) Fonction objectif.



(b) Nombre de contraintes violées.



(c) Maximum des contraintes violées.



(d) Somme des violations de contraintes.

FIGURE VI.11 – Cas test TuBP 2.5 étages à **50 dimensions**. En haut à gauche : Évolution de l'objectif en fonction du nombre d'itérations. En haut à droite : Évolution du nombre de contraintes violées en fonction du nombre d'itérations. En bas à gauche : Évolution de la valeur maximale de contraintes violées en fonction du nombre d'itérations. En bas à droite : Évolution de la somme des violations des contraintes en fonction du nombre d'itérations.

L'algorithme SEGOKPLS+K tente de maximiser le rendement de la turbine comme le montre la figure VI.11a. Nous observons une convergence en moyenne monotone de l'objectif en dépassant très rapidement la solution de référence ; 10 itérations suffisent pour dépasser la valeur de la fonction objective de référence d'après la figure VI.11a.

Parallèlement, nous observons jusqu'à la $22^{ème}$ itération une réduction importante du nombre de contraintes violées : de 13 contraintes violées à l'issue du plan d'expériences initial, nous nous sommes passés à 2 contraintes violées seulement avec une faible somme des violations de contraintes. En résumé, l'algorithme converge de façon satisfaisante jusqu'à la $22^{ème}$ itération. Au delà, nous commençons à constater des échecs de calcul dus essentiellement à l'impossibilité de produire une géométrie ou un maillage correspondant aux valeurs des paramètres géométriques fournies par

l'algorithme SEGOKPLS+K.

Comme l'algorithme SEGOKPLS+K ne dispose pas de solution pour la gestion de ces échecs, cela s'est fatalement traduit par plusieurs tentatives avortées d'enrichissement dans des zones de l'espace de conception où la géométrie n'est pas "réalisable". En conséquence, et en raison de la nature non déterministe de l'enrichissement, les points d'enrichissement qui ont pu être traités par la chaîne sont inévitablement des points présentant de moins bonne qualité d'amélioration de l'objectif et surtout des contraintes, dégradant ainsi l'optimisation. **Ce qui expliquerait qu'à partir de la** $22^{ème}$ **itération, le nombre total de contraintes violées tend à se stabiliser avant d'augmenter. Des travaux complémentaires sont nécessaires pour contourner cette limitation de l'algorithme pour les problèmes industriels. De plus, le nombre limité des points utilisés dans le plan d'expériences favorise l'exploration du domaine de conception, ce qui ralentit énormément l'optimisation**.

En revanche, fait positif, nous observons parallèlement que l'algorithme arrive à réduire la somme totale des violations des contraintes tout en augmentant l'objectif, à défaut de réduire le nombre de contraintes violées, ce qui nous fait dire qu'il a compensé la dégradation du nombre de contraintes respectées par une réduction continue du cumul des violations des contraintes. Ce critère est à prendre en compte car généralement les solutions retenues par les bureaux d'études ne satisfont pas forcément toutes les contraintes du problème d'optimisation. Une certaine marge peut être tolérée sur certaines contraintes. Par conséquent, nous avons obtenu quelques points qui peuvent être intéressants, donnés par la table VI.3.

Remarque 20 *Avant de calculer le pourcentage du nombre de coordonnées des solutions trouvées par l'algorithme SEGOKPLS+K proches des valeurs de la solution de référence, nous avons effectué une transformation de l'espace des entrées dans* $[0,1]^{50}$.

TABLE VI.3 – Valeur de la fonction objectif, nombre de contraintes violées, somme des violations des contraintes et le pourcentage du nombre de coordonnées des solutions trouvées proches de la solution de référence (≤ 0.01 ;$]0.01,0.1]$;>0.1) pour le cas test industriel TuBP 2.5 étages, avec un plan d'expériences initial de 50 points et 80 points d'enrichissement. La solution de référence est donnée en bleu.

Num. d'itération	Valeur de l'objectif à maximiser	Nb. de contraintes violées	Somme des violations des contraintes	Pourcentage de variables avec Card($\lvert\mathbf{x}-\mathbf{x}^{ref}\rvert$) ($\leq 0.01$;$]0.01,0.1]$;>0.1)
Solution référence	**1**	**1**	**0.001117622**	(100 ;0 ;0)
22	1.00015	2	0.047288557	(2,10,88)
35	1.00023	3	0.04551007	(4,12,84)
41	1.00025	5	0.060074966	(2,10,88)
51	1.00026	7	0.032165271	(0,18,82)
52	1.00023	3	0.005123911	(2,18,80)
57	1.00031	7	0.043315866	(0,18,82)
64	1.0002	8	0.016855035	(2,18,80)
65	1.00026	4	0.008680884	(2,12,86)
70	1.00028	7	0.022338087	(2,14,84)
76	1.00033	8	0.08209081	(4,10,86)
77	1.0003	9	0.037626184	(0,16,84)

En effet, une solution violant davantage de contraintes mais avec un cumul de violations très faible sera plus intéressante qu'une autre solution avec un objectif plus faible, violant moins de contraintes, mais avec un cumul de violations beaucoup plus grand. Dans le premier cas, il sera envisageable, en fonction de la nature et de la criticité des contraintes violées, de corriger tout ou partie des contraintes violées en modifiant localement le dessin de la pièce, modification "à la main" non reproductible en automatique dans la chaîne d'optimisation. Cette modification est réalisée avec l'intervention d'un expert métier. Ainsi, en regardant la table VI.3, la solution 52 est préférable à la solution 22 en termes d'objectif et de cumul des violations des contraintes (on est à un ordre de grandeur au dessous par rapport à la solution 22). De plus, les contraintes violées par la solution 52 sont des contraintes géométriques que l'on peut corriger localement d'après l'avis d'un expert. De plus, le gain en rendement de la solution donnée par l'itération 52 est jugé comme un gain moyen par l'expert, mais le temps de restitution de l'optimisation est très intéressant, qui se traduit par le nombre d'évaluation totales de l'optimisation (130 évaluations). D'autre part, en examinant la dernière colonne de la table VI.3, la solution 52 trouve une solution relativement différente de la solution de référence (80% des variables d'entrée ont une différence supérieure à 0.1). L'algorithme semble avoir trouvé une nouvelle zone prometteuse de l'espace de conception permettant ainsi l'exploration de solutions nouvelles et innovantes.

Nous présentons de la même manière que dans le cas MOPTA une cartographie des contraintes donnée par la figure VI.12. Nous rappelons les codes couleurs utilisés dans cette cartographie :

— contraintes faisables en vert ($\leq 10^{-5}$),

— contraintes non faisables avec une faible violation en rouge ($]10^{-5}, 0.01]$),

— contraintes non faisables avec une forte violation en noir (> 0.01).

Remarque 21 *Le nombre de contraintes dans la cartographie donnée par la figure VI.12 est supérieur à 31. En effet, nous avons transformé les contraintes comprises entre deux seuils en deux contraintes séparées. Par exemple, si une contrainte* $g(\mathbf{x})$ *est définie par* $a < g(\mathbf{x}) < b$ *avec* $a < b$ *deux réels, alors nous la transformons en deux contraintes* $a < g(\mathbf{x})$ *et* $g(\mathbf{x}) < b$.

