Métamodélisation introspective pour l'analyse des phénomènes physiques simulés. Formalisation dans le cadre du krigeage et intégration algorithmique en optimisation et inversion.

Nicolas Garland

N°d'ordre NNT : 2020LYSEM017

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
l'Ecole des Mines de Saint-Etienne

Ecole Doctorale N° 488
Sciences, Ingénierie, Santé

Spécialité de doctorat : Mathématiques appliquées

Soutenue publiquement le 22/10/2020, par :
Nicolas Garland

Métamodélisation introspective pour l'analyse des phénomènes physiques simulés.

Formalisation dans le cadre du krigeage et intégration algorithmique en optimisation et inversion.

Devant le jury composé de :

Rokotomamonjy, Alain,	Professeur	Université de Rouen	Président
Bachoc, François,	Maître de conférences	Université Paul Sabatier	Rapporteur
Helbert, Céline,	Maître de conférences	Ecole Centrale de Lyon	Rapporteure
Désidéri, Jean-Antoine,	Directeur de recherche	INRIA Sophia Antipolis-Méditerranée	Examinateur
Durrande, Nicolas,	Ingénieur de recherche	PROWLER.io	Encadrant
Le Riche, Rodolphe	Directeur de recherche	CNRS LIMOS / Mines Saint-Étienne	Directeur de thèse
Richet, Yann	Ingénieur de recherche	IRSN	Invité

Spécialités doctorales	Responsables :	Spécialités doctorales	Responsables
SCIENCES ET GENIE DES MATERIAUX	K. Wolski Directeur de recherche	INFORMATIQUE	O. Boissier, Professeur
MECANIQUE ET INGENIERIE	S. Drapier, professeur	SCIENCES DES IMAGES ET DES FORMES	JC. Pinoli, Professeur
GENIE DES PROCEDES	F. Gruy, Maître de recherche	GENIE INDUSTRIEL	N. Absi, Maitre de recherche
SCIENCES DE LA TERRE	B. Guy, Directeur de recherche	MICROELECTRONIQUE	Ph. Lalevée, Professeur
SCIENCES ET GENIE DE L'ENVIRONNEMENT	D. Graillot, Directeur de recherche		

EMSE : Enseignants-chercheurs et chercheurs autorisés à diriger des thèses de doctorat (titulaires d'un doctorat d'État ou d'une HDR)

ABSI	Nabil	MR	Génie industriel	CMP
AUGUSTO	Vincent	MR	Génie industriel	CIS
AVRIL	Stéphane	PR	Mécanique et ingénierie	CIS
BADEL	Pierre	PR	Mécanique et ingénierie	CIS
BALBO	Flavien	PR	Informatique	FAYOL
BASSEREAU	Jean-François	PR	Sciences et génie des matériaux	SMS
BATTON-HUBERT	Mireille	PR	Sciences et génie de l'environnement	FAYOL
BEIGBEDER	Michel	MA	Informatique	FAYOL
BILAL	Blayac	DR	Sciences et génie de l'environnement	SPIN
BLAYAC	Sylvain	PR	Microélectronique	CMP
BOISSIER	Olivier	PR	Informatique	FAYOL
BONNEFOY	Olivier	PR	Génie des Procédés	SPIN
BORBELY	Andras	DR	Sciences et génie des matériaux	SMS
BOUCHER	Xavier	PR	Génie Industriel	FAYOL
BRUCHON	Julien	PR	Mécanique et ingénierie	SMS
CAMEIRAO	Ana	PR	Génie des Procédés	SPIN
CHRISTIEN	Frédéric	PR	Science et génie des matériaux	SMS
DAUZERE-PERES	Stéphane	PR	Génie Industriel	CMP
DEBAYLE	Johan	MR	Sciences des Images et des Formes	SPIN
DEGEORGE	Jean-Michel	MA	Génie industriel	Fayol
DELAFOSSE	David	PR	Sciences et génie des matériaux	SMS
DELORME	Xavier	PR	Génie industriel	FAYOL
DESRAYAUD	Christophe	PR	Mécanique et ingénierie	SMS
DJENIZIAN	Thierry	PR	Science et génie des matériaux	CMP
BERGER-DOUCE	Sandrine	PR	Sciences de gestion	FAYOL
DRAPIER	Sylvain	PR	Mécanique et ingénierie	SMS
DUTERTRE	Jean-Max	PR	Microélectronique	CMP
EL MRABET	Nadia	MA	Microélectronique	CMP
FAUCHEU	Jenny	MA	Sciences et génie des matériaux	SMS
FAVERGEON	Loïc	MR	Génie des Procédés	SPIN
FEILLET	Dominique	PR	Génie Industriel	CMP
FOREST	Valérie	PR	Génie des Procédés	CIS
FRACZKIEWICZ	Anna	DR	Sciences et génie des matériaux	SMS
GAVET	Yann	MA	Sciences des Images et des Formes	SPIN
GERINGER	Jean	MA	Sciences et génie des matériaux	CIS
GONDRAN	Natacha	MA	Sciences et génie de l'environnement	FAYOL
GONZALEZ FELIU	Jesus	MA	Sciences économiques	FAYOL
GRAILLOT	Didier	DR	Sciences et génie de l'environnement	SPIN
GROSSEAU	Philippe	DR	Génie des Procédés	SPIN
GRUY	Frédéric	PR	Génie des Procédés	SPIN
HAN	Woo-Suck	MR	Mécanique et ingénierie	SMS
HERRI	Jean Michel	PR	Génie des Procédés	SPIN
ISMAILOVA	Esma	MC	Microélectronique	CMP
KERMOUCHE	Guillaume	PR	Mécanique et Ingénierie	SMS
KLOCKER	Helmut	DR	Sciences et génie des matériaux	SMS
LAFOREST	Valérie	DR	Sciences et génie de l'environnement	FAYOL
LERICHE	Rodolphe	DR	Mécanique et ingénierie	FAYOL
LIOTIER	Pierre-Jacques	MA	Mécanique et ingénierie	SMS
MOLIMARD	Jérôme	PR	Mécanique et ingénierie	CIS
MOULIN	Nicolas	MA	Mécanique et ingénierie	SMS
MOUTTE	Jacques	MR	Génie des Procédés	SPIN
NAVARRO	Laurent	MR	Mécanique et ingénierie	CIS
NEUBERT	Gilles	PR	Génie industriel	FAYOL
NIKOLOVSKI	Jean-Pierre	Ingénieur de recherche	Mécanique et ingénierie	CMP
O CONNOR	Rodney Philip	PR	Microélectronique	CMP
PICARD	Gauthier	PR	Informatique	FAYOL
PINOLI	Jean Charles	PR	Sciences des Images et des Formes	SPIN
POURCHEZ	Jérémy	DR	Génie des Procédés	CIS
ROUSSY	Agnès	MA	Microélectronique	CMP
SANAUR	Sébastien	MA	Microélectronique	CMP
SERRIS	Eric	IRD	Génie des Procédés	FAYOL
STOLARZ	Jacques	CR	Sciences et génie des matériaux	SMS
VALDIVIESO	François	PR	Sciences et génie des matériaux	SMS
VIRICELLE	Jean Paul	DR	Génie des Procédés	SPIN
WOLSKI	Krzystof	DR	Sciences et génie des matériaux	SMS
XIE	Xiaolan	PR	Génie industriel	CIS
YUGMA	Gallian	MR	Génie industriel	CMP

Mise à jour : 24/11/2020

Remerciements

Je n'étais pas seul.

Pour réaliser ce travail de thèse, j'étais bien entouré. Tout d'abord à l'IRSN. Je remercie chaleureusement Yann Richet, pour avoir proposé cette thèse et pour m'avoir encadré quotidiennement jusqu'au bout. Il a enseigné la presquitude des choses à mon esprit rigoriste. Je me souviens de son affichette lui rappelant de ne pas pousser la presquitude trop loin. Je remercie également Jean Baccou pour m'avoir aiguillé vers cette thèse. Sans lui je serai passé à côté de cette belle aventure. Mon travail à l'IRSN n'aurait pas eu le même goût sans tous mes joyeux collègues du SNC, les permanents autant que les stagiaires, trop nombreux pour tous les citer. L'ambiance était toujours au beau fixe, chaleureuse et solidaire. Je remercie en particulier les membres des MidiJeux, qui me donnaient une bonne raison d'aller au boulot. Le cadre administratif était en outre bien organisé et rassurant, et je remercie tous ceux et celles qui y ont participé. Je n'oublie évidemment pas mes co-bureau : Antoine et Geoffrey, avec qui j'ai rapidement et durablement sympathisé, puis Guénaël, même si ce fut pour une courte durée. Travailler à l'IRSN c'était aussi les Journées des Thèses et l'occasion de rencontrer d'autres doctorants, travaillant sur des problématiques différentes, et ce fut très enrichissant et très agréable. Je remercie vivement les membres qui faisaient vivre l'association des doctorants de l'IRSN.

Et puis il y avait l'école des Mines de Saint-Étienne. Je remercie de tout cœur mon directeur de thèse, Rodolphe Le Riche, pour avoir accepté de diriger cette thèse et pour le soutien indéfectible dont il a fait preuve pendant de la phase de rédaction, lors des moments difficiles que j'ai traversés. Je remercie également Nicolas Durrande pour son encadrement bienveillant et sa présence constante. Je souhaite remercier aussi le tableau noir de la pause-café et ses énigmes mathématiques, ainsi que tous mes collègues de l'espace Fauriel qui ont gravité autour. J'ai une pensée particulière pour mes co-bureau : Audrey et Andrès, qui m'ont fait me sentir comme chez moi grâce à leur complicité.

Et bien sûr, il y avait ma famille et mes amis. Tout particulièrement, il y avait mes parents qui m'ont soutenu et supporté lorsque j'étais au plus bas. Et puis il y avait mes frères et sœur, toujours présents. Je les en remercie chaleureusement. J'ai pu compter également sur mes amis pour me distraire durant cette période, mes compagnons d'aventure lors de nos séances de jeu de rôle mais également tous les autres. Je remercie Farida pour m'avoir supporté. Et je remercie surtout Maiana sans qui je ne serai jamais allé jusqu'au bout.

Pour finir, je souhaite remercier les rapporteurs pour avoir accepté de relire mon manuscrit, ainsi que tous les membres du jury pour avoir accepté de juger mon travail.

À vous tous qui m'avez accompagné pendant toutes ces années : Merci ! Ce travail est aussi un peu le vôtre. Je n'étais pas seul.

Préambule

À vous qui m'écoutez : mettez-vous à votre aise,
Je m'en vais vous conter le sujet de ma thèse.

Il était un expert en sûreté nucléaire.
On vint lui demander d'étudier un dossier
Pour une installation placée en bord de mer.
Face aux vents et marées, risquait-elle d'exploser ?

Il lui fallait savoir si elle serait bien sûre,
En toute circonstance, en tout cas de figure.
Quel peut être le pire qu'il pourrait arriver ?
Pouvait-on sans danger valider le dossier ?

L'expert pouvait lancer quelques simulations.
Mais pour quels paramètres ? Là était la question.
Les choix sont infinis et les calculs sont longs.
Il ne pouvait tester toutes les combinaisons.

Il fit alors appel au mathématicien
Pour construire un modèle qui relierait les points
Des calculs déjà faits, afin d'évaluer,
Pour chaque simulation, si on peut s'en passer.

L'idée fort ingénieuse du mathématicien
Fut de paramétrer un processus gaussien.
En le conditionnant par toutes les données
Il obtint un modèle faisant ce qu'on voulait.

Le modèle prédisait sa propre imprécision
Du résultat prédit d'une simulation.
Exploitant ses infos, un algo bien pensé
Suggérait à l'expert le calcul à lancer.

L'algorithme était fiable, et surtout efficient,
Mais l'expert insatiable fut encore mécontent :
« Mais moi je sais des choses sur mon code de calcul
Que ton algo néglige, ton modèle est trop nul ! »

« Je sais qu'il est construit de codes enchaînés,
Séparant les physiques. Chaque code est concret.
Une grandeur calculée par le code initial
Me donne une bonne idée du résultat final. »

Ils m'engagèrent donc en tant que doctorant
Pour penser un modèle utile et innovant.
Compulsant à ma guise des travaux publiés,
Je fis une analyse des possibilités.

On pourrait sans soucis, et bien naïvement,
Modéliser chacun des blocs séparément.
Mais comment faire un lien entre plusieurs modèles
Alors que les grandeurs n'ont pas la même échelle ?

Les modèles distincts seraient des blocs latents,
Des processus cachés et tous indépendants.
Les diverses grandeurs alors correspondront
À diverses valeurs de leurs combinaisons.

Et ces combinaisons je peux les structurer
Pour correspondre à un enchaînement donné.
J'ai pu trouver de plus les valeurs optimales
Pour certains paramètres du modèle global.

Je mis ensuite au point une approche nouvelle
Où les sorties données par le code premier
Sont des données d'entrée pour le second modèle.
Les processus gaussiens sont ainsi emboîtés.

Le résultat global de la composition
Nous permet d'obtenir de belles prédictions,
Acquises au prix d'une grande complexité
Qui, grâce à mes calculs, peut être simplifiée.

Ainsi notre modèle s'est vu amélioré.
Mais l'expert ne fut pas suffisamment content
Car les simulations peuvent être arrêtées :
On peut alors chercher à gagner plus de temps.

Mon nouvel algorithme se pose la question,
Pour l'étape d'après d'une simulation,
Suivant le temps qu'il faut pour la voir complétée,
S'il ne vaudrait pas mieux un peu la différer.

Ainsi finit ma thèse ; ces techniques nouvelles
S'ajoutent aux outils pour les métamodèles.

Introduction : Contexte et besoin opérationnel

Cette thèse a été effectuée au sein de l'Institut de Radioprotection et de Sûreté Nucléaire (IRSN), dans le Service de Neutronique et des risques de Criticité (SNC, sous la direction de l'expertise et de sûreté du pôle Sûreté Nucléaire, PSN). Elle a en effet pour but de répondre à un besoin opérationnel des experts en criticité. Ceux-ci étudient le risque d'emballement d'une réaction en chaîne neutronique dans des installations nucléaires (entrepôts, réacteurs, transports, etc.) contenant des matières fissiles. Pour analyser ce risque, ils utilisent des simulations numériques des phénomènes neutroniques induits par la présence des isotopes fissiles (généralement l'uranium et le plutonium).

Ces simulations utilisent des codes de calcul qui restent coûteux en temps de calcul malgré la puissance des ordinateurs actuellement utilisés. Or les experts en sûreté ont souvent des problématiques qui nécessitent d'explorer le domaine des paramètres, par exemple pour trouver la pire configuration neutronique possible (problème d'optimisation), ou pour trouver la limite entre les configurations sous-critiques et surcritique (problème d'inversion). Des problèmes combinant à la fois une nécessité d'inversion et d'optimisation existent également. Afin de limiter le nombre d'appels au code, ils peuvent compter sur des algorithmes qui fonctionnent par modélisation du code de calcul (« métamodèle »). Cette thèse a ainsi également été réalisée avec le département de mathématiques des Mines de Saint-Étienne qui a comme axe de recherche la modélisation statistique de modèles physiques. Les métamodèles les plus efficaces se basent sur des représentations par krigeage, présentées en première partie de ce manuscrit. Les algorithmes à base de krigeage utilisés pour l'optimisation et l'inversion sont présentés dans la suite du chapitre premier.

Le but de cette thèse est d'améliorer ces algorithmes dans le cas où le code de calcul est en réalité une succession de codes. Une telle structure de la simulation est commune à la plupart des chaînes de calcul, notamment celles impliquant plusieurs physiques couplées séquentiellement. À l'IRSN, pour les calculs de criticité, le code de calcul majoritairement utilisé en France, APOLLO-MORET, est constitué de deux codes : le premier fait un prétravail nécessaire avant la transmission de ses données au second. Dans ce cas, la grandeur d'intérêt finalement renvoyée par MORET est le $k_{effectif}$ (un coefficient de multiplication des neutrons), et il se trouve qu'APOLLO peut également générer une grandeur, le k_{infini}, qui est vraisemblablement fortement corrélée avec le $k_{effectif}$ car il s'agit de la même grandeur dans le cas extrême d'un milieu fissile infini. Par ailleurs, il se trouve que le calcul APOLLO seul est beaucoup plus rapide que le calcul APOLLO-MORET.

Les algorithmes actuels de métamodélisation traitent l'ensemble de la chaîne de calcul comme un seul code, une « boîte noire », et oublient de ce fait la partie intermédiaire de l'information qui est rapidement disponible. Ce que l'on va vouloir accomplir dans cette thèse, c'est d'ouvrir la boîte et de profiter des informations supplémentaires. On parlera alors « d'introspection ». Un algorithme adapté, qui exploiterait cette information intermédiaire et qui prendrait en compte la vitesse d'exécution de chaque code, permettrait de faire de substantielles économies de calcul, d'une part en bénéficiant d'une modélisation plus précise (« modélisation introspective »), et d'autre part en ayant la possibilité d'interrompre la chaîne de calcul lorsque le calcul du premier code nous indique que le suivant n'est pas pertinent. Nous appellerons ces algorithmes itératifs adaptés « algorithmes introspectifs ».

Pour aboutir à ces algorithmes adaptés, il a fallu s'intéresser aux différentes modélisations introspectives possibles. Elles constituent le second chapitre de la thèse. En particulier, nous nous sommes intéressés aux variantes du krigeage décrivant plusieurs sorties linéairement corrélées, dont les plus utilisés sont les modèles de cokrigeage. Des travaux ont été effectués pour améliorer l'estimation des paramètres des modèles de cokrigeage. Une alternative non linéaire au cokrigeage, nommée hyperkrigeage, a également

été étudiée. Enfin une comparaison des différents modèles de cokrigeage et d'hyperkrigeage termine le second chapitre.

Le choix d'un métamodèle introspectif permet, dans le troisième chapitre, la conception d'un algorithme d'optimisation adapté, exploitant cette modélisation introspective et la spécificité de la chaîne de calcul (la possibilité de fractionner de calcul). L'algorithme introspectif est d'abord détaillé pour une problématique d'optimisation. Puis on généralise la stratégie de l'algorithme à d'autres problématiques, notamment d'inversion.

Notations

On n'emploiera pas de notation particulière pour discriminer les vecteurs des scalaires. Beaucoup de grandeurs seront vectorielles, ou potentiellement vectorielles en particulier les points d'observation. Le lecteur est invité à faire la distinction en fonction du contexte. Les matrices seront systématiquement en majuscule. Notez que nous ferons un usage abusif de la notation vectoriel : si f est une fonction $\mathbb{R}^d \to \mathbb{R}$ et X est une matrice $N \times d$, alors $f(X)$ est le vecteur de terme général $f(X_i)$. De même, $k(X, X')$ est la matrice $N \times N'$ ayant pour éléments les $k(X_i, X'_j)$.

Tous les objets représentés en gras sont des objets qui peuvent être décomposés suivant les groupes/niveaux.

On utilise un exposant $\cdot^{(a)}$ pour signifier le niveau correspondant à l'objet.

Les exposants $\cdot^{(a)}$ et $\cdot^{(b)}$ sont utilisés pour parcourir les niveaux. Le niveau (1) correspond à la grandeur d'intérêt (haute fidélité). Les niveaux supérieurs correspondent aux grandeurs intermédiaires, par ordre décroissant de fidélité.

$C(x, x')$: covariance après conditionnement.

d : dimension d'entrée du problème.

$\mathbb{D}$: espace des entrées dans lequel évoluent des points d'observations, un hypercube fermé et borné de $\mathbb{R}^d$.

$\hat{\cdot}$: estimateur de l'objet $\cdot$

Les indices i et j seront exclusivement réservés aux parcours des observations ou des objets de même taille (N) s'y référant. Dans les autres cas (parcours des dimensions ou des paramètres par exemple) on utilisera les indices l.

I : matrice identité (taille implicite donnée par le contexte, ou précisée en indice : I_n)

$I_{\{a\}}$: indicatrice du niveau a. Matrice diagonale de taille $N \times N$ dont le i-ème élément diagonal vaut 1 si et seulement si la i-ème observation a été faite au niveau a (0 sinon).

ICJ_M : *Integrated Conditional* J_M, critère issue de la transformation de J_M pour un faire un critère presbyte.

J_M : critère permettant à l'algorithme itératif de trouver un nouveau point, grâce au métamodèle M.

$k(x, x')$: fonction de covariance avant conditionnement.

$k(h) = k(|x - x'|)$: noyau (unidimensionnel), pour processus stationnaire.

$k(x, x') = \sigma_0^2 r(x, x')$: décomposition du noyau avec facteur d'échelle.

$K = k(X, X)$: matrice des $k(X_i, X_j)$.

L,$\mathcal{L}$, $\mathcal{L}_c$: vraisemblance, log-vraisemblance et log-vraisemblance concentrée, respectivement.

$m(\cdot)$: moyenne *a priori* du processus. $m(\cdot) = \sum_{l=1}^{p} \beta_l f_l(\cdot)$: paramétrisation de $m()$. β est un vecteur de taille p et $\mathrm{f}_*()$ est un vecteur de p fonctions $f_l()$ ($l \in [1 \dots p]$).

$\mu(\cdot)$: moyenne *a posteriori* du processus (après conditionnement).

n_g : nombre de groupes/grandeurs. Le niveau (1) correspond à la grandeur d'intérêt (haute fidélité). Le niveau (n_g) correspond au niveau de plus basse fidélité.

n_l : nombre de variables latentes (égal à n_g dans les cas considérés).

$\phi(\cdot)$: densité de distribution de la loi normale centrée réduite.

$\Phi(\cdot)$: fonction de répartition de la loi normale centrée réduite.

$\sigma^2(\cdot) = C(\cdot, \cdot)$: variance de krigeage.

$t^{(a)}$: temps/coût d'exécution du niveau a.

$T_a = \sum_{b=a}^{1} t^{(b)}$: temps d'exécution cumulé du niveau a au niveau 1.

$\boldsymbol{U}(\cdot)$: processus gaussiens multivarié des variables latentes, constitué de n_l processus gaussiens indépendants $U_l()$

x en minuscule : une coordonnée/un point (quelconque, pas forcément associé à une mesure déjà faite), vecteur de taille d ou d-uplet.

x, x' : deux points quelconques.

X : matrice des points d'observations ($N \times d$). Peut être vu également comme une collection de N d-uplets.

X_i : vecteur d'entrée/point d'observation de la i-ème mesure (vecteur de taille d ou d-uplet). De façon générale, l'indice $_i$ désigne le i-ème élément d'un vecteur et la paire d'indices $_{i,j}$ désigne l'élément d'une matrice situé à la ligne i et colonne j.

$\boldsymbol{X} = (X^{(1)}, X^{(2)}, \ldots)$: matrices des observations.

$Y = {}^t(y_1, y_2, \ldots)$: observations/mesures aux points d'observation $X = (X_1, X_2, \ldots)$.

$y(\cdot)$: fonction qui à un point x associe la mesure $y(x)$.

z : valeur quelconque de l'espace des mesures.

$Z(\cdot)$: processus avant conditionnement.

$\tilde{Z}(\cdot)$: processus après conditionnement.

$\boldsymbol{Z}(\cdot)$: processus gaussien multivarié avant conditionnement.

$Z^{(a)}(x) = Z(x, a)$: processus gaussien marginal avant conditionnement.

$\tilde{Z}^{(a)}(x) = \tilde{Z}(x, a)$: processus gaussien marginal après conditionnement.

Sigles et abréviations

AMS : *Adaptative Model Switching*, choix adaptatif de modèle (employé en MDO)

ANOVA, composition : opérateur sur des variables qui les multiplie après les avoir augmentées d'une constante.

APOLLO : code de calcul pour la neutronique, utilisé pour calculer les sections efficaces macroscopiques des matériaux utilisés dans le calcul de criticité. Ce code fait parti de la chaîne CRISTAL.

BFGS : Broyden-Fletcher-Goldfard-Shanno, méthode d'optimisation quasi-newtonienne impliquant des calculs de gradient.

CJ : *Conditional J*, critère J (quelconque) conditionné.

CSSO : *Concurent SubSpace Optimization* (employé en MDO).

CRISTAL : chaîne de calcul pour la neutronique, proposant plusieurs "schémas" de calculs, dont la voie de production industrielle APOLLO-MORET.

DICE : *Deep Inside Computer Experiments*, consortium.

EGO : *Efficient Global Optimization*, algorithme d'optimisation globale impliquant un GP.

EI : *Expected Improvement*, amélioration espérée, critère pour l'optimisation.

EIR : *Expected Improvement Rate*, taux d'amélioration espéré, critère pour l'algorithme itératif SoSo.

GP : *Gaussian Process*, processus gaussien.

IAGO : *Informational Approach to Global Optimization* , approche informationnelle pour optimisation globale, algorithme d'optimisation impliquant un GP.

ICJ : *Integrated Conditional J*, intégrale du critère J (quelconque) conditionné, critère pour algorithme itératif.

IECI :*Integrated Expected Conditional Improvement*, intégrale de l'amélioration espérée conditionnée, critère pour l'optimisation.

IRSN : Institut de Radioprotection et de Sûreté Nucléaire.

LHS : *Latin Hypercube Sample*, échantillonnage par hypercube latin, type de plan d'expériences.

LMC : *Linear Model of Coregionalization*, modèle linéaire de corégionalisation, modèle de cokrigeage.

MDF : *MultiDisciplinary Feasible* (employé en MDO)

MDO : *Multidisciplinary Design Optimisation*, optimisation de conception multidisciplinaire.

MLE : *Maximum Likelihood Estimation*, estimation par maximum de vraisemblance, méthode d'estimation des paramètres du modèle.

MSE : *Mean Square Error*, erreur quadratique moyenne, statistique pour mesurer l'efficacité d'un prédicteur.

MORET : code de calcul pour la neutronique, utilisé pour calculer le k-effectif à partir des sections efficaces macroscopiques des matériaux fournies par APOLLO. Ce code fait parti de la chaîne CRISTAL.

OQUAIDO : Optimisation et QUAntification d'Incertitude pour les Données Onéreuses, chaire en mathématiques appliquées.

PEGO : *Profile Efficient Global Optimization*, optimisation globale efficiente sur profils, algorithme de reconstruction de ligne de crête.

PEI : *Profile Expected Improvement*, amélioration espérée sur profils, critère utilisé par PEGO.

PSN : Pôle Sûreté Nucléaire, partie de l'IRSN.

ReDICE : suite du consortium DICE.

RSM : *Response Surface Method*, méthode de la surface de réponse (employé en MDO).

SNC : Service de Neutronique et des risques de Criticité, appartenant au Pôle Sureté Nucléaire de l'IRSN.

SoS : *Step or Stop*, continuation ou arrêt, famille d'algorithmes itératifs.

SoSi : *Step or Step inversion*, SoS pour inversion.

SoSo : *Step or Stop optimization*, SoS pour optimisation.

SCURR : *Stepwise Complete Uncertainty Reduction Rate*, taux complet de réduction d'incertitude pas-à-pas, variante de SUR en présence de données d'introspection.

SUR : *Stepwise Uncertainty Reduction*, réduction d'incertitude pas-à-pas, famille de critères pour algorithmes itératifs.

SURR : *Stepwise Uncertainty Reduction Rate*, taux de réduction d'incertitude pas-à-pas, variante de SUR en présence de données d'introspection.

TMSE : *Targeted Mean Square Error*, erreur quadratique moyenne ciblée, critère pour l'inversion.

TSEE : *Two Sided Expected Exceedance*, excès espéré bilatéral, critère pour l'inversion.

Table des matières

Remerciements **3**

Préambule **5**

Introduction : Contexte et besoin opérationnel **7**

Notations **9**

Sigles et abréviations **11**

1 Modèles de krigeage et algorithmes associés **17**
1.1 Introduction aux processus gaussiens conditionnels . . . 17
1.2 Choix du modèle de covariance . . . 19
1.2.1 Fonctions de covariance . . . 19
Exemple . . . 20
1.2.2 Estimation des paramètres . . . 20
Exemple . . . 22
Améliorations du krigeage . . . 22
1.3 Algorithmes itératifs . . . 25
1.3.1 Critères pour l'optimisation . . . 27
Expected Improvement (EI) . . . 27
Integrated Expected Conditional Improvement ($IECI$) . . . 28
Informational Approach to Global Optimization (IAGO) . . . 30
1.3.2 Critères pour l'inversion . . . 30
1.3.3 Problème mixte inversion-optimisation . . . 31
Stratégie de parallélisation . . . 31

2 Modèles introspectifs **33**
2.1 Problématique . . . 34
2.1.1 Contexte opérationnel . . . 34
2.1.2 Modèles multifidélités . . . 35
2.1.3 Optimisations de conception multidisciplinaire . . . 36
2.1.4 Variables qualitatives . . . 37
2.1.5 Formalisation . . . 38
2.2 Processus gaussien multivarié . . . 38
2.2.1 Exploitation des valeurs de la dérivée . . . 39
2.2.2 Modèles linéaires de corégionalisation (LMC) : généralités . . . 40
Observation des variables latentes . . . 43
2.2.3 LMC symétrique pour des codes interdépendants . . . 44
2.2.4 LMC markovien pour des codes séquentiels . . . 46
2.2.5 LMC *ad hoc* pour une structure de codes quelconque . . . 48
2.2.6 Autres modèles de cokrigeage . . . 48
2.2.7 Optimisation des paramètres et vraisemblance concentrée . . . 49
Paramètres de tendance $\boldsymbol{\beta}$ . . . 51
Paramètres d'échelle $\boldsymbol{\sigma}$ . . . 52
Application de la vraisemblance concentrée . . . 54

2.3 Hyperkrigeage . . . 59
2.3.1 Approximations du modèle d'hyperkrigeage . . . 60
2.3.2 Hyperkrigeage croisé ou krigeage intriqué . . . 62
2.4 Comparaison des modèles . . . 63
2.4.1 Présentation des cas-tests . . . 63
2.4.2 Résultats sur le cas-test *mesh based* . . . 68
2.4.3 Résultats sur le cas-test Monte-Carlo . . . 71
2.4.4 Résultats sur le cas-test *time step* . . . 73

3 Algorithmes itératifs introspectifs **77**
3.1 Cadre d'étude . . . 77
3.2 Algorithme *Step or Stop* (SoS) . . . 78
3.3 Critère *Expected Improvement Rate* (EIR) pour l'optimisation . . . 80
Critères EI et EIR pour l'hyperkrigeage . . . 82
3.3.1 Illustration de SoSo . . . 83
3.3.2 Efficacité de SoSo . . . 83
3.3.3 Application pratique en sûreté nucléaire . . . 88
3.4 Stratégie *Step or Stop* avec d'autres critères . . . 98

Conclusions et perspectives **99**

A Vraisemblance concentrée (cas classique) **101**
Paramètres de tendance maximisants la vraisemblance, $\hat{\beta}$ . . . 101
Paramètre d'échelle maximisant la vraisemblance, $\hat{\sigma}_0^2$ . . . 102
Log-vraisemblance concentrée . . . 103

B Vraisemblance concentrée (cas multisortie) **105**
B.1 Solution explicite pour 2 groupes . . . 106
B.2 Solution numérique pour un nombre quelconque de groupes . . . 107

Bibliographie **111**

Table des figures

1.1 Processus gaussien, exemple de réalisations. 18
1.2 Interpolation par krigeage, effet des paramètres. 21
1.3 Optimisation de la log-vraisemblance, lignes de niveaux. 23
1.4 Optimisation de la log-vraisemblance, valeurs finales. 23
1.5 Optimisation de la log-vraisemblance, nombre d'appels. 24
1.6 Optimisation de la log-vraisemblance, convergence. 24
1.7 Illustrationde EGO, lignes de niveaux et points ajoutés. 29
1.8 Illustrationde EGO, lignes de niveaux de l'*Expected Improvement*. 29

2.1 Schéma Apollo-Moret. 34
2.2 Schéma de codes de simulation en multifidélité. 35
2.3 Schéma de codes de simulation en couplage total. 37
2.4 Schéma de codes de simulation avec variable qualitative. 37
2.5 Schémas de différentes possibilités de couplage de codes de simulation. 38
2.6 Cokrigeage avec la dérivée, exemple 1D. 40
2.7 Schéma d'un modèle LMC symétrique pour deux sorties. 44
2.8 Modèle LMC symétrique à deux niveaux, exemple 1D. 45
2.9 Schéma d'un modèle LMC markovien pour trois sorties. 45
2.10 Modèle LMC markovien à trois niveaux, exemple 1D. 47
2.11 Schéma de structure de codes complexe, exemple à quatre codes. 48
2.12 Efficacité de la vraisemblance concentrée, données utilisées dans l'exemple. 54
2.13 Efficacité de la vraisemblance concentrée, durées d'optimisation, en secondes. 57
2.14 Efficacité de la vraisemblance concentrée, durées d'optimisation, en nombre d'appels . . . 57
2.15 Efficacité de la vraisemblance concentrée, performance des optimisations, en valeur. 58
2.16 Efficacité de la vraisemblance concentrée, convergence des optimisations. 58
2.17 Hyperkrigeage, exemple 1D à deux niveaux. 61
2.18 Fonction-objectif, fonction de Branin modifiée. 64
2.19 Modification de la fonction-objectif pour une structure *mesh based*. 65
2.20 Modification de la fonction-objectif pour une structure Monte-Carlo. 66
2.21 Modification de la fonction-objectif pour une structure *time step*. 67
2.22 Comparaison de modèles, cas *mesh based*, plan d'expériences. 68
2.23 Comparaison de modèles, cas *mesh based*, erreur quadratique moyenne. 69
2.24 Comparaison de modèles, cas *mesh based*, variance des résidus standardisés. 70
2.25 Comparaison de modèles, cas Monte-Carlo, plan d'expériences. 71
2.26 Comparaison de modèles, cas Monte-Carlo, erreur quadratique moyenne. 72
2.27 Comparaison de modèles, cas Monte-Carlo, variance des résidus standardisés. 73
2.28 Comparaison de modèles, cas *time step*, plan d'expériences. 74
2.29 Comparaison de modèles, cas *time step*, erreur quadratique moyenne. 74
2.30 Comparaison de modèles, cas *timse step*, variance des résidus standardisés. 75

3.1 Schéma Apollo-Moret. 78
3.2 Algorithme SoSo, exemple à trois niveaux, lieux des itérations. 84
3.3 Algorithme SoSo, exemple à trois niveaux, ordre de visite des niveaux. 85
3.4 Algorithme SoSo, comparaison avec EGO, convergence en valeur. 86
3.5 Algorithme SoSo, comparaison avec EGO, convergence en position. 87
3.6 Algorithme SoSo, comparaison avec EGO, écarts à plan d'expériences fixé. 89

3.7 Application en sûreté nucléaire, schéma de la chaîne de calcul. 90
3.8 Application en sûreté nucléaire, stockage de poudre de plutonium. 91
3.9 Application en sûreté nucléaire, configuration des simulations physiques, modèles chimiques. 92
3.10 Application en sûreté nucléaire, configuration des simulations physiques, modèles 1D. . . . 93
3.11 Application en sûreté nucléaire, configuration des simulations physiques, modèles 3D. . . . 94
3.12 Application en sûreté nucléaire, état initial. 95
3.13 Application en sûreté nucléaire, état final. 96
3.14 Application en sûreté nucléaire, itérations de SoSo. 97

Chapitre 1

Modèles de krigeage et algorithmes associés

Les modèles de krigeage ont été développés dans les années 50 (Krige 1951), originellement pour la prédiction de grandeurs géologiques (Matheron 1963) et géostatistiques (Cressie 1992 ; Wackernagel 1998). Dans ces disciplines, les mesures sont sporadiques, mais on voudrait tout de même avoir une idée des valeurs prises par la grandeur d'intérêt quel que soit l'endroit. Idéalement la fiabilité de l'estimation devrait être connue. À cette fin, le krigeage propose de relier les mesures suivant une méthode mathématique particulière. Le modèle ainsi conçu peut être interpolant et prédire exactement les valeurs mesurées aux endroits mesurés ; mais il peut également prendre en considération un bruit temporel (ou « *noise* », variation aléatoire de la réponse à chaque mesure) ou un bruit spatial (ou « *nugget* », variation aléatoire de la réponse entre deux mesures extrêmement proches). La force des modèles de krigeage est de donner une information sur leur propre incertitude : ils ne prédisent pas une simple valeur, mais une loi de probabilité pour la valeur (en l'occurrence une loi normale, caractérisée par sa moyenne et sa variance).

Ces caractéristiques permettent d'employer le krigeage dans des domaines autres que les géostatistiques, par exemple quand les mesures sont générées par des simulations physiques un peu longues. Lorsque le modèle de krigeage sert à modéliser les résultats de simulations physiques (elles-mêmes considérées comme des modélisations de la réalité), on parle de « métamodèle ». Nous nous plaçons dans ce cadre-là pour cette thèse. Le krigeage sert alors à construire des algorithmes, par exemple d'optimisation globale tel que l'EGO (Jones, Schonlau, and Welch 1998 ; Villemonteix, Vazquez, and Walter 2009 ; Richet et al. 2013) (section 1.3).

1.1 Introduction aux processus gaussiens conditionnels

Un processus gaussien est un processus stochastique particulier. Un processus stochastique peut être vu comme un générateur aléatoire de fonctions. Une réalisation du processus correspond à une trajectoire, un ensemble de valeurs prises par la réalisation sur un certain ensemble du domaine. Concrètement les informations sur le processus permettent de générer les valeurs prises par une réalisation pour un nombre arbitrairement grand, mais fini, de points de l'espace d'entrée.

Dans le cas d'un processus gaussien, l'ensemble de ces valeurs suit une loi normale multivariée. Un processus gaussien $Z(\cdot)$ se caractérise alors par une moyenne pour tout point de l'espace d'entrée (qu'on peut représenter comme une fonction $m(\cdot)$), et par une covariance pour tout couple de points de l'espace d'entrée (qu'on peut représenter par une fonction à deux variables $k(\cdot,\cdot)$) :

$$Z(\cdot) \sim GP(m(\cdot), k(\cdot,\cdot)) \ . \tag{1.1}$$

Soit N points de l'espace d'entrée $\mathbb{D}$ (de dimension d, $\mathbb{D} \subset \mathbb{R}^d$), formant l'objet X (collection de N d-uplets, ou matrice de taille $N \times d$). Un tirage sur ces points du processus $Z(X)$ correspond alors à un tirage d'un vecteur sous une loi normale multivariée, ayant pour moyenne le vecteur $m(X)$ et pour

variance-covariance la matrice $K = k(X, X)$. On emploie ici le formalisme usuel : $[m(X)]_i = m(X_i)$ et $[k(X, X)]_{i,j} = k(X_i, X_j)$ pour tous i et j entre 1 et N. La figure 1.1a présente quelques réalisations d'un processus gaussien.

De la même manière qu'une variable aléatoire, un processus peut être conditionné par des évènements. En l'occurrence, on peut conditionner un processus par des points (X) où les valeurs seront fixées (Y). Toutes les trajectoires générées par le processus conditionné passeront alors par ces coordonnées. La figure 1.1b présente quelques réalisations du processus gaussien obtenu en conditionnant le processus gaussien précédent en un point.

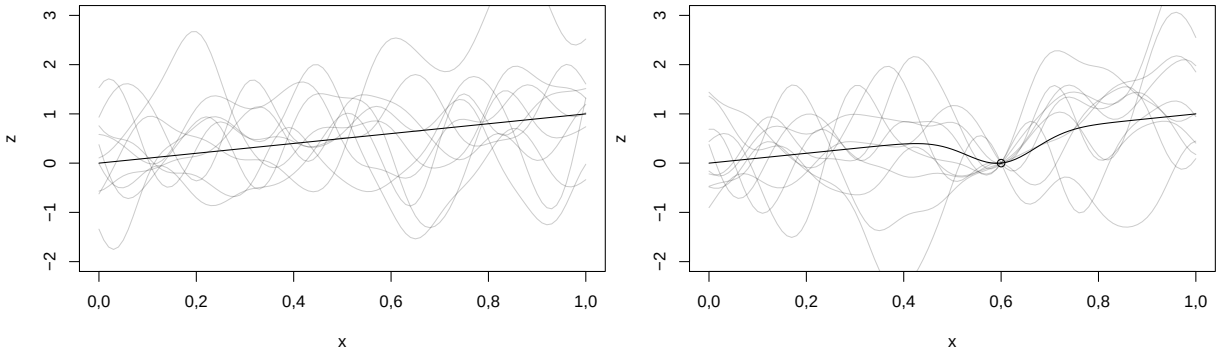
Les calculs [1] montrent alors que le processus conditionné est encore un processus gaussien dont la moyenne et la fonction de covariance sont modifiées comme suit :

$$\mu(x) = m(x) + k(x, X)K^{-1}(Y - m(X)) \ , \tag{1.2}$$

$$C(x, x') = k(x, x') - k(x, X)K^{-1}k(X, x') \ , \tag{1.3}$$

$$\sigma^2(x) = C(x, x) = k(x, x) - k(x, X)K^{-1}k(X, x) \ . \tag{1.4}$$

Pour ces formules, ainsi que pour toutes les autres, on note x ou x' un point quelconque de l'espace d'entrée $\mathbb{D}$, X l'ensemble des N points de l'espace d'entrée pour lesquels on dispose d'une observation et Y l'ensemble des observations correspondantes (vecteur de taille N).



(a) Processus gaussien

(b) Processus gaussien conditionné par un point

FIGURE 1.1 – Exemple de plusieurs réalisations d'un processus gaussien, avec et sans conditionnement, en traits gris. Le trait plein noir correspond à la moyenne du processus.

Dans le cas où le processus gaussien initial est centré autour d'une tendance à paramètres estimés, le modèle est appelé krigeage universel. La tendance $m(\cdot)$ est alors une combinaison linéaire de p fonctions fixées $f_l(\cdot)$, dont les coefficients sont notés β_l et sont les paramètres estimés de la tendance :

$$m(x) = \sum_{l=1}^{p} \beta_l f_l(x) \ . \tag{1.5}$$

On peut alors faire la distinction entre la variance de krigeage $\sigma(x)$ donné par l'équation (1.4) et la variance de prédiction qui comporte un terme supplémentaire pour prendre en compte l'incertitude sur les paramètres de tendance (Santner, Williams, and Notz 2003). Néanmoins cette variance de prédiction ajustée sous-estime encore la variabilité des résultats, du fait de la non-prise en compte de l'incertitude dans l'estimation des autres paramètres. Des approches bayésiennes tentent de combler ces lacunes en injectant de l'information *a priori* dans les paramètres (Helbert, Dupuy, and Carraro 2009). Nous ne nous sommes pas intéressés à ces problématiques au cours de cette thèse.