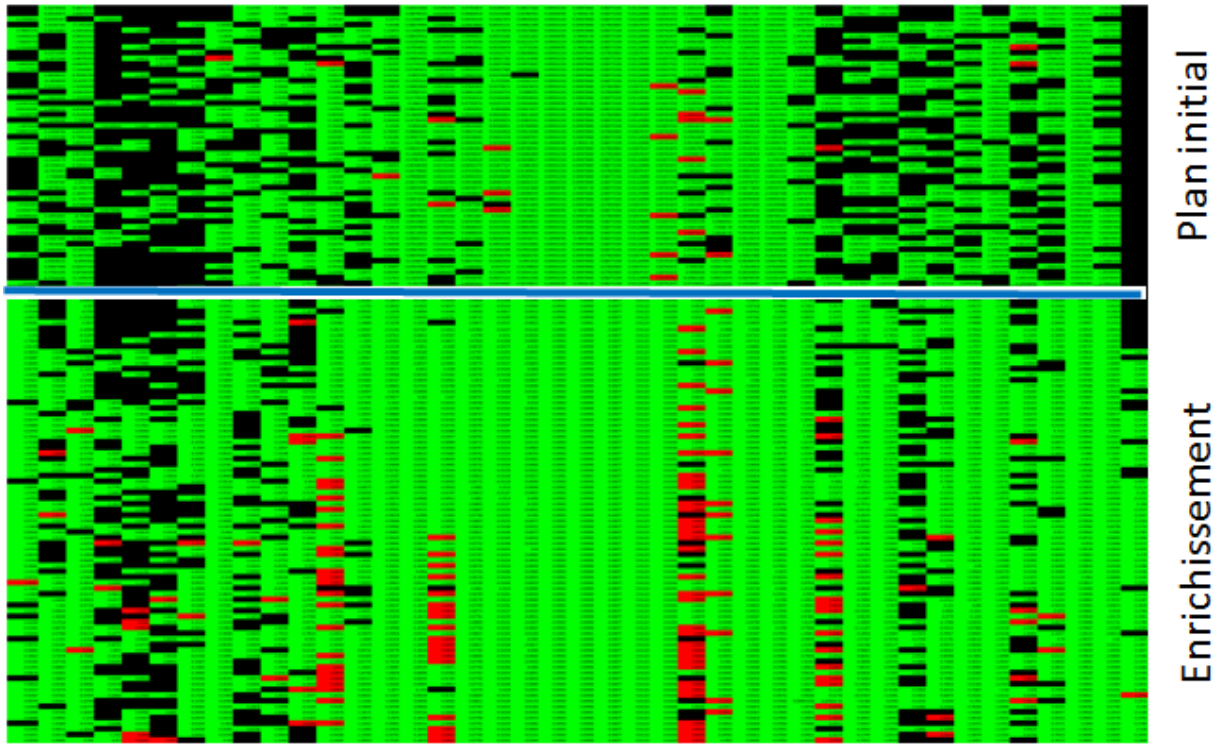


FIGURE VI.12 – Cartographie du cas test TuBP 2.5 étages à **50 dimensions**. Le plan d'expériences initial est au dessus du trait bleu. Les points d'enrichissement sont au dessous du trait bleu. Les contraintes sont données en abscisse et les itérations d'optimisation sont données en ordonnées. Les contraintes faisables sont vertes ($\leq 10^{-5}$), les contraintes non faisables avec une faible violation sont rouges ($]10^{-5}, 0.01]$) et les contraintes non faisables avec une forte violation sont noires (> 0.01).

Malgré une légère amélioration de la faisabilité des points d'enrichissement par rapport aux points du plan d'expériences initial, la cartographie des contraintes montre un antagonisme important entre les contraintes, mettant en lumière la difficulté d'atteindre la faisabilité de ce problème.

VI.2.3. Conclusion du cas TuBP

L'algorithme SEGOKPLS+K a été testé sur un cas test industriel de conception aérodynamique des aubes de turbine basse pression (50 variables de conception et 31 contraintes). L'algorithme a clairement montré une stabilité de la convergence en termes d'objectif et de cumul des violations des contraintes. De 13 contraintes violées à l'issue du plan d'expériences initial, nous avons obtenu des points présentant un intérêt industriel en violant que 2 ou 3 contraintes. En revanche, la convergence en nombre de contraintes violées est apparue relativement instable, voire divergente, en raison d'une part de l'antagonisme des contraintes et d'autre part des échecs de calcul qui ont très certainement "perturbé" la convergence du problème. De plus, le délai imparti pour cette optimisation nous a contraint d'une part à réduire la taille du plan d'expériences initial (avec un nombre de points égal à la dimension du problème), et d'autre part à ne pas considérer de phase intermédiaire de pure recherche de faisabilité. Ces raisons expliquent les résultats, certes prometteurs, mais mitigés

par rapport à l'optimisation de référence, menée à Snecma à l'aide d'une stratégie d'optimisation assistée par métamodèle exploitant un modèle de type RBF. Cela dit, malgré le manque actuel de maturité de la méthode d'optimisation SEGOKPLS+K, l'exploitation du modèle de krigeage PLS dans une stratégie d'optimisation de type EGO a été concluante en termes de qualité des métamodèles employés et de réduction de coût associé.

VI.3. Principaux avantages de l'algorithme SEGOKPLS+K

Les principaux avantages de l'algorithme SEGOKPLS+K observés au cours de ces derniers tests sont :

— Les modèles KPLS+K sont très rapides à construire par rapport aux modèles de krigeage classiques, ce qui nous a permis d'utiliser l'algorithme d'optimisation SEGO pour des problèmes de grande dimension avec un grand nombre de contraintes.

— Une stabilité de l'algorithme SEGOKPLS+K est observée au niveau de sa convergence vers la meilleure solution connue du problème, notamment pour des problèmes d'optimisation allant jusqu'à 50 dimensions. En effet, la solution du problème MOPTA à 12 et 50 dimensions est obtenue à 5 reprises.

— Une stabilité de l'algorithme SEGOKPLS+K est observée au niveau de sa convergence en termes de cumul des violations des contraintes pour tous les tests réalisés.

VI.4. Limitations de l'algorithme SEGOKPLS+K

Plusieurs limitations ont été constatées dans l'algorithme SEGOKPLS+K notamment pour le cas industriel TuBP 2.5 étages :

— La principale difficulté rencontrée par l'algorithme SEGOKPLS+K réside dans le paramétrage de l'optimiseur COBYLA qui n'est pas intuitif comme la majorité des optimiseurs existants de la littérature. À titre d'exemple, nous devons initialiser le pas de recherche de l'algorithme COBYLA, sachant que l'optimisation est très sensible à cette valeur. La valeur de ce paramètre est unique pour toutes les directions de recherche, alors que dans la majorité des problèmes d'optimisation, nous avons généralement des directions de longueurs différentes.

— La version actuelle de l'algorithme SEGOKPLS+K ne peut pas résoudre les problèmes liés aux échecs de simulation ou de construction du modèle géométrique, c'est à dire, les situations où le simulateur numérique ne peut pas effectuer les calculs à cause par exemple d'un maillage impossible à construire pour certaines valeurs de données d'entrée.

— Indépendamment du métamodèle utilisé, l'algorithme d'optimisation SEGO, dans sa version actuelle, n'est pas toujours le mieux adapté pour les problématiques de grande dimension et avec un nombre limité de points dans le plan d'expériences initial. En effet, ce cas de figure (l'utilisation du critère d'enrichissement $wb2$) a tendance à favoriser l'exploration du domaine pour certains problèmes et des critères d'enrichissement plus intelligents et mieux adaptés sont donc à investiguer.

— Pour les problèmes de grande dimension, un nombre de points d'enrichissement relativement grand dans une zone très réduite est généralement nécessaire. Ceci pourrait dégrader le conditionnement de la matrice de covariance des modèles KPLS(+K) au cours des itérations d'enrichissement, ce qui nécessiterait un filtrage des points d'apprentissage au cours de l'optimisation.

VI.5. Conclusion

Dans ce chapitre, nous avons testé l'algorithme SEGOKPLS+K sur deux cas tests industriels, le cas MOPTA de l'automobile dans un premier temps et le cas turbine basse pression 2.5 étages dans un second temps. Le premier cas test traité, avec différentes dimensions, nous a permis d'avoir une idée sur les limitations de l'algorithme notamment en termes de dimensions. L'algorithme SEGOKPLS+K était assez performant sur ce cas test pour les dimensions 12 et 50, avec une convergence très rapide vers la meilleure solution connue du problème. En augmentant le nombre de dimensions à 70 et à 124 dimensions, nous avons rencontré quelques difficultés pour faire converger l'algorithme. Cependant, nous avons réussi à retrouver quelques points avec une légère violation des contraintes et une valeur de la fonction objectif proche de la meilleure solution connue du problème. Le deuxième problème traité nous a permis de tester notre algorithme SEGOKPLS+K dans un vrai contexte industriel, tout en étant confronté aux difficultés techniques et pratiques inhérentes aux problématiques industrielles (échecs de calcul, instabilités du réseau et des serveurs de calcul, etc…).

Ces tests ont montré que les résultats de l'algorithme SEGOKPLS+K sont très prometteurs :

— Convergence de l'algorithme vers des solutions pertinentes en termes d'objectifs et de faisabilité, sur des problématiques de dimension importante et très fortement contraints.

— Intégration réussie et comportement satisfaisant du KPLS+K (qualité et coût de calcul) dans le cadre de la stratégie d'optimisation.

Cependant, des travaux complémentaires seront nécessaires pour améliorer la maturité de la méthode d'optimisation et ainsi permettre son exploitation dans un contexte industriel. Parmi les axes d'amélioration, on peut par exemple citer : la capacité à éviter les zones d'échec du calcul, améliorer la stratégie d'optimisation des critères d'enrichissement, l'enrichissement multi-points, utiliser d'autres optimiseurs pour l'optimisation des critères d'enrichissement, etc…

VII Conclusion et perspectives

Le contexte de cette thèse est celui de l'optimisation multidisciplinaire des aubages de turbomachine via les modèles d'approximation. Les choix du métamodèle et de la méthode d'optimisation se sont portés respectivement sur le modèle de krigeage et la méthode EGO, comme le détaille le chapitre II. Pour améliorer les performances des méthodes choisies et les adapter à notre contexte, un grand travail sur les aspects théoriques de ces méthodes a été réalisé.

Les modèles de krigeage sont connus pour être très couteux à construire pour les problèmes d'optimisation de grande dimension. Une grande partie des travaux de thèse a été consacrée à ce dernier point. La construction des modèles de krigeage repose essentiellement sur un objet central qui est le noyau. En intégrant les informations extraites par la méthode de réduction dimensionnelle PLS, nous avons réussi à construire des nouveaux noyaux KPLS de bonne qualité très rapidement. Cette construction est basée sur un ensemble d'opérations élémentaires sur les noyaux tout en respectant leurs propriétés de symétrie et de positivité. La démonstration détaillée de la validité des noyaux KPLS est donnée dans le chapitre III. Ce chapitre occupe une place importante dans cette thèse, puisque la construction d'un tel métamodèle nous a permis d'utiliser la méthode EGO pour les problèmes d'optimisation de grande dimension.

Cette stratégie d'optimisation a été décrite et adaptée aux problèmes de grande dimension dans le chapitre IV. En effet, l'adaptation de la méthode EGO aux problèmes de grande dimension consiste tout simplement à remplacer les modèles de krigeage classiques par les nouveaux modèles KPLS.

Lorsque les noyaux utilisés appartiennent à la famille des noyaux de type exponentiel, nous avons démontré que les nouveaux noyaux KPLS sont équivalents aux noyaux de krigeage classiques. Cette équivalence nous a permis de rajouter une étape supplémentaire dans la construction des modèles KPLS, dans le but d'améliorer localement le maximum de la fonction de vraisemblance. Ensuite, nous avons couplé ces nouveaux modèles KPLS+K avec la méthode d'optimisation avec contraintes "SEGO", ce qui est décrit en détails dans le chapitre V.

Au final, nous avons utilisé l'algorithme d'optimisation SEGO avec les deux types de métamodèles développés au cours de ces travaux de thèse, qui sont les modèles KPLS et les modèles KPLS+K (avec une utilisation de noyaux exponentiels) comme illustré dans la figure VII.1.

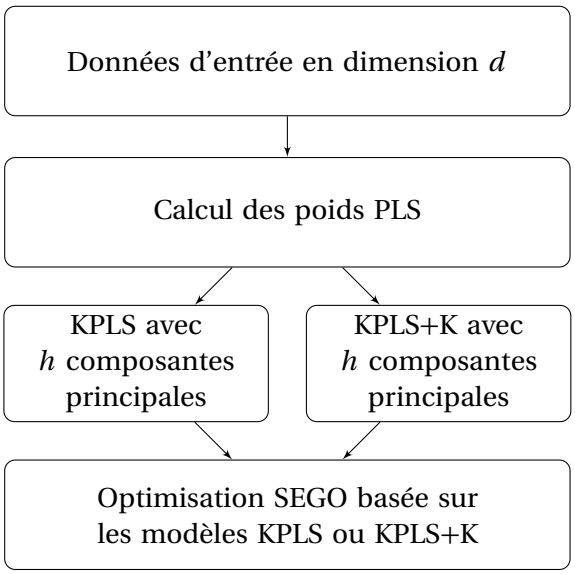


FIGURE VII.1 – Processus d'optimisation développé au cours de ces travaux de thèse avec le choix du modèle KPLS ou KPLS+K.

L'ensemble de ces travaux de thèse a été validé sur des cas académiques et en partie sur des cas industriels. En effet, les problèmes académiques issus de la littérature ont permis de vérifier d'une part la qualité du modèle KPLS et la rapidité de sa construction, et d'autre part la capacité de l'algorithme SEGOKPLS à retrouver les optima. Les méthodes développées ont été également testées sur un problème de l'industrie automobile fourni par la littérature (cas test MOPTA donné dans les sections III.5.3. et VI.1.). Outre le cas test MOPTA, la méthode KPLS a été validée sur des cas industriels Snecma d'aubages de turbomachines dans la section III.5.2.. Des tests complémentaires réalisés par la Snecma ont plus que confirmé les apports bénéfiques de cette évolution du krigeage sur d'autres problématiques industrielles. Il est prévu d'exploiter la méthode KPLS d'une part pour de la prédiction de problèmes de grande taille et très contraints, et d'autre part dans le cadre de stratégies d'optimisation assistée par métamodèle. D'autre part, les tests de l'algorithme d'optimisation sur un cas test industriel de conception des aubages de Turbine Basse Pression (50 dimensions et 31 contraintes), décrits dans la section VI.2., ont montré la pertinence de la méthode proposée (capacité à converger vers un optimum satisfaisant quasiment toutes les contraintes, intérêt de l'usage de modèles KPLS+K dans une stratégie d'optimisation de type EGO) tout en mettant en valeur ses limitations (notamment l'absence de gestion des cas d'échec de calcul) qui n'ont pas permis d'égaler le résultat de référence, rendant par conséquent nécessaire la réalisation de travaux complémentaires pour lever les difficultés rencontrées et pour améliorer la maturité de la méthode proposée.