1. Les calculs sont un peu lourds mais sans difficultés. Ils nécessitent d'utiliser les formules de la loi normale multivariée et des probabilités conditionnelles, ainsi que les formules de déterminant et d'inversion matricielle par blocs.

1.2 Choix du modèle de covariance

Le choix du processus gaussien de base nous laisse beaucoup de latitude, qui se traduit par un large choix dans la fonction de covariance dont la seule contrainte est d'être définie-positive. Dans la pratique, on choisit généralement une fonction paramétrée dont on va ajuster les paramètres. On parle également de « noyau » ou « *kernel* » de covariance.

1.2.1 Fonctions de covariance

Les fonctions usuellement choisies sont les fonctions de type Matérn ou puissance-exponentiel (Santner, Williams, and Notz 2003 ; Rasmussen and Williams 2006), ou composées à partir de celles-ci. Ils comportent un ou plusieurs paramètres de portée (Θ) qui relativisent la distance entre les points, et un paramètre d'échelle (σ_0) qui vient en facteur. Un paramètre de régularité (ν) est généralement fixé par l'utilisateur, il n'est alors pas considéré comme un paramètre, mais il peut être aussi estimé. Les figures 1.1 utilisaient un noyau gaussien, équivalent à un Matérn infini (ou à une puissance-exponentiel d'ordre 2). Le noyau exponentiel, dont les trajectoires sont très irrégulières, correspond à un Matérn 1/2 (ou à une puissance-exponentiel d'ordre 1).

En dimension 1, les différents noyaux classiques s'écrivent ainsi :

- Noyau gaussien : $k(x,x') = \sigma_0^2 \exp\left(-\frac{1}{2}\frac{(x-x')^2}{\theta^2}\right)$;
- Noyau Matérn 5/2 : $k(x,x') = \sigma_0^2 \left(1 + \frac{\sqrt{5}|x-x'|}{\theta} + \frac{5(x-x')^2}{3\theta^2}\right) \exp\left(-\frac{\sqrt{5}|x-x'|}{\theta}\right)$;
- Noyau Matérn 3/2 : $k(x,x') = \sigma_0^2 \left(1 + \frac{\sqrt{3}|x-x'|}{\theta}\right) \exp\left(-\frac{\sqrt{3}|x-x'|}{\theta}\right)$;
- Noyau exponentiel : $k(x,x') = \sigma_0^2 \exp\left(-\frac{|x-x'|}{\theta}\right)$;
- Noyau puissance-exponentiel : $k(x,x') = \sigma_0^2 \exp\left(-\frac{|x-x'|^\nu}{\theta^\nu}\right)$.

On accordera une attention particulière aux paramètres d'échelle et de portée dans la section suivante, et on notera :

$$k(x,x') = \sigma_0^2 r(x,x') \text{ et } K = \sigma_0^2 R \,. \tag{1.6}$$

Les résultats présentés dans le chapitre suivant vont exploiter l'existence d'un paramètre d'échelle factorisé σ_0, ce qui n'est pas toujours le cas, en particulier pour des modèles où le noyau est construit comme une somme de deux (ou plus) noyaux. Un exemple fréquent de modèle additif est le krigeage prenant en compte un effet pépite (*nugget effect*) (Cressie 1992) :

$$k(x,x') = \sigma_0^2 r(x,x') + \tau^2 \delta_x(x') \,. \tag{1.7}$$

Dans de tels cas, pour pouvoir factoriser σ_0, on a besoin de considérer un effet pépite *relatif* :

$$k(x,x') = \sigma_0^2 \left(r(x,x') + \tau^2 \delta_x(x')\right) \,. \tag{1.8}$$

De la sorte, l'effet pépite effectif est proportionnel à σ_0^2. Comme on vient de le voir, l'effet pépite est une discontinuité en $x = x'$ que l'on rajoute au noyau. Cela se traduit dans les formules par une diagonale constante que l'on ajoute à la matrice $K = k(X,X)$. Cependant lorsqu'on fait une prédiction sur un point x autre qu'un point d'observation, le vecteur $k(x,X)$ n'est pas affecté. Le modèle reste interpolant, avec une moyenne qui passe par les points d'observation et une variance qui s'y annule, car $k(X_i,X)K^{-1} = e_i$ (e_i est le vecteur ayant un 1 en i-ème position et 0 ailleurs) ; mais la moyenne et la variance de krigeage sont discontinues aux points d'observation. En revanche, les observations bruitées (*noise effect*) ont un effet légèrement différent. Chaque mesure peut avoir un bruit qui lui est propre. La variance du bruit est ajoutée à la variance *a priori* aux points considérés, c'est-à-dire que l'on ajoute une diagonale ε^2 à la matrice K dont chaque terme est égal à la variance de son observation correspondante. Toutefois,

la fonction $k(x, x')$ n'est pas affectée, même en $x' = x$. Le modèle résultant n'est plus interpolant car $k(X_i, X)(K + \varepsilon^2 I_N)^{-1} \neq e_i$. La moyenne de krigeage ne passe plus aux points d'observation et la variance de krigeage ne s'y annule plus. Davantage d'informations sur les modèles de krigeage pour observations bruitées peuvent être trouvées dans (Picheny, Wagner, and Ginsbourger 2012) avec en particulier une discussion sur leur traitement lorsque le krigeage est utilisé pour optimiser (cf. section 1.3). Comme pour l'effet de pépite, il est possible de définir le bruit de manière *relative* à la variance du noyau, et de factoriser alors le paramètre d'échelle :

$$K = k(X, X) + \sigma_0^2 \varepsilon^2 = \sigma_0^2 \left(R + \varepsilon^2 I_N \right) \ . \tag{1.9}$$

Un exemple simple de noyau et de krigeage unidimensionnel est présenté un peu plus loin. Les noyaux multidimensionnels peuvent par exemple être construits par tensorisation de noyaux, c'est-à-dire en multipliant ensemble des noyaux unidimensionnels :

$$k\left(x, x'\right) = \prod_{l=1}^{d} k_l(x_l, x_l') \ , \tag{1.10}$$

où d est la dimension de l'espace d'entrée $\mathbb{D}$. Les kernels de chaque dimension $k_j(\cdot, \cdot)$ peuvent être différents ou non. Ils peuvent ou non avoir des paramètres en commun. D'autres manières de composer des kernels unidimensionnels pour en faire un multidimensionnel existent, comme l'addition, la transformation isométrique ou la composition ANOVA. Le lecteur peut se référer à (Durrande et al. 2013) pour davantage de détails.

Exemple

La figure 1.2 montre deux modèles de krigeage pour 7 points d'observations en une dimension. La fonction de covariance utilisée est le noyau gaussien (ou Matérn infini, ou exponentielle carrée) avec deux paramètres (σ et θ), et une tendance linéaire :

$$r_\theta\left(x, x'\right) = \exp\left(-\frac{(x - x')^2}{2\theta^2} \right) \ , \tag{1.11}$$

$$k(x, x') = \sigma^2 r_\theta\left(x, x'\right) \ , \tag{1.12}$$

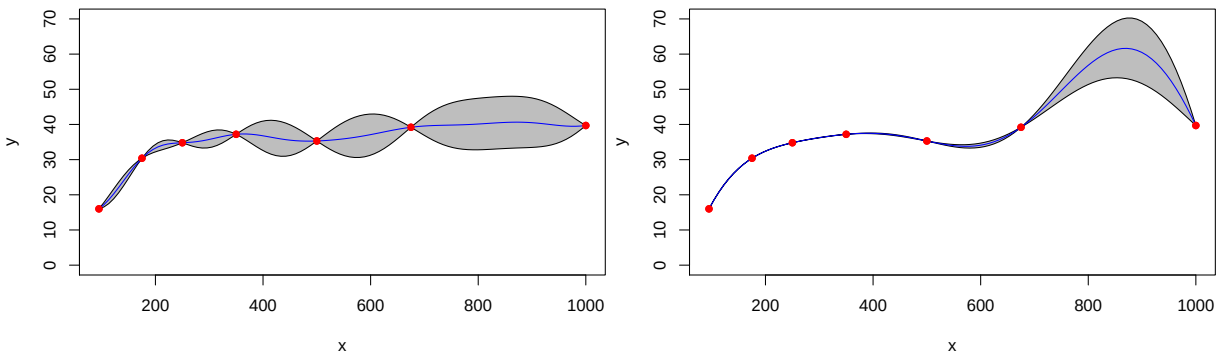
$$m(x) = \beta_0 + \beta_1 x \ . \tag{1.13}$$

Pour chaque modèle, la figure montre les points d'observation, la moyenne de krigeage et l'intervalle de confiance à 95% (pour garder l'interprétation simple, la variance de krigeage ne prend pas en compte l'incertitude lié à l'estimation des paramètres de tendance β_0 et β_1). La seule différence entre les deux modèles tient dans les paramètres du noyau (σ and θ), qui sont fixés arbitrairement à des valeurs différentes, et les paramètres de tendance (β_0 et β_1), qui sont estimés par la même méthode (maximum de vraisemblance, voir plus loin), laquelle donne des résultats différents dans les deux cas.

1.2.2 Estimation des paramètres

Comme on a pu le voir sur la figure 1.2, une estimation correcte des paramètres est essentielle à la construction d'un modèle de krigeage. Les deux approches principales pour estimer les paramètres sont la validation croisée (*Cross Validation*), typiquement dans sa version *Leave One Out* (un point extrait à la fois) (Sundararajan and Keerthi 2001), et l'estimation par maximum de vraisemblance (*Maximum Likelihood Estimation*, MLE). On pourra se référer à (Bachoc 2013) ou (Song, Choi, and Lamb 2013) pour une comparaison des deux méthodes. Dans notre cas, on s'est intéressé uniquement à l'approche par maximum de vraisemblance.

L'estimation des paramètres par MLE exploite l'aspect stochastique du modèle. Puisque le modèle de krigeage est obtenu en conditionnant un processus gaussien par les points connus, on peut ajuster les paramètres de façon à minimiser la déformation du processus induite par le conditionnement. C'est-à-dire que l'on va choisir les paramètres de façon à ce que les points connus soient, simultanément, le plus



(a) Paramètres : $\sigma = 3{,}80$, $\theta = 66{,}95$ (b) Paramètres : $\sigma = 100$, $\theta = 300$

FIGURE 1.2 – Exemple de modèles de krigeage, pour deux différents jeux de paramètres. Les points rouges sont les valeurs connues. Les lignes bleues sont les moyennes prédites et les aires grises représentent les intervalles de confiance à 95%.

vraisemblable possible par le processus gaussien. Cela revient à estimer les paramètres en maximisant la fonction de vraisemblance du modèle. Pour un jeu de données (X, Y), cette vraisemblance s'écrit :

$$L(\beta, \sigma_0^2, \Theta) = \frac{1}{(2\pi)^{N/2} \det(K)^{1/2}} \exp\left(-\frac{1}{2} \, {}^t(Y - m(X)) \, K^{-1} \, (Y - m(X)) \right) \ . \tag{1.14}$$

La matrice K dépend des paramètres σ_0^2 et Θ, mais Ces paramètres peuvent être séparés. Ainsi $K = \sigma_0^2 R$ (équation (1.6)), R ne dépendant que des paramètres Θ. Le maximum de la vraisemblance est atteint pour les mêmes valeurs que le maximum de son logarithme (la « log-vraisemblance »), correspondant à :

$$\begin{aligned} \mathcal{L}(\beta, \sigma_0^2, \Theta) &= \ln(L(\beta, \sigma_0^2, \Theta)) \\ &= -\tfrac{N}{2} \ln(2\pi) - \tfrac{1}{2} \ln(\det(R)) - \tfrac{N}{2} \ln(\sigma_0^2)) - \tfrac{1}{2\sigma_0^2} \, {}^t(Y - F\beta) \, R^{-1} \, (Y - F\beta) \ , \end{aligned} \tag{1.15}$$

avec $F = [f_1(X) \ f_2(X) \ \ldots \ f_p(X)]$ (matrice de taille $N \times p$).

On peut connaître analytiquement les paramètres de tendance β et d'échelle σ_0 qui maximisent la vraisemblance, en fonction des autres paramètres. Ces paramètres optimaux sont bien connus (Park and Baek 2001) et facilement redémontrables (voir Annexe A), et correspondent à :

$$\hat{\beta} = \left({}^tFR^{-1}F \right)^{-1} \ {}^tFR^{-1}Y \ , \tag{1.16}$$

$$\hat{\sigma_0}^2 = \frac{1}{N} \, {}^t\left(Y - F\hat{\beta}\right) R^{-1} \left(Y - F\hat{\beta}\right) \ . \tag{1.17}$$

En réutilisant ces formules dans l'équation de la log-vraisemblance, on obtient la « log-vraisemblance concentrée » (*concentrated log-likelihood*, parfois aussi appelée « *profile log-likelihood* ») :

$$\mathcal{L}_c(\Theta) = -\frac{N}{2} \ln(2\pi) - \frac{1}{2} \ln(\det(R)) - \frac{N}{2} \ln(\hat{\sigma_0}^2)) - \frac{N}{2} \ . \tag{1.18}$$

Le résultat analytique pour les paramètres de tendance est systématiquement utilisé. Il est possible, et fréquent, de prendre en compte l'incertitude de cette estimation dans la variance de prédiction (Santner, Williams, and Notz 2003). Dans nos travaux, nous faisons en sorte de pouvoir également appliquer l'expression obtenue pour $\hat{\sigma}_0^2$. Cela implique de choisir notre modèle de krigeage, et son paramétrage, de façon à pouvoir factoriser le paramètre σ_0. Pour résumer, les paramètres de krigeage sont traditionnellement estimés en maximisant la vraisemblance concentrée (équation 1.18) sur l'ensemble des paramètres de portée Θ admissibles, les paramètres de variance et de tendance étant donnés par les équations 1.17 et 1.16. On peut noter que la log-vraisemblance concentrée a la même complexité numérique que la log-vraisemblance classique (puisque $\hat{\sigma}_0$ nécessite quand même le calcul de R^{-1}).

Les bénéfices de la concentration de la vraisemblance sont doubles. D'une part, cela réduit le nombre de paramètres sur lesquels on fait marcher notre algorithme d'optimisation. C'est important car la log-vraisemblance est une fonction non-convexe des paramètres (sauf pour le paramètre d'échelle), présentant souvent plusieurs optima locaux. La maximisation est de fait non triviale et nécessite un algorithme d'optimisation globale. Localement, des algorithmes de descente de gradient sont souvent utilisés, en particulier quand le calcul analytique du gradient de la log-vraisemblance a été implémenté. D'autre part, les directions pointées par les gradients de la log-vraisemblance et de sa version concentrée sont différents dans l'espace des paramètres. En fait, les gradients de la log-vraisemblance concentrée pointent dans une direction plus intéressante pour l'optimisation car ils prennent en compte les informations d'ordre plus élevé (la courbure) pour le paramètre σ_0. Concentrer la vraisemblance permet de rendre la maximisation plus robuste et plus rapide, comme nous l'illustrons dans l'exemple qui suit.

Exemple

L'exemple de la figure 1.2 est à présent réutilisé pour illustrer les effets de la concentration de la vraisemblance sur l'optimisation des paramètres du modèle σ_0 et θ. Les optimisations sont réalisées par un algorithme utilisant une méthode quasi-Newton de type BFGS (Broyden-Fletcher-Goldfard-Shanno) avec contrainte aux bords (Byrd et al. 1995). La non-convexité de la log-vraisemblance, concentrée ou non, se traduit par une convergence de l'algorithme qui dépend de son initialisation (les paramètres choisis pour démarrer la descente de gradient). Pour cette raison, les optimisations de la log-vraisemblance et de la log-vraisemblance concentrée sont effectuées 100 fois en choisissant à chaque fois les paramètres initiaux de manière aléatoire (pour la version concentrée, le sigma tiré est ignoré, puisque le sigma optimal est calculé à partir du théta choisi). Les résultats sont résumés sur les figures 1.3 à 1.6.

La figure 1.3 montre les lignes de niveau correspondantes aux valeurs de la log-vraisemblance, en tant que fonction de σ_0 et de θ, ainsi que les paramètres obtenus par optimisation. On y voit un plateau, proche de la valeur maximale. Beaucoup d'optimisations s'y arrêtent car elle est reconnue comme un optimum local. Dans cet exemple, le résultat de l'optimisation sur la log-vraisemblance et sur sa version concentrée est sensiblement le même. La figure 1.4 confirme ceci : les histogrammes des valeurs de log-vraisemblance sont les mêmes avec et sans concentration, et un point initial conduit presque toujours au même optimum local ou global.

Toutefois, concentrer la vraisemblance a un impact positif sur la vitesse d'optimisation. La figure 1.5 montre les vitesses de convergence en terme de nombre d'appels à la fonction de vraisemblance. On y voit une différence notable entre les vraisemblances concentrée et non-concentrée : dans 98% des cas, optimiser la vraisemblance concentrée prend moins de temps (moins d'appels à la fonction) que dans le cas de la vraisemblance non-concentrée.

Enfin, on peut synthétiser ces informations en représentant la convergence comme sur la figure 1.6. Pour chaque initialisation, on trace la meilleure valeur de la log-vraisemblance en fonction du nombre d'appels à la fonction qui la calcule. On y distingue clairement le gain de convergence obtenu en concentrant la vraisemblance. Les deux phénomènes y sont visibles : dans le cas concentré comme non concentré, la valeur finale après convergence est distribuée de la même manière entre les optima locaux et globaux, mais dans le cas non concentré (en bleu) on a besoin de plus d'itérations avant d'atteindre cette valeur finale.

Améliorations du krigeage

Depuis les débuts du krigeage, de nombreux travaux ont apporté leur contribution, afin de tenir compte des spécificités de certaines applications. On peut trouver dans (Picheny, Wagner, and Ginsbourger 2012) des informations pour traiter le cas de mesures bruitées par exemple. Ou encore on peut modéliser une fonction soumise à des contraintes de monotonie, de positivité ou de convexité (Maatouk 2015). D'autres personnes (Roustant et al. 2018) se sont intéressées à ce qu'il est possible de faire lorsque l'espace d'entrée comprend des variables qualitatives. De nombreuses améliorations du krigeage concernent la haute dimension, que ce soit la dimension de l'espace des entrées d ou le nombre de points d'observations N. Une des approche de réduction de dimension est de décomposer les d variables en des sous-groupes de variables indépendantes et de sommer les sous-modèles (Hastie 2017). D'autres approches de réduction de dimension construisent le modèle de krigeage dans un sous-espace de l'espace initial qui est déterminé

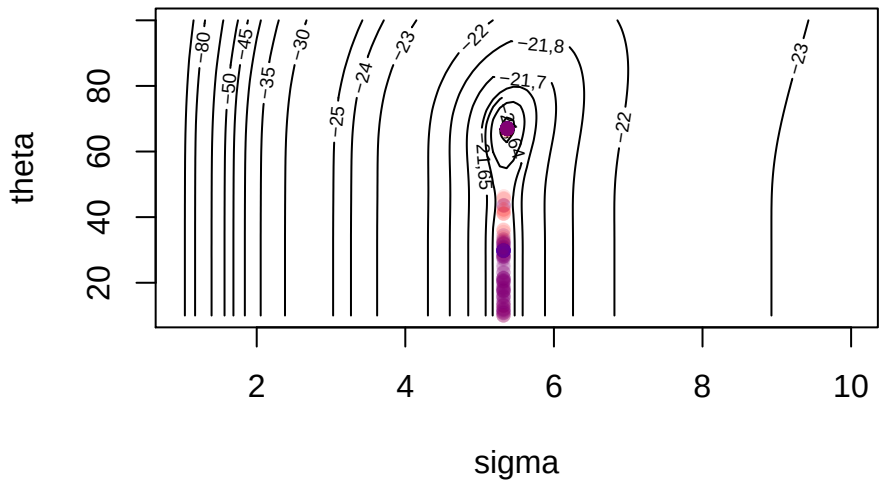


FIGURE 1.3 – Lignes de niveaux de la log-vraisemblance (kernel exponentiel carré, cf. figure 1.2). Les points rouges et bleus sont les résultats de la maximisation, pour diverses initialisations, de la fonction de log-vraisemblance avec et sans concentration, respectivement. Les points sont affichés avec une légère transparence. Notons le plateau proche de l'optimum où beaucoup d'optimisations s'arrêtent.

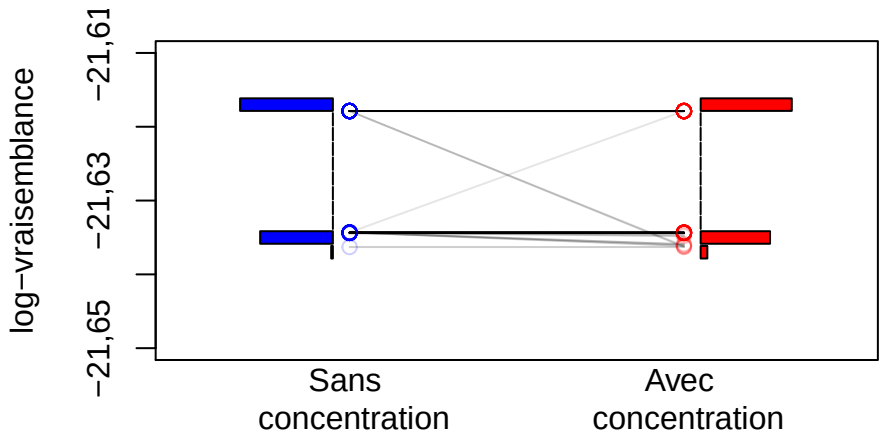


FIGURE 1.4 – Histogrammes des valeurs finales trouvées par la maximisation de la log-vraisemblance concentrée (rouge) ou non (bleu), pour différentes initialisations. Les lignes grises relient les valeurs obtenues dans les deux cas pour une même initialisation. On ne distingue pas de différence significative entre les deux méthodes (98% des différences absolues sont inférieures à 2×10^{-3}).

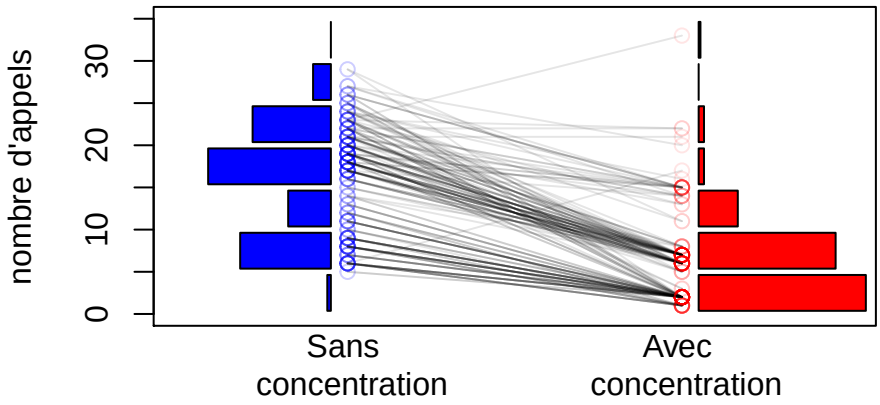


FIGURE 1.5 – Histogrammes des nombres d'appels à la fonction de log-vraisemblance (bleu) et de log-vraisemblance concentrée (rouge) nécessaires pour leur optimisation, pour différentes initialisations. Les lignes grises relient les nombres d'appels des deux cas pour une même initialisation. On observe que la concentration améliore (réduit) presque toujours le nombre d'appels nécessaire (98% d'amélioration strict, 99% de non-détérioration).

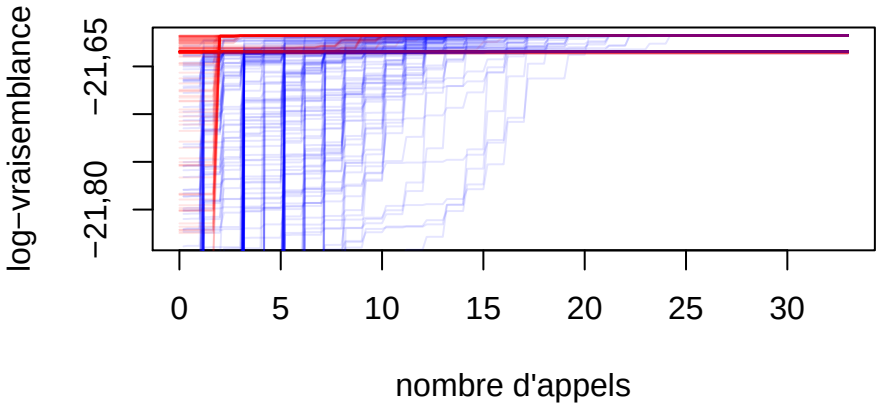


FIGURE 1.6 – Meilleures valeurs de log-vraisemblance trouvées durant son optimisation en fonction du nombre d'appels à la fonction, pour différentes initialisations et pour les cas concentrés (en rouge) et non-concentrés (en bleu). Le nombre d'appels n'inclut pas les appels faits pour approximer la dérivée par différences finies. On voit que concentrer la log-vraisemblance conduit à un taux de convergence bien meilleur.

par *Partial Least Squares* dans (Bouhlel et al. 2016) ou maximum de vraisemblance dans (Tripathy, Bilionis, and Gonzalez 2016). Le grand nombre de points de données N est typiquement abordé par le remplacement des données par des points résumés (*inducing points*) (Hensman, Fusi, and Lawrence 2013), des fonctions de covariance à support restreints (Maurya 2016) ou des combinaisons linéaires de processus gaussiens bâtis sur une partie des données (Rullière et al. 2018). La plupart de ces avancées sont compatibles entre elles. Les projets DICE (2006), ReDICE (2011) et OQUAIDO (2016) ont généré une ensemble d'outils en R pour la modélisation par krigeage : *DiceDesign* (Dupuy, Helbert, and Franco 2015), *DiceKriging* (Roustant, Ginsbourger, and Deville 2012), *kergp* (Deville, Ginsbourger, and Roustant 2015) et *NestedKriging* (Rullière et al. 2018) en particulier. Le cokrigeage, et les autres développements détaillés dans le chapitre 2 ont pour objectif de venir augmenter cette palette d'outils.

1.3 Algorithmes itératifs

L'usage des métamodèles probabilistes, tels que le krigeage, permet d'atteindre un objectif d'ingénierie pour un coût de simulations réduit, grâce à des algorithmes itératifs adéquats. La simulation numérique du phénomène physique peut être interprétée mathématiquement comme une fonction scalaire $y(\cdot)$ de l'espace d'entrée $\mathbb{D}$ vers $\mathbb{R}$. Le simulateur physique est alors traité comme une « boîte noire » : peu importe comment il fonctionne, on s'intéresse à la sortie (scalaire) qu'il produit quand on lui donne un point de l'espace d'entrée. L'espace d'entrée correspond à toutes les valeurs admissibles pour les paramètres de la simulations (à ne pas confondre avec les paramètres du métamodèle). En tout état de cause, cet espace d'entrée peut avoir des variables discrètes ou discontinues, mais nous nous limitons au cas où le domaine $\mathbb{D}$ est un sous-ensemble fini, borné et convexe de $\mathbb{R}^d$, topologiquement semblable à l'hypercube $[0,1]^d$. Cette fonction scalaire correspond à la grandeur d'intérêt, ou grandeur de sortie, de la simulation. Elle n'est connue qu'en un nombre de points fini, notés X. Les valeurs correspondantes sont notées $Y = y(X)$. Le nombre de points connus (noté N) est souvent réduit car l'évaluation de la fonction $y(\cdot)$ est coûteuse, ce qui justifie l'usage d'un métamodèle.

L'objectif d'ingénierie peut prendre des formes diverses. Il correspond souvent à un problème d'optimisation (rechercher la pire ou la meilleure configuration par exemple). Les problèmes de maximisation et de minimisation sont mathématiquement semblables. Il y a également une équivalence entre rechercher la valeur maximale que peut prendre la grandeur simulée et rechercher la position de ce maximum. Le problème d'optimisation peut être formulé comme :

$$x^* = \underset{x \in \mathbb{D}}{\operatorname{argmin}}\, y(x) \ . \tag{1.19}$$

Les ingénieurs ont également souvent affaire à des problèmes d'inversion (parfois appelé problème inverse). Il s'agit alors de trouver l'ensemble des configurations pour lesquelles la grandeur simulée prend une valeur fixée (noté T). De manière équivalente, on peut être amené à rechercher le sous-ensemble du domaine pour lequel la grandeur simulée reste en-dessous ou au-dessus d'une certaine valeur limite T. Un problème d'inversion peut être formulé comme :

$$\text{Trouver l'ensemble } \ \{x^* \in \mathbb{D} \mid y(x^*) = T\} \ . \tag{1.20}$$

Il existe bien d'autres problèmes que peuvent rencontrer les ingénieurs, comme rechercher des optima locaux, trouver les régions du domaines où la grandeur est comprise entre deux valeurs, faire une inversion avec une des variables qui est « pénalisée » (qui doit être dans son pire cas), etc.

L'idée générale des algorithmes itératifs sur base de métamodèle est assez simple, et est donnée par l'algorithme 1.1 : à partir des données on construit un métamodèle dont on se sert pour déduire le point le plus « intéressant » à connaître (pour le problème donné, de maximisation, d'inversion, etc.), on exécute la simulation physique pour ce point-là, on met à jour le métamodèle, et on recommence. Cette démarche vise à économiser autant que possible le nombre d'appels au simulateur, considéré comme coûteux, et à les remplacer par des appels au métamodèle. Ici on se base sur des métamodèles sous forme de processus gaussien conditionné (krigeage), qui sont très populaires depuis (Jones, Schonlau, and Welch 1998).

Algorithme 1.1 : Plan d'expériences itératif pour des simulations numériques, basé sur le critère d'intérêt J_M.

Entrées : nombre initial de points N_i, nombre total de points N_{max}, fonction inconnue y, description du domaine $\mathbb{D}$ (typiquement bornes pour les variables continues), famille paramétrique de métamodèles M, critère à maximiser J_M selon l'objectif à atteindre.

Choisir un échantillon de N_i points (X) uniformément répartis dans $\mathbb{D}$.
Évaluer $Y = f(X)$.
Ajuster un métamodèle M sur (X, Y).
Pour i **de** $N_i + 1$ **jusqu'à** N_{max} **faire :**
  Chercher $X_{new} = \text{argmax}_{x \in \mathbb{D}} \left(J_M(x) \right)$.
  Évaluer $Y_{new} = y(X_{new})$.
  Mettre à jour $X \leftarrow X \cup X_{new}$ et $Y \leftarrow Y \cup Y_{new}$.
  Réajuster le métamodèle M sur (X, Y).
Fin

Résultat : échantillon de points X de l'espace d'entrée $\mathbb{D}$, valeurs Y correspondantes, métamodèle paramétré M.

L'initialisation de l'algorithme nécessite de choisir un premier plan d'expériences et d'évaluer la fonction-objectif sur ces points. Ces points initiaux servent à ajuster le premier métamodèle. Par la suite, les points sont ajoutés un à un, grâce au métamodèle. On choisit généralement le plan initial selon un hypercube latin (LHS pour *Latin Hypercube Sample*). Les plans factoriels présentent en effet des alignements qui peuvent rendre un modèle de krigeage difficile à ajuster (surtout si les dimensions sont combinées de façon tensorielle). L'hypercube latin est optimisé à l'aide du critère *maximin* afin d'assurer une bonne couverture de l'espace. Plus d'information sur les hypercubes latins peuvent être trouvées dans (Stein 1987). Lorsque la dimension n'est pas trop grande, on augmente ce plan avec les bords du domaine. L'incertitude sur ces régions est en effet très grande sans ces points aux extrémités, et l'algorithme les visite alors rapidement. Toutefois, des études récentes tendent à montrer qu'il peut être préférable d'utiliser un premier plan d'expériences aléatoire plutôt que LHS pour pouvoir mieux caractériser les paramètres du métamodèle (Zhang, Cole, and Gramacy 2019).

Le calcul du point « le plus intéressant à connaître » est la partie la plus délicate. Cela se fait en maximisant un critère $J_M(\cdot)$, qui n'est autre qu'une fonction de l'espace d'entrée exploitant les données du métamodèle. Ce critère dépend de l'objectif d'ingénierie bien évidemment (problème d'optimisation ou problème d'inversion par exemple). Le problème initial est décomposé en une suite de problèmes d'optimisation :

$$x^* = \underset{x \in \mathbb{D}}{\text{argmax}}\, J_M(x) \ . \tag{1.21}$$

pour lesquels le critère d'acquisition J_M n'utilise plus la fonction y mais le métamodèle M. Le critère J_M dépend également des poids respectifs que l'on veut donner à l'exploration de l'ensemble de l'espace d'entrée (où la variance de krigeage est élevée) et à l'exploitation des régions localement proches de points déjà calculés dont les valeurs sont intéressantes pour le problème considéré (où la variance de krigeage est faible, mais la moyenne est proche de ce qu'on recherche). Différents exemples de critères peuvent être trouvés dans (Jones 2001) et (Frazier 2018).

L'équilibrage du critère entre exploration et exploitation caractérise le comportement prospectif de l'algorithme. Si le calcul dudit critère vise à valoriser **l'information apportée** sur l'objectif plutôt que **l'atteinte immédiate** de l'objectif, nous parlerons de critère « presbyte » ; et par opposition, un critère valorisant l'atteinte immédiate de l'objectif sera considéré comme « myope ». Le critère réalisera donc un compromis entre la réalisation immédiate d'un gain et la construction du métamodèle. Ce compromis est rendu possible par l'aspect stochastique du métamodèle car il permet de mesurer les gains d'information apportés par les nouveaux points. En pratique, au delà de la définition du critère, son paramétrage heuristique permettra généralement d'accéder à une nuance entre ces différents comportements. En outre, l'objectif d'ingénierie lui-même peut être plus ou moins propice à choisir la stratégie algorithmique, attendu que certains problèmes souffrent de non-identifiabilité et sont donc particulièrement pénalisés par une démarche de myopie (par exemple les problèmes d'inversion déterministes). Différents auteurs se sont intéressés à ces problèmes, parmi lesquels (Schonlau, Welch, and Jones 1998), (Gramacy and Lee 2010) et (Le Riche, Girdziušas, and Janusevskis 2012)

Formellement, si on considère un critère myope $J_M(\cdot)$ dont le maximum sur $\mathbb{D}$ est obtenu pour maximiser l'objectif d'ingénierie, il est possible de construire un critère presbyte associé $ICJ_M(\cdot)$ (*Integrated Conditional* J_M). Celui-ci se définit pour n'importe quel point x comme l'espérance de l'intégrale sur l'ensemble du domaine $\mathbb{D}$ du critère $J_{M'}$, exploitant le métamodèle M' qui correspond au métamodèle M conditionné par les résultats probables au point x :

$$ICJ_M(x) = \int_{z\sim\mathcal{N}(\mu(x),\sigma(x))} \int_{x'\in\mathbb{D}} J_{M'}(x' \mid y(x) = z)dx' \; dz \; . \tag{1.22}$$

Cette démarche a était introduite par (Gramacy and Lee 2010) pour le critère EI, détaillé dans la section suivante. Alors que le critère J_M est maximisé, sa version presbyte ICJ_M est minimisée. L'intégrale du critère J_M sur le domaine $\mathbb{D}$ peut être vu comme une incertitude sur l'objectif d'ingénierie, que l'on cherche alors à minimiser. On cherche donc le point x pour lequel, si on fait une évaluation à cette endroit là, on réduira au maximum cette incertitude (en espérance). C'est le principe de la famille de critères SUR (*Stepwise Uncertainty Reduction*), dont les membres se différencient par leur définition de l'incertitude (Bect et al. 2012).

On pourrait également vouloir minimiser le maximum de J_M plutôt que son intégrale sur tout le domaine, puisque c'est le maximum qui indique le point le « plus intéressant ». Mais l'avantage de prendre pour incertitude l'intégrale de J_M sur tout le domaine est de pouvoir intervertir les intégrales (sous réserve d'un J_M « raisonnable »). On peut alors isoler mathématiquement la quantité CJ_M (*Conditional* J) définie ainsi :

$$CJ_M(x', x) = \int_{z\sim\mathcal{N}(\mu(x),\sigma(x))} J_{M'}(x' \mid y(x) = z)dz \; , \tag{1.23}$$

et qui peut avoir une forme analytique pour certains critères J_M, ce qui facilite grandement le calcul de $ICJ_M(\cdot)$.

1.3.1 Critères pour l'optimisation

Nous n'avons pas vocation à être exhaustif, mais nous présentons ici les principaux critères utilisés pour une recherche d'un minimum ou d'un maximum global.

Expected Improvement (EI)

Pour l'optimisation (par exemple une minimisation), le critère le plus naturel (qui a de surcroît le mérite de ne pas nécessiter de paramètres supplémentaires) est l'amélioration espérée par rapport au meilleur point trouvé (*Expected Improvement*, EI). Pour un point x donné, $EI(x)$ correspond à l'espérance de la différence entre le point le plus bas parmi ceux déjà trouvés (pour une minimisation) et le futur $y(x)$, lorsque cette différence est positive. Pour un problème de minimisation, ce critère s'écrit :

$$EI(x) \;=\; \mathbb{E}\left[\max(0, Y_{min} - Z(x))\right] \quad \text{où} \quad Y_{min} = \min(y(X_1), \ldots, y(X_N)) \; . \tag{1.24}$$

Ce critère EI est un critère myope : il cherche le meilleur candidat pour l'itération en cours. De manière équivalente, il peut être définit comme l'espérance du nouveau minimum si le nouveau point est placé en x, à une transformation affine près. En effet l'équation précédente est équivalente à :

$$EI(x) \;=\; Y_{min} - \mathbb{E}\left[\min(Y_{min}, Z(x))\right] \; . \tag{1.25}$$

Ce critère peut également s'exprimer de manière analytique à l'aide de la variance et de la moyenne de krigeage,

$$EI(x) = \begin{cases} (Y_{min} - \mu(x))\,\Phi\Big(\frac{Y_{min}-\mu(x)}{\sigma(x)}\Big) + \sigma(x)\,\phi\Big(\frac{Y_{min}-\mu(x)}{\sigma(x)}\Big) & \text{lorsque } \sigma(x) > 0 \; , \\ \max\left(0, (Y_{min} - \mu(x))\right) & \text{lorsque } \sigma(x) = 0 \; , \end{cases} \tag{1.26}$$

où $\mu(x)$ et $\sigma(x)$ sont la moyenne et l'écart type de krigeage au point x, et $\Phi()$ et $\phi()$ sont les fonctions de répartition et de densité de la loi normale centrée réduite. Le pseudocode 1.2 décrit l'algorithme

EGO (*Efficient Global Optimisation*, optimisation globale efficiente, (Jones, Schonlau, and Welch 1998)), l'algorithme itératif bayésien exploitant le critère EI. Puisque l'EI est nulle aux points déjà connus ($EI(X_i) = 0$, $\forall\ 1 \leq i \leq N$), l'algorithme construit une série de points dense dans $\mathbb{D}$ et, asymptotiquement, trouvera l'optimum global (Bect, Bachoc, and Ginsbourger 2019). En pratique cependant, les versions standards d'un tel algorithme ne peuvent guère trouver plus d'un millier de points, à cause des problèmes numériques engendrés par l'inversion de la matrice de covariance K qui est de taille $N \times N$ (cf. équations (1.2), (1.3) et (1.4)). Différents travaux étudient la possibilités de construire des modèles de krigeage exploitant un grand nombre de données mais ce n'est pas l'objet de cette thèse (Rullière et al. 2018).

Algorithme 1.2 : Algorithme d'optimisation EGO, pour trouver un minimum.

Entrées : nombre de points total N_{max}, fonction y à minimiser sur le domaine $\mathbb{D}$, processus gaussien (tendance et fonction de covariance paramétrique), plan d'expériences initial $[(X_1, y(X_1))\ ,\ \dots\ ,\ (X_N, y(X_N))]$.

$Y_{min} = \min(y(X_1)\ ,\ \dots\ ,\ y(X_N))$.
Ajuster les paramètres du processus gaussien sur le plan d'expériences.
Tant que $N \leq N_{max}$ **faire :**
  Chercher $X_{new} = \text{argmax}_{x\in\mathbb{D}}\,(EI(x))$.
  Évaluer $Y_{new} = y(X_{new})$.
  Mettre à jour le plan d'expériences avec (X_{new}, Y_{new}).
  $Y_{min} \leftarrow \min(Y_{new}, Y_{min})$.
  Réajuster les paramètres du processus gaussien sur le nouveau plan d'expériences.
Fin

Résultat : Plan d'expériences total, Y_{min}, dernier processus gaussien utilisé (avec ses paramètres).

Pour illustrer le fonctionnement de l'algorithme EGO, nous utilisons comme fonction à minimiser la fonction de Branin, modifiée de sorte qu'elle possède trois minima, dont un global en (0,543, 0,15), dans le domaine $[0, 1]^2$. Cette fonction est alors définit par l'équation suivante :

$$\begin{array}{rl} \overline{x_1} & = 15x_1 - 5 \quad , \quad \overline{x_2} = 15x_2 \\ y(x) & = \left(\overline{x_2} - \frac{5\overline{x_1}^2}{4\pi^2} + \frac{5\overline{x_1}}{\pi} - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(\overline{x_1}) + 11 - \exp(-\frac{(\overline{x_1}-0,5)^2}{15})\ . \end{array} \tag{1.27}$$

Après un premier échantillonnage de 10 points (4 points dans les coins et 6 points en LHS), suivi de 9 itérations de EGO, l'espace d'entrée est couvert comme montré sur la figure 1.7. Notez comment les points de EGO (ronds noirs pleins) ont tendance à se rassembler autour des optima locaux de la fonction de Branin. La fonction EI à la fin des 9 itérations est affichée sous forme de lignes de niveaux en figure 1.8. Le maximum de l'EI nous donne le point de la prochaine itération, dans cet exemple (0,107, 0,792).

Integrated Expected Conditional Improvement ($IECI$)

Considérant le critère myope EI précédent, nous pouvons construire un critère presbyte $IECI$ (*Integrated Expected Conditional Improvement*, intégrale de l'espérance de l'amélioration conditionnée) à l'aide de l'équation (1.22) :

$$IECI(x) = \int_{z\sim\mathcal{N}(\mu(x),\sigma(x))} \int_{x'\in\mathbb{D}} EI(x' \mid y(x) = z)dx'\ dz\ . \tag{1.28}$$

L'appellation retenue est $IECI$ plutôt que $ICEI$ car les intégrales peuvent être interverties et on peut isoler la quantité ECI (*Expected Conditional Improvement*), définie par :

$$ECI(x, x') = \int_{z\sim\mathcal{N}(\mu(x),\sigma(x))} EI(x' \mid y(x) = z)dz\ . \tag{1.29}$$

On aura donc $IECI(x) = \int_{x'\in\mathbb{D}} ECI(x, x')dx'$.

(Gramacy and Lee 2010) exploite le critère $IECI$ et calcule l'ECI par une approche Monte-Carlo. Une forme analytique de l'ECI existe également et est exploité par un package développé par (Richet 2015) à l'IRSN, et disponible sur github où son calcul est également détaillé.

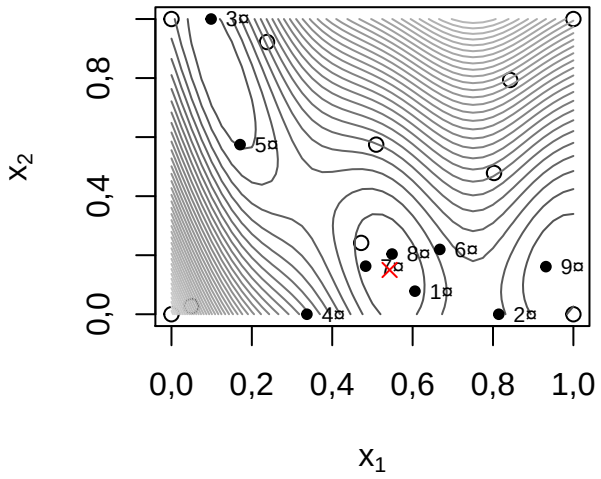


FIGURE 1.7 – Lignes de niveaux de la fonction de Branin modifiée, avec son minimum global en croix rouge, sur laquelle l'algorithme EGO a été lancé. Le plan d'expériences initial est figuré par les ronds creux et les 9 points générés par EGO sont les ronds pleins (numérotés selon leur ordre d'apparition).

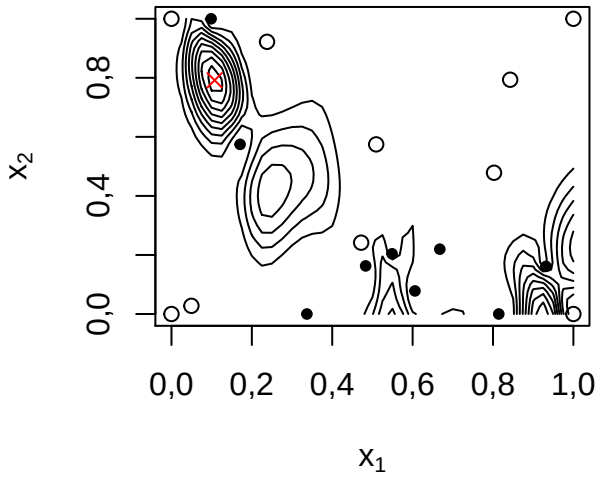


FIGURE 1.8 – Lignes de niveaux de l'EI après les 9 itérations présentées sur la figure 1.7. Noter comment l'EI fait des pics dans les bassins d'attraction de la fonction de Branin mais redescend à 0 près des points déjà évalués. La croix rouge correspond à la position du maximum de l'EI.

***Informational Approach to Global Optimization* (IAGO)**

L'algorithme IAGO (*Informational Approach to Global Optimization*, approche informationnelle à l'optimisation globale) est un exemple de stratégie SUR (*Stepwise Uncertainty Reduction*). Les stratégies SUR visent à quantifier l'incertitude, ou la méconnaissance, concernant l'objectif à atteindre, et cherche le point donnant les meilleurs espoirs de réduction de cette incertitude. IAGO utilise la notion d'entropie de l'information pour définir un critère presbyte.

Le lecteur est invité à se référer à (Villemonteix, Vazquez, and Walter 2009) pour plus d'informations.

1.3.2 Critères pour l'inversion

Les problèmes dits « d'inversion » sont plus complexes que les problèmes d'optimisation. En particulier, le résultat attendu n'est pas un point unique mais un ensemble de points (une ligne de niveau pour les cas 2D), ce qui en fait des problèmes mal-posés au sens de Hadamard. On cherche en effet toutes les valeurs de l'espace d'entrée $\mathbb{D}$ qui conduisent à une valeur ciblée, notée T. Il n'y a pas de critère aussi « naturel » que l'EI. Un package R (Chevalier et al. 2011) exploite certains critères connus pour résoudre de tels problèmes. Leurs auteurs détaillent les critères employés dans (Chevalier, Picheny, and Ginsbourger 2014).

Parmi les critère myopes employés pour l'inversion, on peut citer les critères de Ranjan (Ranjan, Bingham, and Michailidis 2008) et Bichon (Bichon et al. 2008), unifiés par (Bect et al. 2012) :

$$crit_{RB}(x) \;=\; \mathbb{E}\left[\max(0, \alpha^\delta \sigma(x)^\delta - |T-\mu(x)|^\delta\right] \;, \tag{1.30}$$

avec δ égal à 1 pour le critère de Bichon et 2 pour le critère de Ranjan, et α un paramètre du modèle permettant à l'utilisateur d'accorder plus ou moins de poids à l'exploration (α grand) ou à l'exploitation (α petit). Ces deux critères ont chacun une forme analytique, permettant des accès rapides à leurs valeurs.

On peut citer également le critère $TMSE$ (*Targeted Mean Square Error*) de (Picheny et al. 2010), qui nécessite le choix par l'utilisateur d'un paramètre ε (dont une forte valeur incite l'algorithme à davantage d'exploration) :

$$TMSE(x) \;=\; \frac{\sigma(x)^2}{\sqrt{\sigma(x)^2+\varepsilon^2}}\phi\left(\frac{\mu(x)-T}{\sqrt{\sigma(x)^2+\varepsilon^2}}\right) \;. \tag{1.31}$$

Le critère de l'EI a inspiré un autre critère myope pour l'inversion, intégré au package R *KrigInv* (Chevalier et al. 2011, avec la contribution de Yann Richet). Il s'agit du $TSEE$ (*Two Sided Expected Excedance*), qui se calcule comme le produit des espérances des écarts à la valeur ciblée, respectivement en étant au-dessus et en étant en dessous. On peut aussi le voir comme l'espérance du produit des écarts autour de la valeur ciblée de deux réalisations indépendantes du processus gaussien :

$$\begin{array}{rcl}
TSEE(x) & = & \mathbb{E}\left(\max(0, T-Z(x))\right)\mathbb{E}\left(\max(0, Z(x)-T)\right) \\
& = & \mathbb{E}\left(\max(0, T-Z_1(x))\max(0, Z_2(x)-T)\right) \\
& = & \mathbb{E}\left((T-Z_1(x))(Z_2(x)-T) \mid Z_1(x)<T<Z_2(x)\right)\mathrm{P}(Z_1(x)<T<Z_2(x)) \\
& = & \frac{1}{2}\mathbb{E}\left((T-Z_1(x))(Z_2(x)-T) \mid Z_1(x)<T<Z_2(x)\right)\mathrm{P}(Z_1(x)<T<Z_2(x)) \\
& & + \frac{1}{2}\mathbb{E}\left((T-Z_1(x))(Z_2(x)-T) \mid Z_1(x)>T>Z_2(x)\right)\mathrm{P}(Z_1(x)>T>Z_2(x)) \\
& & + \;0\;\mathrm{P}(Z_1(x)>T \;\&\; Z_2(x)>T) \;+\; 0\;\mathrm{P}(Z_1(x)<T \;\&\; Z_2(x)<T) \\
& = & \frac{1}{2}\mathbb{E}\left(\max\left(0, (T-Z_1(x))(Z_2(x)-T)\right)\right) \;,
\end{array} \tag{1.32}$$

avec $Z_1(\cdot)$ et $Z_2(\cdot)$ deux processus gaussiens indépendants et identiquement distribués, similaires au modèle de krigeage $Z(\cdot)$.

Des critères presbytes ont également été développés pour l'inversion. (Bect et al. 2012) en présentent quelques-uns, construits par la stratégie SUR (*Stewise Uncertainty Reduction*), et améliorés par (Chevalier et al. 2014) qui rendent les formules intégrales calculables, reposant sur l'intégrale de la loi normale centrée réduite bivariée supposée exactement connue (en réalité approchée efficacement par (Genz and Bretz 2009)). On peut citer par exemple le critère $TIMSE$ qui est la version intégrale du critère $TMSE$

(Picheny et al. 2010). (Bect et al. 2012) ont également introduit un critère SUR qui utilise comme incertitude à réduire la variance de la variable de Bernoulli « $Z(x)$ est-il au-dessus de la valeur ciblée T ? » après l'ajout d'un point.

1.3.3 Problème mixte inversion-optimisation

Les ingénieurs rencontrent parfois des problèmes encore plus complexes que ceux d'optimisation globale ou d'inversion. On peut citer à titre d'exemples les problèmes consistant à rechercher l'ensemble des optima locaux, à trouver les régions du domaine où la grandeur est comprise entre deux valeurs, à faire une inversion avec une variable « de nuisance » (la variable doit être dans son pire cas), etc. Chaque problème nécessite une approche spécifique. Le principe général des algorithmes reste toutefois le même, et les critères nouvellement développés sont souvent des variantes de critères déjà connus.

L'article de (Ginsbourger et al. 2014) traite par exemple un problème de reconstruction de « ligne de crête », c'est-à-dire qu'on cherche le maximum global suivant certaines dimensions tandis qu'on parcourt les autres sur l'ensemble de leur domaine. Le domaine $\mathbb{D}$ se décompose en $\mathbb{D} = \mathbb{A} \times \mathbb{V}$, avec $\mathbb{A}$ les dimensions à parcourir et $\mathbb{V}$ les dimensions à maximiser. Le problème peut s'écrire :

$$\text{Trouver la fonction } \; v^* : \alpha \in \mathbb{A} \to v^*(\alpha) \in \mathbb{V} \; \text{ telle que } \; \forall \alpha \in \mathbb{A} \; , \; y(\alpha, v^*(\alpha)) = \max_{w \in \mathbb{V}} y(\alpha, w) \; . \tag{1.33}$$

La problématique comporte également la volonté de minimiser le nombre d'appels à la fonction-objectif, et le sujet est traité en utilisant les processus gaussiens. Les auteurs (dont je fais partie) développent alors une variante de l'algorithme EGO, appelée PEGO (*Profile Efficient Global Optimization*), adaptée à leur problème. L'algorithme PEGO repose sur un critère PEI (*Profile Expected Improvement*), variante du critère EI, et définit ainsi :

$$\begin{aligned} PEI(\alpha, v) &= \mathbb{E}\left[\max\left(0 \; , \; Z(\alpha, v) - Y_{max}(\alpha)\right)\right] \; , \\ \text{où} \quad Y_{max}(\alpha) &= \min\left(\max_{w \in \mathbb{V}} \mu(\alpha, w) \; , \max_{1 \leq i \leq N} y(X_i) \; \right) \; . \end{aligned} \tag{1.34}$$

On essaie ainsi d'améliorer la connaissance de la ligne de crête par rapport à un maximum local prédit par le modèle ($\max_{w \in \mathbb{V}} \mu(\alpha, w)$) et limité par le maximum global ($\max_{1 \leq i \leq N} y(X_i)$) (afin d'éviter de trop favoriser l'exploration).

Lors de cette thèse, j'ai contribué à implémenter pour l'IRSN cet algorithme PEGO, dans le cadre de l'étude d'algorithmes permettant de résoudre des problèmes de dimensionnement sous nuisance de sûreté-criticité (Garland 2017).

Stratégie de parallélisation

Dans le milieu industriel, les ingénieurs font calculer leurs simulations (générant les valeurs $y(\cdot)$) par des serveurs et ont la possibilité de lancer plusieurs calculs en même temps. La puissance de calcul des serveurs est néanmoins limitée et la volonté de faire appel à des optimisations parcimonieuses reste importante. Afin de répondre à leur problème d'ingénierie plus rapidement, il est souvent intéressant de faire appel à un algorithme qui ne leur indiquera pas **le** prochain calcul à effectuer, mais **les** prochains calculs à effectuer **simultanément**.

Les critères faciles à évaluer sont des critères myopes, qui cherchent à atteindre l'objectif immédiatement. Leur myopie peut être corrigée avec la méthode SUR qui permet de construire un critère intégral presbyte mais cela a un coût, le nouveau critère étant numériquement plus lourd à évaluer (et parfois beaucoup trop, au point de le rendre inutilisable en pratique). Une alternative pour corriger la myopie d'un critère est d'en construire une version permettant de proposer un ensemble de nouveaux points de calcul plutôt qu'un seul. Ainsi, l'algorithme peut viser un objectif à plus long terme, en explorant et exploitant plusieurs zones d'intérêt simultanément, évitant alors le risque de passer trop de temps à exploiter une zone qui se révélera finalement moins intéressante qu'une autre.

Certains critères peuvent être directement adaptés en une version « parallèle ». C'est notamment le cas de certains critères SUR qui, au lieu de regarder une espérance conditionnée à **un** nouveau point, peuvent être

formulés pour considérer une espérance conditionnée à **plusieurs** nouveaux points. (Chevalier, Picheny, and Ginsbourger 2014) donnent des informations sur la mise en pratique de certains de ces critères. On pourra trouver dans (Chevalier and Ginsbourger 2013) et (Ginsbourger, Le Riche, and Carraro 2010) des informations plus spécifiques sur les stratégies de parallélisation des algorithmes itératifs.

Chapitre 2

Modèles introspectifs

Dans le chapitre 1, nous avons vu ce qu'était le krigeage : une méthode de modélisation par interpolation des données. Nous avons vu que le krigeage possède de nombreux atouts. En particulier il donne intrinsèquement l'incertitude de sa prédiction ; celle-ci ne dépend pas seulement de l'incertitude sur les paramètres du modèle, comme c'est le cas avec une régression linéaire par exemple. Il permet également de prendre en compte de nombreuses informations supplémentaires, comme par exemple du bruit de mesure (changeant l'interpolation en régression), un effet pépite (un bruit blanc spatial), mais également selon des travaux plus récents (López-Lopera et al. 2018), des bornes pour les mesures, de la monotonie *a priori*, et d'autres choses encore, déjà évoquées dans le chapitre 1.

Il est également possible d'étendre le krigeage de façon à prendre en compte d'autres informations, que l'on va qualifier de « localisées » (chaque valeur est associée à une coordonnée dans l'espace d'entrée), par exemple des valeurs en certains points de la dérivée de la fonction à interpoler. La présence, effective ou potentielle, d'autres informations localisées est très fréquente dans les cas d'application du krigeage. En géologie, on imagine sans peine que des mesures peuvent être effectuées au niveau du sol en plus des mesures effectuées en forant, et qu'un même forage permet de mesurer plusieurs grandeurs. Lors de simulations physiques, il n'est pas rare que le code de calcul renvoie les valeurs de plusieurs grandeurs. Les simulations physiques un peu sophistiquées utilisent même plusieurs codes enchaînés, chacun renvoyant des grandeurs différentes, porteuses d'information sur la grandeur d'intérêt (par exemple multiphysique ou multiéchelles).

Il est donc naturel de ne pas vouloir se contenter de la grandeur d'intérêt et de ne pas prendre la mesure comme une simple « boîte noire ». On veut « ouvrir la boîte noire » et construire un *modèle introspectif* qui regarde à l'intérieur, exploitant toutes les informations pertinentes.

Le cokrigeage répond à ce besoin. En connaissant ou estimant la corrélation entre les différentes grandeurs, on construit un modèle similaire au krigeage via des processus gaussiens conditionnés (section 2.2).

Une autre approche peut être de considérer une forme de causalité entre les grandeurs et de former un modèle par composition. En intégrant l'information supplémentaire comme s'il s'agissait d'une variable d'entrée, on construit un modèle appelé « hyperkrigeage » (section 2.3).

Chaque modèle a son lot de paramètres à fixer ou à estimer. Pour l'utilisateur, il peut être difficile de choisir le modèle. La section 2.4 traite de ce choix et tente de comparer les modèles.

Dans le chapitre suivant, on ira plus loin en utilisant les modèles introspectifs de ce chapitre dans des algorithmes tels que ceux présentés dans le chapitre précédent (section 1.3), mais également en construisant de nouveaux algorithmes eux-mêmes introspectifs (c'est-à-dire ayant la connaissance des différentes grandeurs et de leurs coûts d'acquisition).

Principales contributions de ce chapitre

- Un lien entre la structure des codes de simulation et le processus gaussien multivarié associé (section 2.2.5).
- Une explicitation des variables latentes après conditionnement (section 2.2.2).
- Une formule quasi-analytique de certains paramètres des modèles de cokrigeage (section 2.2.7).
- Un modèle introspectif différent du cokrigeage, l'hyperkrigeage (section 2.3).

— Une étude comparative des différents modèles introspectifs décrits dans ce chapitre (section 2.4).

2.1 Problématique

2.1.1 Contexte opérationnel

Focalisé sur l'intérêt d'utiliser des métamodèles de krigeage pour étudier un modèle numérique coûteux, l'article fondateur de (Jones, Schonlau, and Welch 1998) reposait sur une hypothèse simplificatrice : l'absence d'informations (autres qu'entrées/sorties) relatives au modèle sous-jacent à émuler. Ce prérequis présente une contrainte très faible, et cela a de nombreux avantages tels que la versatilité applicative et la simplicité du cadre mathématique. On peut néanmoins constater que nombre d'applications opérationnelles fournissent plus d'informations exploitables que les seules entrées/sorties du modèle numérique. Des hypothèses fondamentales telles que la dérivabilité ou les conditions aux limites du domaine sont ainsi ignorées et le métamodèle mathématique construit est alors inconsistant avec la physique considérée (cf. (Maatouk 2015)). De plus, le contexte opérationnel classique d'usage des simulateurs numériques est de plus en plus fréquemment tourné vers des applications complexes de type multiphysiques, ou encore à évolution spatio-temporelle par exemple. Conventionnellement nommés « schémas numériques », ces outils de calculs sont typiquement constitués de plusieurs modèles distincts, liés séquentiellement ou parallèlement par leurs entrées-sorties. On peut par exemple évoquer :

— les schémas multiphysiques à couplage (i.e. résolutions successives des modèles physiques sur un maillage unique par exemple)
— les schémas itératifs temporels (i.e. chaque pas de temps définit la condition initiale du suivant)
— les schémas hétérogènes par intégration homogène (i.e. discrétisation spatiale des zones homogènes, ensuite agrégées pour simuler la globalité hétérogène du système)

C'est cette dernière configuration à laquelle font face les ingénieurs de l'IRSN, où les sorties d'un premier modèle numérique servent d'entrées pour un second (figure 2.1), qui a motivé cette thèse. Nous exploiterons alors plusieurs des sorties du premier modèle comme des informations valorisables pour métamodéliser la grandeur finale issue du second calcul. L'approche cartésienne consiste naturellement à exploiter cette décomposition en sous-problèmes, avec l'espoir implicite et souvent raisonnable que la métamodélisation d'une physique isolée soit plus simple et accessible que celle d'un système multiphysique complet (hypothèse épistémologique dépassant toutefois le cadre d'interrogations du présent travail).

La chaîne de calcul CRISTAL [1] (figure 2.1), utilisée pour les expertises de sûreté-criticité en France, permet d'évaluer le risque d'occurrence d'une réaction nucléaire en chaîne incontrôlée, typiquement par la simulation numérique du coefficient effectif de multiplication neutronique (grandeur physique usuellement nommée $k_{effectif}$ ou k_{eff}). Une expertise de sûreté-criticité consiste à vérifier que dans toute configuration possible (modification géométrique ou chimique des matières fissiles étudiées), ce k_{eff} reste strictement inférieur à 1,0 (ou 0,95 en incluant une marge « administrative ») ce qui garantit un contrôle de la population neutronique (i.e. un neutron génère une descendance « moyenne » [2] inférieure à un neutron). Le besoin nominal d'une telle expertise consiste donc à rechercher les configurations pénalisantes (i.e. celle dont le k_{eff} est le plus élevé) ou encore identifier les configurations dépassant le seuil de sûreté (i.e. celles dont le k_{eff} excède 0,95). Les grandeurs b_2 et k_{inf} issues du premier code APOLLO sont considérées par les experts comme fortement corrélées avec la grandeur d'intérêt k_{eff}.

FIGURE 2.1 – Schéma Apollo-Moret.

1. voir http://oecd-nea.org/tools/abstract/detail/nea-1903/ .
2. « moyenne » au sens ergodique du processus de la simulation *Markov chain Monte-Carlo* du k_{eff}

2.1.2 Modèles multifidélités

La notion de « fidélité » pour un code de calcul ou une simulation physique, désigne la précision ou la fiabilité du résultat. Un code « haute fidélité » sera très précis et le résultat sera très proche du réel, mais il sera généralement très coûteux (en temps ou en ressources informatiques). À l'inverse, un code « basse fidélité » sera moins précis, soit parce qu'on aura utilisé des simplifications concernant la physique à simuler, soit parce qu'on n'aura pas attendu la convergence complète de l'algorithme. Mais le résultat imprécis pourra être obtenu beaucoup plus rapidement. La fidélité d'un code est rarement linéaire par rapport à son coût. La figure 2.2 illustre le choix qu'il y a à faire entre appeler un code haute fidélité ou un code basse fidélité. Les modèles introspectifs tels que celui à deux maillons illustré sur la figure 2.1 sont un cas particulier de fidélité multiple où la haute fidélité est composée de l'enchaînement complet des codes, $y^{(2)}(y^{(1)}(x))$ dans l'exemple, et les basses fidélités sont des simulations incomplètes, $y^{(1)}(x)$ ici. Historiquement la fidélité renvoie à l'idée que le code produit presque la bonne valeur avec des niveaux de précision ou de biais variables. Souvent la « fidélité » prendra un sens plus large, et c'est le cas dans cette thèse. Les résultats de basse fidélité pourront être de nature différente (unité ou grandeur physique différente) des résultats de plus haute fidélité qui constituent l'objectif. Mathématiquement, les outils utilisés ici, essentiellement la corrélation linéaire, seront indépendants du lien de nature entre la haute et la basse fidélité. En particulier, un code basse fidélité peut sans problème être anticorrélée avec le code haute fidélité.

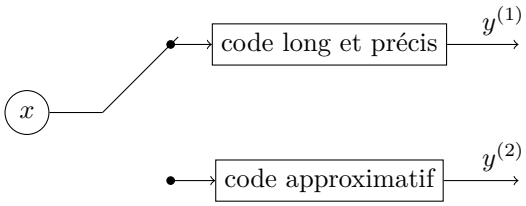


FIGURE 2.2 – Multifidélité : switching entre 2 codes.

Les modèles multifidélités visent à combiner des modèles de différentes fidélités. En exploitant les informations provenant des codes basse fidélités, on va améliorer les prédictions sur le niveau de haute fidélité. Le but étant d'obtenir une précision acceptable à moindre coût.

Le thème de la multifidélité pour les plans d'expériences est déjà riche en contributions. (Peherstorfer, Willcox, and Gunzburger 2018) et (Fernández-Godino et al. 2016) ont réalisé deux enquêtes sur les méthodes multifidélités. L'analyse de systèmes complexes tels que la propagation de l'incertitude, l'analyse de sensibilité ou l'optimisation nécessite des évaluations de modèles répétées à différents endroits de l'espace de conception, ce qui est généralement impossible avec des modèles haute fidélité, pour des raisons de coût. Les techniques multifidélités visent à accélérer ces analyses en capitalisant sur des modèles de différentes précisions. Les méthodologies multifidélités assurent la gestion des modèles, c'est-à-dire qu'elles équilibrent les niveaux de fidélité afin de réduire les coûts et d'assurer la précision des analyses. Dans l'examen de Peherstorfer *et al.*, les auteurs classent les techniques de multifidélité en trois catégories : adaptation, fusion et filtrage. La catégorie adaptation englobe les méthodes qui améliorent le modèle basse fidélité avec les résultats des modèles haute fidélité pendant le calcul. Un exemple est donné par l'approche de correction de modèle (Kennedy and O'Hagan 2000) où un processus autorégressif est utilisé pour refléter la hiérarchie entre la précision des différentes sorties. Cette technique est reprise dans (Huang et al. 2006). Les techniques de fusion visent à créer des modèles en combinant des sorties de modèle basse et haute fidélité. Deux exemples de techniques de fusion sont le cokrigeage (Myers 1982 ; Perdikaris et al. 2015) et la collocation stochastique à plusieurs niveaux (Teckentrup et al. 2015). Enfin, le filtrage consiste à appeler des modèles de basse fidélité pour décider quand utiliser des modèles de haute fidélité (par exemple, échantillonnage sur plusieurs niveaux). Le filtrage est un ingrédient fondamental des techniques d'optimisation avec deux niveaux de fidélité, techniques plus connues sous le nom d'algorithmes d'optimisation avec *surrogate* ou surface de réponse (Jones 2001 ; A. I. Forrester and Keane 2009 ; A. I. J. Forrester, Sóbester, and Keane 2007). Le *surrogate* est le modèle basse fidélité. Bien que ce soit le sujet du chapitre 3, nous citerons deux exemples. En optimisation évolutionnaire avec *surrogates* (Loshchilov 2013), différentes stratégies sont décrites pour filtrer des points candidats à l'évaluation haute fidélité au moyen du modèle basse fidélité (le *surrogate*). Dans le cas particulier ou le contrôle de la fidélité du

modèle est celui d'une convergence de solveur Monte-Carlo, (Picheny et al. 2013) ont développé une version modifiée de l'algorithme d'optimisation EGO tirant parti de la possibilité de définir la précision du résultat de la simulation (l'incertitude du résultat dépend du temps laissé au solveur pour converger).

2.1.3 Optimisations de conception multidisciplinaire

L'optimisation de conception multidisciplinaire (MDO pour *Multidisciplinary Design Optimisation*) est un domaine d'ingénierie qui utilise des méthodes d'optimisation afin de résoudre des problèmes de conception mettant en œuvre plusieurs disciplines physiques. Par exemple, un projet de conception de véhicule aérien nécessite de faire simultanément appel à l'aérodynamique, la mécanique des structures, la propulsion et l'économie. La MDO permet aux concepteurs d'incorporer les effets de chacune des disciplines en même temps. L'optimum global ainsi trouvé est meilleur que la configuration trouvée en optimisant chaque discipline indépendamment des autres (sauf dans le cas particulier où le modèle intégré est additif selon toutes les sous-disciplines), car l'on prend en compte les interactions entre les disciplines. Cependant cela entraîne un surcoût au niveau du temps calcul et de la complexité du problème. Le problème est d'autant plus compliqué qu'il fait souvent appel à plusieurs solveurs conçus indépendamment (voir figure 2.3). Ces techniques sont utilisées dans plusieurs domaines d'application, dont la conception automobile, navale, électronique et informatique, pour n'en citer que quelques-uns.

Les modèles multifidélités sont une manière de résoudre les problèmes relevant de la MDO. Le sujet est traité dans (Garland et al. 2020).

(Sellar, Batill, and Renaud 1996) ont présenté un CSSO (*Concurent SubSpace Optimization*) basé sur la surface de réponse pour réduire les coûts de calcul, tandis que (Simpson et al. 2001) ont exploré la combinaison du krigeage et du *multidisciplinary feasible* (MDF). (Sobieski and Kroo 2000) décrivent une formulation d'optimisation collaborative qui utilise la méthode de la surface de réponse (RSM). Paiva *et al.* ont comparé les surfaces de réponse polynomiales, les processus gaussiens et les réseaux neuronaux pour un processus MDF (Paiva et al. 2010). Même si ces études MDO remplacent les modèles haute fidélités par des modèles de substitution, elles ne gèrent pas vraiment la fidélité des modèles. Seul un nombre limité d'études, citées ci-après, porte sur l'adaptation de la multifidélité à des problèmes multidisciplinaires.

(Allaire, Willcox, and Toupet 2010) ont proposé une approche bayésienne de la MDO multifidélité. La méthode met l'accent sur la quantification probabiliste de l'inadéquation du modèle, qui est liée à la fidélité du modèle. La fidélité du modèle est gérée en utilisant la conviction qu'un modèle est vrai, étant donné que le véritable modèle se trouve dans l'ensemble des modèles considérés. L'approche consiste à résoudre un problème déterministe MDO avec un niveau de modélisation fixe pour les différentes disciplines. Ensuite, sur la base de l'optimum estimé, une évaluation des variances de performance et de contrainte est effectuée. Si les écarts sont trop importants, un deuxième problème est résolu pour identifier la discipline qui influe le plus sur les incertitudes de la modélisation (en utilisant les mesures de Sobol). Le niveau de fidélité de ces disciplines est augmenté et l'approche est répétée. Allaire *et al.* soulignent que cette approche ne garantit pas la détermination d'un optimum global, mais facilite la gestion de plusieurs disciplines et la modélisation de niveaux dans un seul problème. Cette approche a été étendue par (Christensen 2012) et (Kordonowy et al. 2012) pour prendre en compte de manière appropriée le couplage interdisciplinaire dans une approche bayésienne de la MDO multifidélité. Le processus proposé casse la boucle de couplage en une série d'évaluations séquentielles unidisciplinaires qui se nourrissent des évaluations précédentes.
La méthode est une première tentative de prise en compte des sous-systèmes couplés et de la multifidélité, et facilite l'estimation des objectifs et des contraintes. Mais elle ne fournit qu'une approximation des incertitudes de couplage car le problème de rétroaction est décomposé, ce qui empêche de pouvoir garantir la consistance du résultat.

(Zadeh and Toropov 2002) ont décrit une formulation d'optimisation collaborative qui résout les problèmes de MDO avec des modèles haute et basse fidélités. L'optimisation au niveau de chaque discipline se fait en utilisant un modèle de simulation basse fidélité corrigé, qui combine les simulations haute et basse fidélités dans la construction du modèle (basé sur le plan d'expériences et des approximations mathématiques). L'agrégation mathématique des simulations est réalisée avec des polynômes, et des moindres carrés pour la détermination des hyperparamètres polynomiaux. Cependant, la construction des modèles multifidélités utilisés dans l'optimisation collaborative se fait à l'avance, sans aucune mise à jour du modèle au cours

de l'algorithme d'optimisation.

(March and Willcox 2012) ont proposé deux méthodes pour paralléliser une MDO. La première stratégie décompose le processus MDO en plusieurs optimisations de sous-système résolues en parallèle. La deuxième technique définit un ensemble de plans d'expériences à évaluer avec des simulations coûteuses en calcul, exécute ces évaluations en parallèle, puis résout un problème de MDO basé sur un métamodèle se substituant aux simulations coûteuses. Les deux méthodes sont des exemples d'optimisation multifidélité pour la MDO.

(Wang et al. 2018) ont développé un cadre MDO multifidélité avec un mécanisme de déplacement entre les différents niveaux de fidélité. Tout d'abord, une formulation initiale de la MDO et un niveau de fidélité sont sélectionnés pour lancer le processus de recherche. Au cours des itérations MDO, si le critère de déplacement est rempli, le processus d'optimisation s'arrête, incrémente la fidélité du modèle et met à jour l'architecture MDO (en modifiant si nécessaire la formulation de la MDO). Les auteurs utilisent le modèle de déplacement adaptatif (*adaptive model switching*, AMS) (Mehmani et al. 2015). Ce critère estime si l'incertitude associée au niveau actuel de fidélité de la sortie du modèle domine la dernière amélioration de la fonction-objectif normalisée.

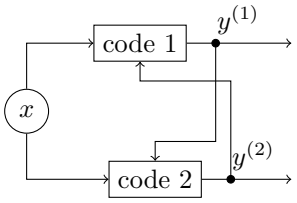


FIGURE 2.3 – Deux physiques couplées.

2.1.4 Variables qualitatives

Les récents travaux sur des modèles de krigeage prenant en compte des variables qualitatives utilisent une méthode similaire (Roustant et al. 2018). Il apparaît une grande similarité conceptuelle entre l'existence de différentes grandeurs et l'existence de différentes modalités. Les variables qualitatives vont déterminer le *type* de résultat, comme s'il était produit par différents codes de calcul (figure 2.4).

Bien qu'il n'y ait qu'un seul type de grandeur, la difficulté pour construire un modèle de krigeage se retrouve dans le choix (ou la construction) de la fonction de covariance, laquelle doit être correctement définie entre toutes les modalités des variables qualitatives. Les possibilités de fonctions de covariance sont les mêmes que celles que l'on va décrire plus bas et il y a une équivalence entre le krigeage avec des variables qualitatives et le cokrigeage. De la même manière que les grandeurs mesurées peuvent être hiérarchisées ou non, les modalités d'une variables qualitatives peuvent être ordonnées ou non. Il convient d'utiliser un modèle adapté selon le cas.

Une des principales particularités réside dans la moyenne *a priori* lorsque celle-ci est paramétrée, puisque dans le cas du krigeage avec variables qualitatives on aura bien souvent des paramètres et des tendances communes à toutes les modalités (ce qu'on n'a pas nécessairement avec des grandeurs de natures différentes).

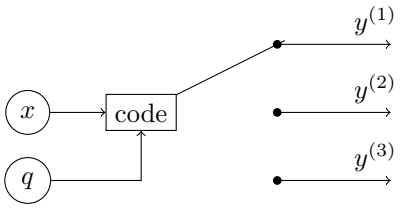


FIGURE 2.4 – Switching entre 3 réponses en fonction de la valeur de la variable qualitative q.

2.1.5 Formalisation

Ces différentes approches peuvent être regroupées sous une même représentation formelle. À chaque fois, on a un ou plusieurs codes pouvant calculer différentes grandeurs (figure 2.5). Même s'il peut arriver qu'un ingénieur se retrouve face à un problème dans lequel ce n'est pas différents codes de calcul qui entrent en jeu pour calculer les grandeurs (ça peut être un seul code qui donne plusieurs grandeurs, ou des résultats expérimentaux), on se référera dans la suite à des « codes » pour désigner les générateurs d'observations.

La façon dont sont reliés les codes va influencer l'utilisateur dans le choix du modèle introspectif à mettre en place. D'un point de vue mathématique on pourra tout de même écrire les formules sous la même forme.

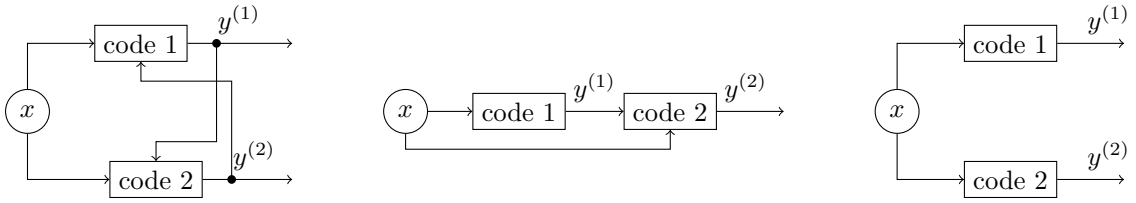


FIGURE 2.5 – Possibilités de couplage pour 2 codes.

On va noter N le nombre total d'observations. Mais chaque observation ne correspond pas systématiquement à la même grandeur. n_g grandeurs sont mesurées, et pour chaque grandeur on dispose de N_a observations ($a \in [\![1, n_g]\!]$, $\sum N_a = N$). La $j^{ième}$ mesure de la grandeur a sera notée $y_j^{(a)}$ et on note les coordonnées correspondantes $x_j^{(a)}$. La valeur d'une fonction f dépendant à la fois des coordonnées x et de la grandeur a pourra être notée $f(x^{(a)})$ ou $f^{(a)}(x)$, le terme $x^{(a)}$ désignant les coordonnées x associées à la grandeur a. Le processus gaussien marginal associé à la grandeur a sera noté $Z^{(a)}(\cdot)$ avant le conditionnement et $\tilde{Z}^{(a)}(\cdot)$ après. Le processus gaussien multivarié sera quant à lui noté en gras $\boldsymbol{Z}$. Il en sera de même pour les objets résultant de la concaténation sur toutes les grandeurs. L'ensemble des mesures de la grandeur a sera noté $Y^{(a)}$ et leurs coordonnées $X^{(a)}$. $y(x)$ désigne une mesure effectuée aux coordonnées x. $\boldsymbol{Y}$ et $\boldsymbol{X}$ désignent respectivement l'ensemble des mesures sur toutes les grandeurs et les coordonnées associées à chacune ($\boldsymbol{Y} = y(\boldsymbol{X})$). En accord avec ce qu'on vient de préciser, $\boldsymbol{X}_i$ contient l'information sur le type de grandeur mesurée lors de la i-ème observation. On note $\boldsymbol{G}$ le vecteur qui, à chaque élément de $\boldsymbol{X}$, associe sa grandeur correspondante, et $\underline{\boldsymbol{X}}$ les éléments privées de leur grandeur : $\boldsymbol{X} = \underline{\boldsymbol{X}} \cup \boldsymbol{G}$ (ces notations seront utilisées dans le chapitre suivant).

On utilise également $I_{\{a\}}$ pour la matrice (de taille $N \times N$) indicatrice de la grandeur a, une matrice diagonale n'ayant que des 1 ou des 0 selon que la cellule correspond bien à la grandeur a ou non.

Dans une vision plus mathématique, on pourra être amené à parler de « groupes » à la place de « grandeurs ». On parle également de « niveaux » pour les groupes lorsque ceux-ci peuvent être ordonnés. C'est notamment le cas pour les problèmes de multifidélité ou certaines variables qualitatives. Par abus de langage, le terme « niveaux » peut parfois être employé même sans ordonnancement naturel des grandeurs. D'un point de vue algorithmique, on numérote souvent les groupes. Le niveau de plus haute fidélité, la grandeur principale, sera appelé *niveau d'intérêt* ou *grandeur d'intérêt*. Dans le reste de ce document, la grandeur d'intérêt portera le numéro 1. Ainsi les mesures de la grandeur d'intérêt seront toujours les $Y^{(1)}$, quel que soit le nombre de grandeurs considérées. Notons que dans la littérature c'est souvent le choix inverse qui est fait : la grandeur 1 correspondant à la plus basse fidélité et la grandeur n_g à la plus haute.

2.2 Processus gaussien multivarié

Le krigeage se base sur le conditionnement d'un processus gaussien. Avec des informations supplémentaires *localisées* (des valeurs associées à des x de l'espace des entrées, par exemple des valeurs de dérivée ou des

grandeurs intermédiaires calculées par les codes informatiques), on peut étendre ce conditionnement à un processus gaussien multivarié pour intégrer ces informations et construire un modèle dit de « cokrigeage ».

Un processus gaussien multivarié se construit de la même manière qu'un processus gaussien univarié. Tout ce dont on a besoin pour le définir entièrement est une fonction pour la moyenne et une fonction de covariance. La seule subtilité est que les fonctions de moyenne et de covariance (*a priori* **et** *a posteriori*) ne dépendent pas seulement de la valeur des entrées x mais également de la grandeur (ou sortie) considérée (des deux grandeurs pour la fonction de covariance). Plus encore que pour le krigeage présenté en Partie I, le choix de la fonction de covariance est crucial. Il passe bien souvent par le choix d'un ou plusieurs *kernels* tels que ceux utilisés pour le krigeage, et par le choix de la manière de combiner ces *kernels* selon les groupes de sortie. Deux manières de les combiner existent : soit on définit une relation entre les groupes, notamment via des variables cachées ou latentes (voir section 2.2.2), et on en déduit la fonction de covariance correspondante, soit on construit directement une fonction de covariance *ad hoc* qui sera alors définie positive par construction (voir section 2.2.6).

Pour la moyenne *a priori* du processus, il convient d'avoir une fonction pour chaque groupe. Une fois la moyenne et la fonction de covariance *a priori* établies, les formules du conditionnement de processus gaussien peuvent s'appliquer, et les équations du cokrigeage sont ainsi similaires à celles du krigeage :

$$\mu^{(a)}(x) = \mu(x^{(a)}) = m(x^{(a)}) + k(x^{(a)}, \boldsymbol{X})\boldsymbol{K}^{-1}(\boldsymbol{Y} - m(\boldsymbol{X})) \ , \tag{2.1}$$

$$\mathrm{Cov}(\tilde{Z}^{(a)}(x), \tilde{Z}^{(b)}(x')) = C(x^{(a)}, x'^{(b)}) = k(x^{(a)}, x'^{(b)}) - k(x^{(a)}, \boldsymbol{X})\boldsymbol{K}^{-1}k(\boldsymbol{X}, x'^{(b)}) \ , \tag{2.2}$$

$$\sigma^{2^{(a)}}(x) = \sigma^2(x^{(a)}) = C(x^{(a)}, x^{(a)}) = k(x^{(a)}, x^{(a)}) - k(x^{(a)}, \boldsymbol{X})\boldsymbol{K}^{-1}k(\boldsymbol{X}, x^{(a)}) \ , \tag{2.3}$$

où $k(\cdot, \cdot)$ est la fonction de covariance et $\boldsymbol{K}$ la matrice des $k(\boldsymbol{X}, \boldsymbol{X})$.

La problématique de l'estimation des paramètres de krigeage a été évoquée dans le chapitre précédent. Dans le cas du cokrigeage, la fonction de covariance est généralement plus riche, plus complexe et possède ainsi davantage de paramètres. La problématique de l'estimation des paramètres se pose ainsi à nouveau. Un résultat développé au cours de cette thèse permet de réduire le nombre de paramètres à estimer. Ce résultat est détaillé dans la section 2.2.7.

2.2.1 Exploitation des valeurs de la dérivée

Si on peut supposer une relation entre les différentes grandeurs de sortie, on peut aisément en déduire une fonction de covariance cohérente. En effet lorsqu'il existe une relation linéaire connue entre les grandeurs, on peut choisir une fonction *kernel* pour l'une d'entre elles, et les fonctions de moyenne et de covariance *a priori* des autres peuvent alors être déduites par linéarité. C'est le cas par exemple lorsqu'on dispose de valeurs de dérivée en plus de la grandeur d'intérêt.

L'exploitation des informations sur la dérivée est un cas simple et probant de l'efficacité du cokrigeage. Ce type particulier de cokrigeage est parfois appelé « krigeage amélioré par gradient » (*Gradient-Enhanced Kriging*, voir Morris, Mitchell, and Ylvisaker (1993)). En effet, on est capable d'écrire explicitement la covariance entre un processus gaussien et sa dérivée. La dérivée d'un processus gaussien est encore un processus gaussien, grâce à la linéarité de l'opérateur de dérivation. Pour pouvoir considérer la dérivée d'un processus gaussien, il faut que le noyau de covariance soit deux fois dérivable (notamment en 0).