La méthode KPLS a été codée initialement sous le logiciel Matlab. Pour des raisons d'exploitation en environnement industriel, nous l'avons ré-implémentée en langage Python et son industrialisation

au sein de Snecma est actuellement en cours. De plus, nous l'avons intégrée sous la plateforme OpenMDAO [Gray et al., 2010] à l'ONERA pour une utilisation en multifidélité. Cette intégration a été faite au cours du stage réalisé en collaboration avec l'ISAE-SUAPERO [Vauclin, 2014]. D'autre part, l'algorithme SEGOKPLS a été implémenté en langage Python et il a été couplé à la plateforme *OPTIMUS* via l'interface *sdk Python*.

En termes de communications scientifiques, un premier article sur les modèles KPLS intitulé "Improving kriging surrogates of high-dimensional design models by Partial Least Squares dimension reduction" a été publié dans le journal *Structural Multidisciplinary Optimization*. De plus, un deuxième article sur les modèles KPLS+K intitulé "New approach to estimate kriging hyper-parameters for high-dimensional problems through KPLS method" a été soumis dans le journal *Mathematical Problems in Engineering*. Enfin, un troisième article sur la méthode SEGOKPLS est en cours de rédaction en collaboration avec le professeur R. Régis du département mathématique de l'Université Saint-Joseph de Philadelphie. En outre, une demande de dépôt de brevet intitulée "Procédé de conception de pièces mécaniques, notamment d'aubes de turbomachines" a été déposée en juillet 2015 par la Snecma. Outre les articles et la demande de dépôt de brevet, nous avons présenté nos travaux de thèse dans trois conférences internationales :

— La $8^{ème}$ conférence internationale "PLS2014" sur la méthode PLS et les méthodes connexes à Paris en 2014.

— La $1^{ère}$ conférence internationale "Opti2014" sur les méthodes d'optimisation appliquées en ingénierie à Kos Island en Grèce en 2014.

— La $4^{ème}$ conférence internationale "EngOpt" sur les méthodes d'optimisation à Lisbonne en 2014.

Pour finir avec les perspectives de ces travaux, nous avons testé dans le chapitre V la méthode KPLS+K seulement sur la fonction académique de Griewank sur l'intervalle [-5,5]. Par ailleurs, l'usage d'autres fonctions académiques et l'extraction de l'évolution de l'erreur relative en fonction du ratio $\frac{n}{d}$ est à investiguer. De plus, la gestion des échecs/succès du calcul dans l'algorithme SEGOKPLS pourrait améliorer l'évolution de l'optimisation. Des techniques de gestion des échecs/succès de la littérature pouvant être intégrées dans notre algorithme [Forrester et al., 2006]. D'autres techniques alternatives auraient pu être élaborées et testées avec notre approche comme par exemple :

— Tester d'autres optimiseurs au lieu de l'algorithme COBYLA. Au sein de l'ONERA, un projet de stage est en cours de préparation pour utiliser l'algorithme CMA-ES avec notre algorithme SEGOKPLS [Hansen and Ostermeier, 2001].

— Utiliser la méthode SEGOKPLS dans une stratégie hybride. À titre d'exemple, nous pouvons coupler la méthode SEGOKPLS avec la méthode CMA-ES (nous pouvons citer le travail de [Mohammadi et al., 2015] où l'optimiseur CMA-ES a été utilisé dans la méthode EGO).

— Utiliser les modèles KPLS pour construire des modèles de mélange d'experts comme décrit par [Bettebghor et al., 2011, Liem et al., 2015].

— Intégrer une stratégie d'enrichissement multi-points en parallélisant les calculs, Cf. par exemple [Ginsbourger, 2009].

— Tester d'autres critères d'enrichissement comme le critère de réduction d'incertitudes utilisé par [Picheny, 2014].

A Annexes

I.1. Définition de noyaux séparables

Un noyau k est séparable si : $\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^d, \exists k_1, \cdots, k_d$ noyaux de $\mathbb{R}^d$ dans $\mathbb{R}$, tel que :

$$k\left(\mathbf{x}, \mathbf{x}'\right) = \prod_{i=1}^{d} k_i\left(\mathbf{x}, \mathbf{x}'\right). \tag{A.1}$$

I.2. Définition des problèmes d'optimisation

Dans cette section, nous définissons les cas tests académiques utilisés pour l'optimisation (Cf. [Regis, 2014a] pour plus de détails). Quelques fonctions contraintes sont modifiées soit par une division par une constante positive, soit par une transformation logarithmique, sans changer la géométrie représentative de l'espace faisable. La transformation logarithmique utilisée est introduite par [Regis and Shoemaker, 2013b] et elle est donnée par l'équation (A.2).

$$plog(x) = \begin{cases} log(1+x) & \text{if} \quad 0 \leq x \\ -log(1-x) & \text{if} \quad x \leq 0. \end{cases} \tag{A.2}$$

Welded Beam (WB4) [Deb, 1998]

Pour $P = 6000$, $L = 14$, $E = 30e^6$, $G = 12e^6$, $t_{max} = 13600$, $s_{max} = 30000$, $x_{max} = 10$, $d_{max} = 0.25$, $M = P(L + \frac{x_2}{2})$, $R = \sqrt{0.25(x_2^2 + (x_1 + x_3)^2)}$, $J = \sqrt{2}x_1x_2(\frac{x_2^2}{12} + 0.25(x_1 + x_3)^2)$, $Pc = \frac{4.013E}{6L^2}x_3x_4^3(1 - 0.25x_3\frac{\sqrt{\frac{E}{G}}}{L})$,

$t_1 = \frac{P}{\sqrt{2}x_1x_2}$, $t_2 = M\frac{R}{J}$, $t = \sqrt{t_1^2 + t_1t_2\frac{x_2}{R} + t_2^2}$, $s = 6P\frac{L}{x_4x_3^2}$ et $d = 4P\frac{L^3}{Ex_4x_3^3}$, nous avons :

$$\begin{cases} \min 1.10471x1^2x_2 + 0.04811x_3x_4(14+x_2) \\ \texttt{s.c.} \\ g_1 = \frac{t-t_{max}}{t_{max}} \leq 0 \\ g_2 = \frac{s-s_{max}}{s_{max}} \leq 0 \\ g_3 = \frac{x_1-x_4}{x_{max}} \leq 0 \\ g_4 = \frac{0.10471x_1^2+0.04811x_3x_4(14+x_2)-5}{5} \leq 0 \\ g_5 = \frac{d-d_{max}}{d_{max}} \leq 0 \\ g_6 = \frac{P-Pc}{P} \leq 0 \\ 0.125 \leq x_1 \leq 10,\ 0.1 \leq x_i \leq 10,\ i = 2,3,4. \end{cases} \tag{A.3}$$

Pressure Vessel Design (PVD4) [Carlos and Efrén, 2002]

$$\begin{cases} \min 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \\ \texttt{s.c.} \\ g_1 = -x_1 + 0.0193x_3 \leq 0 \\ g_2 = -x_2 + 0.00954x_3 \leq 0 \\ g_3 = plog(-\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000) \leq 0 \\ 0 \leq x_1, x_2 \leq 1,\ 0 \leq x_3 \leq 50,\ 0 \leq x_4 \leq 240. \end{cases} \tag{A.4}$$

Gas Transmission Compressor Design (GTCD) [Beightler and Phillips, 1976]

$$\begin{cases} \min 8.61e^5x_1^{\frac{1}{2}}x_2x_3^{\frac{-2}{3}}x_4^{\frac{-1}{2}} + 3.69e^4x_3 + 7.72e^8x_1^{-1}x_2^{0.219} - 765.43e^6x_1^{-1} \\ \texttt{s.c.} \\ g_1 = x_4x_2^{-2} + x_2^{-2} - 1 \leq 0 \\ 20 \leq x_1 \leq 50,\ 1 \leq x_2 \leq 10,\ 20 \leq x_3 \leq 50,\ 0.1 \leq x_4 \leq 60. \end{cases} \tag{A.5}$$