Par exemple en $d = 1$ dimension, la covariance entre la grandeur d'intérêt $Z^{(1)}$ en un point x et sa dérivée $Z^{(2)}$ en un point x' sera donnée par :

$$\begin{aligned} \mathrm{Cov}(Z^{(1)}(x), Z^{(2)}(x')) &= \mathrm{Cov}\left(Z^{(1)}(x), \tfrac{dZ^{(1)}(x')}{dx'}\right) \\ &= \tfrac{\partial}{\partial x'}\mathrm{Cov}(Z^{(1)}(x), Z^{(1)}(x')) \\ &= \tfrac{\partial}{\partial x'}\left(k(x - x')\right) \\ &= -\tfrac{dk(h)}{dh} \ , \end{aligned} \tag{2.4}$$

où $h = x - x'$ et $k(h)$ est le noyau de covariance de la grandeur d'intérêt ($k(x - x') = \mathrm{Cov}\left(Z^{(1)}(x), Z^{(1)}(x')\right)$). Puisque cette fonction noyau est paire ($k(h) = k(-h)$), sa dérivée est impaire et on a bien :

$$\begin{aligned}\mathrm{Cov}(Z^{(2)}(x'), Z^{(1)}(x)) &= \mathrm{Cov}\left(\frac{dZ^{(1)}(x')}{dx'}, Z^{(1)}(x)\right) \\ &= \frac{\partial}{\partial x'}\mathrm{Cov}(Z^{(1)}(x'), Z^{(1)}(x)) \\ &= \frac{\partial}{\partial x'}(k(x'-x)) \\ &= \frac{dk(-h)}{dh} = -\frac{dk(h)}{dh}\ .\end{aligned} \tag{2.5}$$

La covariance entre deux valeurs de dérivée vaut alors :

$$\begin{aligned}\mathrm{Cov}(Z^{(2)}(x), Z^{(2)}(x')) &= \mathrm{Cov}\left(\frac{dZ^{(1)}(x)}{dx}, \frac{dZ^{(1)}(x')}{dx'}\right) \\ &= \frac{\partial}{\partial x}\frac{\partial}{\partial x'}\mathrm{Cov}(Z^{(1)}(x), Z^{(1)}(x')) \\ &= -\frac{d^2k(h)}{dh^2}\ .\end{aligned} \tag{2.6}$$

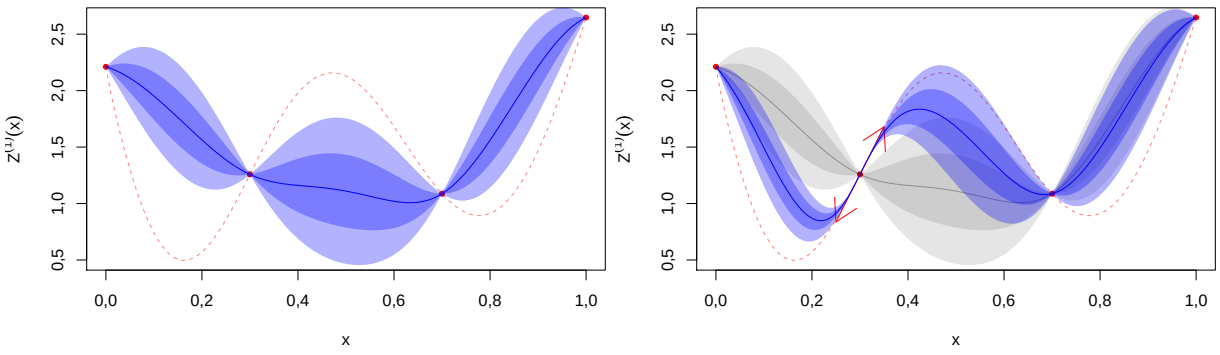
Si le noyau de covariance est deux fois dérivable, la fonction de covariance sera ainsi parfaitement bien définie et respectera par construction toutes les contraintes qu'une fonction de covariance doit respecter (en particulier la positivité).

Des contraintes similaires existent quand on considère des dérivées d'ordre supérieur : pour pouvoir considérer la n-ième dérivée, il faut que le noyau soit $n+1$ fois dérivable (Laurent et al. 2017 ; Rasmussen and Williams 2006).

Lorsque nous avons les expressions des covariances entre la grandeur d'intérêt et sa dérivée et entre deux valeurs de la dérivée, nous pouvons appliquer les formules connues de conditionnement du processus gaussien.

Dans le cas multidimensionnel, chaque dimension donnera lieu à une grandeur différente pour représenter la dérivée partielle dans sa direction. Le raisonnement est similaire et on parvient à construire la fonction de covariance sans difficulté.

La figure 2.6 présente un exemple 1D d'un krigeage auquel on ajoute une information sur la dérivée.



(a) Krigeage classique

(b) Krigeage avec une information sur la dérivée

FIGURE 2.6 – Exemple 1D d'amélioration d'un modèle de krigeage à l'aide d'une information sur la dérivée. La ligne pointillée rouge correspond à la vraie fonction qu'on cherche à modéliser. Les points rouges sont les points connus. En bleu sont représentés la moyenne de prédiction et les intervalles de confiance à 1 et 2 sigmas. En gris est représenté le modèle de krigeage (intervalle de confance à 1 et 2 sigmas) classique.

2.2.2 Modèles linéaires de corégionalisation (LMC) : généralités

À présent, on s'intéresse au cas où on ne connaît pas exactement la relation entre les grandeurs. Les différentes grandeurs vont être modélisées par des processus gaussiens non indépendants. L'idée du modèle linéaire de corégionalisation (LMC pour *Linear Model of Coregionalization*) est de décomposer le processus gaussien en *variables latentes* indépendantes, linéairement combinées. Une description du

LMC peut être trouvée dans (Fricker, Oakley, and Urban 2013). Un cas particulier de LMC sert à une modélisation multifidélité dans (Huang et al. 2006) (avec Q une matrice triangulaire de 1 et autant de variables latentes que de niveaux de fidélité, cf. explications ci-après et sous-section 2.2.4). On va ainsi supposer que les grandeurs possèdent des parties communes : mathématiquement on va écrire que les n_g groupes sont des combinaisons linéaires de n_l variables latentes indépendantes, ou plus exactement que nos n_g processus gaussiens *dépendants* peuvent s'écrire comme des combinaisons linéaires de n_l processus *indépendants* :

$$\boldsymbol{Z}(x) = \begin{pmatrix} Z^{(1)}(x) \\ Z^{(2)}(x) \\ \dots \\ Z^{(n_g)}(x) \end{pmatrix} = \begin{pmatrix} m^{(1)}(x) \\ m^{(2)}(x) \\ \dots \\ m^{(n_g)}(x) \end{pmatrix} + Q \begin{pmatrix} U_1(x) \\ U_2(x) \\ \dots \\ U_{n_l}(x) \end{pmatrix} = \boldsymbol{m}(x) + Q\boldsymbol{U}(x) \ . \tag{2.7}$$

Les $U_b(\cdot)$ sont les « variables latentes ». Ce sont des processus gaussiens stationnaires, non observés, indépendants, centrés et de variance unitaire. Q est la matrice $n_g \times n_l$ des coefficients des combinaisons linéaires.

Par la suite, nous « oublierons » $\boldsymbol{m}(\cdot)$ qui ne constitue qu'une simple translation du processus $\boldsymbol{Z}(\cdot)$. Nous reviendrons dessus lorsque nous aborderons la question de l'estimation des paramètres, puisque, comme dans le cas du krigeage, $\boldsymbol{m}(\cdot)$ est une fonction paramétrée lorsqu'elle n'est pas nulle. De même, nous allons simplifier l'écriture en considérant des facteurs d'échelle σ_a propres à chaque groupe. On écrira alors $Z^{(a)}_{/\sigma} = \frac{Z^{(a)}(x)}{\sigma_a}$ et

$$\boldsymbol{Z}_{/\sigma}(x) = \begin{pmatrix} Z^{(1)}_{/\sigma} \\ Z^{(2)}_{/\sigma} \\ \dots \\ Z^{(n_g)}_{/\sigma} \end{pmatrix} = \begin{pmatrix} \frac{Z^{(1)}(x)}{\sigma_1} \\ \frac{Z^{(2)}(x)}{\sigma_2} \\ \dots \\ \frac{Z^{(n_g)}(x)}{\sigma_{n_g}} \end{pmatrix} \ . \tag{2.8}$$

De même, on définira le vecteur des observations réduites $\boldsymbol{Y}_{/\sigma}$ comme étant le vecteur des observations $\boldsymbol{Y}$ où chaque élément aura été divisé par le σ_a correspondant à sa grandeur. En appelant $D_\sigma^{(n_g)}$ la matrice purement diagonale de taille $n_g \times n_g$ (l'exposant sera négligé lorsqu'il n'y aura pas d'ambiguïté sur sa dimension) contenant les σ_a, nous allons définir la matrice P telle que :

$$Q = D_\sigma P \ . \tag{2.9}$$

Les coefficients de P seront notés $\rho_{a,b}$ (la lettre p sera gardée pour les paramètres de la tendance $\boldsymbol{m}(\cdot)$, cf. section 2.2.7). Nous pouvons alors écrire que :

$$\boldsymbol{Z}_{/\sigma}(x) = D_\sigma^{-1}\boldsymbol{Z} = P\ \boldsymbol{U}(x) \ . \tag{2.10}$$

Comme nous le verrons par la suite (section 2.2.7), les facteurs σ_a pourront être estimés indépendamment des autres paramètres du modèle. Il ne s'agit pas exactement d'un écart type et $\frac{Z^{(a)}(\cdot)}{\sigma_a}$ n'est pas nécessairement de variance unitaire (cf. équation (2.18)).

Si la covariance spatiale, entre différents x de l'espace d'entrée, va être portée par les $U_b(\cdot)$ et leurs noyaux de covariance, notés $r_b(\cdot,\cdot)$ (avec $r_b(x,x) = 1$ par définition), la matrice Q va déterminer la covariance intergroupes entre les processus $Z^{(a)}(\cdot),\ a = 1, \dots, n_g$. On définit ainsi la matrice de covariance intergroupes :

$$\boldsymbol{\Sigma}_2 = \mathrm{Cov}(\boldsymbol{Z}(x), \boldsymbol{Z}(x)) = Q\,\mathrm{Cov}(\boldsymbol{U}(x), \boldsymbol{U}(x))\ {}^tQ = Q\ {}^tQ \ . \tag{2.11}$$

Cette matrice, de taille $n_g \times n_g$, ne prend pas en compte les covariances spatiales et ne doit pas être confondue avec la matrice de covariance $\boldsymbol{K}$ (de taille $N \times N$ avec $N = \sum_a^{n_g} N_a$) :

$$\boldsymbol{K} = \mathrm{Cov}\left(\begin{pmatrix} Z(X^{(1)}) \\ Z(X^{(2)}) \\ \dots \\ Z(X^{(n_g)}) \end{pmatrix}, \begin{pmatrix} Z(X^{(1)}) \\ Z(X^{(2)}) \\ \dots \\ Z(X^{(n_g)}) \end{pmatrix}\right) = \begin{bmatrix} \mathrm{Cov}(Z(X^{(1)}), Z(X^{(1)}) & \dots & \mathrm{Cov}(Z(X^{(1)}), Z(X^{(n_g)})) \\ \vdots & \ddots & \vdots \\ \mathrm{Cov}(Z(X^{(n_g)}), Z(X^{(n_g)})) & \dots & \mathrm{Cov}(Z(X^{(n_g)}), Z(X^{(n_g)})) \end{bmatrix} , \tag{2.12}$$

où chaque $\mathrm{Cov}(Z(X^{(a)}), Z(X^{(b)}))$ est la sous-matrice $N_a \times N_b$ de terme général $\mathrm{Cov}(Z(X_k^{(a)}), Z(X_l^{(b)}))$ (pour la ligne k, colonne l), $X^{(a)}$ et $X^{(b)}$ étant les coordonnées des observations effectuées pour les groupes a et b respectivement.

La fonction de covariance du processus multivarié va pouvoir s'écrire dans le cas général :

$$k(x^{(a)}, x'^{(b)}) = \{\boldsymbol{k}(x, x')\}_{a,b} \ , \tag{2.13}$$

$$\begin{array}{rcl} \boldsymbol{k}(x, x') & = & \mathrm{Cov}(Z(x), Z(x')) \\ & = & Q \ \mathrm{Cov}(U(x), U(x')) \ {}^tQ \\ & = & Q \ \mathrm{diag}(r_b(x, x')) \ {}^tQ \\ & = & \sum_{b=1}^{n_l} r_b(x, x') Q_{\cdot,b} \ {}^tQ_{\cdot,b} \\ & = & \sum_{b=1}^{n_l} r_b(x, x') \boldsymbol{V}_b \ . \end{array} \tag{2.14}$$

Les matrices $\boldsymbol{V}_b = Q_{\cdot,b} \ {}^tQ_{\cdot,b}$, de taille $n_g \times n_g$, sont appelées « matrices de corégionalisation », et donnent le nom de ces modèles. On voit dans l'équation (2.14) que le rang de $\boldsymbol{k}(x, x')$ est au maximum $\min(n_l, n_g)$. On a l'identité $\sum_{b=1}^{n_l} \boldsymbol{V}_b = \boldsymbol{\Sigma_2}$. Ainsi la matrice de covariance du processus multivarié en un point donné correspond à la matrice de covariance intergroupes ($\boldsymbol{k}(x, x) = \boldsymbol{\Sigma_2}$). Celle-ci doit donc être définie positive. Pour des raisons numériques, une positivité stricte est préférable sans quoi il faut faire appel à des méthodes de pseudo-inverse (Mohammadi et al. 2016). Le reste de ce document va supposer $\boldsymbol{\Sigma_2}$ inversible.

Nous allons même aller plus loin en imposant $n_l = n_g$. En théorie, le modèle permet un nombre quelconque de variables latentes. En pratique, l'usage d'une matrice Q carrée est presque toujours de mise. Il y a différentes raisons à cela. Fixer $n_l < n_g$, i.e. mettre moins de variables latentes que de grandeurs observées, nécessite de faire l'hypothèse qu'il y a de la redondance entre les grandeurs, que certaines n'apportent pas d'information. C'est une hypothèse forte qui ne peut pas être posée dans le cas général. De plus, comme expliqué dans le paragraphe précédent, la matrice de covariance intergroupes $\boldsymbol{\Sigma_2}$ ne serait plus inversible (son rang serait inférieur à n_g) et cela impose d'utiliser des méthodes de pseudo-inverse pour calculer $\boldsymbol{K}^{-1}$, ce qui n'est pas notre cadre ici. À l'inverse, fixer $n_l > n_g$, mettre plus de variables latentes que de grandeurs, pose des problèmes d'identifiabilité. En effet, en l'absence d'information, les $U_b(\cdot)$ auront des *kernels* similaires qui ne différeront que par leurs paramètres. Des jeux de paramètres différents conduiront ainsi à des modèles identiques. Cette problématique est déjà présente lorsque $n_l = n_g$ et justifie que l'on rajoute des contraintes sur Q comme nous le verrons par la suite. Le même problème apparaît clairement lorsque qu'on veut utiliser dans un modèle de krigeage un noyau qui est la somme de deux noyaux : il faut que les deux noyaux soient distinguables sans quoi deux jeux de paramètres différents pourrait conduire exactement au même modèle. Dans la pratique, ce problème n'empêchera pas le modèle de fonctionner correctement mais il soulève une autre question, qui est celle de l'interprétabilité des variables latentes. Cette question peut être considérée comme cruciale pour des ingénieurs, des physiciens ou des décideurs. Dans la suite de ce chapitre, la matrice Q sera supposée carrée et le nombre de variables latentes égal au nombre de grandeurs ($n_l = n_g$).

La factorisation de la matrice de covariance intergroupes $\boldsymbol{\Sigma}_2$ donnée par l'équation (2.11) n'est pas unique (il y a par exemple la décomposition de Cholesky ou la décomposition par les vecteurs et valeurs propres). Chaque factorisation conduit à une matrice Q différente, donc à des matrices de corégionalisation $\boldsymbol{V}_i$ différentes, et donc à des noyaux $k(\cdot, \cdot)$ différents (cf. équation (2.14)). Comme nous allons le voir, changer la matrice Q conduit à changer considérablement le modèle même si $\boldsymbol{\Sigma}_2$ reste identique. L'utilisateur a besoin d'une méthode pour choisir la matrice Q, et donc la matrice P qui en découle. L'idée que nous développons ici est de définir la forme de P à partir de la manière dont les grandeurs sont organisées, c'est-à-dire, d'un point de vue ingénieur, à partir de l'architecture des codes de calcul, des connaissances ou des hypothèses. Une fois la forme de la matrice P définie, il conviendra d'en ajuster les paramètres en même temps que les autres paramètres du modèle.

En regardant comment les codes sont imbriqués, ou comment les grandeurs sont liées, on peut arriver à des relations de la forme :

$$Z_{/\sigma}^{(a)}(\cdot) = \sum_{b \neq a} A_{a,b} Z_{/\sigma}^{(b)}(\cdot) + \alpha_a U_a(\cdot) \ . \tag{2.15}$$

Le coefficient $A_{a,b}$ est un scalaire représentant le lien entre les grandeurs a et b (leur corrélation s'il n'y a que ces deux groupes). Cette équation traduit le fait que la grandeur i peut être vue comme une

combinaison linéaire des autres grandeurs, plus une perturbation qui lui est propre. Avec cette vision des choses, le nombre de variables latentes correspond très exactement au nombre de grandeurs ($n_l = n_g$).

Un ingénieur peut bien sûr rencontrer des situations où les grandeurs sont liées de façon non linéaire, mais il pourra presque toujours considérer une transformation de l'une ou l'autre des grandeurs pour aboutir à une relation quasiment linéaire, et la relation entre la grandeur et sa transformée pourra être étudiée directement (comme dans le cas de la dérivée).

En notant D_α la matrice diagonale contenant les α_a, et A la matrice $n_g \times n_g$ de terme général $A_{a,b}$, en ayant fixé $A_{a,a} = 0$, on peut faire apparaître le lien entre A et P. En effet l'équation (2.15) peut alors s'écrire sous forme matricielle, et en utilisant l'égalité (2.10), il vient :

$$\begin{array}{rcl} \boldsymbol{Z}_{/\sigma}(\cdot) & = & A\ \boldsymbol{Z}_{/\sigma}(\cdot) + D_\alpha \boldsymbol{U}(\cdot) \\ P\ \boldsymbol{U}(\cdot) & = & AP\ \boldsymbol{U}(\cdot) + D_\alpha \boldsymbol{U}(\cdot)\ . \end{array} \tag{2.16}$$

En résolvant cette équation pour $\boldsymbol{U}(\cdot)$ on trouve :

$$(I - A)P = D_\alpha\ . \tag{2.17}$$

Avec les bonnes contraintes, nous ferons en sorte que P et $(I - A)$ soient toutes les deux inversibles. La non-inversibilité de P se répercuterait sur $\boldsymbol{\Sigma_2}$ à travers Q (cf. équation (2.9)) et poserait des difficultés numériques.

La fonction de covariance va alors pouvoir s'écrire de façon détaillée :

$$\begin{array}{rcl} k(x^{(a)}, x'^{(b)}) & = & \{\boldsymbol{k}(x,x')\}_{a,b} = \mathrm{Cov}(Z^{(a)}(x), Z^{(b)}(x')) \\ & = & \mathrm{Cov}(\sigma_a P_{a,\cdot}\boldsymbol{U}(x), \sigma_b P_{b,\cdot}\boldsymbol{U}(x')) = \sigma_a\sigma_b P_{a,\cdot}\,\mathrm{Cov}(\boldsymbol{U}(x), \boldsymbol{U}(x'))\ {}^t P_{b,\cdot} \\ & = & \sigma_a\sigma_b P_{a,\cdot}\ \mathrm{diag}(r_\cdot(x,x'))\ \ {}^t P_{b,\cdot} \\ & = & \sigma_a\sigma_b \sum_{l=1}^{n_g} \rho_{a,l}\rho_{b,l} r_l(x,x')\ . \end{array} \tag{2.18}$$

Observation des variables latentes

Il pourra intéresser certains utilisateurs de savoir à quoi ressemblent les variables latentes. Plus exactement il est possible de caractériser un processus gaussien $\tilde{U}_b(\cdot)$ correspondant au processus $U_b(\cdot)$ conditionné par les observations $\boldsymbol{Y}$. La covariance entre $U_b(x)$ et $Z_a(x')$ vaut en effet $Q_{ab} r_b(x,x')$, et on peut définir le vecteur :

$$k_{UZ}(x^{(a)}, \boldsymbol{X}) = \mathrm{Cov}(U_a(x), Z(\boldsymbol{X}))\ . \tag{2.19}$$

En appliquant les formules de conditionnement, il vient :

$$\mu_{\tilde{U}_a}(x) = k_{UZ}(x^{(a)}, \boldsymbol{X})\boldsymbol{K}^{-1}(\boldsymbol{Y} - m(\boldsymbol{X}))\ , \tag{2.20}$$

$$\mathrm{Cov}(\tilde{U}_a(x), \tilde{U}_b(x')) = r_a(x,x')\delta_{ab} - k_{UZ}(x^{(a)}, \boldsymbol{X})\boldsymbol{K}^{-1}\ {}^t k_{UZ}(x'^{(b)}, \boldsymbol{X})\ , \tag{2.21}$$

$$\sigma^2_{\tilde{U}_a}(x) = 1 - k_{UZ}(x^{(a)}, \boldsymbol{X})\boldsymbol{K}^{-1}\ {}^t k_{UZ}(x^{(a)}, \boldsymbol{X})\ , \tag{2.22}$$

où δ_{ab} est le symbole de Kronecker (valant 1 si et seulement si $a = b$). Ces formules permettent de tracer des représentations des variables latentes. L'interprétation des variables latentes *a posteriori* dépend très fortement du problème d'ingénierie considéré.

La question qui se pose à présent est de savoir comment fixer les $\rho_{a,b}$. Des choix de paramétrisation différents vont conduire à des modèles LMC différents. Nous allons présenter dans ce qui suit deux LMC basiques, puis nous verrons comment on peut les combiner pour avoir une cohérence entre la forme de la matrice P (ou Q) et la connaissance que l'on a de l'agencement des codes.

2.2.3 LMC symétrique pour des codes interdépendants

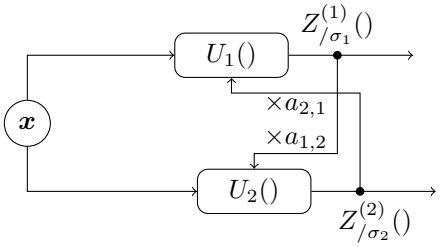


FIGURE 2.7 – Un modèle LMC pour deux sorties interconnectées.

Prenons d'abord le cas de deux codes. Le schéma 2.7 représente le modèle LMC en faisant apparaître les variables latentes. La relation (2.15) entre les processus va pouvoir s'écrire :

$$\begin{array}{rcl} Z^{(1)}_{/\sigma}(\cdot) & = & A_{12} Z^{(2)}_{/\sigma}(\cdot) + \alpha_1 U_1(\cdot) \\ Z^{(2)}_{/\sigma}(\cdot) & = & A_{21} Z^{(1)}_{/\sigma}(\cdot) + \alpha_2 U_2(\cdot) \ . \end{array} \tag{2.23}$$

Ce qui se traduit par la relation matricielle suivante :

$$\begin{pmatrix} Z^{(1)}_{/\sigma}(\cdot) \\ Z^{(2)}_{/\sigma}(\cdot) \end{pmatrix} = \frac{1}{1 - A_{12}A_{21}} \begin{bmatrix} \alpha_1 & A_{12}\alpha_1 \\ A_{21}\alpha_2 & \alpha_2 \end{bmatrix} \begin{pmatrix} U_1(\cdot) \\ U_2(\cdot) \end{pmatrix} = \begin{bmatrix} \rho_{11} & \rho_{12} \\ \rho_{21} & \rho_{22} \end{bmatrix} \begin{pmatrix} U_1(\cdot) \\ U_2(\cdot) \end{pmatrix} \ . \tag{2.24}$$

Il s'agit du cas $n_g = 2$ de l'équation $D_\sigma^{-1}\boldsymbol{Z}(\cdot) = (I - A)^{-1} D_\alpha \boldsymbol{U}(\cdot) = P\boldsymbol{U}(\cdot)$ (cf. (2.17)).

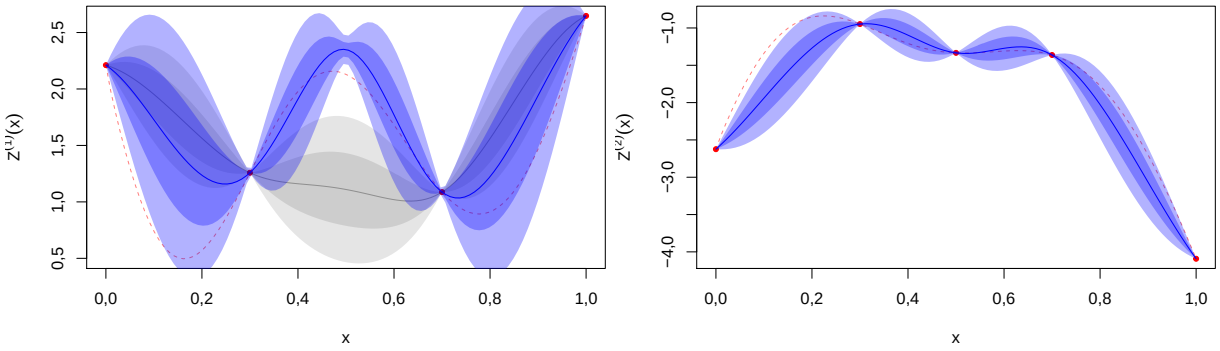
Le modèle LMC symétrique consiste à fixer tous les α_a à 1 (donc $D_\alpha = I$) et à considérer que la matrice A est symétrique. La matrice P est alors également symétrique par la relation $P = (I - A)^{-1} D_\alpha$. On va également imposer que $Q_{a,a} = \sigma_a$ en fixant $\rho_{a,a} = 1$. Rappelons que les σ_a ne sont pas des écart types mais des facteurs d'échelles, c'est-à-dire qu'il s'agit d'une partie arbitraire des paramètres de Q qui a été factorisée en $Q = D_\sigma P$. Comme les σ_a n'ont pas de raisons d'être égaux, la propriété de symétrie de P ne se vérifie pas pour Q. Ces contraintes imposées permettent de réduire le nombre de paramètres du modèle.

Les différents modèles LMC peuvent être associés aux différentes manières de choisir une matrice Q tel que $\boldsymbol{\Sigma_2} = Q\ {}^tQ$ (équation (2.11)). Ici, le choix d'une matrice P symétrique correspond à une factorisation de $(D_\sigma^{-1}\boldsymbol{\Sigma_2}D_\sigma^{-1})$ par ses valeurs propres. En effet si on nomme λ et M les matrices des valeurs propres et des vecteurs propres de $(D_\sigma^{-1}\boldsymbol{\Sigma_2}D_\sigma^{-1})$, on peut voir une équivalence entre les relations $D_\sigma^{-1}\boldsymbol{\Sigma_2}D_\sigma^{-1} = M\lambda\ {}^tM = (M\lambda^{1/2}\ {}^tM)(M\lambda^{1/2}\ {}^tM)$ et $D_\sigma^{-1}\boldsymbol{\Sigma_2}D_\sigma^{-1} = P\ {}^tP = P^2$. La relation entre les valeurs propres de $(D_\sigma^{-1}\boldsymbol{\Sigma_2}D_\sigma^{-1})$ et de $\boldsymbol{\Sigma_2}$ n'est pas triviale (et on ne cherche pas à l'expliciter ici). Dans notre cas $n_g = 2$, on obtient $\rho_{12} = \rho_{21} = \frac{A_{12}}{1 - {A_{12}}^2}$, et l'équation (2.18) devient :

$$\begin{array}{rcl} k(x^{(1)}, x'^{(1)}) & = & \sigma_1^2 (r_1(x, x') + \rho_{12}^2 r_2(x, x')) \\ k(x^{(2)}, x'^{(2)}) & = & \sigma_2^2 (\rho_{12}^2 r_1(x, x') + r_2(x, x')) \\ k(x^{(1)}, x'^{(2)}) & = & \sigma_1 \sigma_2 \rho_{12} (r_1(x, x') + r_2(x, x')) \ . \end{array} \tag{2.25}$$

On comprends bien à partir de ce cas $n_g = 2$ que le modèle LMC symétrique accorde le même poids à $U_a(\cdot)$ et $U_b(\cdot)$ quand on observe la covariance entre les niveaux a et b. On note qu'il aurait été possible de fixer les α_a différemment, notamment de façon à ce que les σ_a correspondent à des écart types propres à chaque niveau, mais cela conduit à des $\rho_{a,b}$ ne pouvant pas être simplement bornés entre -1 et 1 et cela pose des problèmes au moment de l'évaluation des paramètres.

La figure 2.8 présente un exemple de cokrigeage LMC symétrique sur un cas 1D à deux niveaux.



(a) Prédictions sur la grandeur d'intérêt

(b) Prédictions sur la grandeur secondaire

FIGURE 2.8 – Exemple 1D de cokrigeage utilisant un modèle LMC symétrique. La grandeur d'intérêt se trouve à gauche, la grandeur secondaire est à droite. La ligne pointillé rouge correspond à la vraie fonction qu'on cherche à modéliser. Les points rouges sont les points connus. En bleu sont représentés la moyenne de prédiction et les intervalles de confiance à 1 et 2 sigmas. En gris est représenté le modèle de krigeage classique (intervalle de confiance à 1 et 2 sigmas).

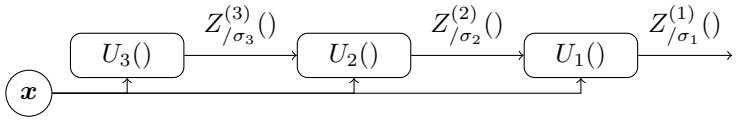


FIGURE 2.9 – Un LMC markovien pour trois sorties en cascade.

2.2.4 LMC markovien pour des codes séquentiels

La factorisation de $\mathbf{\Sigma_2}$ précédemment utilisée n'est pas la seule manière. Puisque $\mathbf{\Sigma_2}$ est symétrique et définie positive, on peut la décomposer à l'aide d'une décomposition de Cholesky. Dans ce cas les décompositions de $(D_\sigma^{-1}\mathbf{\Sigma_2}D_\sigma^{-1})$ et de $\mathbf{\Sigma_2}$ sont liées de manière triviale. Du point de vue des grandeurs, cela va correspondre à des relations de la sorte (ici pour 3 niveaux) :

$$\begin{array}{rcl} Z^{(1)}_{/\sigma}(\cdot) & = & A_{12}Z^{(2)}_{/\sigma}(\cdot) + \sqrt{1-A_{12}^2}U_1(\cdot) \\ Z^{(2)}_{/\sigma}(\cdot) & = & A_{23}Z^{(3)}_{/\sigma}(\cdot) + \sqrt{1-A_{23}^2}U_2(\cdot) \\ Z^{(3)}_{/\sigma}(\cdot) & = & U_3(\cdot)\ . \end{array} \tag{2.26}$$

Ici on a fixé les coefficients α_a à $\sqrt{1-A_{a,a+1}}$ (et α_{n_g} à 1). Nous verrons par la suite que ce choix va permettre de faire en sorte que les σ_a correspondent aux variances propres des différentes grandeurs. Les σ_a estimés seront ainsi mieux interprétables. De plus nous pouvons borner les paramètres $A_{a,a+1}$ entre -1 et $+1$, ce qui peut s'avérer important lors de l'estimation des paramètres.

On peut faire correspondre les relations entre les niveaux avec des codes qui seraient enchaînés comme sur la figure 2.9. Ces relations peuvent se lire « $Z^{(1)}$ se compose de $Z^{(2)}$ plus un processus qui lui est indépendant ; $Z^{(2)}$ se compose de $Z^{(3)}$ plus un processus qui lui est indépendant ; etc. ». Ces contraintes correspondent au LMC markovien. Cette appellation provient de ce que chaque niveau dépend uniquement de celui qui lui est immédiatement inférieur quand celui-ci est connu, comme un processus markovien. Dans la littérature, ce modèle est également connu sous le nom de cokrigeage multifidélité (Le Gratiet 2013), krigeage autorégressif (Kennedy and O'Hagan 2000) ou modèle LMC par décomposition de Cholesky (Fricker, Oakley, and Urban 2013). Il n'est pas rare de le trouver dans un ordre inversé, c'est-à-dire avec $Z^{(1)}()$ qui ne dépend que de $U_1()$ et les niveaux suivant qui dépendent chacun du précédent.

En fait ce modèle est plus restrictif qu'une décomposition de Cholesky de $\mathbf{\Sigma_2}$ puisqu'on va drastiquement réduire le nombre de paramètres (« restrictif » dans le sens où toutes les matrices $\mathbf{\Sigma_2}$ imaginables ne seront pas forcément atteignables). On va ainsi passer de n_g^2 paramètres de la matrice Q à $2n_g - 1$ paramètres (dont n_g pour les σ_a). À partir de l'équation précédente, on en déduit la matrice P :

$$P = \begin{bmatrix} \sqrt{1-A_{12}^2} & A_{12}\sqrt{1-A_{23}^2} & A_{12}A_{23} \\ 0 & \sqrt{1-A_{23}^2} & A_{23} \\ 0 & 0 & 1 \end{bmatrix}\ . \tag{2.27}$$

Dans le cas général, les coefficients de P sont (avec la convention $A_{n_g,n_g+1} = 0$) :

$$\begin{array}{rcll} \rho_{a,b} & = & \prod_{l=a}^{b-1} A_{l,l+1}\sqrt{1-A_{b,b+1}^2} & \text{if } a < b\ , \\ & = & \sqrt{1-A_{a,a+1}^2} & \text{if } a = b\ , \\ & = & 0 & \text{if } a > b\ . \end{array} \tag{2.28}$$

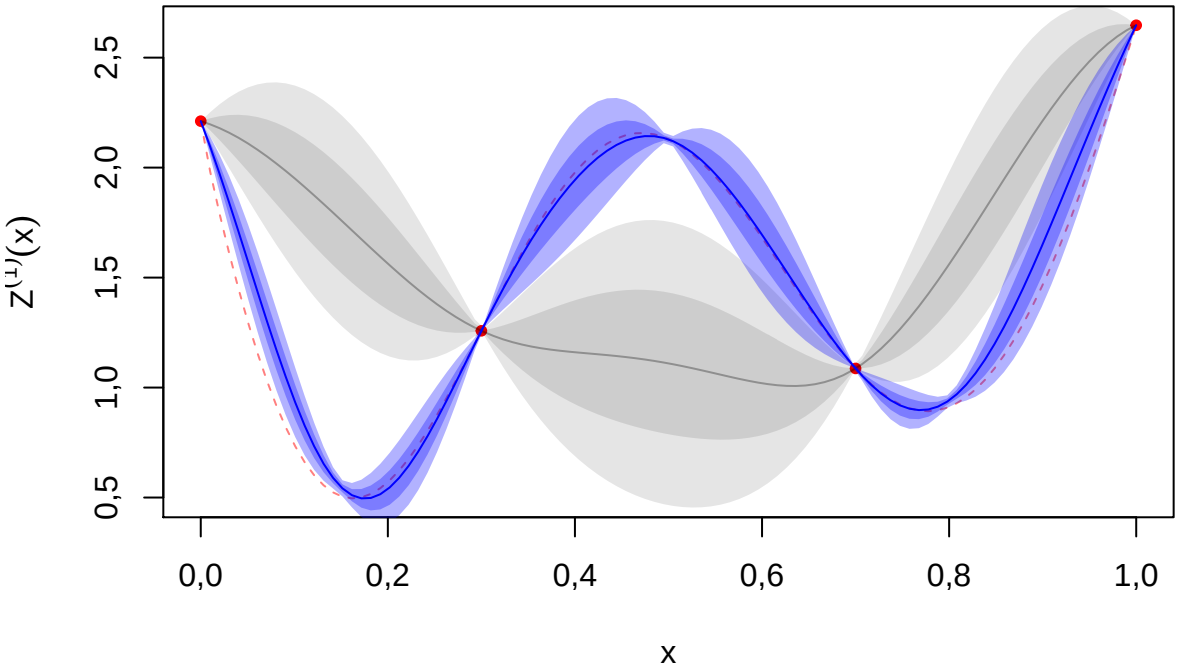
La matrice P, et avec elle la matrice Q, est bien triangulaire.

Toujours dans le cas à 3 niveaux, la fonction de covariance de l'équation (2.18) devient :

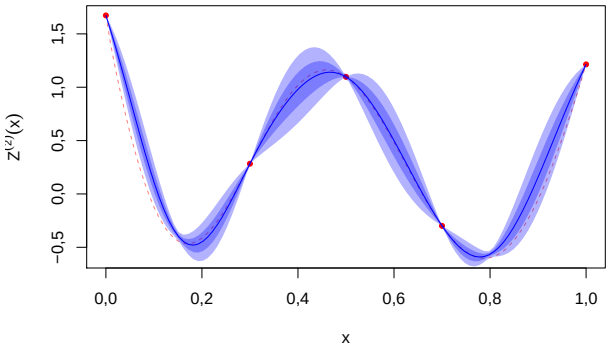
$$\begin{array}{rcl} k(x^{(1)}, x'^{(1)}) & = & \sigma_1^2\ \left(\left(1-A_{12}^2\right) r_1(x,x') + A_{12}^2\left(\left(1-A_{23}^2\right) r_2(x,x') + A_{23}^2 r_3(x,x')\right)\right) \\ k(x^{(2)}, x'^{(2)}) & = & \sigma_2^2\ \left(\left(1-A_{23}^2\right) r_2(x,x') + A_{23}^2 r_3(x,x')\right) \\ k(x^{(3)}, x'^{(3)}) & = & \sigma_3^2\ r_3(x,x') \\ k(x^{(1)}, x'^{(2)}) & = & \sigma_1\sigma_2\ A_{12}\left(\left(1-A_{23}^2\right) r_2(x,x') + A_{23}^2 r_3(x,x')\right) \\ k(x^{(2)}, x'^{(3)}) & = & \sigma_2\sigma_3\ A_{23}\ r_3(x,x') \\ k(x^{(1)}, x'^{(3)}) & = & \sigma_1\sigma_3\ A_{12}A_{23}\ r_3(x,x')\ . \end{array} \tag{2.29}$$

Il est important de noter que l'ordre dans lequel on aura placé les grandeurs a une importance non négligeable sur le modèle.

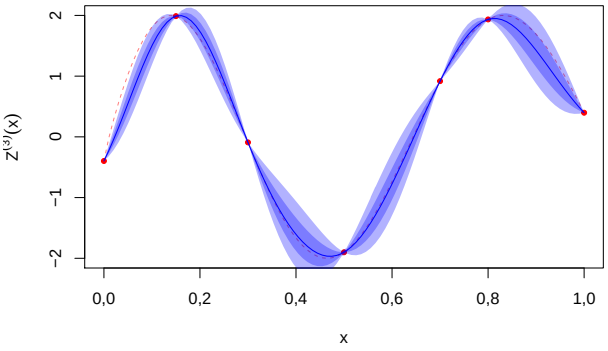
La figure 2.10 présente un exemple de cokrigeage LMC markovien sur un cas 1D à trois niveaux.



(a) Prédiction sur la grandeur d'intérêt



(b) Prédiction sur le deuxième niveau



(c) Prédiction sur le troisième niveau

FIGURE 2.10 – Exemple 1D à 3 niveaux de cokrigeage utilisant un modèle LMC markovien. La grandeur d'intérêt se trouve en haut (a), les grandeurs secondaires sont en bas (b et c). La ligne pointillé rouge correspond à la vraie fonction qu'on cherche à modéliser. Les points rouges sont les points connus. En bleu sont représentés la moyenne de prédiction et les intervalles de confiance à 1 et 2 sigmas. En bleu est représenté le modèle de krigeage classique (intervalle de confiance à 1 et 2 sigma).

2.2.5 LMC *ad hoc* pour une structure de codes quelconque

Ce n'est pas une nécessité, mais faire correspondre la paramétrisation de la covariance du cokrigeage avec la structure des codes est une bonne pratique qui permet de simplifier le modèle tout en le rendant interprétable. Par exemple, lorsqu'il existe une hiérarchie dans les sorties, le modèle LMC markovien est plus simple et permet d'interpréter les variables latentes comme un calcul spécifique effectué par chaque code. Il est toujours possible de modéliser n'importe quelle structure de codes avec n'importe quel modèle statistique multisortie. Mettre une structure markovienne sur un code qui ne correspond pas à cette hypothèse signifie créer une hiérarchie qui n'existe pas au sein des variables latentes. L'utilisation d'un modèle symétrique pour un code en cascade génère une complexité de calcul qui aurait pu être évitée. Et bien sûr, on peut avoir recours à des processus gaussiens indépendants pour chaque sortie, mais cela annule tout effort de partage d'informations entre ceux-ci.

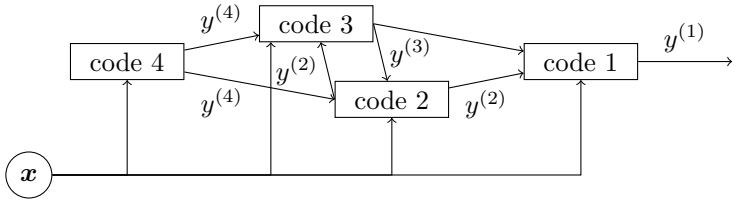


FIGURE 2.11 – Exemple de mélange de 4 modules de code en parallèle et en série.

Une bonne pratique consiste donc à baser la structure de la covariance sur les interactions des modules du code. Cette paramétrisation de la covariance peut être déduite de la combinaison de blocs de codes en série et en parallèle. A titre d'exemple, considérons les modules de code dessinés dans la figure 2.11. Une traduction directe des interactions en termes de notre modèle statistique s'écrit comme suit :

$$\begin{aligned} Z^{(1)}_{/\sigma}(\cdot) &= A_{12}Z^{(2)}_{/\sigma}(\cdot) + A_{13}Z^{(3)}_{/\sigma}(\cdot) + \alpha_1 U_1(\cdot) \\ Z^{(2)}_{/\sigma}(\cdot) &= A_{23}Z^{(3)}_{/\sigma}(\cdot) + A_{24}Z^{(4)}_{/\sigma}(\cdot) + \alpha_2 U_2(\cdot) \\ Z^{(3)}_{/\sigma}(\cdot) &= A_{32}Z^{(2)}_{/\sigma}(\cdot) + A_{34}Z^{(4)}_{/\sigma}(\cdot) + \alpha_3 U_3(\cdot) \\ Z^{(4)}_{/\sigma}(\cdot) &= \alpha_4 U_4(\cdot)\ . \end{aligned} \tag{2.30}$$

En ajoutant la symétrie entre (2) et (3) ($A_{23} = A_{32}$, $A_{12} = A_{13}$ et $A_{34} = A_{24}$), et en fixant les α_a à 1, on obtient pour $P = (I - A)^{-1}$:

$$P = \begin{bmatrix} 1 & \frac{A_{12}}{1-A_{23}} & \frac{A_{12}}{1-A_{23}} & \frac{2A_{12}A_{34}}{1-A_{23}} \\ 0 & \frac{1}{1-A_{23}^2} & \frac{A_{23}}{1-A_{23}^2} & \frac{A_{34}}{1-A_{23}} \\ 0 & \frac{A_{23}}{1-A_{23}^2} & \frac{1}{1-A_{23}^2} & \frac{A_{34}}{1-A_{23}} \\ 0 & 0 & 0 & 1 \end{bmatrix}\ . \tag{2.31}$$

On peut reconnaître dans cette matrice les parties parallèles et séquentielles de la structure des codes : P est presque triangulaire à part pour le bloc central 2×2 qui correspond aux codes parallèles 2 et 3. À une constante de normalisation près, on retrouve en ρ_{14} le produit $A_{12}A_{34}$, typique des modèles markoviens.

Les modèles combinant ainsi une partie symétrique et une partie markovienne n'ont pas été étudiés au-delà de la théorie présentée ici. Ces modèles n'ont notamment pas été inclus dans l'étude des performances décrites plus loin (section 2.4).

2.2.6 Autres modèles de cokrigeage

Il existe d'autres modèles de cokrigeage en-dehors des modèles LMC. On peut notamment construire une fonction de covariance qui sera définie positive *par construction*, comme le font (Gneiting, Kleiber, and Schlather 2010).

Les modèles de cokrigeage à covariances séparables (Conti and O'Hagan 2010) partent de la matrice de covariance intergroupes $\mathbf{\Sigma_2}$, parfois notée T (Roustant et al. 2018). Ils combinent ensuite cette matrice avec un noyau de covariance dépendant uniquement de l'espace d'entrée. La combinaison peut être additive, multiplicative, ou autre. Cela peut donner lieu à des équations de la forme :

$$\text{(cokrigeage additif)} \qquad k\left(x_1^{(a)}, x_2^{(b)}\right) = \sigma_a\sigma_b\left(r_x(x_1, x_2) + T_{a,b}\right) \tag{2.32}$$

$$\text{(cokrigeage multiplicatif)} \qquad k\left(x_1^{(a)}, x_2^{(b)}\right) = \sigma_a\sigma_b\left(r_x(x_1, x_2)T_{a,b}\right) \tag{2.33}$$

$$\text{(cokrigeage ANOVA)} \qquad k\left(x_1^{(a)}, x_2^{(b)}\right) = \sigma_a\sigma_b\left(1 + r_x(x_1, x_2)\right)\left(1 + T_{a,b}\right) \tag{2.34}$$

La façon de combiner la covariance spatiale avec la covariance intergroupes s'ajoute à la complexité de combiner les différentes dimensions dans la covariance spatiale et peut donner lieu à des modèles divers et variés.

Dans ces modèles, remonter aux relations entre les grandeurs peut être ardu, mais certains de ces modèles conduisent à des relations linéaires et peuvent être interprétés comme des cas particuliers de modèles LMC (tout comme les modèles LMC symétrique et markovien n'en étaient que des cas particuliers). On pourra par exemple aller voir Seeger, Teh, and Jordan (2004) et Micchelli and Pontil (2005) pour des modèles développés pour des besoins en *machine learning* qui s'avèrent être des cas de LMC.

Néanmoins, il existe d'autres modèles de cokrigeage complètement différents, et qui ne peuvent s'interpréter comme des modèles LMC. Le plus populaire est probablement celui basé sur une convolution de processus stochastiques (dont les kernels sont ici notés Γ_a) (Álvarez and Lawrence 2011 ; Fricker, Oakley, and Urban 2013 ; Phillip and Marcus 2005) :

$$Z_a(x) = \int \Gamma_a(x - s)U(s)\,\mathrm{d}s\ . \tag{2.35}$$

Cette approche permet d'obtenir des sorties corrélées, même si elles présentent une longueur de portée et une régularité très différentes. À l'inverse des LMC, les paramètres de corrélations spatiales sont plus facilement interprétables, mais les paramètres contrôlant les interactions intergroupes le sont moins. Un inconvénient important est l'absence d'une écriture analytique pour la convolution, à moins de n'utiliser que des noyaux bien spécifiques (des noyaux gaussiens par exemple). Plus de détails peut être trouvé dans Álvarez, Rosasco, and Lawrence (2012)

Enfin, plusieurs extensions au cokrigeage existent générant des modèles dont les prédictions ne suivraient pas des distributions gaussiennes. Un exemple d'un telle construction peut être trouvé dans Marque-Pucheu, Perrin, and Garnier (2017) ou dans Le Gratiet (2013), où des simulateurs emboîtés $(Z_1(Z_2(x), x))$ sont étudiés. Dans la section 2.3, nous présentons d'ailleurs une méthode (appelée « hyper-krigeage ») de métamodélisation introspectif par composition de processus gaussiens, dont les prédictions ne correspondent pas à des lois normales.

2.2.7 Optimisation des paramètres et vraisemblance concentrée

La problématique de l'estimation des paramètres est aussi présente pour le cokrigeage qu'elle l'était pour le krigeage (section 1.2.2). Malheureusement, le modèle étant plus riche qu'un krigeage, il contient davantage de paramètres. Cela a pour conséquence de complexifier la recherche des « meilleurs » paramètres (au sens du maximum de vraisemblance, comme nous l'avons vu dans la Partie I). Il est bien connu que les performances des optimiseurs décroissent généralement lorsque le nombre de dimensions de l'espace de recherche augmente.

Un modèle LMC a besoin de n_l ($= n_g$ sous nos hypothèses) paramètres de portée pour décrire les U_b, tant qu'on leur attribue des noyaux simples [3] et isotropes [4], et n_g autres paramètres pour définir les σ_a, en plus des paramètres nécessaires pour décrire la matrice de covariance intergroupes ($n_g - 1$ pour un LMC

3. par opposition à un noyau composé, par exemple d'une somme de noyau de base

4. beaucoup de problème d'ingénierie peuvent néanmoins requérir l'usage de noyaux anisotrope, ayant des dimensions du domaine d'entrée qui ont des sensibilités différentes

markovien, ou $n_g(n_g-1)$ pour un LMC symétrique). À ces paramètres peut éventuellement s'ajouter un paramètre de bruit ou de *nugget*. De plus, il faut également paramétrer la tendance (la moyenne *a priori*, cf. équation (2.7)), qu'on a ignorée dans la section précédente, ce qui est plus complexe dans le cas du cokrigeage que du krigeage puisqu'on a une tendance différente pour chacune des n_g grandeurs. Pour reprendre les notations du krigeage, nous allons considérer une moyenne *a priori* $\boldsymbol{m}(\cdot)$ qui sera paramétrée comme une combinaison linéaire de fonctions fixées $f_l^{(a)}(x)$, mais avec la contrainte qu'il n'y a pas de fonction commune à plusieurs grandeurs (i.e. les tendances pour chaque grandeurs seront entièrement distinctes) :

$$m(x^{(a)}) = m^{(a)}(x) = \sum_{l=1}^{p_a} \beta_l^{(a)} f_l^{(a)}(x) \ . \tag{2.36}$$

Comme dans le cas du krigeage, le vecteur $\beta^{(a)}$ contient les paramètres à estimer et le vecteur ${\mathrm{f}_*}^{(a)}(\cdot)$ les fonctions choisies a priori qui composent la tendance pour la grandeur a. La contrainte imposée (l'absence de tendance commune à estimer) permettra de réduire l'espace de recherche lors de l'optimisation des paramètres, comme expliqué plus loin. C'est une contrainte assez légère, d'une part parce qu'il est rare de vouloir une tendance commune à différentes grandeurs et d'autre part parce qu'il est toujours possible d'estimer la partie commune de la tendance par un moyen quelconque (régression linéaire par exemple) et de la soustraire directement aux observations.

Généralisons l'équation précédente pour l'ensemble des grandeurs et accédons à une écriture matricielle. Comme pour le krigeage on notera $\boldsymbol{F} = \mathbf{f}_*(\boldsymbol{X})$ la matrice $(N \times p) = (\sum_{a=1}^{n_g} N_a) \times (\sum_{a=1}^{n_g} p_a)$ de terme général $\boldsymbol{F}_{i,l} = \boldsymbol{f}_l(\boldsymbol{X}_i)$. Par convention cette matrice contient un zéro chaque fois de l'observation i appartient à une grandeur différente de celle associée au paramètre l. Nous notons $p = \sum_{a=1}^{n_g} p_a$, $\boldsymbol{\beta} = (\beta^{(1)}, \beta^{(2)}, \ldots)$ et $\boldsymbol{f}(\cdot) = (f^{(1)}(\cdot), f^{(2)}(\cdot), \ldots)$ (tous les deux sont des vecteurs de tailles p). On obtient :

$$m(x^{(a)}) = \sum_{l=1}^{p} \boldsymbol{\beta}_l \boldsymbol{f}_j(x^{(a)}) \ , \tag{2.37}$$

et, pour l'ensemble des observations,

$$m(\boldsymbol{X}) = \boldsymbol{F}\boldsymbol{\beta} \ . \tag{2.38}$$

Exemple en 1D (x scalaire) avec $n_g = 2$, $N_1 = N_2 = 2$ ($N = 4$), $m^{(1)}(x) = ax+b$, $m^{(2)}(x) = cx^2+dx+e$:

On a alors $\beta^{(1)} = (a,b)$ et $\beta^{(2)} = (c,d,e)$, et $f_1^{(1)}(x) = x$, $f_2^{(1)}(x) = 1$, $f_1^{(2)}(x) = x^2$, $f_2^{(2)}(x) = x$ et $f_3^{(2)}(x) = 1$, et par convention $f_{1,2}^{(1)}(x^{(2)}) = 0$ et $f_{1,2,3}^{(2)}(x^{(1)}) = 0$. Si le vecteur $\boldsymbol{X}$ est telle que les deux premières mesures correspondent à la grandeur (1) et les deux dernière à la grandeur (2), la matrice $\boldsymbol{F}$ s'écrira alors :

$$\boldsymbol{F} \ = \ \begin{bmatrix} f_1^{(1)}(\boldsymbol{X}_1) & f_2^{(1)}(\boldsymbol{X}_1) & f_1^{(2)}(\boldsymbol{X}_1) & f_2^{(2)}(\boldsymbol{X}_1) & f_3^{(2)}(\boldsymbol{X}_1) \\ f_1^{(1)}(\boldsymbol{X}_2) & f_2^{(1)}(\boldsymbol{X}_2) & f_1^{(2)}(\boldsymbol{X}_2) & f_2^{(2)}(\boldsymbol{X}_2) & f_3^{(2)}(\boldsymbol{X}_2) \\ f_1^{(1)}(\boldsymbol{X}_3) & f_2^{(1)}(\boldsymbol{X}_3) & f_1^{(2)}(\boldsymbol{X}_3) & f_2^{(2)}(\boldsymbol{X}_3) & f_3^{(2)}(\boldsymbol{X}_3) \\ f_1^{(1)}(\boldsymbol{X}_4) & f_2^{(1)}(\boldsymbol{X}_4) & f_1^{(2)}(\boldsymbol{X}_4) & f_2^{(2)}(\boldsymbol{X}_4) & f_3^{(2)}(\boldsymbol{X}_4) \end{bmatrix} \ = \ \begin{bmatrix} \boldsymbol{X}_1 & 1 & 0 & 0 & 0 \\ \boldsymbol{X}_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & {\boldsymbol{X}_3}^2 & \boldsymbol{X}_3 & 1 \\ 0 & 0 & {\boldsymbol{X}_4}^2 & \boldsymbol{X}_4 & 1 \end{bmatrix} \ . \tag{2.39}$$

Et on aura également :

$$m(\boldsymbol{X}) \ = \ \begin{bmatrix} m(\boldsymbol{X}_1) \\ m(\boldsymbol{X}_2) \\ m(\boldsymbol{X}_3) \\ m(\boldsymbol{X}_4) \end{bmatrix} \ = \ \boldsymbol{F}\boldsymbol{\beta} \ = \ \begin{bmatrix} a\boldsymbol{X}_1 + b \\ a\boldsymbol{X}_2 + b \\ c{\boldsymbol{X}_3}^2 + d\boldsymbol{X}_3 + e \\ c{\boldsymbol{X}_4}^2 + d\boldsymbol{X}_4 + e \end{bmatrix} \ . \tag{2.40}$$

Un modèle de cokrigeage peut ainsi facilement avoir beaucoup plus de paramètres qu'un modèle de krigeage. Ce qui suit va présenter une extension de la notion de vraisemblance concentrée, vue pour le krigeage en 1.2.2, au cas du cokrigeage, et ainsi obtenir les valeurs « optimales » au sens du maximum de vraisemblance, d'abord des p paramètres de tendance $\boldsymbol{\beta}$ et ensuite des n_g paramètres d'échelle $\boldsymbol{\sigma}$.

Commençons par remarquer que les facteurs d'échelle peuvent toujours être factorisés. D'un point de vue pratique, cela nécessite de faire attention si on utilise un bruit ou un effet pépite dans le modèle à ce que ceux-ci soient relatifs à la grandeur, et non absolu comme c'est parfois le cas dans les codes informatiques. La fonction de covariance peut s'écrire en fonction de la fonction de « covariance réduite » :

$$k\left(x^{(a)}, x'^{(b)}\right) = k_{ab}(x, x') = \sigma_a \sigma_b r(x^{(a)}, x'^{(b)}) \ . \tag{2.41}$$

$r(x^{(a)}, x'^{(b)})$ est une fonction différente des $r_b(\cdot,\cdot)$, qui désignaient les noyaux des variables latentes $U_b(\cdot)$ et dont on ne reparlera pas. L'expression de la covariance réduite $r(x^{(a)}, x'^{(b)})$ en fonction des noyaux des variables latentes se retrouve à partir de l'équation (2.18). Le point important est que $r(\cdot,\cdot)$ est indépendant des $\boldsymbol{\sigma}$.

Ainsi la matrice $\boldsymbol{K}$, matrice de covariance de l'ensemble des observations, va pouvoir s'écrire :

$$\boldsymbol{K} = D_\sigma^{(N)} \boldsymbol{R} D_\sigma^{(N)} \ , \tag{2.42}$$

avec $D_\sigma^{(N)}$ la matrice diagonale $N \times N$ contenant en $i^{ème}$ position le σ correspondant à la grandeur de $\boldsymbol{X}_i$, et $\boldsymbol{R}$ la matrice de covariance réduite $\boldsymbol{R} = r(\boldsymbol{X}, \boldsymbol{X})$. Là encore, $\boldsymbol{R}$ est indépendant des paramètres $\boldsymbol{\sigma}$. De plus, on constate que $\boldsymbol{K}$ est définie positive en même temps que $\boldsymbol{R}$. On peut donc toujours effectuer cette factorisation, même avec des modèles de cokrigeage différents de ceux présentés précédemment, quitte à réécrire la paramétrisation de la fonction de covariance. De plus, cela permet d'avoir une interprétation claire des $\boldsymbol{\sigma}$, cette factorisation est donc recommandée. Si on s'arrange pour que $r(x, x) = 1$, les $\boldsymbol{\sigma}$ pourront même s'interpréter comme la variance propre à chaque niveau.

On remarquera, et ce sera utile par la suite, que la matrice $\boldsymbol{F}$ peut commuter avec la matrice diagonale D_σ. Évidemment la matrice $\boldsymbol{F}$ n'étant pas carrée, il ne s'agit pas d'une vraie commutation au sens stricte du terme, mais on a :

$$D_\sigma^{(N)} \boldsymbol{F} = \boldsymbol{F} D_\sigma^{(p)} \ , \tag{2.43}$$

avec $D_\sigma^{(p)}$ la matrice diagonale $p \times p$ contenant en $l^{ème}$ position le σ correspondant à la même grandeur que $\boldsymbol{\beta}_l$. Cette commutation est rendue possible par le fait que toutes les cellules de $\boldsymbol{F}$ qui associent une observation d'une grandeur a avec une fonction de tendance d'une grandeur b sont nulles si $a \neq b$ (cf. l'exemple au-dessus, équation (2.39)).

Paramètres de tendance $\boldsymbol{\beta}$

La formule utilisée pour estimer les paramètres de tendance du krigeage est toujours valable :

$$\hat{\boldsymbol{\beta}} = \left({}^t\boldsymbol{F} \boldsymbol{K}^{-1} \boldsymbol{F} \right)^{-1} {}^t\boldsymbol{F} \boldsymbol{K}^{-1} \boldsymbol{Y} \ . \tag{2.44}$$

En utilisant les équations (2.42) et (2.43), il vient :

$$\begin{aligned} \hat{\boldsymbol{\beta}} &= \left({}^t\boldsymbol{F} D_\sigma^{(N)-1} \boldsymbol{R}^{-1} D_\sigma^{(N)-1} \boldsymbol{F} \right)^{-1} {}^t\boldsymbol{F} D_\sigma^{(N)-1} \boldsymbol{R}^{-1} D_\sigma^{(N)-1} \boldsymbol{Y} \\ &= \left(D_\sigma^{(p)-1} \, {}^t\boldsymbol{F} \boldsymbol{R}^{-1} \boldsymbol{F} D_\sigma^{(p)-1} \right)^{-1} D_\sigma^{(p)-1} \, {}^t\boldsymbol{F} \boldsymbol{R}^{-1} \boldsymbol{Y}_{/\boldsymbol{\sigma}} \\ &= D_\sigma^{(p)} \left({}^t\boldsymbol{F} \boldsymbol{R}^{-1} \boldsymbol{F} \right)^{-1} {}^t\boldsymbol{F} \boldsymbol{R}^{-1} \boldsymbol{Y}_{/\boldsymbol{\sigma}} \ . \end{aligned} \tag{2.45}$$

Notons B la matrice $p \times N$ correspondant à $\left({}^t\boldsymbol{F} \boldsymbol{R}^{-1} \boldsymbol{F} \right)^{-1} {}^t\boldsymbol{F} \boldsymbol{R}^{-1}$. La matrice B est indépendante des valeurs des σ (qu'elles soient fixées ou évaluées). Le lien entre $\hat{\boldsymbol{\beta}}$ et les σ s'explicite clairement :

$$\hat{\boldsymbol{\beta}} = D_\sigma^{(p)} B \boldsymbol{Y}_{/\sigma} \ . \tag{2.46}$$

Ainsi $\hat{\boldsymbol{\beta}}$ se calcule immédiatement à partir des paramètres $\boldsymbol{\sigma}$ et des autres paramètres (présents dans la matrice B).

Paramètres d'échelle $\boldsymbol{\sigma}$

Par l'égalité (2.42) on a :

$$\begin{aligned} \ln(|\boldsymbol{K}|) &= \ln(|\boldsymbol{R}|) + \ln(\left|D_\sigma^{(N)}\right|^2) \\ &= \ln(|\boldsymbol{R}|) + 2\sum_{a=1}^{n_g}(N_a \ln(\sigma_a)) \ . \end{aligned} \tag{2.47}$$

Et par les équations (2.43) et (2.46), on obtient :

$$\begin{aligned} \left(\boldsymbol{Y} - \boldsymbol{F}\hat{\boldsymbol{\beta}}\right) &= \left(\boldsymbol{Y} - \boldsymbol{F}D_\sigma^{(p)}B\boldsymbol{Y}_{/\sigma}\right) \\ &= \left(\boldsymbol{Y} - D_\sigma^{(N)}\boldsymbol{F}B\boldsymbol{Y}_{/\sigma}\right) \\ &= D_\sigma^{(N)}\left(D_\sigma^{(N)-1}\boldsymbol{Y} - \boldsymbol{F}B\boldsymbol{Y}_{/\sigma}\right) \\ &= D_\sigma^{(N)}\left(I_N - \boldsymbol{F}B\right)\boldsymbol{Y}_{/\sigma} \ . \end{aligned} \tag{2.48}$$

Ce qui nous permet d'écrire :

$$\begin{aligned} {}^t\left(\boldsymbol{Y} - \boldsymbol{F}\hat{\boldsymbol{\beta}}\right)\boldsymbol{K}^{-1}\left(\boldsymbol{Y} - \boldsymbol{F}\hat{\boldsymbol{\beta}}\right) &= {}^t\left(\boldsymbol{Y} - \boldsymbol{F}\hat{\boldsymbol{\beta}}\right)D_\sigma^{(N)-1}\boldsymbol{R}^{-1}D_\sigma^{(N)-1}\left(\boldsymbol{Y} - \boldsymbol{F}\hat{\boldsymbol{\beta}}\right) \\ &= {}^t\boldsymbol{Y}_{/\sigma}\,{}^t(I_N - \boldsymbol{F}B)\,\boldsymbol{R}^{-1}\,(I_N - \boldsymbol{F}B)\,\boldsymbol{Y}_{/\sigma} \\ &= \sum_{a=1}^{n_g}\sum_{b=1}^{n_g}\frac{1}{\sigma_a}\frac{1}{\sigma_b}\,{}^t\boldsymbol{Y}I_{\{a\}}\,{}^t(I_N - \boldsymbol{F}B)\,\boldsymbol{R}^{-1}\,(I_N - \boldsymbol{F}B)\,I_{\{b\}}\boldsymbol{Y} \\ &= \sum_{a=1}^{n_g}\sum_{b=1}^{n_g}\frac{1}{\sigma_a}\frac{1}{\sigma_b}w_{ab} \ . \end{aligned} \tag{2.49}$$

Ici $I_{\{a\}}$ désigne la matrice $N \times N$ indicatrice de la grandeur a, une matrice diagonale ayant un 1 pour la i-ème valeur de la diagonale si et seulement si la i-ème observation a bien été faite sur la grandeur a (0 sinon). On a $\sum_{a=1}^{n_g} I_{\{a\}} = I_N$ et on a donc l'égalité $\boldsymbol{Y}_{/\sigma} = \sum_{a=1}^{n_g}\frac{1}{\sigma_a}I_{\{a\}}\boldsymbol{Y}$ dont on vient de se servir. On note alors $\boldsymbol{Y}_{\{a\}}$ le vecteur de taille N comportant les observations de la grandeur a et ayant 0 pour valeur pour toutes les observations des autres grandeurs ($\boldsymbol{Y}_{\{a\}} = I_{\{a\}}\boldsymbol{Y}$). La concaténation des $\boldsymbol{Y}_{\{a\}}$ en une matrice de taille $N \times n_g$ sera notée $\mathbb{Y}$. L'équation précédente introduit les w_{ab} ($w_{ab} = {}^t\boldsymbol{Y}_{\{a\}}\,{}^t(I_N - \boldsymbol{F}B)\,\boldsymbol{R}^{-1}\,(I_N - \boldsymbol{F}B)\,\boldsymbol{Y}_{\{b\}}$) qui constituent la matrice W, symétrique, définie positive et de taille $n_g \times n_g$:

$$W = {}^t\mathbb{Y}\,{}^t(I_N - \boldsymbol{F}B)\,\boldsymbol{R}^{-1}\,(I_N - \boldsymbol{F}B)\,\mathbb{Y} \ . \tag{2.50}$$

On obtient donc pour l'expression de la log-vraisemblance (vu pour le krigeage en section 1.2.2) :

$$\begin{aligned} \mathcal{L}(\hat{\boldsymbol{\beta}}, \boldsymbol{\sigma}, \boldsymbol{\Theta}) &= -\tfrac{N}{2}\ln(2\pi) - \tfrac{1}{2}\ln(|\boldsymbol{K}|) - \tfrac{1}{2}\,{}^t\left(\boldsymbol{Y} - \boldsymbol{F}\hat{\boldsymbol{\beta}}\right)\boldsymbol{K}^{-1}\left(\boldsymbol{Y} - \boldsymbol{F}\hat{\boldsymbol{\beta}}\right) \\ &= -\tfrac{N}{2}\ln(2\pi) - \tfrac{1}{2}\ln(|\boldsymbol{R}|) - \sum_{a=1}^{n_g}(N_a\ln(\sigma_a)) - \tfrac{1}{2}\sum_{a=1}^{n_g}\sum_{b=1}^{n_g}\frac{1}{\sigma_a}\frac{1}{\sigma_b}w_{ab} \ . \end{aligned} \tag{2.51}$$

Cette écriture est rendue possible par l'absence de tendance commune à plusieurs grandeurs (condition *sine qua non* à l'égalité (2.43)). Elle nous permet de rechercher analytiquement les paramètres $\boldsymbol{\sigma}$ qui maximisent la vraisemblance. Les résultats présentés ci-après sont nouveaux et permettent d'étendre la notion de **vraisemblance concentrée** (vu pour le krigeage en section 1.2.2) au cas du cokrigeage.

Les $\hat{\boldsymbol{\sigma}}$ qui maximisent la vraisemblance vont annuler la dérivée de la log-vraisemblance :

$$\left\{\frac{\partial\mathcal{L}}{\partial\sigma_a} = 0\right\}_{a\in[1\ldots n_g]} \ . \tag{2.52}$$

Ainsi nous cherchons les $\hat{\sigma}_a$ qui sont solution du système suivant :

$$\left\{ -\frac{N_a}{\sigma_a} + \frac{1}{\sigma_a^2} \sum_{b=1}^{n_g} \frac{1}{\sigma_b} w_{ab} = 0 \right\}_{a \in [1 \dots n_g]} . \tag{2.53}$$

On ne s'intéresse qu'aux solutions strictement positives pour tous les σ_a et le système peut s'écrire de manière équivalente :

$$\left\{ N_a \sigma_a - \sum_{b=1}^{n_g} \frac{1}{\sigma_b} w_{ab} = 0 \right\}_{a \in [1 \dots n_g]} . \tag{2.54}$$

Sans résoudre le système, on peut déjà remarquer (grâce à l'équation (2.54)) que les $\hat{\sigma}_a$ qui maximisent la vraisemblance permettent d'aboutir à la formule de la log-vraisemblance concentrée car on a :

$$\begin{aligned} \sum_{a=1}^{n_g} \sum_{b=1}^{n_g} \tfrac{1}{\hat{\sigma}_a} \tfrac{1}{\hat{\sigma}_b} w_{ab} &= \sum_{a=1}^{n_g} \tfrac{1}{\hat{\sigma}_a} \left(\sum_{b=1}^{n_g} \tfrac{1}{\hat{\sigma}_b} w_{ab} \right) \\ &= \sum_{a=1}^{n_g} \tfrac{1}{\hat{\sigma_a}} N_a \hat{\sigma_a} \\ &= \sum_{a=1}^{n_g} N_a = N \ . \end{aligned} \tag{2.55}$$

Et ainsi :

$$\begin{aligned} \mathcal{L}_c(\boldsymbol{\Theta}) &= -\tfrac{N}{2} \ln(2\pi) - \tfrac{1}{2} \ln(|\boldsymbol{R}|) - \sum_{a=1}^{n_g} (N_a \ln(\hat{\sigma_a})) - \tfrac{1}{2} \sum_{a=1}^{n_g} \sum_{b=1}^{n_g} \tfrac{1}{\sigma_a} \tfrac{1}{\sigma_b} w_{ab} \\ &= -\tfrac{N}{2} \ln(2\pi) - \tfrac{1}{2} \ln(|\boldsymbol{R}|) - \sum_{a=1}^{n_g} (N_a \ln(\hat{\sigma_a})) - \tfrac{N}{2} \ . \end{aligned} \tag{2.56}$$

Solution analytique pour $n_g = 2$

Dans le cas où on n'a que 2 grandeurs, le système (2.54) admet une solution analytique (unique dans le domaine des $\boldsymbol{\sigma}$ positifs). En manipulant le système, on peut en effet le convertir en deux équations quadratiques pour chacun des deux $\boldsymbol{\sigma}^2$. Le détail des calculs se trouve en Annexe B.1. La solution s'écrit alors :

$$\left\{ \begin{array}{rcl} \hat{\sigma}_1^2 &=& \frac{(N_2 - N_1) w_{12}^2 + 2 N_1 w_{11} w_{22} + w_{12} \sqrt{\Delta}}{2 N_1^2 w_{22}} \\ \hat{\sigma}_2^2 &=& \frac{(N_1 - N_2) w_{12}^2 + 2 N_2 w_{11} w_{22} + w_{12} \sqrt{\Delta}}{2 N_2^2 w_{11}} \end{array} \right. , \tag{2.57}$$

avec $\Delta = (N_1 - N_2)^2 w_{12}^2 + 4 N_1 N_2 w_{11} w_{22}$.

Solution numérique

Le système (2.54) peut être réécrit sous forme matricielle ;

$$D_N D_\sigma \mathbb{1}_{n_g} - W D_\sigma^{-1} \mathbb{1}_{n_g} = \mathbb{0}_{n_g} \ , \tag{2.58}$$

avec D_σ la matrice diagonale $n_g \times n_g$ des σ_a, D_N la matrice diagonale $n_g \times n_g$ des N_a, et W la matrice des w_{ab}. $\mathbb{1}_{n_g}$ et $\mathbb{0}_{n_g}$ désignent un vecteur de taille n_g rempli respectivement de 1 ou de 0.

On ne sait pas résoudre cette équation dans le cas général, mais on peut construire, par une méthode de Newton-Raphson, une suite de vecteurs convergeant vers le vecteur solution. On peut s'assurer que la suite reste dans le domaine des valeurs positives, soit avec une vérification numérique et une réduction arbitraire de la progression de la suite (équation de récurrence (2.59)), soit en appliquant la méthode avec une transformation logarithmique (équation de récurrence (2.60)). Les calculs sous-jacents aux expressions qui suivent ainsi que l'existence et l'unicité de la solution sont donnés en Annexe B.2. Avec la continuité

et la dérivabilité des fonctions dans le domaine d'intérêt, l'existence et l'unicité de la solution garantissent la convergence de la méthode de Newton-Raphson.

$$\begin{array}{c} S_{n+1} = 2\left(\mathrm{diag}(S_n)W^{-1}D_N + \mathrm{diag}(S_n)^{-1}\right)^{-1}\mathbb{1}_{n_g}\ , \\ \textit{tant que}\ :\ (\min(S_{n+1}) \leq 0)\ ,\ \textit{faire}\ :\ \left(S_{n+1} \leftarrow \tfrac{1}{2}(S_n + S_{n+1})\right)\ . \end{array} \tag{2.59}$$

$$\log(S_{n+1}) = \log(S_n) - \mathbb{1}_{n_g} + 2\left(\mathrm{diag}(S_n)W^{-1}D_N\,\mathrm{diag}(S_n) + I_{n_g}\right)^{-1}\mathbb{1}_{n_g}\ . \tag{2.60}$$

Application de la vraisemblance concentrée

Nous allons à présent illustrer l'intérêt d'utiliser la vraisemblance concentrée avec un cas simple. Pour ce faire, nous allons utiliser le dataframe *CO2* du package de base *datasets* du logiciel R. Les données viennent d'une expérience visant à évaluer l'impact du froid sur l'absorption du CO_2 de l'herbe *Echinochloa crus-galli* (Potvin, Lechowicz, and Tardif 1990). L'expérience porte sur 12 plants répartis en 4 groupes : les plants proviennent soit du Québec soit du Mississippi et ont été ou non refroidis pendant la nuit précédant les mesures. Leurs capacités d'absorption du CO_2 en fonction de sa concentration dans l'air sont représentées sur la figure 2.12. Pour chaque plant, 7 mesures ont été effectuées à différentes valeurs de concentration de CO_2. On dispose donc de 21 mesures pour chaque groupe ($N_1 = N_2 = N_3 = N_4 = 21$). On constate des comportements différents pour les plantes du Mississippi qui ont été refroidies (en noir, groupe 'Mc'), les plantes du Mississippi qui n'ont pas été refroidies (en rouge, groupe 'Mn'), les plantes du Québec qui ont été refroidies (en vert, groupe 'Qc'), et les plantes du Québec qui n'ont pas été refroidies (en bleu, groupe 'Qn'). Ces différents comportements semblent fortement corrélés.

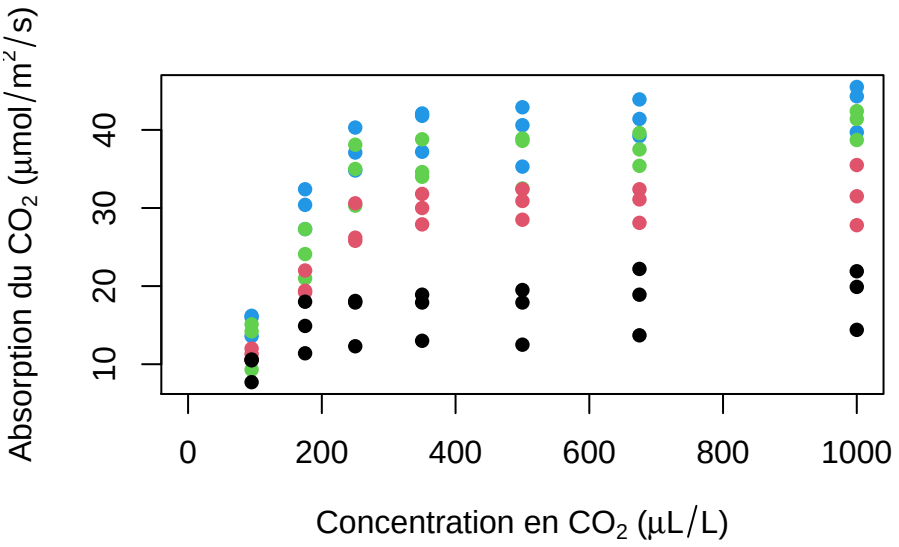


FIGURE 2.12 – Données utilisée pour l'application : absorption du CO_2 de 12 plants différents pour différentes teneur en CO_2. Les plantes du Québec sont en vert (celles ayant été refroidies) et bleu (non refroidies), celles du Mississippi sont en noir (refroidies) et en rouge (non refroidies).

Construction du modèle

Servons-nous de ces données pour construire un LMC à 4 groupes : Mc, Mn, Qc, Qn (respectivement numéroté 1, 2, 3, 4). Une tendance linéaire est estimée pour chaque groupe. Dans les formules qui suivent, la variable d'entrée x correspond à la concentration en CO_2 dans l'air.

$$\begin{pmatrix} Z^{(Mc)}(x) \\ Z^{(Mn)}(x) \\ Z^{(Qc)}(x) \\ Z^{(Qn)}(x) \end{pmatrix} = \begin{pmatrix} \beta_0^{(Mc)} + x\beta_1^{(Mc)} \\ \beta_0^{(Mn)} + x\beta_1^{(Mn)} \\ \beta_0^{(Qc)} + x\beta_1^{(Qc)} \\ \beta_0^{(Qn)} + x\beta_1^{(Qn)} \end{pmatrix} + Q \begin{pmatrix} U_1(x) \\ U_2(x) \\ U_3(x) \\ U_4(x) \end{pmatrix}\ . \tag{2.61}$$

La matrice Q vaut :

$$Q = D_\sigma P = \begin{pmatrix} \sigma_{Mc} & 0 & 0 & 0 \\ 0 & \sigma_{Mn} & 0 & 0 \\ 0 & 0 & \sigma_{Qc} & 0 \\ 0 & 0 & 0 & \sigma_{Qn} \end{pmatrix} \begin{pmatrix} 1 & \rho_{12} & \rho_{13} & \rho_{14} \\ \rho_{12} & 1 & \rho_{23} & \rho_{24} \\ \rho_{13} & \rho_{23} & 1 & \rho_{34} \\ \rho_{14} & \rho_{24} & \rho_{34} & 1 \end{pmatrix} . \tag{2.62}$$

Le modèle comporte ainsi 4 paramètres de variance intragroupe (σ_{Mc}, σ_{Mn}, σ_{Qc}, σ_{Qn}) et 6 paramètres de corrélation intergroupe (ρ_{12}, ρ_{13}, ρ_{14}, ρ_{23}, ρ_{24}, ρ_{34}).

On choisit d'utiliser le même noyau gaussien pour toutes les variables latentes (le paramètre de portée θ est le même pour tous) :

$$\forall a \in [1 \dots 4] \quad U_a(\cdot) \sim GP\left(0 \ , \ r(x_1, x_2) = e^{-(x_1 - x_2)^2/\theta^2}\right) \ , \tag{2.63}$$

$$\text{et :} \quad cov(U_a(x_1), U_b(x_2)) = 0 \ \text{ si } a \neq b \quad . \tag{2.64}$$

On a alors la fonction de covariance de $\boldsymbol{Z}(\cdot)$:

$$k\left(x_1^{(a)}, x_2^{(b)}\right) = \sigma_a \sigma_b \left(\sum_{l=1}^{4} \rho_{al}\rho_{bl}\right) r_\theta(x_1, x_2) \ . \tag{2.65}$$

Enfin, chaque groupe contenant 3 plantes, on a 3 valeurs d'absorption pour chaque concentration en CO_2. On interprète cela en supposant un bruit de mesure ϵ^2, relatif à la variance de chaque groupe :

$$K = D_\sigma \left(R + \epsilon^2 \ I\right) D_\sigma \ . \tag{2.66}$$

Notre modèle comporte donc 20 paramètres à estimer : 8 paramètres β pour les tendances, 4 paramètres σ pour les variances intragroupes, 6 paramètres ρ pour les corrélation intergroupes, 1 paramètre θ de portée et 1 paramètre ϵ de bruit.

Résultats

Nous cherchons les paramètres de ce modèle qui optimisent la vraisemblance dans 4 cas de figure :

- on ne concentre pas la vraisemblance (cas « 0 sigma »),
- on concentre la vraisemblance pour un paramètre σ_0 factorisant l'ensemble du modèle (cas « 1 sigma »),
- on concentre pour 2 paramètres σ_M et σ_Q, respectivement pour les plantes du Québec et pour les plantes du Mississippi, en utilisant la formule analytique (cas « 2 sigmas »),
- on concentre pour les 4 sigmas : σ_{Mc}, σ_{Mn}, σ_{Qc} et σ_{Qn}, en utilisant la solution numérique calculant le logarithme des sigmas (cas « 4 sigmas »).

Pour retrouver les relations entre σ_0 (respectivement σ_M et σ_Q) et $\{\sigma_{Mc} \ , \ \sigma_{Mn} \ , \ \sigma_{Qc} \ , \ \sigma_{Qn}\}$ dans le cas « 1 sigma » (respectivement dans le cas « 2 sigmas »), on va se servir du fait que la valeur maximale de la vraisemblance (et de la log-vraisemblance) ne change pas selon la paramétrisation.

Pour le cas « 1 sigma », on va utiliser la reparamétrisation suivante :

$$\begin{aligned} \sigma_{Mc} &= \sigma_0 \ \tilde{\sigma_1} \ , \\ \sigma_{Mn} &= \sigma_0 \ \tilde{\sigma_2} \ , \\ \sigma_{Qc} &= \sigma_0 \ \tilde{\sigma_3} \ , \\ \sigma_{Qn} &= \sigma_0 \ \tilde{\sigma_4} \ , \end{aligned} \tag{2.67}$$

associée à la contrainte qui assure que la matrice $\boldsymbol{R}$ reste inchangée :

$$\sigma_0{}^{(N_1+N_2+N_3+N_4)} = {\sigma_{Mc}}^{N_1} \ {\sigma_{Mn}}^{N_2} \ {\sigma_{Qc}}^{N_3} \ {\sigma_{Qn}}^{N_4} \ , \tag{2.68}$$

avec $N_1 = N_{Mc}$, $N_2 = N_{Mn}$, $N_3 = N_{Qc}$ et $N_4 = N_{Qn}$. Ce qui nous permet de déduire $\frac{1}{\tilde{\sigma_1}^{N_1}} = \tilde{\sigma_2}^{N_2}\ \tilde{\sigma_3}^{N_3}\ \tilde{\sigma_4}^{N_4}$. Les paramètres qui sont optimisés par l'algorithme sont alors $\tilde{\sigma_2}$, $\tilde{\sigma_3}$ et $\tilde{\sigma_4}$, tandis que σ_0 est donné par la formule 1.17.

Pour le cas « 2 sigmas », on va utiliser la reparamétrisation suivante :

$$\begin{array}{rcl} \sigma_{Mc} & = & \sigma_M\ \tilde{\sigma_1}\ , \\ \sigma_{Mn} & = & \sigma_M\ \tilde{\sigma_2}\ , \\ \sigma_{Qc} & = & \sigma_Q\ \tilde{\sigma_3}\ , \\ \sigma_{Qn} & = & \sigma_Q\ \tilde{\sigma_4}\ , \end{array} \tag{2.69}$$

associée aux contraintes qui assurent que la matrice $\boldsymbol{R}$ reste inchangée :

$$\begin{array}{rcl} {\sigma_M}^{(N_1+N_2)} & = & {\sigma_{Mc}}^{N_1}\ {\sigma_{Mn}}^{N_2}\ , \\ {\sigma_Q}^{(N_3+N_4)} & = & {\sigma_{Qc}}^{N_3}\ {\sigma_{Qn}}^{N_4}\ . \end{array} \tag{2.70}$$

Ce qui nous permet de déduire ${\tilde{\sigma_1}}^{N_1} = \frac{1}{{\tilde{\sigma_2}}^{N_2}}$ et ${\tilde{\sigma_3}}^{N_3} = \frac{1}{{\tilde{\sigma_4}}^{N_4}}$. Les paramètres qui sont optimisés par l'algorithme sont alors $\tilde{\sigma_2}$ et $\tilde{\sigma_4}$, tandis que σ_M et σ_Q sont donnés par la formule (2.57).