Hesse [Hesse, 1973]

$$\begin{cases} \min -25(x_1-2)^2-(x_2-2)^2-(x_3-1)^2-(x_4-4)^2-(x_5-1)^2-(x_6-4)^2 \\ \texttt{s.c.} \\ g_1=\frac{2-x_1-x_2}{2}\leq 0 \\ g_2=\frac{x_1+x_2-6}{6}\leq 0 \\ g_3=\frac{-x_1+x_2-2}{2}\leq 0 \\ g_4=\frac{x_1-3x_2-2}{2}\leq 0 \\ g_5=\frac{4-(x_3-3)^2-x_4}{4}\leq 0 \\ g_6=\frac{4-(x_5-3)^2-x_6}{4}\leq 0 \\ 0\leq x_1\leq 5,\ 0\leq x_2\leq 4,\ 1\leq x_3\leq 5 \\ 0\leq x_4\leq 6,\ 1\leq x_5\leq 5,\ 0\leq x_6\leq 10. \end{cases} \tag{A.6}$$

SR7 [Floudas and Pardalos, 1990]

Pour $A=3.3333x_3^2+14.9334x_3-43.0934$, $B=x_6^2+x_7^2$, $C=x_6^3+x_7^3$, $D=x_4x_6^2+x_5x_7^2$, $A1=[(745\frac{x_4}{x_2x_3})^2+16.91e^6]^{0.5}$, $B1=0.1x_6^3$, $A2=[(745\frac{x_5}{x_2x_3})^2+157.5e^6]^{0.5}$ et $B2=0.1x_7^3$, nous avons

$$\mathbf{f}(x_{opt})=\begin{cases} \min 0.7854x_1x_2^2A-1.508x_1B+7.477C+0.7854D \\ \texttt{s.c.} \\ g_1=\frac{27-x_1x_2^2x_3}{27}\leq 0 \\ g_2=\frac{397.5-x_1x_2^2x_3^2}{397.5}\leq 0 \\ g_3=\frac{1.93-\frac{x_2x_6^4x_3}{x_4^3}}{1.93}\leq 0 \\ g_4=\frac{1.93-\frac{x_2x_7^4x_3}{x_5^3}}{1.93}\leq 0 \\ g_5=\frac{\frac{A1}{B1}-1100}{1100}\leq 0 \\ g_6=\frac{\frac{A2}{B2}-850}{850}\leq 0 \\ g_7=\frac{x_2x_3-40}{40}\leq 0 \\ g_8=\frac{5-\frac{x_1}{x_2}}{5}\leq 0 \\ g_9=\frac{\frac{x_1}{x_2}-12}{12}\leq 0 \\ g10=\frac{1.9+1.5x_6-x_4}{1.9}\leq 0 \\ g_{11}=\frac{1.9+1.1x_7-x_5}{1.9}\leq 0 \\ 2.6\leq x_1\leq 3.6,\ 0.7\leq x_2\leq 0.8 \\ 17\leq x_3\leq 28,\ 7.3\leq x_4,x_5\leq 8.3 \\ 2.9\leq x_6\leq 3.9,\ 5\leq x_7\leq 5.5. \end{cases} \tag{A.7}$$

g_{02} **[Michalewicz and Schoenauer, 1996]**

$$\begin{cases} \min -\left|\frac{\sum_{i=1}^{d}\cos^4(x_i)-2\prod_{i=1}^{d}\cos^2(x_i)}{\sqrt{\sum_{i=1}^{d} i x_i^2}}\right| \\ \texttt{s.c.} \\ g_1 = \frac{plog(-\prod_{i=1}^{d} x_i+0.75)}{plog(10^d)} \leq 0 \\ g_2 = \frac{\sum_{i=1}^{d} x_i-7.5d}{2.5d} \leq 0 \\ 0 \leq x_i \leq 10,\ i = 1,\cdots,d. \end{cases} \tag{A.8}$$

g_{03} **[Michalewicz and Schoenauer, 1996]**

$$\begin{cases} \min -plog\left[(\sqrt{d})^d \prod_{i=1}^{d} x_i\right] \\ \texttt{s.c.} \\ g_1 = \sum_{i=1}^{d} x_i^2 - 1 \leq 0 \\ 0 \leq x_i \leq 1,\ i = 1,\cdots,d. \end{cases} \tag{A.9}$$

g_{04} **[Michalewicz and Schoenauer, 1996]**

Pour $u = 85.334407+0.0056858x_2x_5+0.0006262x_1x_4-0.0022053x_3x_5$, $v = 80.51249+0.0071317x_2x_5+0.0029955x_1x_2+0.0021813x_3^2$, $w = 9.300961+0.0047026x_3x_5+0.0012547x_1x_3+0.0019085x_3x_4$, nous avons

$$\begin{cases} \min 5.3578547x_3^2+0.8356891x_1x_5+37.293239x_1-40792.141 \\ \texttt{s.c.} \\ g_1 = -u \leq 0 \\ g_2 = u-92 \leq 0 \\ g_3 = -v+90 \leq 0 \\ g_4 = v-110 \leq 0 \\ g_5 = -w+20 \leq 0 \\ g_6 = w-25 \leq 0 \\ 78 \leq x_1 \leq 102,\ 33 \leq x_2 \leq 45,\ 27 \leq x_i \leq 45,\ i = 3,4,5. \end{cases} \tag{A.10}$$

g_{05} **[Michalewicz and Schoenauer, 1996]**

$$\begin{cases} \min 3x_1 + e^{-6}x_1^3 + 2x_2 + \frac{2e^{-6}}{3}x_2^3 \\ \texttt{s.c.} \\ g_1 = x_3 - x_4 - 0.55 \leq 0 \\ g_2 = x_4 - x_3 - 0.55 \leq 0 \\ g_3 = 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 \leq 0 \\ g_4 = 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 \leq 0 \\ g_5 = 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 \leq 0 \\ 0 \leq x_1, x_2 \leq 1200, \ -0.55 \leq x_3, x_4 \leq 0.55. \end{cases} \tag{A.11}$$

g_{06} **[Michalewicz and Schoenauer, 1996]**

$$\begin{cases} \min(x_1 - 10)^3 + (x_2 - 20)^3 \\ \texttt{s.c.} \\ g_1 = \frac{(-(x_1-5)^2 - (x_2-5)^2 + 100)}{100} \\ g_2 = \frac{((x_1-6)^2 - (x_2-5)^2 + 82.81)}{82.81} \\ 13 \leq x_1 \leq 100, 0 \leq x_2 \leq 100. \end{cases} \tag{A.12}$$

g_{07} **[Michalewicz and Schoenauer, 1996]**

$$\begin{cases} \min x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3-10)^2 + 4(x_4-5)^2 + (x_5-3)^2 \\ \quad +2(x_6-1)^2 + 5x_7^2 + 7(x_8-11)^2 + 2(x_9-10)^2 + (x_{10}-7)^2 + 45 \\ \texttt{s.c.} \\ g_1 = \frac{4x_1+5x_2-3x_7+9x_8-105}{105} \leq 0 \\ g_2 = \frac{10x_1-8x_2-17x_7+2x_8}{370} \leq 0 \\ g_3 = \frac{-8x_1+2x_2+5x_9-2x_{10}-12}{158} \leq 0 \\ g_4 = \frac{3(x_1-2)^2+4(x_2-3)^2+2x_3^2-7x_4-120}{1258} \leq 0 \\ g_5 = \frac{5x_1^2+8x_2+(x_3-6)^2-2x_4-40}{816} \leq 0 \\ g_6 = \frac{0.5(x_1-8)^2+2(x_2-4)^2+3x_5^2-x_6-30}{834} \leq 0 \\ g_7 = \frac{x_1^2+2(x_2-2)^2-2x_1x_2+14x_5-6x_6}{788} \leq 0 \\ g_8 = \frac{-3x_1+6x_2+12(x_9-8)^2-7x_{10}}{4048} \leq 0 \\ -10 \leq x_i \leq 10, \ i = 1, \cdots, 10. \end{cases} \tag{A.13}$$