La vraisemblance est à chaque fois optimisée par un algorithme de type quasi-Newton sous contraintes (Byrd et al. 1995), lequel est relancé 100 fois pour différent jeux de paramètres initiaux. Les paramètres optimaux pour les tendances (β) sont à chaque fois calculés à l'aide de leur formule analytique (2.46). L'espace des paramètres pour l'optimisation est donc de dimension 12 dans le premier cas, de dimension 11 dans le deuxième cas, de dimension 10 dans le troisième cas et de dimension 8 dans le dernier cas.

Les résultats sont représentés sur les figures ci-après.

Les figures 2.13 et 2.14 nous confirment que de concentrer un paramètre σ global permet un net gain quasi-systématique dans l'optimisation des paramètres, en temps de calcul et en nombre d'appels à la fonction de vraisemblance. Elles nous apprennent également que le fait de concentrer 2 des σ plutôt qu'un avec les formules analytiques permet statistiquement un gain de temps d'optimisation de 10-20%. Enfin, on y voit le gain très important obtenu en concentrant les 4 σ : le temps de calcul et le nombre d'appels sont approximativement divisés par 3 par rapport au cas sans concentration. On voit de plus que le gain est systématique, et pas seulement statistique, par rapport au cas à 2 σ concentrés.

De plus, la figure 2.15 montre que concentrer la vraisemblance permet d'améliorer la précision du résultat final de l'optimisation (au sens de la valeur de la vraisemblance). Dans notre cas, concentrer un seul σ global permet d'être suffisamment précis et en concentrer davantage ne permet pas d'améliorer davantage l'optimisation.

Enfin les courbes de convergence présentées sur la figure 2.16 confirment l'efficacité de la méthode qui concentre tous les σ : les courbes en violet qui sont associées à une concentration sur toutes les variances intragroupes montent plus vite vers le maximum de vraisemblance que les autres couleurs. Outre leurs lenteurs, les courbes en bleu (qui correspondent à aucune concentration) stagnent plus souvent dans des optima locaux.

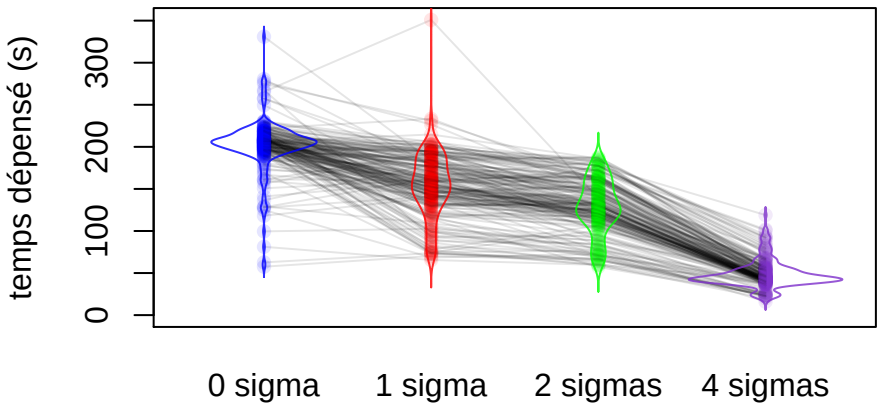


FIGURE 2.13 – Temps passé pour la maximisation de la vraisemblance, pour différentes initialisations de l'algorithme, en maximisant la log-vraisemblance complète (bleu), la log-vraisemblance concentrée pour 1 paramètre (rouge), pour 2 paramètres (vert) ou pour 4 paramètres (violet). Les lignes grises relient les temps dépensés par chaque méthode pour une même initialisation.

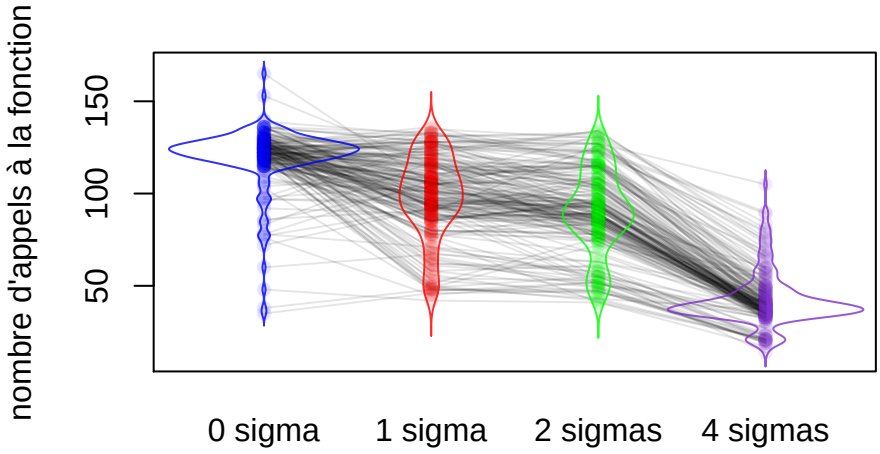


FIGURE 2.14 – Nombre d'appels effectués par l'algorithme de maximisation à la fonction calculant la log-vraisemblance (bleu), la log-vraisemblance concentrée pour 1 paramètre (rouge), la log-vraisemblance concentrée pour 2 paramètres (vert) ou la log-vraisemblance concentrée pour 4 paramètres (violet). Les lignes grises relient les nombres d'appels effectués par chaque méthode pour une même initialisation.

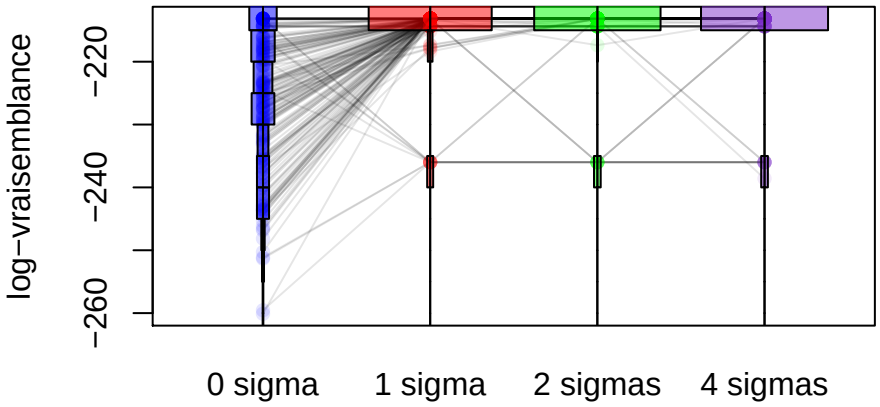


FIGURE 2.15 – Valeur finale obtenue par l'algorithme de maximisation, pour différentes initialisations, en maximisant la log-vraisemblance complète (bleu), la log-vraisemblance concentrée pour 1 paramètre (rouge), pour 2 paramètres (vert) ou pour 4 paramètres (violet). Les lignes grises relient les valeurs trouvées par chaque méthode pour une même initialisation.

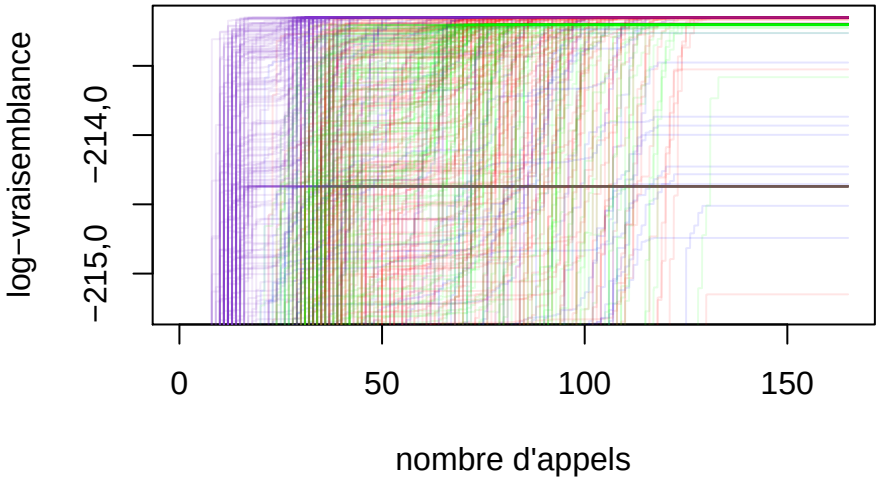


FIGURE 2.16 – Convergence de l'algorithme de maximisation (meilleure valeur trouvée au bout de tant d'appels), pour différentes initialisations, en maximisant la log-vraisemblance complète (bleu), la log-vraisemblance concentrée pour 1 paramètre (rouge), pour 2 paramètres (vert) ou pour 4 paramètres (violet). Le nombre d'appels à la fonction-objectif ne tient pas compte des appels faits pour évaluer les dérivées, mais ces appels ne feront qu'augmenter l'avantage des méthodes de concentration.

2.3 Hyperkrigeage

Le cokrigeage permet de prendre en compte des grandeurs supplémentaires, mais d'autres approches peuvent exister pour créer des modèles introspectifs. Dans cette section, nous allons présenter ce qu'on va appeler « l'hyperkrigeage ». Les cas d'application de l'hyperkrigeage sont plus restreints que ceux du cokrigeage : les grandeurs doivent être hiérarchisées et toutes les observations mesurées en haute fidélité doivent également avoir été mesurées aux niveaux de fidélité inférieurs. Ces cas d'applications correspondent par exemple à des codes enchaînés qui ne peuvent pas être lancés indépendamment, ou à des valeurs intermédiaires obtenues dans un même calcul.

Le parti pris de l'hyperkrigeage est de construire des krigeages emboîtés. À chaque niveau correspondra un processus gaussien, indépendant des autres, mais à partir du deuxième, l'espace d'entrée du processus sera augmenté d'une dimension correspondant à la grandeur précédente. Cette augmentation de la dimension est à l'origine de l'appellation « hyperkrigeage » :

$$\begin{array}{rcl} Z^{(n_g)}(x) & \sim & GP(x) \ , \\ Z^{(n_g-1)}(x, y^{(n_g)}) & \sim & GP(x, y^{(n_g)}) \ , \\ & \cdots & \\ Z^{(1)}(x, y^{(2)}) & \sim & GP(x, y^{(2)}) \ . \end{array} \tag{2.71}$$

Nous rappelons que dans nos notations, la sortie de plus haute fidélité est notée $y^{(1)}$ et celle de plus basse fidélité $y^{(n_g)}$. De tels krigeages emboîtés sont étudiés dans (Marque-Pucheu, Perrin, and Garnier 2017) (sans appellation spécifique du modèle). La notion de *« deep-GP »* couvre également des krigeages emboîtés, mais dont les niveaux intermédiaires sont virtuels et ne correspondent pas à des grandeurs mesurables (Salimbeni and Deisenroth 2017 ; Dutordoir 2017 ; Damianou and Lawrence 2013). Cette construction de modèle est assez intuitive pour représenter des fonctions composées, néanmoins elle présente l'inconvénient que les prédictions ne suivent plus des distributions gaussiennes. En effet si $Z^{(1)}(x, y^{(2)})$ est bien un processus gaussien, on ignore la valeur prise par $y^{(2)}$ en x et la prédiction du niveau (1) en x doit tenir compte de cette incertitude. On note avec un tilde $\tilde{Z}^{(i)}(x)$ le processus (non gaussien) au niveau i ne dépendant que de x, et résultant du modèle d'emboîtement. Dans la suite, on emploie les notations $\mathrm{P}[A]$ pour désigner la probabilité d'un évènement A, et $\mathrm{dP}[A = a]$ pour désigner la densité de probabilité en a d'une variable aléatoire continue A.

Illustrons ceci par un cas à deux niveaux. Calculons la distribution prédite par le modèle en une coordonnée x. Pour le niveau (2), $Z^{(2)}(\cdot)$ est un processus gaussien et $Z^{(2)}(x)$ suit bien une loi normale de moyenne $\mu^{(2)}(x)$ et de variance $\sigma^{2^{(2)}}(x)$ comme décrit en partie I :

$$\begin{array}{rcl} Z^{(2)}(x) & \sim & \mathcal{N}(\mu^{(2)}(x), \sigma^{2^{(2)}}(x)) \ , \\ \mu^{(2)}(x) & = & m(x^{(2)}) + k(x^{(2)}, X^{(2)}) K^{(2)-1}(Y^{(2)} - m(X^{(2)})) \ , \\ \sigma^{2^{(2)}}(x) & = & C(x^{(2)}, x^{(2)}) = k(x^{(2)}, x^{(2)}) - k(x^{(2)}, X^{(2)}) K^{(2)-1} k(X^{(2)}, x^{(2)}) \ . \end{array} \tag{2.72}$$

En revanche pour la grandeur d'intérêt (le niveau (1)), l'une des dimensions d'entrée du processus gaussien est incertaine, et la distribution prédite en x (correspondant à la probabilité de présence de la vraie valeur) ne suit plus une loi normale. Par la formule des probabilités totales (pour le cas continu), on peut calculer :

$$\begin{array}{rcl} \mathrm{P}\left[\tilde{Z}^{(1)}(x) < z_1\right] & = & \int_{-\infty}^{+\infty} \mathrm{P}\left[Z^{(1)}(x, Z^{(2)}(x)) < z_1 \ \& \ Z^{(2)}(x) = y\right] \mathrm{d}y \\ & = & \int_{-\infty}^{+\infty} \mathrm{P}\left[Z^{(1)}(x, Z^{(2)}(x)) < z_1 \mid Z^{(2)}(x) = y\right] \mathrm{dP}[Z^{(2)}(x) = y] \, \mathrm{d}y \\ & = & \int_{-\infty}^{+\infty} \mathrm{P}\left[Z^{(1)}(x, y) < z_1\right] \frac{1}{\sigma^{(2)}(x)} \phi\left(\frac{y - \mu^{(2)}(x)}{\sigma^{(2)}(x)}\right) \mathrm{d}y \\ & = & \int_{-\infty}^{+\infty} \Phi\left(\frac{z_1 - \mu^{(1)}(x,y)}{\sigma^{(1)}(x,y)}\right) \frac{1}{\sigma^{(2)}(x)} \phi\left(\frac{y - \mu^{(2)}(x)}{\sigma^{(2)}(x)}\right) \mathrm{d}y \ , \end{array} \tag{2.73}$$

avec $\phi(\cdot)$ et $\Phi(\cdot)$ la densité de distribution et la fonction de répartition de la loi normale centrée réduite. Cette intégrale ne peut pas être simplifiée davantage car $\mu^{(1)}(x, y)$ et $\sigma^{(1)}(x, y)$ peuvent varier de façon quelconque avec y. Notons que la distribution suit à nouveau une loi normale si on a une information sur le niveau supérieur :

$$\begin{aligned} \mathrm{P}\big[\tilde{Z}^{(1)}(x) < z_1 \mid \tilde{Z}^{(2)}(x) = z_2\big] &= \mathrm{P}\big[Z^{(1)}(x, z_2) < z_1\big] \\ &= \Phi\left(\tfrac{z_1 - \mu^{(1)}(x,z_2)}{\sigma^{(1)}(x,z_2)}\right) . \end{aligned} \tag{2.74}$$

On a une formule similaire pour la densité de probabilité d'une prédiction, ainsi que pour la moyenne de prédiction :

$$\begin{aligned} \mathrm{dP}\big[\tilde{Z}^{(1)}(x) = z_1\big] &= \tfrac{\partial}{\partial z_1}\left(\mathrm{P}\big[\tilde{Z}^{(1)}(x) < z_1\big]\right) \\ &= \int_{-\infty}^{+\infty} \tfrac{1}{\sigma^{(1)}(x,y)} \phi\left(\tfrac{z_1 - \mu^{(1)}(x,y)}{\sigma^{(1)}(x,y)}\right) \tfrac{1}{\sigma^{(2)}(x)} \phi\left(\tfrac{y - \mu^{(2)}(x)}{\sigma^{(2)}(x)}\right) \mathrm{d}y \ , \end{aligned} \tag{2.75}$$

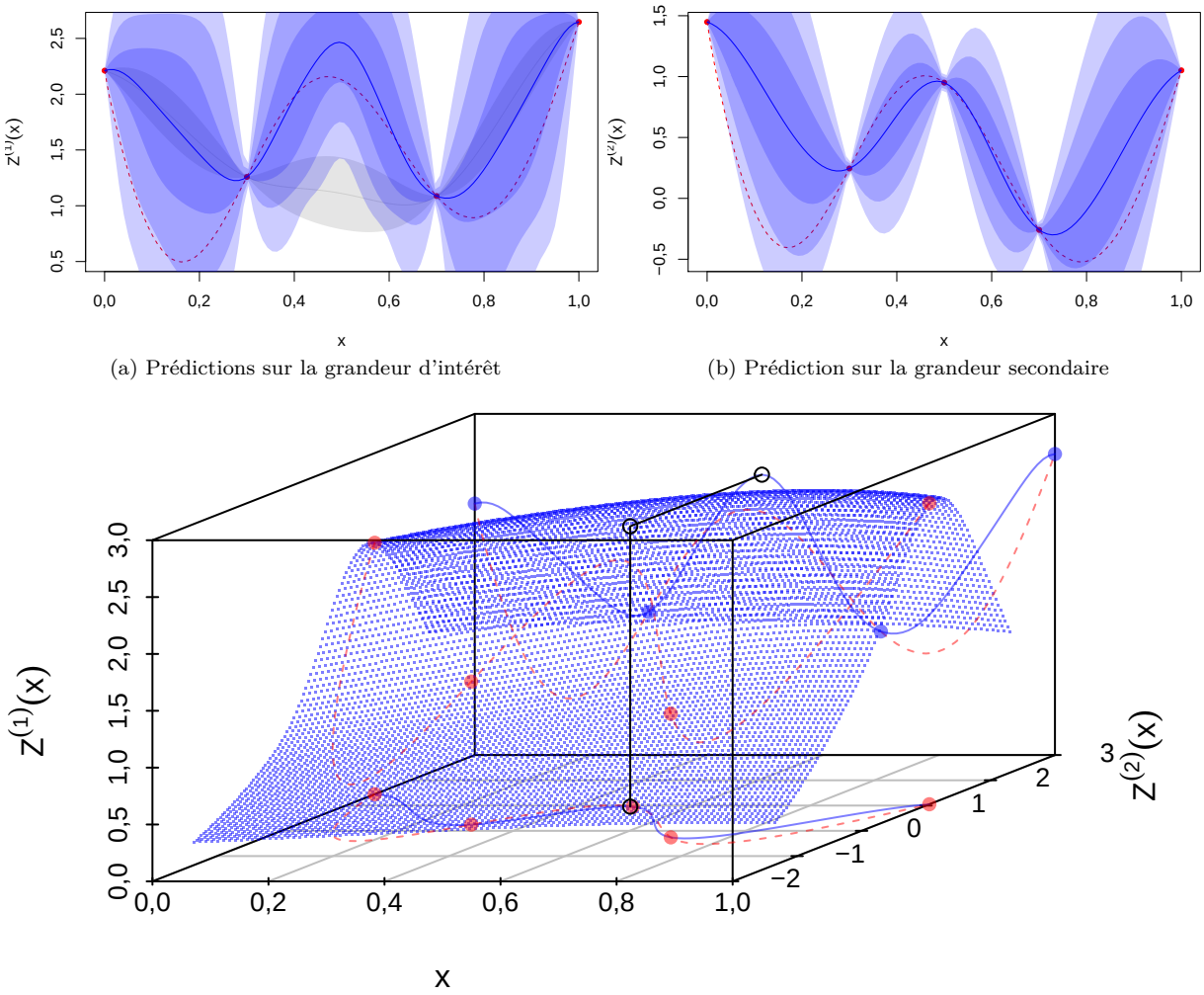
$$\begin{aligned} \tilde{\mu}^{(1)}(x) &= \int_{-\infty}^{+\infty} z \, \mathrm{dP}\big[\tilde{Z}^{(1)}(x) = z\big] \, \mathrm{d}z \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} z \tfrac{1}{\sigma^{(1)}(x,y)} \phi\left(\tfrac{z - \mu^{(1)}(x,y)}{\sigma^{(1)}(x,y)}\right) \tfrac{1}{\sigma^{(2)}(x)} \phi\left(\tfrac{y - \mu^{(2)}(x)}{\sigma^{(2)}(x)}\right) \mathrm{d}y \, \mathrm{d}z \\ &= \int_{-\infty}^{+\infty} \mu^{(1)}(x,y) \tfrac{1}{\sigma^{(2)}(x)} \phi\left(\tfrac{y - \mu^{(2)}(x)}{\sigma^{(2)}(x)}\right) \mathrm{d}y \ . \end{aligned} \tag{2.76}$$

La variance d'hyperkrigeage présente moins d'intérêt dans la mesure où elle ignore la potentielle asymétrie de la distribution (et dans un cadre de recherche d'optimum par exemple, cette asymétrie est importante). Les fonctions quantiles, plus intéressantes, gagnent également en complexité, de même que les formules de probabilités jointes de plusieurs prédictions simultanées. Ces formules complexes ne sont pas présentées dans ce document. La présence d'intégrales qui ne se réduisent pas à des formules analytiques rend l'usage des modèles d'hyperkrigeage plus difficile à mettre en œuvre que celui des modèles de cokrigeage. Cependant les modèles d'hyperkrigeage ne sont pas sans intérêt. D'une part, lorsque les hypothèses sont les bonnes, que l'on cherche effectivement à modéliser une composition de fonctions, un modèle d'hyperkrigeage est le plus pertinent. D'autre part, le cokrigeage vu précédemment possède des limites. Pour fonctionner efficacement, il a besoin d'une corrélation linéaire entre les grandeurs. Or en pratique, il arrive pour beaucoup de phénomènes physiques que deux grandeurs fortement corrélées le soit de manière non linéaire ou linéairement par morceaux (avec des coefficients de linéarité qui diffèrent selon la portion du domaine considérée). Un modèle de cokrigeage peut ainsi être rendu inefficace simplement en considérant un domaine trop grand, la relation entre les grandeurs changeant de comportement. Un modèle d'hyperkrigeage ne présente pas cet inconvénient. Le paramètre de portée estimé sur la dimension correspondant à l'autre grandeur ($y^{(2)}$ dans l'exemple à deux composants ci-dessus) va décrire à quel point les grandeurs conservent un même comportement vis-à-vis l'une de l'autre.

Cela justifie que l'on s'intéresse à l'hyperkrigeage. La section suivante présente les approximations qui peuvent être faites afin de retrouver des formules analytiques qui approcheront les vraies valeurs à la précision souhaitée. Bien sûr une intégrale sous une gaussienne peut également être calculée par n'importe quelle méthode numérique éprouvée mais cela a un coût (en temps de calcul) qui est contre-productif par rapport à l'intérêt d'un métamodèle (qui réside justement dans la possibilité d'obtenir un grand nombre de prédictions très rapidement). Les statistiques complexes, comme les fonctions quantiles, peuvent également être calculées par des méthodes de Monte-Carlo car la simulation d'une trajectoire d'un modèle d'hyperkrigeage est aussi simple que de simuler des trajectoires de krigeage successives. Bien que facile à mettre en œuvre, simuler un nombre suffisant de trajectoires peut parfois se révéler trop coûteux. En effet, si la complexité pour simuler une trajectoire est simplement proportionnelle au nombre de niveaux, le nombre de trajectoires suffisantes pour obtenir une bonne statistique augmente de façon exponentielle. Si on a besoin de $o(s)$ simulations pour couvrir une gaussienne avec la précision souhaitée, une trajectoire décrite sur n points nécessitera $o(s^n)$ simulations dans le pire cas (corrélations nulles), et une trajectoire d'hyperkrigeage à n_g niveaux nécessitera alors $o(s^{n\ n_g})$ simulations. On travaille en effet sur une variable aléatoire multivariée de dimension $n \times n_g$. Dans la pratique, on essaie de travailler sur des statistiques ponctuelles ($n = 1$), éventuellement sur des couples de points ($n = 2$), et lorsque le nombre de niveaux est faible (souvent $n_g = 2$ ou 3), le recours à des méthodes de Monte-Carlo pour utiliser l'hyperkrigeage reste une solution attirante lors des premières implémentations destinées à valider l'intérêt de la méthode.

2.3.1 Approximations du modèle d'hyperkrigeage

Une intégrale sous une gaussienne présente des particularités qui permettent de calculer rapidement des approximations à la précision souhaitée. Si on considère une gaussienne de moyenne μ et de variance σ^2, il est aisé de démontrer (par intégration par partie) que :



(a) Prédictions sur la grandeur d'intérêt

(b) Prédiction sur la grandeur secondaire

(c) Grandeur d'intérêt vue comme une fonction de x et de la grandeur secondaire

FIGURE 2.17 – Exemple 1D à 2 niveaux de modèle d'hyperkrigeage. La grandeur d'intérêt se trouve en haut à gauche, la grandeur secondaire est en haut à droite. La ligne pointillée rouge correspond à la vraie fonction qu'on cherche à modéliser. Les points rouges sont les points connus. En bleu sont représentés la moyenne de prédiction et les intervalles de confiance à 1, 2 ou 3 sigmas. En gris est représenté le modèle de krigeage (à 1 sigma) classique. Le graphe en 3D représente le modèle de krigeage construit comme si la grandeur d'intérêt était fonction des entrées et de la grandeur secondaire (seules les moyennes sont représentées). Le graphe en haut à gauche se construit par composition de la nappe 3D avec le graphe en haut à droite.

$$\int_{-\infty}^{+\infty} \left(\frac{y-\mu}{\sigma}\right)^n \frac{1}{\sigma}\phi\left(\frac{y-\mu}{\sigma}\right) \mathrm{d}y = \begin{cases} 0 & \text{si } n = 2k+1 \text{ (impair)} \\ \frac{(2k)!}{k!\ 2^k} & \text{si } n = 2k \text{ (pair)} . \end{cases} \tag{2.77}$$

Par ailleurs, rappelons la formule de la série de Taylor d'une fonction $f(y)$:

$$f(y) \overset{y\simeq\mu}{=} f(\mu) + \sum_{n=1}^{+\infty} \frac{(y-\mu)^n}{n!} \frac{\partial^n f}{\partial y^n}(\mu) . \tag{2.78}$$

Ce qui nous donne une formule sous forme de série infinie pour l'intégrale d'une fonction sous une gaussienne :

$$\int_{-\infty}^{+\infty} f(y) \frac{1}{\sigma}\phi\left(\frac{y-\mu}{\sigma}\right) \mathrm{d}y = f(\mu) + \sum_{k=1}^{+\infty} \frac{\sigma^{2k}}{k!\ 2^k} \frac{\partial^{2k} f}{\partial y^{2k}}(\mu) . \tag{2.79}$$

On pourra donc approximer l'intégrale en tronquant la série de Taylor. La convergence sera d'autant plus rapide que σ^2 est faible. Ces formules vont nous permettre d'approximer convenablement les modèles d'hyperkrigeage, sous réserve d'utiliser un noyau suffisamment dérivable. Un noyau n fois dérivable va nous permettre d'obtenir les n premières dérivées partielles par rapport à y de $\mu^{(1)}(x,y)$ et $\sigma^{(1)}(x,y)$ (par les formules (2.72)). Ces dérivées sont nécessaires pour avoir les dérivées de $\Phi\left(\frac{z_1-\mu^{(1)}(x,y)}{\sigma^{(1)}(x,y)}\right)$ et approximer correctement les équations (2.73) et (2.75).

De la même manière, on pourra approximer d'autres statistiques intéressantes comme la moyenne d'hyperkrigeage $\tilde{\mu}^{(1)}(x)$ (2.76) ou l'*Expected Improvement* (vu au chapitre 1, détaillé à la section 3.3) pour un besoin de recherche d'optimum :

$$\begin{aligned} \tilde{\mu}^{(1)}(x) &= \int_{-\infty}^{+\infty} \mu^{(1)}(x,y) \tfrac{1}{\sigma^{(2)}(x)} \phi\left(\tfrac{y-\mu^{(2)}(x)}{\sigma^{(2)}(x)}\right) \mathrm{d}y \\ &= \mu^{(1)}(x,\mu^{(2)}(x)) + \textstyle\sum_{k=1}^{+\infty} \frac{\sigma^{(2)}(x)^{2k}}{k!\ 2^k} \frac{\partial^{2k}\mu^{(1)}}{\partial y^{2k}}(x,\mu^{(2)}(x)) \\ &= \mu^{(1)}(x,\mu^{(2)}(x)) + \tfrac{\sigma^{(2)2}(x)}{2} \tfrac{\partial^2\mu^{(1)}}{\partial y^2}(x,\mu^{(2)}(x)) + \tfrac{\sigma^{(2)4}(x)}{8} \tfrac{\partial^4\mu^{(1)}}{\partial y^4}(x,\mu^{(2)}(x)) + \ \ldots \end{aligned} \tag{2.80}$$

Les quantiles peuvent être approximés également, par exemple en utilisant la relation suivante entre une fonction $f(y)$ et sa réciproque $f_r(x)$:

$$\frac{\mathrm{d}f_r(x)}{\mathrm{d}x} = \left(\frac{\mathrm{d}f(y)}{\mathrm{d}y}(y = f_r(x))\right)^{-1} . \tag{2.81}$$

Ce calcul ne sera pas détaillé dans ce mémoire. La mise en place de ces approximations n'a pas été menée jusqu'au bout durant cette thèse, la priorité ayant été donnée à d'autres applications. Lors de mes tests et applications, j'ai employé une méthode Monte-Carlo pour approximer l'intégrale. L'exploitation du modèle d'hyperkrigeage s'en est trouvée très ralentie.

On remarque tout de même que l'approximation au premier ordre, très simple à mettre en œuvre, néglige l'incertitude sur le niveau bas (l'inconnue y est remplacée par $\mu^{(2)}(x)$). Si le niveau 2 est très bien connu (si le premier code est très peu cher et qu'on a échantillonné finement tout le domaine), alors on peut utiliser un « hyperkrigeage léger ». L'hyperkrigeage léger est à nouveau un processus gaussien et tous les résultats connus peuvent s'y appliquer. L'équation (2.73) devient en effet :

$$\mathrm{P}\left[\tilde{Z}^{(1)}(x) < z_1\right] = \Phi\left(\frac{z_1 - \mu^{(1)}(x,\mu^{(2)}(x))}{\sigma^{(1)}(x,\mu^{(2)}(x))}\right) . \tag{2.82}$$

2.3.2 Hyperkrigeage croisé ou krigeage intriqué

Il est possible d'envisager des relations plus complexes entre les niveaux, et les modèles d'hyperkrigeage pourront encore être utilisés. Par exemple pour 2 niveaux non-hiérarchisés, on va définir les processus gaussiens interdépendants :

$$\begin{array}{rcl} Z^{(2)}(x,y^{(1)}) & \sim & GP(x,y^{(1)})\ , \\ Z^{(1)}(x,y^{(2)}) & \sim & GP(x,y^{(2)})\ . \end{array} \tag{2.83}$$

Cela conduit à des formules encore plus complexes pour calculer les prédictions du modèle. Ces prédictions n'ont pas plus de raisons de suivre des lois normales que dans le cas de l'hyperkrigeage hiérarchique :

$$\begin{array}{rcl} \mathrm{dP}\big[\tilde{Z}^{(1)}(x)=z_1\big] & = & \left(\frac{1}{\mathrm{dP}\left[\tilde{Z}^{(1)}(x)=z_1\right]}\right)^{-1} \\ & = & \left(\frac{1}{\mathrm{dP}\left[\tilde{Z}^{(1)}(x)=z_1\right]}\ \int_{-\infty}^{+\infty}\mathrm{dP}\big[\tilde{Z}^{(2)}(x)=z_2\big]\,\mathrm{d}z_2\right)^{-1} \\ & = & \left(\int_{-\infty}^{+\infty}\frac{\mathrm{dP}\left[\tilde{Z}^{(2)}(x)=z_2\right]}{\mathrm{dP}\left[\tilde{Z}^{(1)}(x)=z_1\right]}\,\mathrm{d}z_2\right)^{-1} \\ \textit{(th. de Bayes)} & = & \left(\int_{-\infty}^{+\infty}\frac{\mathrm{dP}\left[\tilde{Z}^{(2)}(x)=z_2\ \mid\ \tilde{Z}^{(1)}(x)=z_1\right]}{\mathrm{dP}\left[\tilde{Z}^{(1)}(x)=z_1\ \mid\ \tilde{Z}^{(2)}(x)=z_2\right]}\,\mathrm{d}z_2\right)^{-1} \\ & = & \left(\int_{-\infty}^{+\infty}\frac{\mathrm{dP}\left[Z^{(2)}(x,z_1)=z_2\right]}{\mathrm{dP}\left[Z^{(1)}(x,z_2)=z_1\right]}\,\mathrm{d}z_2\right)^{-1} \\ & = & \left(\int_{-\infty}^{+\infty}\frac{\frac{1}{\sigma^{(2)}(x,z_1)}\phi\left(\frac{z_2-\mu^{(2)}(x,z_1)}{\sigma^{(2)}(x,z_1)}\right)}{\frac{1}{\sigma^{(1)}(x,z_2)}\phi\left(\frac{z_1-\mu^{(1)}(x,z_2)}{\sigma^{(1)}(x,z_2)}\right)}\,\mathrm{d}z_2\right)^{-1}\ . \end{array} \tag{2.84}$$

Des approximations sont encore possibles, toujours en exploitant le développement de Taylor. Des modèles d'hyperkrigeage sont envisageables pour des situations encore plus compliquées, par exemple avec certains niveaux en parallèle et d'autres en série. Néanmoins cela dépasse le cadre de cette thèse et nous n'avons pas développé davantage ces possibilités.

2.4 Comparaison des modèles

2.4.1 Présentation des cas-tests

Dans le but d'évaluer l'efficacité des modèles introspectifs présentés plus haut nous allons les appliquer à la fonction de Branin (couramment utilisée pour évaluer des méthodes d'optimisation), modifiée de trois façons pour l'adapter à des problèmes à plusieurs niveaux. Tout d'abord, la fonction est légèrement changée pour n'avoir qu'un seul minimum global en $(0,543;0,150)$ et 2 minima locaux :

$$\begin{array}{rcl} \overline{x_1} & = 15x_1-5 \quad , \quad \overline{x_2}=15x_2 & \\ y^{(1)}(x) & = \left(\overline{x_2}-\frac{5\overline{x_1}^2}{4\pi^2}+\frac{5\overline{x_1}}{\pi}-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos(\overline{x_1})+11-\exp(-\frac{(\overline{x_1}-0{,}5)^2}{15})\ . \end{array} \tag{2.85}$$

La fonction est représentée sur la figure 2.18. La fonction de Branin modifiée va ensuite être altérée de trois manières afin d'émuler des simulateurs de types *mesh based* (à base de maillage), Monte-Carlo et *time step* (pas-à-pas), qui correspondent à trois comportements différents de convergence classiques de simulations.

Les simulations *mesh based* (comme les solveurs basés sur la méthode des éléments finis) ont une convergence généralement régulière avec la taille du maillage. Nous admettons qu'elles génèrent des résultats qui évoluent continûment avec la finesse des détails et qui tendent asymptotiquement vers la fonction-objectif. Bien sûr, la régularité de cette évolution suivant le maillage est généralement altérée par d'éventuelles instabilités numériques dues aux équations résolues, on encore par des effets de discrétisation lorsque les mailles sont inadaptées. Ici, cette convergence sera émulée par une somme pondérée entre la fonction-objectif et un polynôme quadratique vers lequel la simulation tend lorsque le nombre de nœuds diminue. La pondération utilisée suit le logarithme du nombre de nœuds du maillage (voir figure 2.19).

Les simulations de type Monte-Carlo convergent typiquement vers la fonction-objectif polluée par un bruit blanc additif. Ce bruit blanc est d'autant plus faible que la taille de l'échantillonnage est importante, et sa variance décroit comme l'inverse de la taille de l'échantillon. Ce bruit blanc n'a aucune corrélation spatiale.

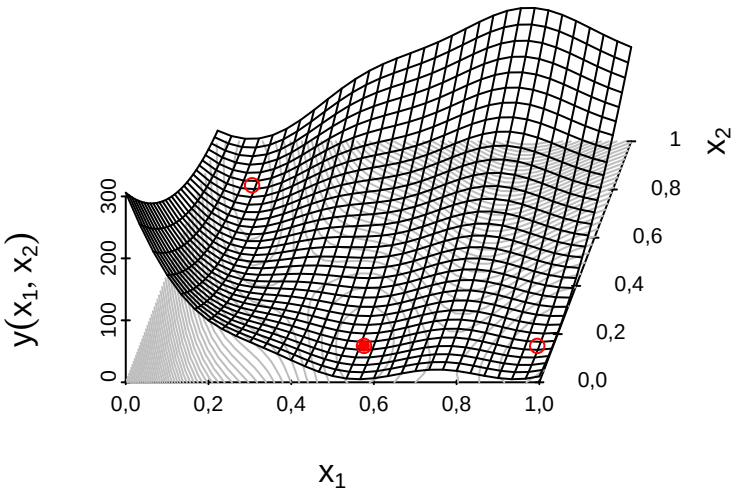
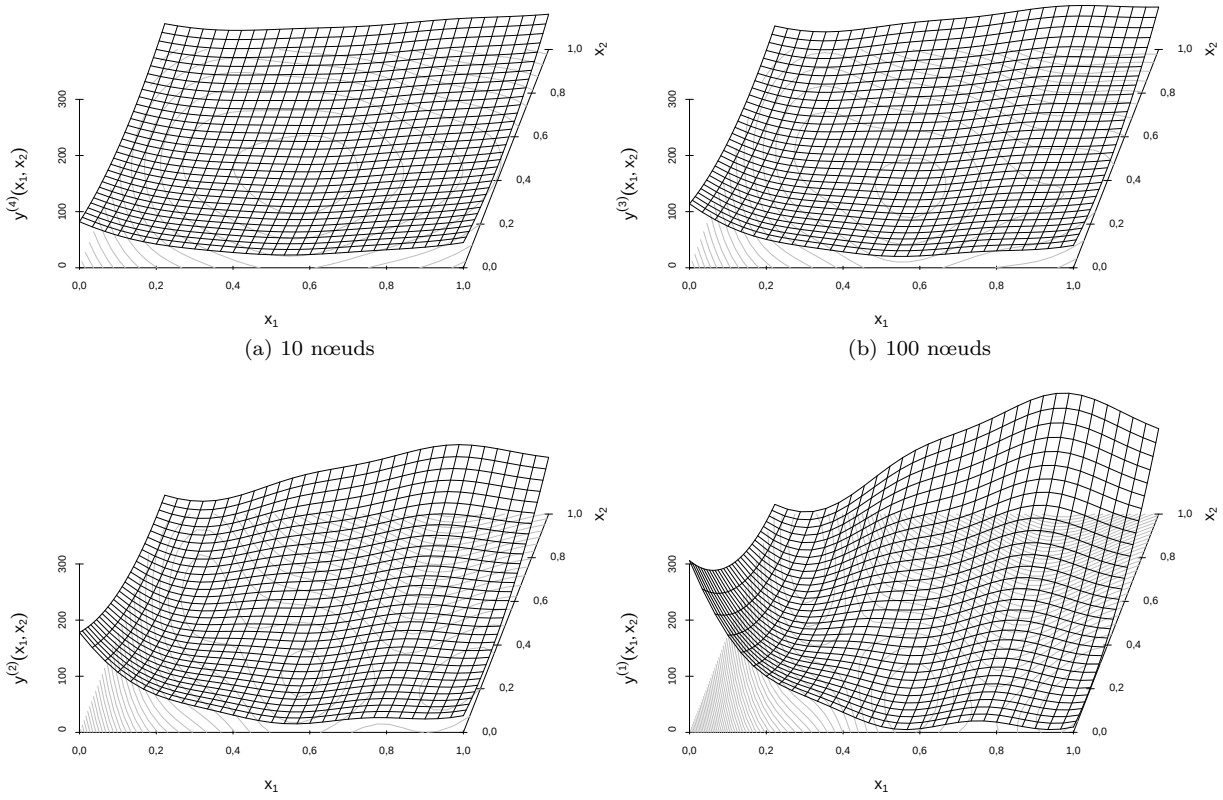


FIGURE 2.18 – Fonction de Branin modifiée (formule en 2.85) utilisée en tant que simulation haute fidélité. Le cercle plein est l'optimum global, les cercles vides sont les optima locaux.

Les effets de ce type de multifidélité sont représentés en figure 2.20. Cette figure doit être comprise comme un échantillon des fonctions avec les différents niveaux de bruit. Une réévaluation d'un point conduirait à une valeur différente, mais générée par le bruit blanc gaussien.

Dans les deux exemples précédents, le niveau de fidélité correspond au degré de convergence de la simulation. Il existe également des solveurs par itérations spatiales ou temporels (dits *time step*). Il s'agit de simulations dans lesquelles le résultat d'une étape donne les conditions limites de la suivante (comme par exemple un solveur multidisciplinaire à temps discret). De telles simulations pas-à-pas ne convergent pas vers une solution asymptotique, dans le sens où chaque étape ne produit pas simplement le résultat final altéré d'une « erreur » qui décroîtrait à chaque étape. Néanmoins, le comportement intrinsèquement markovien de telles simulations se prête bien à l'usage de modèles introspectifs. Le troisième cas-test analytique est ainsi construit par un processus autorégressif à 4 étapes (AR1) dont la dernière itération est la fonction-objectif « haute fidélité » (voir figure 2.21).



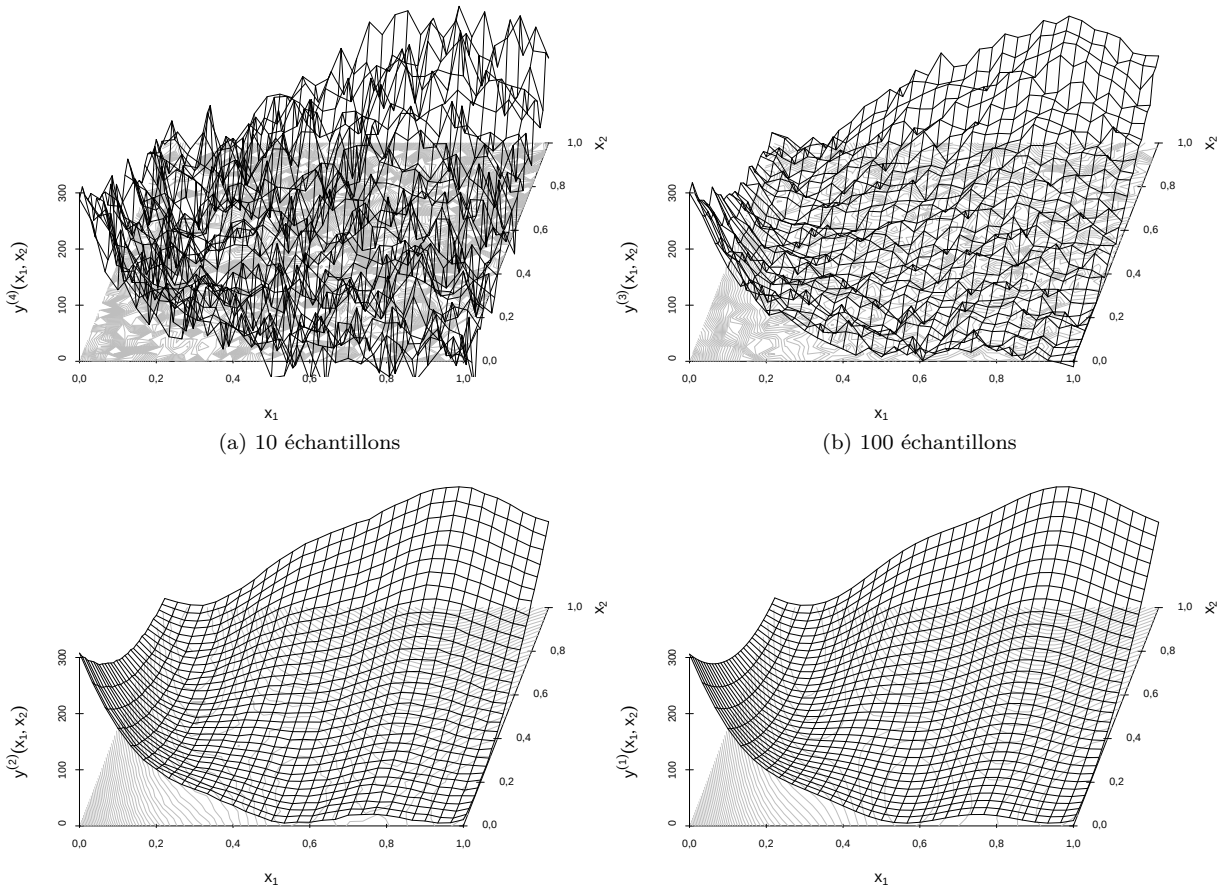
(a) 10 nœuds

(b) 100 nœuds

(c) 10^4 nœuds

(d) Fonction-objectif (10^8 nœuds)

FIGURE 2.19 – Ersatz de simulations multifidélité *mesh based* par déformation progressive de la fonction-objectif (Branin modifiée).



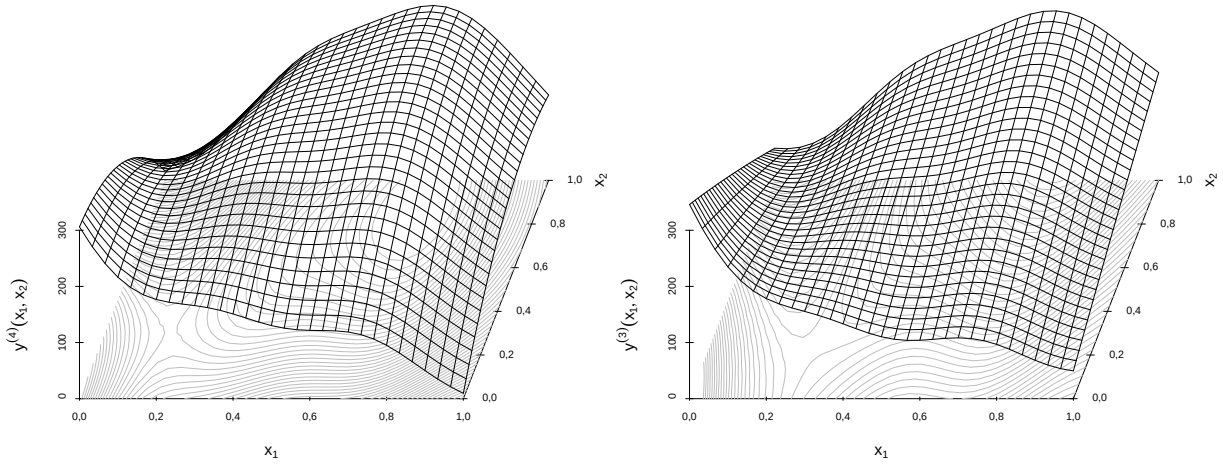
(a) 10 échantillons

(b) 100 échantillons

(c) 10^4 échantillons

(d) Fonction-objectif (10^8 échantillons)

FIGURE 2.20 – Ersatz de simulations multifidélité Monte-Carlo par adjonction de bruit blanc à la fonction-objectif (Branin modifiée).



(a) Étape 1 (b) Étape 2

(c) Étape 3 (d) Fonction-objectif (étape 4)

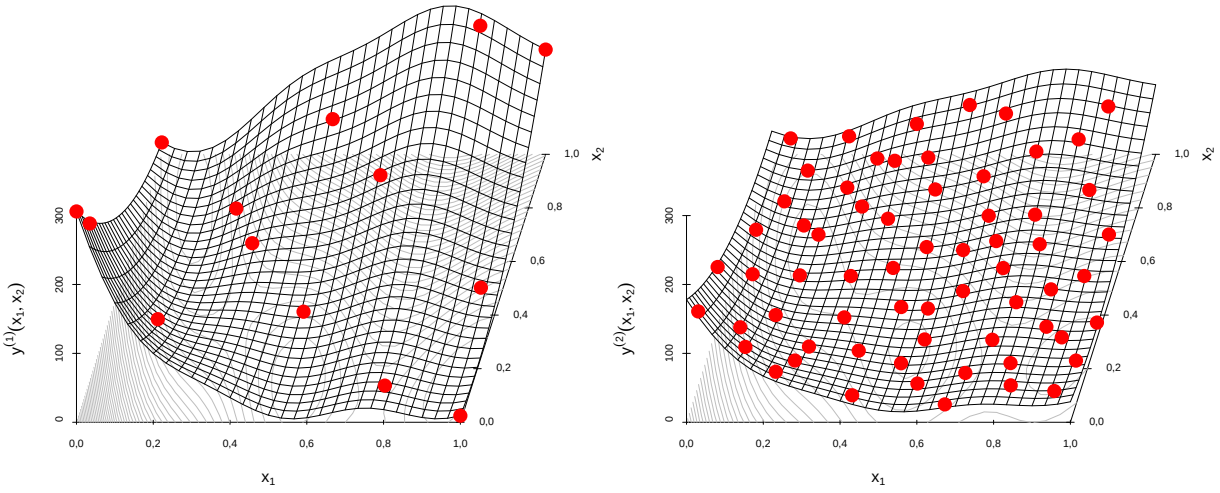
FIGURE 2.21 – Illustration d'un processus autorégressif en 4 étapes constituant notre troisième cas-test.

2.4.2 Résultats sur le cas-test *mesh based*

On compare à présent les différents modèles sur le cas-test *mesh based* présenté en figure 2.19. Le maillage à 10^8 nœuds est considéré comme notre fonction-objectif. Le modèle de krigeage est ajusté sur la base d'un plan d'expériences à 14 points (10 points placés selon un hypercube latin (LHS), et 4 autres dans les coins du domaine) sur cette fonction-objectif. Deux modèles de cokrigeage, un symétrique et un markovien, tels que présentés précédemment, sont ajustés sur ce même plan d'expériences augmenté de 64 points en LHS sur le niveau de fidélité du maillage à 10^4 nœuds (considéré comme étant bien moins coûteux à évaluer). Un modèle d'hyperkrigeage est également ajusté sur ces mêmes données. Les modèles de krigeage et de cokrigeage incluent un paramètre de *nugget* (effet de pépite), estimé d'après le plan d'expériences, et relatif au paramètre d'échelle. L'effet pépite absolu, pour un niveau (a), correspond à la valeur estimée, $\boldsymbol{\tau}$, fois le paramètre d'échelle pour le niveau considéré :

$$\tau^{(a)} = \boldsymbol{\tau}\ \sigma^{(a)}\ . \tag{2.86}$$

Le modèle d'hyperkrigeage utilise deux processus gaussiens indépendants, $Z^{(2)}(x_1, x_2)$ pour modéliser le niveau à 10^4 nœuds, et $Z^{(1)}(x_1, x_2, y^{(2)})$ pour modéliser la fonction-objectif, et estime un paramètre de *nugget* pour chacun d'eux. Dans les cas examinés ici, le *nugget* est généralement estimé à des valeurs proches de la précision machine. Un paramètre de tendance constante est également estimé pour tous les modèles et pour chacun des niveaux, suivant les formules données en section 2.2.7. Le modèle d'hyperkrigeage La figure 2.22 présente un exemple des plans d'expériences utilisés.



(a) Fonction-objectif (10^8 nœuds) (b) 10^4 nœuds

FIGURE 2.22 – Exemple de plan d'expériences utilisé pour ajuster les modèles de krigeage, de cokrigeage et d'hyperkrigeage, pour le cas *mesh based*.

La construction des modèles est répétée 100 fois, pour des plans d'expériences différents. On mesure alors l'erreur quadratique moyenne (MSE, pour *Mean Square Error*) de la moyenne de prédiction par rapport à la fonction-objectif, sur 10 000 points pris au hasard, uniformément, dans le domaine.

$$\begin{aligned} MSE_{théorique} &= \int_{x\in\mathbb{D}} \left(\mu^{(1)}(x) - y^{(1)}(x)\right)^2 \mathrm{d}x \\ MSE_{empirique} &= \sum_{x_l\in\mathbb{X}} \left(\mu^{(1)}(x_l) - y^{(1)}(x_l)\right)^2 \qquad \text{avec } \mathbb{X} \sim {U_{\mathbb{D}}}^{10000} \end{aligned} \tag{2.87}$$

Les 10 000 points de test sont différents à chaque nouveau plan d'expériences, mais sont les mêmes pour tous les modèles. La figure 2.23 présente les résultats avec une échelle logarithmique. Pour le modèle de krigeage, on trouve une valeur médiane de la MSE de 337,318, et une valeur moyenne de 495,096. Le cokrigeage symétrique améliore grandement la prédiction et sa MSE médiane descend à 47,05 (moyenne à 85,268). Le modèle de cokrigeage markovien est encore un peu meilleur avec une MSE médiane de 19,292 (moyenne à 61,112). L'hypergrigeage est en revanche moins bon, tout en restant meilleur que le krigeage, et sa MSE une médiane vaut 168,711 (moyenne à 208,491).

Notons que l'estimation des paramètres du modèle peut parfois se tromper lourdement. La MSE d'un prédicteur grossier, prédisant par le moyenne des données de niveau 1, tourne autour de 2900, pour un

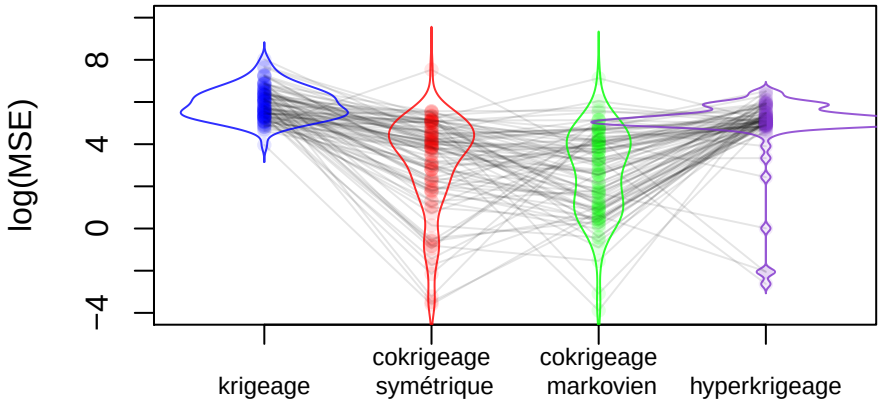


FIGURE 2.23 – Logarithme de la MSE (erreur quadratique moyenne) pour le cas *mesh based* pour les différents modèles considérés (de gauche à droite : krigeage (bleu), cokrigeage symétrique (rouge), cokrigeage markovien (vert), hyperkrigeage (violet)). La MSE est mesurée sur 10 000 points aléatoires de la fonction-objectif, uniformément répartis. Une MSE faible indique que le modèle prédit une valeur proche de la vraie valeur. Les MSE sont mesurées pour 100 plans d'expériences différents. Les lignes colorés représentent les densités. Les traits gris relient les MSE d'un même plan d'expériences.

logarithme approchant 8,0 (variable selon le plan d'expérience). Les modèles introspectifs qui atteignent ces valeurs ont estimé leurs paramètres à de mauvaises valeurs. Quelques uns ont même une MSE de l'ordre de 10000, leurs paramètres sont très mal estimés, et même leurs paramètres de tendances ont des valeurs aberrantes (à cause d'instabilités numériques dans leurs calculs). On a exclu de l'étude de la MSE ces derniers modèles, en l'occurrence sur les 100 répétitions : 2 modèles de krigeage et 1 modèle de cokrigeage markovien.

On peut vouloir comparer les modèles à plan d'expériences fixé. La figure 2.23 relie le logarithme de la MSE de chaque modèle pour un même plan d'expérience. Ici, il n'y a pas d'effet systématique notable. Les comparaisons deux à deux des modèles sont pratiquement les mêmes selon que l'on compare pour un même plan d'expériences ou pour deux plans d'expériences quelconques. Nous avons choisi de ne présenter que les données de comparaison à plan d'expériences identique. Le tableau ci-dessous résume les écarts de MSE entre les différents modèles pour chacun des plans d'expériences utilisés. Pour chaque paire de modèle, on affiche les quantiles et la moyenne des écarts de MSE. Un écart positif indique que la MSE du premier modèle est supérieure à celle du deuxième (le premier modèle est donc moins bon). « kg », « sym », « mark » et « hk » désignent les modèles de krigeage, de cokrigeage symétrique, de cokrigeage markovien et d'hyperkrigeage respectivement.

	kg - sym	kg - mark	kg - hk	sym - mark	sym - hk	mark - hk
5%	52,1	85,8	-166,60	-125,16	-389,3	-421,1
25%	154,9	185,3	1,34	-20,68	-202,8	-217,3
Médiane	262,6	296,8	128,44	7,52	-131,1	-140,2
Moyenne	408,3	437,7	285,81	24,75	-123,2	-147,1
75%	534,0	548,4	419,44	53,83	-53,0	-94,6
95%	1314,0	1367,0	1220,32	166,41	84,6	74,1

Il apparaît clairement que l'amélioration des modèles introspectifs sur le modèle de krigeage est quasi-systématique (colonnes 1, 2 et 3), de même que l'amélioration des modèles de cokrigeage par rapport au modèle d'hyperkrigeage (colonnes 5 et 6).

On pourrait arguer que la MSE n'est pas le meilleur critère de comparaison des métamodèles. En effet, un des avantages de ces métamodèles est qu'ils prédisent leur propre imprécision, laquelle n'est pas utilisée dans le calcul de la MSE. Pour évaluer si l'imprécision prédite correspond à l'imprécision observée, nous calculons les résidus standardisés $\alpha(x)$ (l'écart de prédiction rapporté à l'écart type de prédiction) :

$$\alpha(x) = \frac{\mu(x) - y(x)}{\sigma(x)} \ . \tag{2.88}$$

Si le modèle prédit correctement en moyenne et en incertitude, les résidus standardisés doivent se répartir suivant une loi normale centrée réduite. Pour l'hyperkrigeage, les prédictions ne suivent pas des lois

normales, et le résidu standardisé est calculé ainsi :

$$\alpha_{hk}(x) = q_\phi\left(\mathrm{P}\left[\tilde{Z}^{(1)}(x) < y(x)\right]\right) \ , \tag{2.89}$$

avec $q_\phi(\cdot)$ la fonction quantile de la loi normale centrée réduite. La quantité $\mathrm{P}\left[\tilde{Z}^{(1)}(x) < y(x)\right]$ est calculé grâce à l'équation (2.73). Pour une fonction de répartition de $\tilde{Z}^{(1)}(x)$ qui suivrait une loi normale, cette formule aboutit à la même valeur que le résidu standardisé habituel en théorie. On a donc utilisé cette formule pour tous les résidus standardisés, pas seulement ceux de l'hyperkrigeage. Néanmoins cette manière de calculer les résidus standardisés augmente les instabilités numériques : en utilisant la fonction de répartition puis la fonction quantile de la loi normale, le α calculé devient infini quand sa valeur théorique dépasse 8,3, générant ainsi des `NaN` (*Not a Number*) numériques lors du calcul de la variance. Sur nos 100 répétitions, on a ainsi dû écarter 2 instances pour le modèle de krigeage, 5 instances pour le modèle de cokrigeage markovien et 10 instances pour le modèle d'hyperkrigeage pour les calculs de variance des résidus standardisés.

Comme on peut le voir sur la figure 2.24, qui présente les variances des résidus standardisés des différents plans d'expériences pour chacun des modèles, les résidus standardisés sont souvent moins étalés que la normale attendue (variances inférieures à 1). Cela correspond à une incertitude prédite surestimée (en effet plus l'incertitude estimée est grande, plus la standardisation des résidus les rassemble autour du zéro). On en déduit que la fonction-objectif est, dans cet exemple, plus lisse et régulière que ce que les modèles estiment. Cela s'explique aussi par le choix d'un plan d'expériences LHS avec le critère *maximin* (au lieu d'un plan uniforme) qui ne permet pas au modèle de correctement évaluer ses paramètres (il lui manque des informations sur les très courtes distances). La moyenne prédite est alors plus proche de la vraie fonction que ce que prévoit la variance de prédiction. Les modèles peuvent dans ce cas être considérés comme conservatifs. Il est en effet préférable d'avoir une incertitude surestimée plutôt que sous-estimée, car cela évite un excès de confiance dans les prédictions, qui pourrait nuire au bon déroulement des algorithmes (tels que ceux présentés dans le prochain chapitre de ce mémoire). Dans le cas présenté ici, le modèles de krigeage et de cokrigeage markovien ont un comportement similaire de leurs incertitudes prédites par rapport à leurs incertitudes observées. Les modèles d'hyperkrigeage et de cokrigeage symétrique surestiment un peu moins leurs incertitudes. Une variance faible dans les résidus standardisés peut aussi provenir d'une prédiction meilleure et d'une incertitude observée particulièrement faible. Ces informations sont à mettre en relation avec les information sur la MSE (figure 2.23), qui informent de l'incertitude observée.

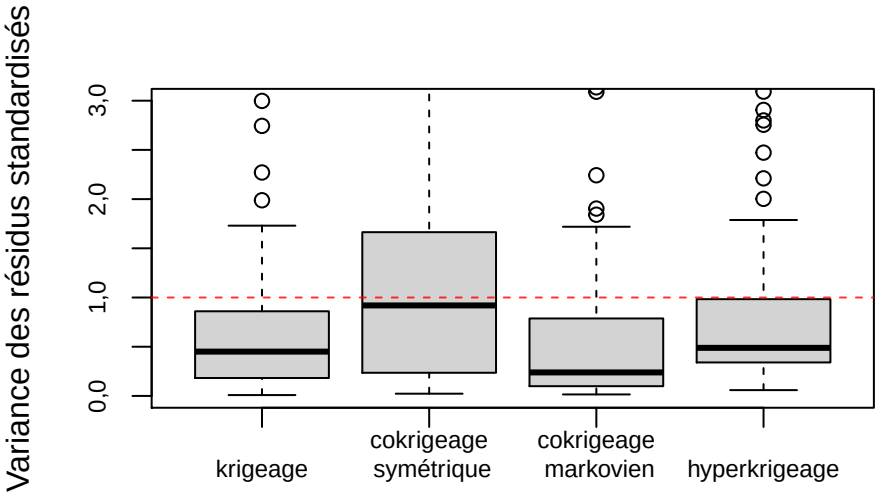
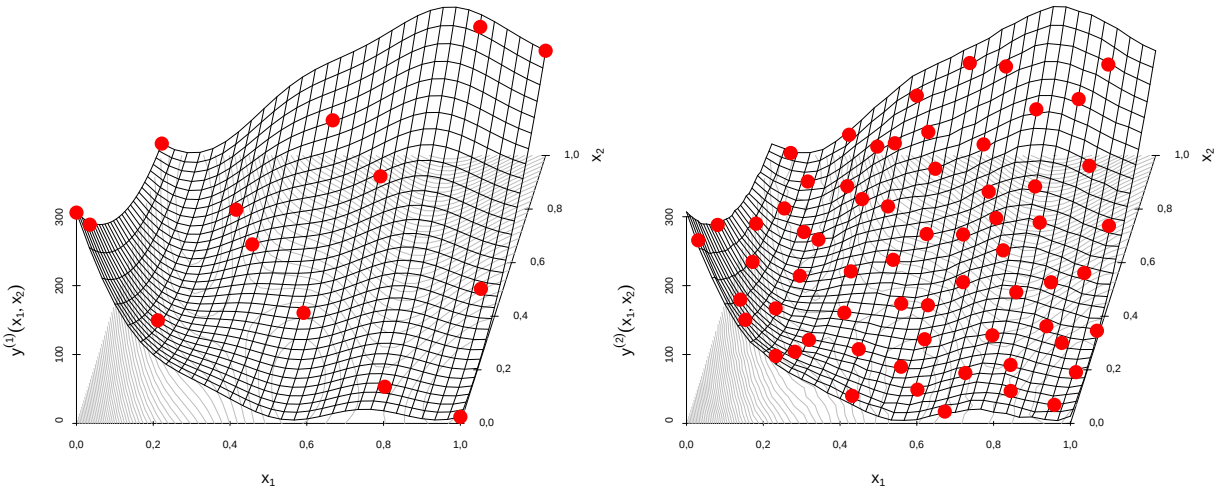


FIGURE 2.24 – Variance des résidus standardisés pour le cas *mesh based* pour les différents modèles considérés. Les résidus standardisés sont mesurés sur 10 000 points aléatoires de la fonction-objectif, uniformément répartis. Les boîtes à moustaches résument les variances des résidus standardisés pour 100 plans d'expériences différents. La variance unitaire est représentée en pointillé rouge.

2.4.3 Résultats sur le cas-test Monte-Carlo

On compare à présent les différents modèles sur le cas-test Monte-Carlo présenté en figure 2.20. L'échantillonnage de taille 10^8 fait office de fonction-objectif qui est ainsi également bruitée, quoique que très peu. Le modèle de krigeage est ajusté sur la base d'un plan d'expériences à 14 points (10 points placés selon un hypercube latin (LHS), et 4 autres dans les coins du domaine) sur cette fonction-objectif. Deux modèles de cokrigeage, un symétrique et un markovien, tels que présentés précédemment, sont ajustés sur ce même plan d'expériences augmenté de 64 points en LHS sur le niveau de fidélité à 10^4 échantillons (considéré comme étant bien moins coûteux à évaluer). Un modèle d'hyperkrigeage est également ajusté sur ces mêmes données. Comme dans la section précédente, les modèles de krigeage et de cokrigeage incluent un paramètre de *nugget* (effet de pépite), estimé d'après le plan d'expériences, et relatif au paramètre d'échelle (l'effet de pépite absolu correspond à la valeur estimée multipliée par le paramètre d'échelle pour le niveau considéré). Le modèle d'hyperkrigeage estime un paramètre de *nugget* différent pour chacun des niveaux. Dans ce cas-test Monte-Carlo, le *nugget* est généralement estimé à des valeurs très faibles, voire négligeables. D'après ce qu'on a observé, le fait que les fonctions à métamodéliser soient bruitées n'a pas eu d'impact significatif sur l'estimation du *nugget*. Un paramètre de tendance constante est également estimé pour tous les modèles et pour chacun des niveaux, suivant les formules données en section 2.2.7. La figure 2.25 présente un exemple des plans d'expériences utilisés.



(a) Fonction-objectif (10^8 échantillons) (b) 10^4 échantillons

FIGURE 2.25 – Exemple de plan d'expériences utilisé pour ajuster les modèles de krigeage, de cokrigeage et d'hyperkrigeage, pour le cas Monte-Carlo.

La construction des modèles est répétée 100 fois, pour des plans d'expériences différents. On utilise les mêmes plan d'expériences que ceux pour le cas-test *mesh based*. On mesure alors l'erreur quadratique moyenne (MSE, pour *Mean Square Error*, cf. équation (2.87)) de la moyenne de prédiction par rapport à la fonction-objectif, sur 10 000 points pris au hasard, uniformément, dans le domaine. La figure 2.26 présente les résultats avec une échelle logarithmique. Pour le modèle de krigeage, on trouve une valeur médiane de la MSE de 376,116 (et une valeur moyenne à 537,03). Le cokrigeage markovien améliore grandement la prédiction et sa MSE médiane descend à 4,555 (moyenne à 101,538). Le modèle de cokrigeage symétrique est marginalement plus efficace et conduit à une MSE médiane de 3,653 (moyenne à 22,29). La MSE de l'hypergrigeage est encore meilleure (plus proche de 0), avec une médiane à 1,521, et une moyenne à 31,611.

Comme pour le cas-test précédent, l'estimation des paramètres du modèle peut parfois se tromper, plus ou moins gravement. Pour ce cas-test Monte-Carlo, sur les 100 répétitions, 4 modèles de cokrigeage symétrique ont été exclus de l'étude de la MSE, car conduisant à des valeurs de MSE aberrantes (supérieures à 35000, quand la prochaine plus haute valeur est à 1275). Ces valeurs aberrantes de la MSE nous permettent de détecter des modèles pour lesquels l'estimation des paramètres est restée bloquée dans un optimum local particulièrement loin de l'optimum global.

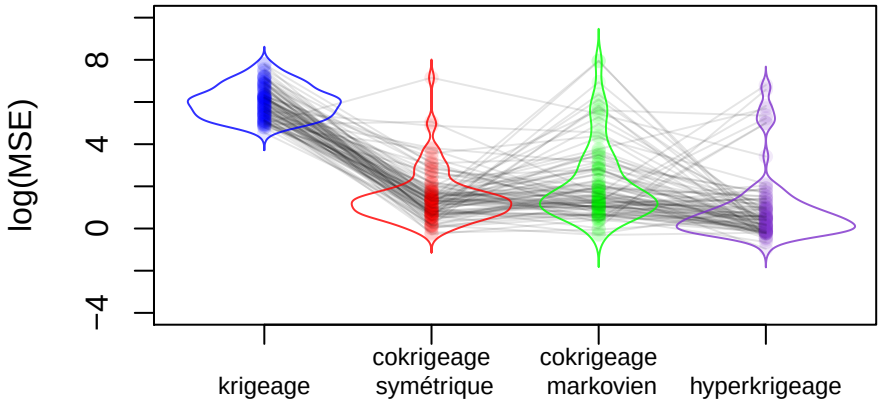


FIGURE 2.26 – Logarithme de la MSE (erreur quadratique moyenne) pour le cas Monte-Carlo pour les différents modèles considérés (de gauche à droite, krigeage (bleu), cokrigeage symétrique (rouge), cokrigeage markovien (vert), hyperkrigeage (violet)). La MSE est mesurée sur 10 000 points aléatoires de la fonction-objectif, uniformément répartis. Une MSE faible indique que le modèle prédit une valeur proche de la vraie valeur. Les MSE sont mesurées pour 100 plans d'expériences différents. Les lignes colorés représentent les densités. Les traits gris relient les MSE d'un même plan d'expériences.

On compare les modèles à plan d'expériences fixé. La figure 2.26 relie le logarithme de la MSE de chaque modèle pour un même plan d'expérience. Ici aussi, il n'y a pas d'effet systématique notable. Les comparaisons deux à deux des modèles sont pratiquement les mêmes selon que l'on compare pour un même plan d'expériences ou pour deux plans d'expériences quelconques. Nous avons choisi de ne présenter que les données de comparaison à plan d'expériences identique. Le tableau ci-dessous résume les écarts de MSE entre les différents modèles pour les plans d'expériences testés. Pour chaque paire de modèle, on affiche les quantiles et la moyenne des écarts de MSE. Un écart positif indique que la MSE du premier modèle est supérieure à celle du deuxième (le premier modèle est donc moins bon). « kg », « sym », « mark » et « hk » désignent les modèles de krigeage, de cokrigeage symétrique, de cokrigeage markovien et d'hyperkrigeage respectivement.

	kg - sym	kg - mark	kg - hk	sym - mark	sym - hk	mark - hk
5%	117	-142	111	-366,704	-187,69	-149,218
25%	213	169	200	-10,080	-0,34	0,637
Médiane	397	347	361	-0,964	1,05	2,719
Moyenne	530	435	505	-82,995	-10,59	69,928
75%	701	593	591	1,443	3,81	15,264
95%	1434	1302	1389	23,453	36,20	344,112

Il apparaît clairement que l'amélioration des modèles introspectifs sur le modèle de krigeage est quasi-systématique (colonnes 1, 2 et 3). Le modèle symétrique semble très légèrement meilleur que le modèle markovien (colonne 4). Le modèle d'hyperkrigeage apporte quant à lui une nette amélioration par rapport aux autres modèles (colonnes 5 et 6).

Nous regardons maintenant les résidus standardisés $\alpha(\cdot)$ et $\alpha_{hk}(\cdot)$ (équations (2.88) et (2.89)) et leurs variances, pour juger à la fois de la qualité de la prédiction et de l'incertitude des modèles. Pour ce cas-test Monte-Carlo, sur les 100 répétitions, 6 instances ont été écartées du fait d'instabilités numériques liées à des variances quasi-nulles pour le modèle de cokrigeage symétrique et 8 pour le modèle de cokrigeage markovien.

Nous observons sur la figure 2.27 que les résidus standardisés sont souvent moins étalés que la normale attendue (variances inférieures à 1). Cela correspond à une incertitude prédite surestimée. En effet plus l'incertitude estimée est grande, plus la standardisation des résidus les rassemble autour du zéro. Dans cet exemple comme dans le *mesh based* précédent, la fonction-objectif au niveau 1 serait plus régulière que ce que les modèles estiment. Le choix d'un plan d'expériences LHS optimisé par critère *maximin* (au lieu d'un plan uniforme) ne permet pas d'acquérir les informations sur les très courtes portées, ce qui peut expliquer cette surestimation des incertitudes. Les modèles appris sont donc conservatifs. Les modèles de cokrigeage, symétrique et markovien, surestiment plus leurs incertitudes que l'hyperkrigeage qui lui même est plus conservatif que le krigeage. La faible variance des résidus standardisés du cokrigeage et de

l'hyperkrigeage s'explique aussi par leur meilleure prédiction (cf. figure 2.26).

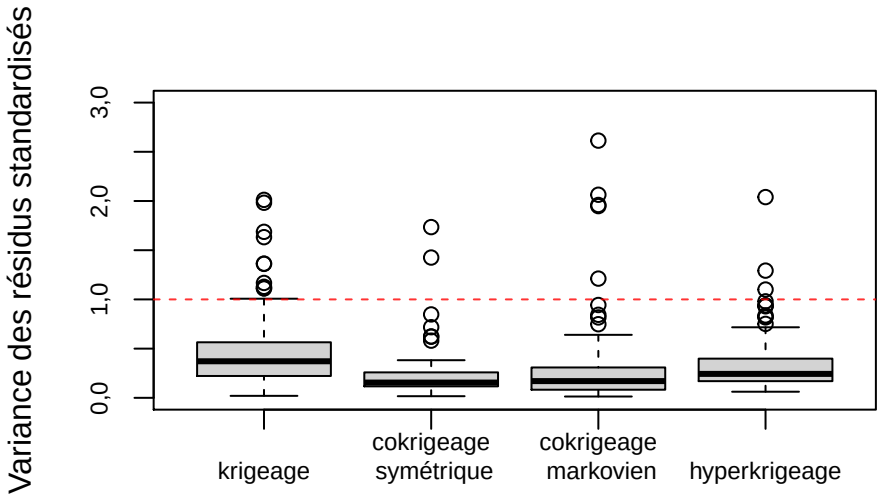


FIGURE 2.27 – Variance des résidus standardisés pour le cas Monte-Carlo pour les différents modèles considérés. Les résidus standardisés sont mesurés sur 10 000 points aléatoires uniformément répartis. Les boîtes à moustaches résument les variances des résidus standardisés pour 100 plans d'expériences différents. La variance unitaire est représentée en pointillé rouge.

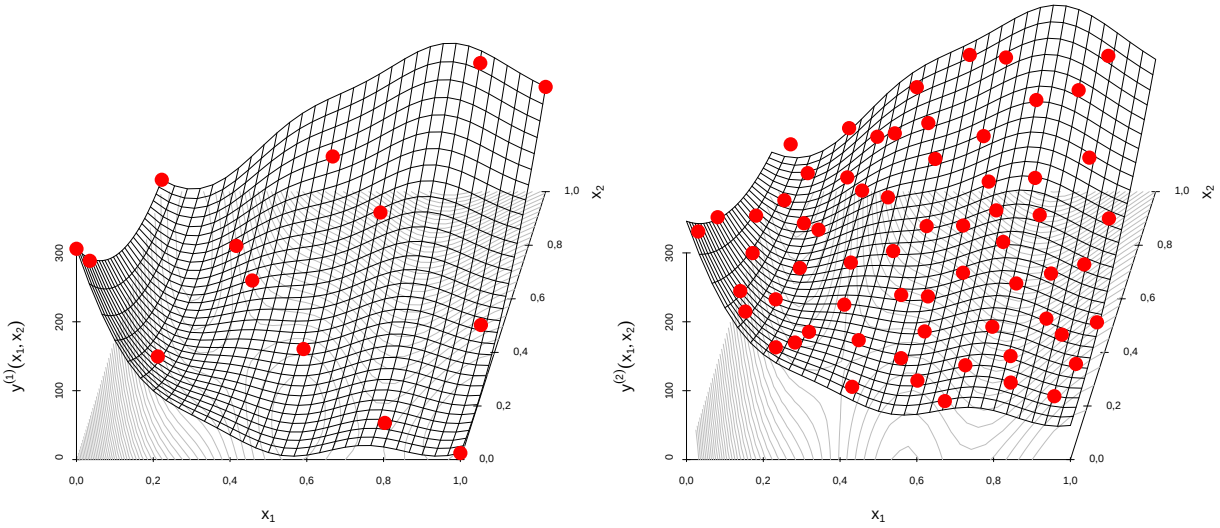
2.4.4 Résultats sur le cas-test *time step*

On compare à présent les différents modèles sur le cas-test *time step* présenté en figure 2.21. L'étape 4 du processus autorégressif est considérée comme notre fonction-objectif. Les modèles statistiques sont construits et testés en suivant le même protocole expérimental que celui des cas-tests *mesh based* et Monte-Carlo. La figure 2.28 présente un exemple des plans d'expériences utilisés.

La figure 2.29 donne les MSE en échelle logarithmique. Pour le modèle de krigeage, on trouve une valeur médiane de la MSE de 465,624 (et une valeur moyenne à 598,569). Le cokrigeage symétrique améliore grandement la prédiction et sa MSE médiane descend à 65,356 (moyenne à 110,338). Le modèle de cokrigeage markovien est un peu plus efficace et conduit à une MSE médiane de 40,986 (moyenne à 105,81). La MSE de l'hyperkrigeage est dans les mêmes ordres de grandeur, avec une médiane à 52,188, et une moyenne à 75,505.

On a exclu de l'étude 5 modèles de cokrigeage symétrique. Ces modèles ont une MSE supérieure à 50000, là où prédire avec une constante raisonnable (comprise entre le minimum et le maximum des observation) conduit à une MSE inférieure à 5000. La MSE la plus élevée suivante, parmi tous les modèles de cokrigeage symétrique, est à environ 660.

La figure 2.29 permet de comparer les modèles à plan d'expériences fixé en suivant les trais gris. Sur ce cas-test *time step*, comme sur les deux précédents, il n'y a pas d'effet systématique notable. Les comparaisons deux à deux de modèles ne varient pas lorsqu'elles sont réalisées avec un même plan d'expériences ou deux plans d'expériences quelconques. Nous avons choisi de ne présenter que les données de comparaison à plan d'expériences identique. Le tableau ci-dessous résume les écarts de MSE entre les différents modèles sur l'ensemble des plans d'expériences testés sous la forme de quantiles et de moyenne pour chaque paire de modèle. Un écart positif indique que la MSE du premier modèle est supérieure à celle du second. « kg », « sym », « mark » et « hk » désignent les modèles de krigeage, de cokrigeage symétrique, de cokrigeage markovien et d'hyperkrigeage respectivement.



(a) Fonction-objectif (étape 4)

(b) Étape 3

FIGURE 2.28 – Exemple de plan d'expériences utilisé pour ajuster les modèles de krigeage, de cokrigeage et d'hyperkrigeage, pour le cas *time step*.

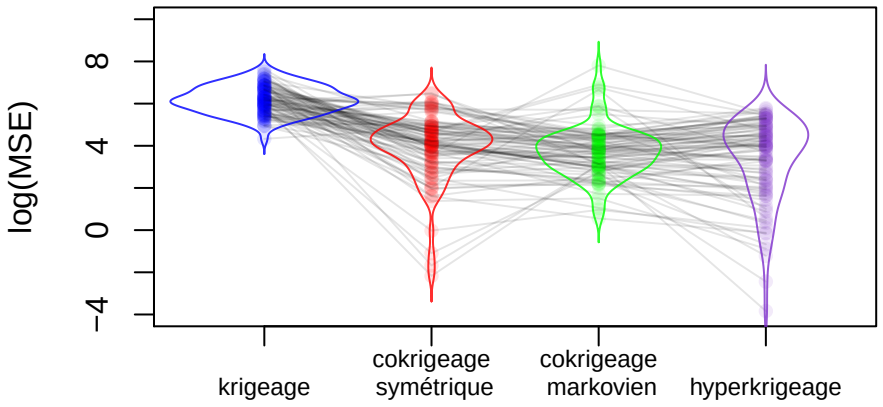


FIGURE 2.29 – Logarithme de la MSE (erreur quadratique moyenne) pour le cas *time step* pour les différents modèles considérés (de gauche à droite, krigeage (bleu), cokrigeage symétrique (rouge), cokrigeage markovien (vert), hyperkrigeage (violet)). La MSE est mesurée sur 10 000 points aléatoires de la fonction-objectif, uniformément répartis. Une MSE faible indique que le modèle prédit une valeur proche de la vraie valeur. Les MSE sont mesurées pour 100 plans d'expériences différents. Les lignes colorés représentent les densités. Les traits gris relient les MSE d'un même plan d'expériences.

	kg - sym	kg - mark	kg - hk	sym - mark	sym - hk	mark - hk
5%	-116	69,2	39,2	-248,98	-96,94	-145,322
25%	204	213,1	251,3	-3,72	-5,65	-46,563
Médiane	376	398,1	389,6	29,77	6,49	0,975
Moyenne	492	492,8	523,1	2,43	32,04	30,305
75%	732	712,2	705,1	55,91	40,08	16,195
95%	1298	1283,2	1313,6	328,83	304,56	321,920

Il apparaît clairement que l'amélioration des modèles introspectifs sur le modèle de krigeage est quasi-systématique (colonnes 1, 2 et 3). Le modèle symétrique semble légèrement moins bon que le modèle markovien (colonne 4). Ceci est peut-être dû au fait que la structure markovienne du modèle corresponde à la forme auto-regressive du cas test *time step*. Le modèle d'hyperkrigeage apporte quant à lui une petite amélioration par rapport aux autres modèles (colonnes 5 et 6).

Nous nous intéressons aux résidus standardisés pour ce cas-test *time step*. Sur les 100 répétitions, certains résidus standardisés n'étaient pas calculables (division par une trop petite variance) et leurs variances ont été écartées : 6 instances pour le modèle de cokrigeage symétrique, 6 instances pour le modèle de cokrigeage markovien et 5 pour le modèle d'hyperkrigeage. Une nouvelle fois, comme pour les deux autres cas tests et pour les mêmes raisons, les résidus standardisés représentés sur la figure 2.30 sont moins dispersés que la normale de l'hypothèse gaussienne.

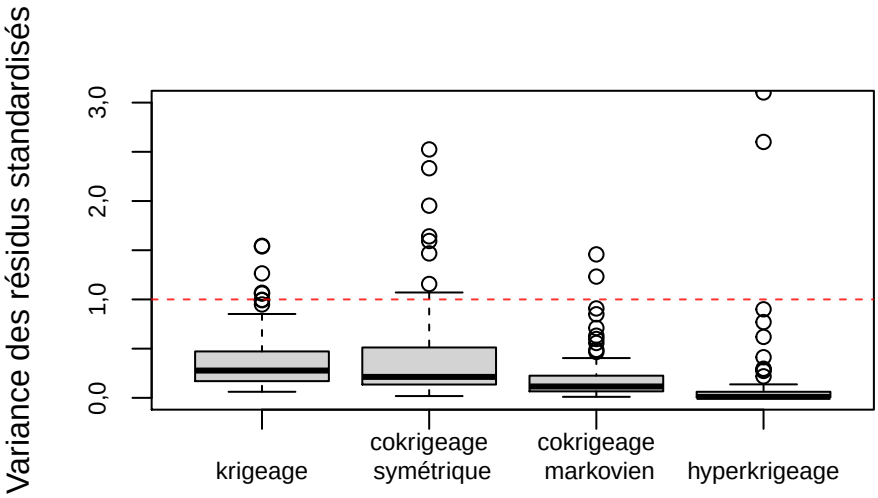


FIGURE 2.30 – Variance des résidus standardisés pour les différents modèles considérés. Les résidus standardisés sont mesurés sur 10 000 points aléatoires de la fonction-objectif, uniformément répartis. Les boîtes à moustaches résument les variances des résidus standardisés pour 100 plans d'expériences différents. La variance unitaire est représentée en pointillé rouge.

Nous avons jusqu'à présent étudié différents métamodèles capables de prendre en compte des données supplémentaires. Dans le prochain chapitre, nous allons nous intéresser à des algorithmes d'optimisation et d'inversion capables de prendre en compte des données supplémentaires, algorithmes pour lesquels ces nouveaux métamodèles sont nécessaires. Les trois cas-tests présentés dans cette section seront alors utilisés pour comparer l'efficacité des algorithmes.