g_{08} **[Michalewicz and Schoenauer, 1996]**

$$\begin{cases} \min \frac{-\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)} \\ \texttt{s.c.} \\ g_1 = x_1^2 - x_2 + 1 \leq 0 \\ g_2 = 1 - x_1 + x_2^2 \leq 0 \\ -10 \leq x_1, x_2 \leq 10. \end{cases} \tag{A.14}$$

g_{09} **[Michalewicz and Schoenauer, 1996]**

$$\begin{cases} \min(x_1-10)^2 + 5(x_2-12)^2 + x_3^4 + 3(x_4-11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 \\ \texttt{s.c.} \\ g_1 = \frac{2x_1^2+3x_2^4+x_3+4x_4^2+5x_5-127}{127} \leq 0 \\ g_2 = \frac{7x_1+3x_2+10x_3^2+x_4-x_5-282}{282} \leq 0 \\ g_3 = \frac{23x_1+x_2^2+6x_6^2-8x_7-196}{196} \leq 0 \\ g_4 = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0 \\ -10 \leq x_i \leq 10,\ i = 1, \cdots, 7. \end{cases} \tag{A.15}$$

g_{10} **[Michalewicz and Schoenauer, 1996]**

$$\begin{cases} \min x_1 + x_2 + x_3 \\ \texttt{s.c.} \\ g_1 = -1 + 0.0025(x_4 + x_6) \leq 0 \\ g_2 = -1 + 0.0025(-x_4 + x_5 + x_7) \leq 0 \\ g_3 = -1 + 0.01(-x_5 + x_8) \leq 0 \\ g_4 = plog(100x_1 - x_1x_6 + 833.33252x_4 - 83333.333) \leq 0 \\ g_5 = plog(x_2x_4 - x_2x_7 - 1250x_4 + 1250x_5) \leq 0 \\ g_6 = plog(x_3x_5 - x_3x_8 - 2500x_5 + 1250000) \leq 0 \\ 10^2 \leq x_1 \leq 10^4,\ 10^3 \leq x_2, x_3 \leq 10^4, \\ 10 \leq x_i \leq 10^3,\ i = 4, 5, 6, 7, 8. \end{cases} \tag{A.16}$$

I.3. Résultats de la fonction de Griewank sur l'intervalle [-5,5]

Les statistiques (stat.), la moyenne (mean) et l'écart-type (std), de 10 répétitions de la fonction de Griewank sur l'intervalle [-5,5] sont reportés dans les tables A.1 et A.2.

TABLE A.1 – Résultats de la fonction de Griewank en $20D$ sur l'intervalle $[-5,5]$. 10 répétitions sont réalisées pour chaque cas test (50, 100, 200 and 300 points d'apprentissage).

modèle	stat.	50 points		100 points		200 points		300 points	
		ER (%)	CPU	ER (%)	CPU	ER (%)	CPU	ER (%)	CPU
Kriging	mean	0.62	30.43 s	0.43	40.09 s	0.15	120.74 s	0.16	94.31 s
	std	0.03	9.03 s	0.04	11.96 s	0.02	27.49 s	0.06	21.92 s
KPLS1	mean	0.54	0.05 s	0.53	0.12 s	0.48	0.43 s	0.45	0.89 s
	std	0.03	0.007 s	0.03	0.02 s	0.03	0.08 s	0.03	0.02 s
KPLS2	mean	0.52	0.11 s	0.48	1.04 s	0.42	1.14 s	0.38	2.45 s
	std	0.03	0.05 s	0.04	0.97 s	0.04	0.92 s	0.04	1 s
KPLS3	mean	0.51	1.27 s	0.46	3.09 s	0.37	3.56 s	0.35	3.52 s
	std	0.03	1.29 s	0.06	3.93 s	0.03	2.75 s	0.06	1.38 s

TABLE A.2 – Résultats de la fonction de Griewank en $60D$ sur l'intervalle $[-5,5]$. 10 répétitions sont réalisées pour chaque cas test (50, 100, 200 and 300 points d'apprentissage).

modèle	stat.	50 points		100 points		200 points		300 points	
		ER (%)	CPU	ER (%)	CPU	ER (%)	CPU	ER (%)	CPU
Kriging	mean	1.39	560.19 s	1.04	920.41 s	0.83	2015.39 s	0.65	2894.56 s
	std	0.15	200.27 s	0.05	231.34 s	0.04	239.11 s	0.03	728.48 s
KPLS1	mean	0.92	0.07 s	0.87	0.10 s	0.82	0.37 s	0.79	0.86 s
	std	0.02	0.02 s	0.02	0.007 s	0.02	0.02 s	0.03	0.04 s
KPLS2	mean	0.91	0.43 s	0.87	0.66 s	0.78	2.92 s	0.74	1.85 s
	std	0.03	0.54s	0.02	1.06 s	0.02	2.57 s	0.03	0.51 s
KPLS3	mean	0.92	1.57 s	0.86	3.87 s	0.78	6.73 s	0.70	20.01 s
	std	0.04	1.98 s	0.02	5.34 s	0.02	10.94 s	0.03	26.59 s

TABLE A.3 – Résultats de la fonction de Griewank en $20D$ sur l'intervalle $[-5,5]$. 10 répétitions sont réalisées pour chaque cas test (50, 100, 200 and 300 points d'apprentissage).

modèle	stat.	50 points		100 points		200 points		300 points	
		ER (%)	CPU	ER (%)	CPU	ER (%)	CPU	ER (%)	CPU
Kriging	mean	0.62	30.43 s	0.43	40.09 s	0.15	120.74 s	0.16	94.31 s
	std	0.03	9.03 s	0.04	11.96 s	0.02	27.49 s	0.06	21.92 s
KPLS1+K	mean	0.59	1.20 s	0.45	2.42 s	0.20	8.00 s	0.17	19.07 s
	std	0.04	0.16 s	0.07	0.44 s	0.04	1.51 s	0.07	3.19 s
KPLS2+K	mean	0.58	1.28 s	0.42	3.38 s	0.18	9.71 s	0.16	19.89 s
	std	0.04	0.15 s	0.05	1.06 s	0.02	1.29 s	0.05	2.67 s
KPLS3+K	mean	0.58	2.45 s	0.41	5.61 s	0.16	11.67 s	0.16	20.49 s
	std	0.03	1.32 s	0.05	3.99 s	0.02	3.88 s	0.05	3.46 s

TABLE A.4 – Résultats de la fonction de Griewank en $60D$ sur l'intervalle $[-5,5]$. 10 répétitions sont réalisées pour chaque cas test (50, 100, 200 and 300 points d'apprentissage).

modèle	stat.	50 points		100 points		200 points		300 points	
		ER (%)	CPU	ER (%)	CPU	ER (%)	CPU	ER (%)	CPU
Kriging	mean	1.39	560.19 s	1.04	920.41 s	0.83	2015.39 s	0.65	2894.56 s
	std	0.15	200.27 s	0.05	231.34 s	0.04	239.11 s	0.03	728.48 s
KPLS1+K	mean	0.99	2.14 s	0.90	2.90 s	0.76	9.88 s	0.66	22.00 s
	std	0.03	0.72 s	0.03	0.03 s	0.03	0.06 s	0.02	0.15 s
KPLS2+K	mean	0.98	2.44 s	0.88	3.44 s	0.75	12.38 s	0.60	23.03 s
	std	0.04	0.63s	0.02	1.06 s	0.03	2.56 s	0.03	0.50 s
KPLS3+K	mean	0.99	3.82 s	0.88	6.68 s	0.74	16.18 s	0.61	41.13 s
	std	0.05	2.33 s	0.03	5.34 s	0.03	10.95 s	0.03	26.59 s

Bibliographie

[Alberto and González, 2012] Alberto, P. R. and González, F. G. (2012). Partial least squares regression on symmetric positive-definite matrices. *Revista Colombiana de Estadística*, 36(1) :177–192.

[Audet and Denis, 2006] Audet, C. and Denis, J. E. J. (2006). Mesh adaptive direct search algorithms for constrained optimizaton. *SIAM Journal on Optimization*, 17(2) :188–217.

[Bachoc, 2013] Bachoc, F. (2013). Cross Validation and Maximum Likelihood estimation of hyper-parameters of Gaussian processes with model misspecification. *Computational Statistics and Data Analysis*, 66 :55–69.

[Beightler and Phillips, 1976] Beightler, C. S. and Phillips, D. T. (1976). *Applied geometric programming*. Wiley.

[Bellman, 1961] Bellman, R. E. (1961). *Adaptive control processes - A guided tour*. Princeton University Press, Princeton, New Jersey, U.S.A.

[Bennett and Campbell, 2000] Bennett, K. P. and Campbell, C. (2000). Support vector machines : Hype or hallelujah ? *SIGKDD Explor. Newsl.*, 2(2) :1–13.

[Bettebghor et al., 2011] Bettebghor, D., Bartoli, N., Grihon, S., Morlier, J., and Samuelides, M. (2011). Surrogate modeling approximation using a mixture of expert based on EM joint estimation. *Structural and Multidisciplinary Optimization*, 43(2) :243–259.

[Carlos and Efrén, 2002] Carlos, A. C. and Efrén, M. (2002). Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics*, 16 :2002.

[Chen et al., 2012] Chen, L. L., Liao, C.and Lin, W., Chang, L., and Zhong, X. M. (2012). Hybrid-surrogate-model-based efficient global optimization for high-dimensional antenna design. *Progress In Electromagnetics Research*, 124 :85–100.

[Cheng et al., 2015] Cheng, G. H., Younis, A., Hajikolaei, K. H., and Wang, G. G. (2015). Trust region based mode pursuing sampling method for global optimization of high dimensional design problems. *Journal of Mechanical Design*, 137(2) :021407.

[Cristianini and Shawe-Taylor, 2000] Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines : And Other Kernel-based Learning Methods.* Cambridge University Press, New York, NY, USA.

[De Jong, 1993] De Jong, S. (1993). SIMPLS : An alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 18(3) :251–263.

[Deb, 1998] Deb, K. (1998). An efficient constraint handling method for genetic algorithms. In *Computer Methods in Applied Mechanics and Engineering*, pages 311–338.

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

[Floudas and Pardalos, 1990] Floudas, C. A. and Pardalos, P. M. (1990). *A Collection of Test Problems for Constrained Global Optimization Algorithms*. Lecture Notes in Computer Science. Springer.

[Forrester et al., 2008] Forrester, A., Sobester, A., and Keane, A. (2008). *Engineering Design via Surrogate Modelling : A Practical Guide*. Wiley.

[Forrester et al., 2006] Forrester, A. I. J., Sóbester, A., and Keane, A. J. (2006). Optimization with missing data. *Proceedings of the Royal Society of London A : Mathematical, Physical and Engineering Sciences*, 462(2067) :935–945.

[Franco, 2008] Franco, J. (2008). *Exploratory Designs for Computer Experiments of Complex Physical Systems Simulation*. Thèse, Ecole Nationale Supérieure des Mines de Saint-Etienne.

[Friedman, 1991] Friedman, J. H. (1991). Multivariate adaptive regression splines. *Annals of Statistics*.

[Ginsbourger, 2009] Ginsbourger, D. (2009). *Multiples métamodèles pour l'approximation et l'optimisation de fonctions numériques multivariables*. Thèse, Ecole National Supérieure des Mines de Saint-Etienne.

[Ginsbourger and Le Riche, 2009] Ginsbourger, D. and Le Riche, R. (2009). Towards GP-based optimization with finite time horizon.

[Goddard, 1920] Goddard, R. H. (1920). A method of reaching extreme altitudes. *Nature*, 105 :809–811.

[Gray et al., 2010] 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*.

[Hansen and Ostermeier, 2001] Hansen, N. and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, 9(2) :159–195.

[Hastie et al., 2001] Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer, New York, NY, USA.

[Haykin, 1998] Haykin, S. (1998). *Neural Networks : A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition.

[Hesse, 1973] Hesse, R. (1973). A heuristic search procedure for estimating a global solution of nonconvex programming problems. *Operations Research*, 21 :1267–1280.

[Hock and Schittkowski, 1981] Hock, W. and Schittkowski, K. (1981). *Test Examples for Nonlinear Programming Codes*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Hopfield, 1982] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79(8) :2554–2558.

[Jin et al., 2005] 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.

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

[Jones, 2008] Jones, D. R. (2008). Large-scale multi-disciplinary mass optimization in the auto-industry. presented at the *Modeling and Optimization : Theory and Application (MOPTA) 2008 Conference*. Ontario, Canada.

[Jones et al., 1998] 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.

[Keane and Nair, 2005] Keane, A. J. and Nair, P. B. (2005). *Computational approaches to aerospace design : The pursuit of excellence*. John Wiley.

[Kleijnen et al., 2012] Kleijnen, J., Beers, W., and Nieuwenhuyse, I. (2012). Expected improvement in efficient global optimization through bootstrapped Kriging. *Journal of Global Optimization*, 54(1) :59–73.

[Kleijnen et al., 2010] Kleijnen, J., van Beers, W., and van Nieuwenhuyse, I. (2010). Constrained optimization in expensive simulation : Novel approach. *European Journal of Operational Research*, 202(1) :164 – 174.

[Krige, 1951] Krige, D. G. (1951). A statistical approach to some basic mine valuation problems on the Witwatersrand. *Journal of the Chemical, Metallurgical and Mining Society*, 52 :119–139.

[Lanczos, 1950] Lanczos, C. (1950). An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45(4) :255–282.

[Laurenceau, 2008] Laurenceau, J. (2008). *Kriging based response surfaces for aerodynamic shape optimization*. Thèse, Institut National Polytechnique de Toulouse - INPT.

[Le Digabel, 2011] Le Digabel, S. (2011). Algorithm 909 : NOMAD : Nonlinear Optimization with the MADS Algorithm. *ACM Transaction Mathematical Software*, 37(4).

[Liem et al., 2015] Liem, R. P., Mader, C. A., and Martins, J. R. R. A. (2015). Surrogate models and mixtures of experts in aerodynamic performance prediction for aircraft mission analysis. *Aerospace Science and Technology*, 43 :126–151. 10.1016/j.ast.2015.02.019.

[Liuzzi et al., 2010] Liuzzi, G., Lucidi, S., and Sciandrone, M. (2010). Sequential penalty derivative-free methods for nonlinear constrained optimization. *SIAM Journal on Optimization*, 20(5) :2614–2635.

[Lophaven et al., 2002] Lophaven, S. N., Nielsen, H. B., and Søndergaard, J. (2002). Dace-a matlab Kriging toolbox, version 2.0. Technical report.

[Manne, 1987] Manne, R. (1987). Analysis of two partial-least-squares algorithms for multivariate calibration. *Chemometrics and Intelligent Laboratory Systems*, 2(1-3) :187–197.

[Martins and Lambe, 2013] Martins, J. R. R. A. and Lambe, A. B. (2013). Multidisciplinary design optimization : A survey of architectures. *AIAA journal*, 51(9) :2049–2075.

[Matheron, 1963] Matheron, G. (1963). Principles of geostatistics. *Economic Geology*, 58(8) :1246–1266.

[McCulloch and Pitts, 1988] McCulloch, W. S. and Pitts, W. (1988). Neurocomputing : Foundations of research. chapter A Logical Calculus of the Ideas Immanent in Nervous Activity, pages 15–27. MIT Press, Cambridge, MA, USA.

[Mcculloch et al., 1959] Mcculloch, W. S., Pitts, W. H., Lettvin, J. Y., and Maturana, H. R. (1959). What the frog's eye tells the frog's brain. *Proceedings of the IRE*, 47(11) :1940–1951.

[Mezura-Montes and Cetina-Domínguez, 2012] Mezura-Montes, E. and Cetina-Domínguez, O. (2012). Empirical analysis of a modified artificial bee colony for constrained numerical optimization. *Applied Mathematics and Computation*, 218(22) :10943 – 10973.

[Michalewicz and Schoenauer, 1996] Michalewicz, Z. and Schoenauer, M. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, 4 :1–32.

[Mohammadi et al., 2015] Mohammadi, H., Le Riche, R., and Touboul, E. (2015). Making EGO and CMA-ES complementary for global optimization. In *Learning and Intelligent Optimization - 9th International Conference, LION 9, Lille, France, January 12-15, 2015. Revised Selected Papers*, pages 287–292.

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

[Noesis, 2015] Noesis, S. (2015). *OPTIMUS, http ://www.noesissolutions.com/Noesis/*.

[on Simulation-Based Engineering Science, 2006] on Simulation-Based Engineering Science, N. S. F. U. B. R. P. (2006). *Revolutionizing Engineering Science Through Simulation : A Report of the National Science Foundation Blue Ribbon Panel on Simulation-Based Engineering Science*. National Science Foundation.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Rettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research*, 12 :2825–2830.

[Picheny, 2014] Picheny, V. (2014). A stepwise uncertainty reduction approach to constrained global optimization. In *AISTATS*, pages 787–795.

[Powell, 1994] Powell, M. (1994). A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in optimization and numerical analysis : Proceedings of the Sixth Workshop on Optimization and Numerical Analysis, Oaxaca, Mexico*, page 51. Kluwer Academic Pub.

[Powell, 1992] Powell, M. J. D. (1992). *The Theory of Radial Basis Function Approximation in 1990*, pages 105–210. Oxford University Press, USA.

[Rasmussen and Williams, 2006] Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA.

[Refaeilzadeh et al., 2009] Refaeilzadeh, P., Tang, L., and Liu, H. (2009). Cross-validation. In Liu, L. and Özsu, M. T., editors, *Encyclopedia of Database Systems*, pages 532–538. Springer US.

[Regis, 2014a] Regis, R. G. (2014a). Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. *Engineering Optimization*, 46(2) :218–243.

[Regis, 2014b] Regis, R. G. (2014b). Evolutionary programming for high-dimensional constrained expensive black-box optimization using radial basis functions. *IEEE Trans. Evolutionary Computation*, 18(3) :326–347.

[Regis and Shoemaker, 2007] Regis, R. G. and Shoemaker, C. A. (2007). A stochastic radial basis function method for the global optimization of expensive functions. *INFORMS Journal on Computing*, 19(4) :479–509.

[Regis and Shoemaker, 2013a] Regis, R. G. and Shoemaker, C. A. (2013a). Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization. *Engineering Optimization*, 45(5) :529–555.

[Regis and Shoemaker, 2013b] Regis, R. G. and Shoemaker, C. A. (2013b). A quasi-multistart framework for global optimization of expensive functions using response surface models. *Journal of Global Optimization*, 56(4) :1719–1753.

[Roustant et al., 2012] Roustant, O., Ginsbourger, D., and Deville, Y. (2012). Dicekriging, Diceoptim : Two R packages for the analysis of computer experiments by Kriging-based metamodeling and optimization. *Journal of Statistical Software*, 51(1) :1–55.

[Sacks et al., 1989a] Sacks, J., Schiller, S. B., and Welch, W. J. (1989a). Designs for computer experiments. *Technometrics*, 31(1) :41–47.

[Sacks et al., 1989b] Sacks, J., Welch, W. J., Mitchell, W. J., and Wynn, H. P. (1989b). Design and analysis of computer experiments. *Statistical Science*, 4(4) :409–435.

[Sakata et al., 2004] Sakata, S., Ashida, F., and Zako, M. (2004). An efficient algorithm for Kriging approximation and optimization with large-scale sampling data. *Computer methods in applied mechanics and engineering*, 193(3) :385–404.

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

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

[Schonlau, 1998] Schonlau, M. (1998). *Computer Experiments and Global Optimization*. PhD thesis, University of Waterloo, Canada.

[Sendin et al., 2009] Sendin, J. O. H., Banga, J. R., and Csendes, T. (2009). Extensions of a multistart clustering algorithm for constrained global optimization problems. *Industrial and Engineering Chemistry Research*, 48(6) :3014–3023.

[Sobieszczanski-Sobieski and Haftka, 1997] Sobieszczanski-Sobieski, J. and Haftka, R. T. (1997). Multidisciplinary aerospace design optimization : survey of recent developments. *Structural Optimization*, 14 :1–23.

[Tenenhaus, 1998] Tenenhaus, M. (1998). *La régression PLS : théorie et pratique*. Éd. Technip.

[Thevenin, 2004] Thevenin, J. C. (2004). *Moteur des avions à réaction*. Association Aéronautique et Astronautique de France.

[Toal et al., 2011] Toal, D. J. J., Bressloff, N. W., Keane, A. J., and Holden, C. M. E. (2011). The development of a hybridized particle swarm for kriging hyperparameter tuning. *Engineering optimization*, 43(6) :675–699.

[Ugray et al., 2007] Ugray, Z., Lasdon, L., Plummer, J., Glover, F., Kelly, J., and Martí, R. (2007). Scatter search and local NLP solvers : A multistart framework for global optimization. *INFORMS Journal on Computing*, 19(3) :328–340.

[Vauclin, 2014] Vauclin, R. (2014). Développement de modèles réduits multifidélité en vue de l'optimisation de structures aéronautiques. *Rapport Institut Supérieure de l'Aéronautique et de l'Espace – École Nationale Supérieure des Mines de Saint-Etienne*.

[Viana et al., 2013] Viana, F., 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.

[Welch et al., 1992] Welch, W. J., Buck, R. J., Sacks, J., Wynn, H. P., Mitchell, T. J., and Morris, M. D. (1992). Screening, predicting, and computer experiments. *Technometrics*, 34(1) :15–25.

[Wold, 1966] Wold, H. (1966). *Estimation of Principal Components and Related Models by Iterative Least squares*, pages 391–420. Academic Press, New York.

[Wold, 1975] Wold, H. (1975). Soft Modeling by Latent Variables ; the Nonlinear Iterative Partial Least Squares Approach. *Perspectives in Probability and Statistics. Papers in Honour of M. S. Bartlett*, pages 117–142.