Chapitre 3

Algorithmes itératifs introspectifs

Dans le premier chapitre, nous avons décrit les modèles de krigeage et les algorithmes itératifs qu'ils permettent de mettre en œuvre, notamment l'algorithme EGO pour résoudre des problèmes d'optimisation. Dans le deuxième chapitre, nous avons vu comment améliorer les modèles de krigeage lorsqu'on a accès à des sorties complémentaires à la sortie d'intérêt. Nous avons vu qu'un modèle prenant en compte les informations d'introspection permet d'obtenir de meilleures prédictions. De ce fait, l'usage d'un modèle introspectif améliore l'efficacité d'un algorithme itératif tel que EGO. Dans ce troisième chapitre, nous allons plus loin. Nous présentons et mettons en œuvre un nouvel algorithme : l'algorithme *Step or Stop optimization* (SoSo) est une variante de EGO spécialement conçue pour exploiter les possibilités d'introspection afin de réduire le coût de calcul des observations. Certains calculs seront ainsi effectués seulement partiellement car les sorties intermédiaires seront suffisamment informatives. Un exemple d'application de ce nouvel algorithme est présenté, suivi d'une comparaison avec EGO pour différents cas de figure. On généralise ensuite le concept pour d'autres problématiques, notamment l'inversion.

Principales contributions de ce chapitre

- Un cadre général pour les algorithmes itératifs « SoS » exploitant la possibilité de ne faire certains calculs que partiellement (section 3.2),
- Le critère EIR qui permet de construire l'algorithme « SoSo », déclinaison du cadre « SoS » pour traiter les problèmes d'optimisation (section 3.3),
- La généralisation du cadre « SoS » à la construction d'algorithmes traitant les problèmes d'inversion (section 3.4).

3.1 Cadre d'étude

Dans les sections 2.1.2 et 2.1.3, nous avons décrit un certain nombre de travaux de la littérature où, non seulement des modèles capables de gérer plusieurs niveaux ont été développés, mais également des algorithmes y ont été associés en fonction des besoins de leurs auteurs respectifs. Il est en effet difficile de dissocier les modèles introspectifs des algorithmes qui les exploitent, et la bibliographie reflète cet état de fait. On renverra donc le lecteur aux sections susnommées où nous avons regroupé ces références. Chacun de ces travaux traite d'un cas d'étude particulier, avec des objectifs qui lui sont propres et des particularités inhérentes. Il convient de préciser le cadre et le type de problèmes pour lesquels les algorithmes développés dans cette thèse peuvent s'appliquer.

Nous nous plaçons ici dans un cas de figure où une simulation physique est calculée par un enchaînement séquentiel de différents codes de calcul. Nous avons accès aux sorties successives des codes. C'était le cadre du chapitre 2 pour les modèles introspectifs, mais nous sommes plus restrictifs puisque nous imposons ici de considérer l'enchaînement comme strict : il est impossible de faire la dernière étape de calcul sans avoir fait les précédentes. Cela implique que le plan d'expériences du niveau de plus haute fidélité soit une sous-partie du plan d'expériences du niveau précédent, lui-même étant une sous-partie du plan utilisé au niveau précédent, et ainsi de suite. En anglais, on utilise la notion de « *nested design* » pour désigner cette structure de plans d'expériences imbriqués. Cette structure est exploitée dans (Le Gratiet and Cannamela 2015), où les auteurs choisissent le prochain point et le niveau jusqu'où le calculer en

anticipant la réduction de variance sur la fonction-objectif, pondérée par le temps de calcul nécessaire à l'obtention du point. Leur critère permet un apprentissage global de la fonction-objectif, en ciblant systématiquement les régions les plus incertaines pour le métamodèle. D'autres travaux (Huang et al. 2006 ; puis Sacher 2018 ; Sacher et al. 2020 ; et Kandasamy et al. 2019) ont étudié le cas de codes de calcul multifidélités où l'utilisateur pouvait choisir à sa guise le niveau calculé. Dans (Huang et al. 2006), pour choisir en même temps le prochain itéré et son niveau de fidélité, les auteurs multiplient un critère de progrès espéré défini au niveau le plus fin (l'EI, *expected improvement*, voir section 3.3) par des coefficients liés au niveau considéré : il y a un coefficient proportionnel à l'inverse du temps de calcul, un coefficient lié à sa variance et un coefficient lié à la corrélation entre le niveau et celui de plus haute fidélité. De la même façon, dans (Sacher et al. 2020), le progrès espéré au niveau le plus fin est multiplié par un coefficient proportionnel au temps de calcul et un coefficient dépendant de la réduction de variance attendue par ajout d'un point à un niveau quelconque. Dans (Kandasamy et al. 2019), les auteurs proposent une stratégie d'optimisation en utilisant un métamodèle sur le niveau de basse fidélité pour éliminer les régions inintéressantes et un autre métamodèle sur le niveau de haute fidélité dont le domaine est ainsi restreint. Dans notre cas de *nested design*, le calcul d'un certain niveau exige le calcul de tous les niveaux précédents. Les ingénieurs désignent cette propriété avec le nom de « code Markovien ». Il s'agit d'une situation fréquemment rencontrée par l'IRSN, et d'une manière générale par les utilisateurs de toutes les chaînes de simulations industrielles (cf. figure 3.1).

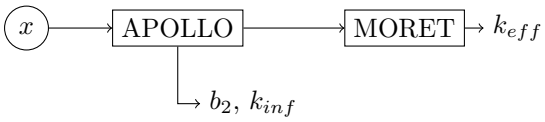


FIGURE 3.1 – Schéma Apollo-Moret.

Comme pour EGO, vu au chapitre 1 (1.3), nous considérons que l'objectif est de trouver la valeur minimale que peut prendre la grandeur d'intérêt dans l'ensemble du domaine d'étude. Le problème de la recherche de la valeur maximale (rencontré à l'IRSN, voir section 3.3.3) est un problème équivalent et on peut se ramener à une recherche de minimum en multipliant par -1 la fonction-objectif. Comme pour EGO, nous voulons réduire, autant que faire se peut, les appels aux codes de calcul afin d'en réduire les coûts (en temps/puissance CPU). Précisons que dans notre cas de figure, les informations supplémentaires (introspectives) sont générées séquentiellement et il est possible d'interrompre le calcul avant la fin (ou de manière équivalente de ne faire qu'une partie du calcul). Nous considérons que l'utilisateur est non seulement capable d'interrompre des simulations à l'étape de son choix, mais il peut également relancer des simulations précédemment interrompues. Enfin il est admis que l'utilisateur possède une connaissance (même approximative) du coût de calcul de chaque étape, lequel ne dépend pas du point simulé, et qu'il dispose d'un budget fini pour trouver le maximum recherché. Dans les exemples, les coûts seront exprimés avec le symbole monétaire « ¤ », prévu pour désigner une monnaie non spécifiée (ou au caractère typographique inexistant).

Comme cela a été posé précédemment, nous nous plaçons dans un cas de figure où les appels aux simulations physiques sont coûteux. On va donc chercher à en faire le moins possible, et pour cela nous allons utiliser des métamodèles stochastiques, comme les modèles de krigeage présentés au chapitre 1 ou les modèles introspectifs présentés au chapitre 2.

Enfin, nous supposons dans un premier temps que les simulations physiques sont non-bruitées. Deux appels identiques à la chaîne de calcul donnent ainsi des résultats parfaitement identiques. Un bruit spatial peut néanmoins être présent (de type « effet de pépite », où deux points extrêmement proches peuvent donner des résultats significativement différents) sans que cela n'affecte les algorithmes présentés.

3.2 Algorithme *Step or Stop* (SoS)

Dans un premier temps, nous nous plaçons dans un cadre plus général et nous visons un objectif d'ingénierie quelconque pourvu que, à chaque itération, il puisse s'exprimer sous la forme d'un critère à maximiser. Nous revenons sur l'optimisation plus loin (section 3.3). Comme dans la section 1.3 du premier chapitre, l'objectif sera poursuivi à travers la construction de métamodèles lors des itérations.

Pour transformer un algorithme itératif « standard » en algorithme introspectif, la première chose à faire est de mettre à jour le modèle utilisé avec les informations supplémentaires. En l'occurrence, on va utiliser un modèle stochastique introspectif tel que ceux présentés dans le chapitre précédent.

Ce simple changement de modèle ne suffit pas. Nous allons également proposer un algorithme plus adapté. L'algorithme 3.1 présenté ici, appelé *Step or Stop* (SoS) [1], est un algorithme itératif qui reprend les bases de EGO, mais qui exploite les possibilités offertes par l'introspection. L'algorithme *Step or Stop* doit son nom au fait qu'on va se poser la question, pour un point donné dont on a commencé les calculs, de savoir s'il faut en calculer l'étape suivante (*Step*) ou s'il faut s'arrêter là et aller chercher un autre point (*Stop*). La situation initiale correspond à un plan d'expériences donné (par exemple un plan *Latin Hypercube Sampling*), et les calculs correspondants déjà faits du niveau n_g (le premier niveau calculé par la chaîne) jusqu'au niveau 1 (le dernier niveau). Les coûts de chaque étape de calcul peuvent éventuellement être évalués à partir de cette initialisation s'ils ne sont pas connus par ailleurs. Sur la base du plan d'expériences initial, on ajuste les paramètres du métamodèle introspectif choisi (parmi ceux présentés dans le chapitre précédent par exemple). Le métamodèle global est noté $\boldsymbol{M}$ et les modèles prédicteurs qui en sont issus pour chacun des niveaux $1, \ldots, n_g$ sont notés $M^{(1)}, \ldots, M^{(n_g)}$. On note $\boldsymbol{G}$ le vecteur qui, à chaque élément de $\boldsymbol{X}$, associe l'indice de sa grandeur correspondante. On note également $\underline{\boldsymbol{X}}$ les éléments privées de leur grandeur : $\boldsymbol{X} = (\underline{\boldsymbol{X}}\ ,\ \boldsymbol{G})$.

Algorithme 3.1 : Algorithme générique SoS pour itérer des simulations multisorties.

Entrées : plan d'expériences initial $(\boldsymbol{X}, \boldsymbol{Y})$ et N sa taille, coûts des différents niveaux $t^{(1)}, \ldots, t^{(n_g)}$, budget maximal t_{max}, chaîne de calcul symbolisée par les fonctions $(y^{(n_g)}(x), \ldots, y^{(1)}(x))$, métamodèle introspectif $\boldsymbol{M}$, critère à maximiser $J_{\boldsymbol{M}}(x,t)$ selon l'objectif à atteindre (fonction de $x \in \mathbb{D}$ et du temps de calcul t).

Initier le budget dépensé $t \leftarrow 0$.
Calculer les coûts cumulés pour finir un calcul, $\left\{T_a = \sum_{b=1}^{a} t^{(b)}\right\}_{a=1,\ \ldots,\ n_g}$.
Ajuster un métamodèle $\boldsymbol{M}$ sur $(\boldsymbol{X}, \boldsymbol{Y})$.
Tant que $t \leq t_{max}$ **faire :**
  Chercher $x^* = \text{argmax}_{x \in \mathbb{D}} \left(J_{\boldsymbol{M}}(x, T_{n_g})\right)$
  Chercher $i^* = \text{argmax}_{1 \leq i \leq N} \left(J_{\boldsymbol{M}}(\underline{\boldsymbol{X}}_i, T_{\boldsymbol{G}_i - 1})\right)$
  Si $J_{\boldsymbol{M}}(\underline{\boldsymbol{X}}_{i^*}, T_{\boldsymbol{G}_{i^*}-1}) > J_{\boldsymbol{M}}(x^*, T_{n_g})$ `/* (Stop) */`
    $\underline{\boldsymbol{X}}_{new} \leftarrow \underline{\boldsymbol{X}}_{i^*}$
    $\boldsymbol{G}_{new} \leftarrow \boldsymbol{G}_{i^*} - 1$
    $\boldsymbol{X}_{new} = \left(\underline{\boldsymbol{X}}_{new}\ ,\ \boldsymbol{G}_{new}\right)$
    $N \leftarrow N + 1$
  Sinon `/* (Step) */`
    $\underline{\boldsymbol{X}}_{new} \leftarrow x^*$
    $\boldsymbol{G}_{new} \leftarrow n_g$
    $\boldsymbol{X}_{new} = \left(\underline{\boldsymbol{X}}_{new}\ ,\ \boldsymbol{G}_{new}\right)$
    $N \leftarrow N + 1$
  Évaluer $\boldsymbol{Y}_{new} = \boldsymbol{y}(\boldsymbol{X}_{new})$ (ou $\boldsymbol{Y}_{new} = y^{(\boldsymbol{G}_{new})}(\underline{\boldsymbol{X}}_{new})$)
  Mettre à jour $\boldsymbol{X} \leftarrow \boldsymbol{X} \cup \boldsymbol{X}_{new}$, $\boldsymbol{G} \leftarrow \boldsymbol{G} \cup \boldsymbol{G}_{new}$ et $\boldsymbol{Y} \leftarrow \boldsymbol{Y} \cup \boldsymbol{Y}_{new}$
  Mettre à jour $t \leftarrow t + t^{(\boldsymbol{G}_{new})}$
  Réajuster le métamodèle $\boldsymbol{M}$ sur $(\boldsymbol{X}, \boldsymbol{Y})$
Fin

Résultat : Plan d'expériences final $(\boldsymbol{X}, \boldsymbol{Y})$, métamodèle $\boldsymbol{M}$

À partir du métamodèle $\boldsymbol{M}$, on cherche le point qui maximise le critère $J_{\boldsymbol{M}}$ parmi tous les points du domaine. Le critère est à définir en fonction de l'objectif d'ingénierie ciblé (optimisation, inversion, ou autre). Il doit s'agir d'une fonction scalaire, à calculer sur la base du métamodèle, et dépendant de la position x pour laquelle il est calculé et du temps théoriquement nécessaire à l'obtention de la connaissance

1. Dans quelques cas de la littérature (par exemple dans (Sanson, Le Maître, and Congedo 2019)), l'acronyme « SoS » peut parfois être utilisé pour désigner *« system of solvers »* (chaîne de calcul).

parfaite du point x. Ici, le temps nécessaire pour finir les calculs d'un point $\underline{\boldsymbol{X}_i}$ déjà évalué jusqu'au niveau $\boldsymbol{G}_i$ s'écrit $T_{\boldsymbol{G}_i-1}$, et le temps nécessaire pour faire entièrement un calcul non commencé vaut T_{n_g}. Le critère doit être une fonction décroissante du temps de calcul restant (plus le temps nécessaire au calcul est faible, plus le critère est favorable). Par exemple pour un objectif d'optimisation, qui nous intéresse tout particulièrement, on va pouvoir utiliser le critère EIR (*Expected Improvement Rate*) décrit dans la section suivante. Nous évoquons d'autres possibilités en section 3.4.

Du fait de sa dépendance au temps de calcul restant, le critère n'est pas une fonction continue, notamment aux points (dénombrables) pour lesquels on a déjà les premières étapes de calcul effectuées. Donc en pratique, on procède en deux étapes : d'une part on va rechercher le maximum du critère sur tout le domaine en faisant comme si aucun point n'avait été commencé (le temps de calcul restant est fixé à T_{n_g} partout) ; et d'autre part on va calculer le critère sur tous les points connus pour lesquels on a déjà des calculs partiellement effectués et on en prend le maximum. Le maximum du critère global sera le maximum des deux étapes. Si le meilleur critère correspond à un point déjà commencé, on le continue avec l'étape de calcul suivante (phase de *Step*). S'il correspond à un nouveau point, on effectue la première étape de calcul correspondant à ce point. C'est la phase de *Stop* car on suspend les évaluations déjà en cours. On a alors augmenté le plan d'expérience d'un calcul (ou plutôt d'une étape de calcul) et on peut réitérer l'algorithme. Cette stratégie sera expliquée plus en détails dans le cas de l'EIR dans la prochaine section.

L'algorithme tourne jusqu'à ce que le budget alloué ait été consommé. En pratique, SoS itère normalement tant que le budget restant est supérieur au coût d'un calcul complet ($T_{n_g} = \sum_{a=1}^{n_g} t^{(a)}$). Lorsque ce n'est plus le cas, on va restreindre notre recherche du maximum du critère, J_M, aux points déjà commencés, et suffisamment commencés pour que le budget restant puisse permettre de les finir. La dernière itération de l'algorithme consiste ainsi toujours à finir le calcul d'un point jusqu'au premier niveau. Cette subtilité n'est pas précisée dans le pseudocode 3.1 afin de ne pas l'alourdir.

3.3 Critère *Expected Improvement Rate* (EIR) pour l'optimisation

Pour répondre à un objectif d'optimisation, nous introduisons un nouveau critère, le critère EIR (*Expected Improvement Rate*, taux d'amélioration espéré). Ce critère est dérivé du critère EI pour EGO. On en rappelle ici la formule, déjà donnée dans le premier chapitre (1.3.1). Elle est écrite ici pour un problème de minimisation. La formule de l'EI est spécifiée pour un métamodèle à plusieurs niveaux :

$$EI(x) \;=\; \mathbb{E}\left(\max(0, Y_{\min} - Z^{(1)}(x))\right) \quad \text{où} \quad Y_{\min} = \min(Y^{(1)}) \;, \tag{3.1}$$

$$EI(x) = \begin{cases} (Y_{\min} - \mu^{(1)}(x))\,\Phi\Big(\frac{Y_{\min}-\mu^{(1)}(x)}{\sigma^{(1)}(x)}\Big) + \sigma^{(1)}(x)\,\phi\Big(\frac{Y_{\min}-\mu^{(1)}(x)}{\sigma^{(1)}(x)}\Big) & \text{lorsque } \sigma^{(1)}(x) > 0 \;, \\ \max\big(0, \big(Y_{\min} - \mu^{(1)}(x)\big)\big) & \text{lorsque } \sigma^{(1)}(x) = 0 \;. \end{cases} \tag{3.2}$$

On remarque que la moyenne et la variance de krigeage, $\mu^{(1)}$ et $\sigma^{(1)}$, sont celles du niveau 1, celui qui succède à tous les calculs. En effet, le cadre des modèles introspectifs qui est le nôtre est plus général que celui de la multifidélité et seul le niveau 1 peut représenter la fonction à minimiser. L'objectif d'ingénierie (ici la recherche d'un minimum) est supposé porter sur le niveau 1 uniquement. Les niveaux le précédant sont possiblement de nature différente, quoique – c'est notre hypothèse – corrélés.

Notre nouveau critère EIR est défini comme l'espérance du taux d'amélioration. Ce qu'on appelle « taux d'amélioration » est l'amélioration divisée par le temps nécessaire à l'obtention de cette amélioration :

$$EIR(x) \;=\; \mathbb{E}\left(\max\left(0 \;,\; \frac{Y_{\min} - Z^{(1)}(x)}{T_{g(x)-1}}\right)\right) \;, \tag{3.3}$$

où $t^{(a)}$ correspond au temps (coût) de calcul de l'étape passant du niveau $a+1$ au niveau a, $T_b = \sum_{a=1}^{b} t^{(a)}$ et $g(x)$ est la fonction qui nous donne le dernier niveau calculé pour le point x. Pour rappel, dans notre

convention, la première étape de calcul produit le niveau n_g, la dernière étape produit le niveau 1. Par convention, on fixe $g(x) = n_g + 1$ si il n'y a pas encore eu de calcul pour le point x. Pour un point connu $x \in \underline{\boldsymbol{X}}$, $g(x) = \min_{i\in[1\dots N]}(\boldsymbol{G_i} \mid \underline{\boldsymbol{X}}_{\boldsymbol{i}} = \boldsymbol{x})$. La somme $T_{g(x)-1} = \sum_{a=1}^{g(x)-1} t^{(a)}$ correspond donc au temps restant pour finir les calculs au point x. Pour un point encore non exploré, T_{n_g} correspond au temps total pour faire entièrement un calcul. On fixe également $EIR(x) = 0$ si $g(x) = 1$, c'est-à-dire si x est parfaitement connu (les calculs ont déjà été menés jusqu'au bout pour ce point).

Par la linéarité de l'espérance, et puisqu'on considère que les temps de calcul sont figés, on peut écrire :

$$\begin{aligned} EIR(x) &= \frac{\mathbb{E}\big(\max\big(0\ ,\ Y_{\min} - Z^{(1)}(x)\big)\big)}{T_{g(x)-1}} \\ &= \frac{EI(x)}{T_{g(x)-1}}\ . \end{aligned} \tag{3.4}$$

La continuité de la fonction $EI(x)$ est liée à la continuité du noyau utilisé. Dans la plupart des cas (pour les noyaux présentés en 1.2.1), $EI(x)$ est continue sur le domaine $\mathbb{D}$ des x. En effet, $\mu(x)$ et $\sigma(x)$, la moyenne et l'écart type de krigeage, sont des combinaisons linéaire et quadratique des noyaux, $k(x, X_i)$; donc si $k(x, X_i)$ est continu, elles le sont aussi. $EI(x)$ est une composition de fonctions continues, donc elle est essentiellement continue. Lorsque $\sigma(x)$ s'annule, aux points de données X_i, on peut se demander si l'EI est encore continue du fait des termes en $1/\sigma(x)$. Cependant, dans ces cas $EI(x) = \max(0, Y_{min} - \mu(x))$ (Mohammadi et al. 2018) et l'on retrouve la continuité. Toutefois ,si on ajoute un effet pépite au modèle (*nugget effect*), $k(x, X_i)$ n'est plus continu en X_i, $\mu(x)$ et $\sigma(x)$ deviennent discontinu aux points d'observation, et $EI(x)$ également. En dehors des points d'observation, la continuité est cependant conservée. Dans le cas multisortie qui nous intéresse, $EI(x)$ dépend de $\mu^{(1)}(x)$ et de $\sigma^{(1)}(x)$ et la continuité n'est perdue que pour les observations faites jusqu'au niveau 1. La fonction $g(x)$, pour sa part, est discontinue à tous les points d'observation de par sa définition. La fonction $EIR(\cdot)$ n'est donc pas continue aux points où des calculs ont été partiellement effectués. En mettant ces discontinuités en évidence, on peut écrire :

$$EIR(x) = \begin{cases} \frac{EI(x)}{T_{n_g}} & \text{si } x \notin \underline{\boldsymbol{X}}\ , \\ \frac{EI(x)}{T_{g(x)-1}} & \text{si } x \in \underline{\boldsymbol{X}}\ . \end{cases} \tag{3.5}$$

Il convient de noter que $T_{g(x)-1} = \sum_{a=g(x)-1}^{1} t^{(a)} < T_{n_g} = \sum_{a=n_g}^{1} t^{(a)}$, puisque tous les temps sont positifs. On en déduit que pour un point partiellement connu, le critère EIR donne une valeur supérieure à ce qu'il aurait donné pour un point totalement inconnu (à métamodèle identique), ce qui favorise la progression d'un point déjà connu par rapport à l'exploration d'un point nouveau trop proche du premier :

$$\frac{EI(x)}{T_{g(x)-1}} > \frac{EI(x)}{T_{n_g}} \qquad \text{si } x \in \underline{\boldsymbol{X}}\ . \tag{3.6}$$

On voit également que pour les points totalement inconnus, le maximum de l'EIR correspond (en position) au maximum de l'EI :

$$\begin{aligned} \operatorname*{argmax}_{x\in\mathbb{D}\setminus\underline{\boldsymbol{X}}} EIR(x) &= \operatorname*{argmax}_{x\in\mathbb{D}\setminus\underline{\boldsymbol{X}}} \frac{EI(x)}{T_{n_g}} \\ &= \operatorname*{argmax}_{x\in\mathbb{D}\setminus\underline{\boldsymbol{X}}} EI(x)\ . \end{aligned} \tag{3.7}$$

Puisque l'EI est une fonction continue, en dehors éventuellement des points connus jusqu'au niveau 1 où l'EI est nulle, si son maximum sur tout le domaine $\mathbb{D}$ tombe sur un point partiellement connu (par exemple $\underline{\boldsymbol{X}}_j$) alors le maximum de l'$EI$ sur le domaine restreint $\mathbb{D} \setminus \underline{\boldsymbol{X}}$ en sera infiniment proche. Par (3.6), le maximum de l'EIR sur le domaine des points inconnus $\mathbb{D} \setminus \underline{\boldsymbol{X}}$ sera alors inférieur à $EIR(\underline{\boldsymbol{X}}_j)$ et par conséquent le maximum de l'EIR se trouvera sur l'un des points connus (partiellement). La méthode présentée dans la section précédente 3.2, consistant à chercher le maximum de l'EIR sur tout le domaine $\mathbb{D}$ comme si aucun point n'était connu, puis à le comparer au maximum de l'EIR sur tous les points connus, est alors parfaitement justifiée.

Cela nous conduit à l'algorithme 3.2, nommé *Step or Stop Optimization* (SoSo). L'algorithme SoSo est une amélioration de l'algorithme EGO pour les cas d'introspection.

Algorithme 3.2 : Algorithme d'optimisation SoSo pour simulateur multisortie.

Entrées : plan d'expériences initial $(\boldsymbol{X}, \boldsymbol{Y})$, coûts des différents niveaux $t^{(1)}, \ldots, t^{(n_g)}$, budget maximal t_{max}, chaîne de calcul générant les fonctions $(y^{(n_g)}(x), \ldots, y^{(1)}(x))$, métamodèle introspectif $\boldsymbol{M}$.

Initier le budget dépensé $t \leftarrow 0$.
Calculer les sommes cumulées $T_a = \sum_{b=1}^{a} t^{(b)}$ (par convention $T_0 = 0$).
Ajuster un métamodèle $\boldsymbol{M}$ sur $(\boldsymbol{X}, \boldsymbol{Y})$.
Tant que $t \leq t_{max}$ **faire :**
  Chercher $x^* = \text{argmax}_{x \in \mathbb{D}} \left(EI(x)\right)$
  Chercher $i^* = \text{argmax}_{1 \leq i \leq N} \left(\frac{EI(\underline{\boldsymbol{X}_i})}{T_{\boldsymbol{G}_i - 1}} \right)$
  Si $\frac{EI(\underline{\boldsymbol{X}_i})}{T_{\boldsymbol{G}_i - 1}} > \frac{EI(x^*)}{T_{n_g}}$
    $\underline{X_{new}} \leftarrow \underline{\boldsymbol{X}_{i^*}}$
    $G_{new} \leftarrow \boldsymbol{G}_{i^*} - 1$
    $X_{new} = \left(\underline{X_{new}}\ ,\ G_{new}\right)$
  Sinon
    $\underline{X_{new}} \leftarrow x^*$
    $G_{new} \leftarrow n_g$
    $X_{new} = \left(\underline{X_{new}}\ ,\ G_{new}\right)$
  Évaluer $Y_{new} = y^{(G_{new})}(\underline{X_{new}})$
  Mettre à jour $\boldsymbol{X} \leftarrow \boldsymbol{X} \cup X_{new}$, $\boldsymbol{G} \leftarrow \boldsymbol{G} \cup G_{new}$ et $\boldsymbol{Y} \leftarrow \boldsymbol{Y} \cup Y_{new}$
  Mettre à jour $t \leftarrow t + t^{(G_{new})}$
  Réajuster le métamodèle $\boldsymbol{M}$ sur $(\boldsymbol{X}, \boldsymbol{Y})$
Fin

Résultat : Plan d'expériences final $(\boldsymbol{X}, \boldsymbol{Y})$, métamodèle $\boldsymbol{M}$

Critères EI et EIR pour l'hyperkrigeage

Un métamodèle d'hyperkrigeage est construit sur des processus gaussien mais le résultat n'est pas un processus gaussien. La formule analytique de l'EI ne peut donc pas s'appliquer directement. Nous pouvons repartir de la définition de l'EI pour en chercher une formule analytique. Voyons le cas d'un hyperkrigeage à deux niveaux. L'EI (pour un problème de recherche de minimum) s'écrit ainsi :

$$\begin{aligned}
EI(x) &= \mathbb{E}\left(\max(0, Y_{\min} - \tilde{Z}^{(1)}(x))\right) \quad \text{où} \quad Y_{\min} = \min(Y^{(1)}) \\
&= \int_{-\infty}^{+\infty} \max(0, Y_{\min} - z)\, \mathrm{dP}\left[\tilde{Z}^{(1)}(x) = z\right] \mathrm{d}z \\
\text{(par 2.75)} \quad &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \max(0, Y_{\min} - z)\, \mathrm{dP}\left[Z^{(1)}(x,y) = z \mid Z^{(2)} = y\right] \mathrm{dP}\left[Z^{(2)}(x) = y\right] \mathrm{d}y\, \mathrm{d}z \\
&= \int_{-\infty}^{+\infty} \left(\int_{-\infty}^{+\infty} \max(0, Y_{\min} - z)\, \mathrm{dP}\left[Z^{(1)}(x,y) = z \mid Z^{(2)} = y\right] \mathrm{d}z \right) \mathrm{dP}\left[Z^{(2)}(x) = y\right] \mathrm{d}y \\
&= \int_{-\infty}^{+\infty} \mathbb{E}\left(\max(0, Y_{\min} - Z^{(1)}(x,y))\right) \mathrm{dP}\left[Z^{(2)}(x) = y\right] \mathrm{d}y \\
&= \int_{-\infty}^{+\infty} EI^{(1)}(x,y)\, \mathrm{dP}\left[Z^{(2)}(x) = y\right] \mathrm{d}y\ ,
\end{aligned} \tag{3.8}$$

avec $EI^{(1)}(x,y) = (Y_{\min} - \mu^{(1)}(x,y))\, \Phi\left(\frac{Y_{\min} - \mu^{(1)}(x,y)}{\sigma^{(1)}(x,y)}\right) + \sigma^{(1)}(x,y)\, \phi\left(\frac{Y_{\min} - \mu^{(1)}(x,y)}{\sigma^{(1)}(x,y)}\right)$. On peut appliquer la formule (2.79) pour avoir une approximation de l'EI :

$$EI(x) = EI^{(1)}(x, \mu^{(2)}(x)) + \sum_{k=1}^{+\infty} \frac{\sigma^{(2)}(x)^{2k}}{k!\ 2^k}\ \frac{\partial^{2k} EI^{(1)}}{\partial y^{2k}}(x, \mu^{(2)}(x))\ . \tag{3.9}$$

Les dérivées successives de $EI^{(1)}$ se déduisent de son expression en fonction de $\mu^{(1)}(x,y)$, $\sigma^{(1)}(x,y)$, et des dérivées de ces derniers que l'on peut déduire de leurs expressions en fonction du noyau k (dont les

dérivées sont supposées connues).

$$\begin{aligned} EI^{(1)}(x,y) &= (Y_{\min} - \mu^{(1)}(x,y))\,\Phi\Big(\tfrac{Y_{\min}-\mu^{(1)}(x,y)}{\sigma^{(1)}(x,y)}\Big) + \sigma^{(1)}(x,y)\,\phi\Big(\tfrac{Y_{\min}-\mu^{(1)}(x,y)}{\sigma^{(1)}(x,y)}\Big) \\ &= \sigma^{(1)}(x,y)\,\alpha(x,y)\,\Phi(\alpha(x,y)) + \sigma^{(1)}(x,y)\,\phi(\alpha(x,y)) \ , \end{aligned} \tag{3.10}$$

$$\begin{aligned} \tfrac{\partial EI^{(1)}}{\partial y} &= \tfrac{\partial \sigma^{(1)}}{\partial y}\,\alpha(x,y)\,\Phi(\alpha(x,y)) + \sigma^{(1)}(x,y)\,\tfrac{\partial \alpha}{\partial y}\,\Phi(\alpha(x,y)) + \sigma^{(1)}(x,y)\,\alpha(x,y)\,\tfrac{\partial \alpha}{\partial y}\,\phi(\alpha(x,y)) \\ &\quad + \tfrac{\partial \sigma^{(1)}}{\partial y}\,\phi(\alpha(x,y)) - \sigma^{(1)}(x,y)\,\tfrac{\partial \alpha}{\partial y}\,\alpha(x,y)\,\phi(\alpha(x,y)) \\ &= \tfrac{\partial \sigma^{(1)}}{\partial y}\,\alpha(x,y)\,\Phi(\alpha(x,y)) + \sigma^{(1)}(x,y)\,\tfrac{\partial \alpha}{\partial y}\,\Phi(\alpha(x,y)) + \tfrac{\partial \sigma^{(1)}}{\partial y}\,\phi(\alpha(x,y)) \\ &= \tfrac{\partial}{\partial y}\left(\sigma^{(1)}(x,y)\,\alpha(x,y)\right)\,\Phi(\alpha(x,y)) + \tfrac{\partial \sigma^{(1)}}{\partial y}\,\phi(\alpha(x,y)) \\ &= \left(Y_{\min} - \tfrac{\partial \mu^{(1)}}{\partial y}\right)\,\Phi(\alpha(x,y)) + \tfrac{\partial \sigma^{(1)}}{\partial y}\,\phi(\alpha(x,y)) \ , \end{aligned} \tag{3.11}$$

$$\begin{aligned} \tfrac{\partial^2 EI^{(1)}}{\partial y^2} &= \left(Y_{\min} - \tfrac{\partial^2 \mu^{(1)}}{\partial y^2}\right)\,\Phi(\alpha(x,y)) + \left(Y_{\min} - \tfrac{\partial \mu^{(1)}}{\partial y}\right)\,\tfrac{\partial \alpha}{\partial y}\,\phi(\alpha(x,y)) \\ &\quad + \tfrac{\partial^2 \sigma^{(1)}}{\partial y^2}\,\phi(\alpha(x,y)) - \tfrac{\partial \sigma^{(1)}}{\partial y}\,\alpha(x,y)\,\tfrac{\partial \alpha}{\partial y}\,\phi(\alpha(x,y)) \\ &= \left(Y_{\min} - \tfrac{\partial^2 \mu^{(1)}}{\partial y^2}\right)\,\Phi(\alpha(x,y)) + \left(\alpha(x,y)\tfrac{\partial \sigma^{(1)}}{\partial y} + \tfrac{\partial \alpha}{\partial y}\sigma^{(1)}(x,y)\right)\,\tfrac{\partial \alpha}{\partial y}\,\phi(\alpha(x,y)) \\ &\quad + \tfrac{\partial^2 \sigma^{(1)}}{\partial y^2}\,\phi(\alpha(x,y)) - \tfrac{\partial \sigma^{(1)}}{\partial y}\,\alpha(x,y)\,\tfrac{\partial \alpha}{\partial y}\,\phi(\alpha(x,y)) \\ &= \left(Y_{\min} - \tfrac{\partial^2 \mu^{(1)}}{\partial y^2}\right)\,\Phi(\alpha(x,y)) + \left(\left(\tfrac{\partial \alpha}{\partial y}\right)^2\,\sigma^{(1)}(x,y) + \tfrac{\partial^2 \sigma^{(1)}}{\partial y^2}\right)\,\phi(\alpha(x,y)) \ . \end{aligned} \tag{3.12}$$

Si la dérivée première de l'EI est simple (et peut être utilisée lors de la recherche de son maximum même dans un krigeage classique), les dérivées suivantes sont de plus en plus complexes. Nous n'avons pas mis en œuvre ces approximations dans nos applications. Nous avons choisi d'approximer l'intégrale par une méthode de Monte-Carlo.

3.3.1 Illustration de SoSo

Nous présentons ici un exemple d'utilisation de l'algorithme SoSo afin d'en illustrer le fonctionnement. Nous réutilisons la fonction Branin modifiée présentée au chapitre précédent, dans le cas *mesh based* sur 3 niveaux (nombre de nœuds à 100, 10000 et 10^8). Le plan d'expérience initial est constitué de 10 points répartis selon un *LHS* et évalués jusqu'au niveau 1. La condition d'arrêt de l'algorithme est fixé par un budget maximal de 10¤ (équivalent à 10 calculs complets). Les coûts de calcul pour chaque niveau sont 0,01¤, 0,1¤ et 0,89¤ (pour les niveau 3, 2 et 1 respectivement).

La figure 3.2 présente les points successivement ajoutés par SoSo. Les lignes de niveau des fonctions de chaque niveau sont également dessinées. Notez que les minima des niveaux 3 et 2 ne coïncident pas avec le minimum du niveau d'intérêt (le niveau 1).

On peut noter que les itérations sur les niveaux 2 et 3 ne sont pas nécessairement proches des minima de ces niveaux, puisque seul le minimum du niveau 1 est visé. Sur cet exemple, le point trouvé le plus proche du vrai minimum est (0,531 , 0,159). Il a été trouvé après un coût de 1,27¤, après quoi l'algorithme a continué de placer des points afin de chercher un meilleur minimum. Les points positionnés par SoSo visitent correctement les minima locaux du niveau 1. L'algorithme a ici placé 28 points au niveau 3, 10 points au niveau 2 et 9 points au niveau 1, pour un budget total de 9,29¤. La figure 3.3 montre comment ces points de différents niveaux ont été appelés au fur et à mesure des itérations. Sur cet exemple, de nombreuses évaluations sont restées au niveau 3 mais un seul point de niveau 2 n'a pas été poursuivi jusqu'au niveau 1. Comme pour un algorithme EGO, les premières itérations sont exploratoires, et dans notre cas l'algorithme se dispense de poursuivre ces calculs jusqu'au bout.

3.3.2 Efficacité de SoSo

Nous allons à présent évaluer l'efficacité de l'algorithme SoSo en comparant ses performances avec celles d'un EGO classique à travers une répétition de 50 optimisations indépendantes. Le plan d'expérience

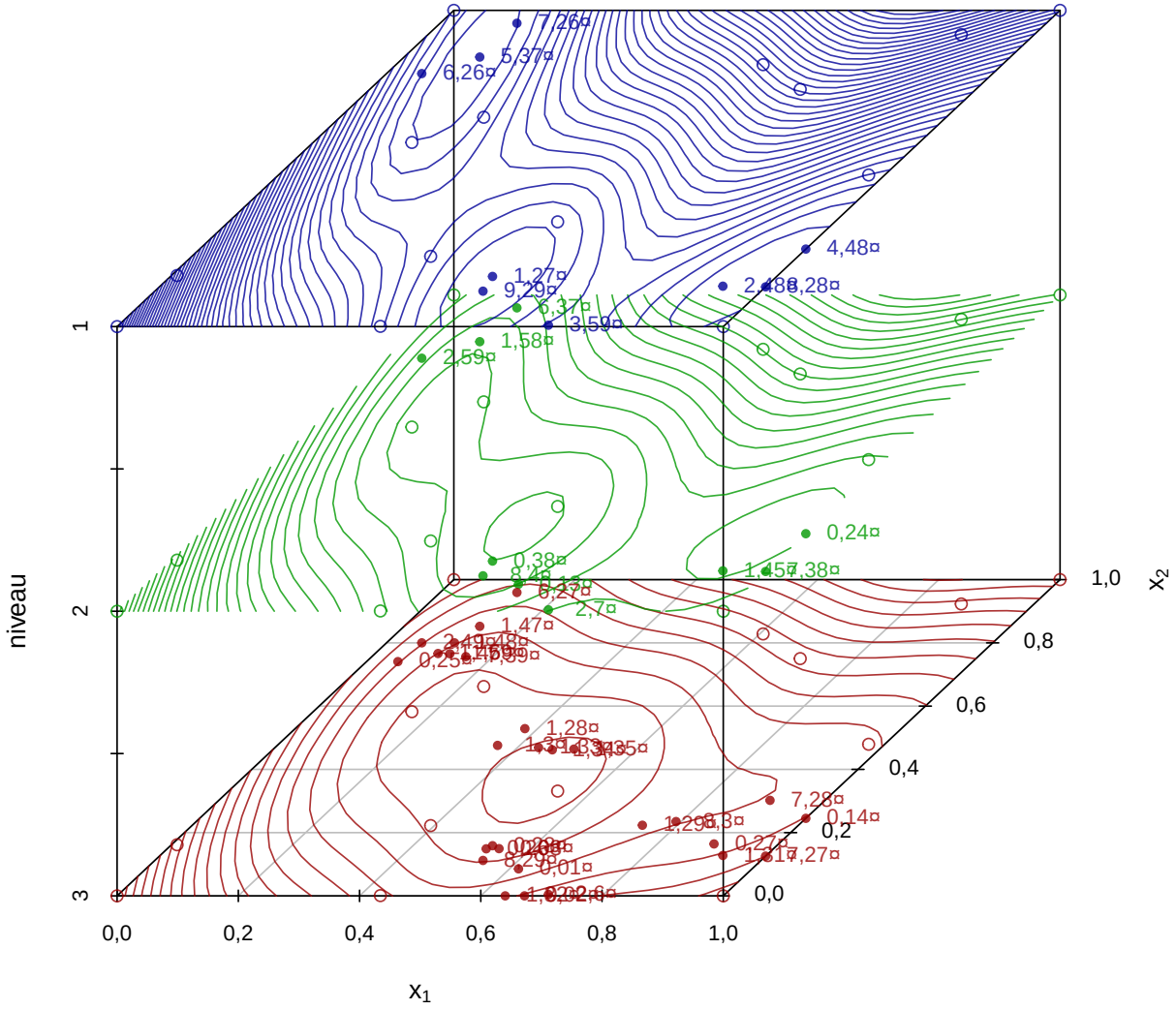


FIGURE 3.2 – Illustration du fonctionnement de SoSo sur un cas *mesh based* à 3 niveaux. Les cercles vides et pleins sont respectivement les points du plan d'expérience initial et les points ajoutés par l'algorithme. Les lignes de niveaux correspondent à la fonction de chaque niveau.

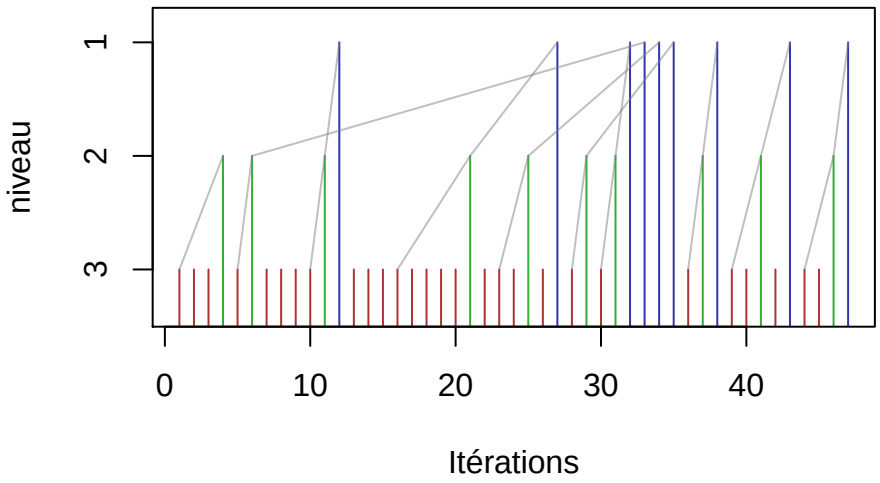


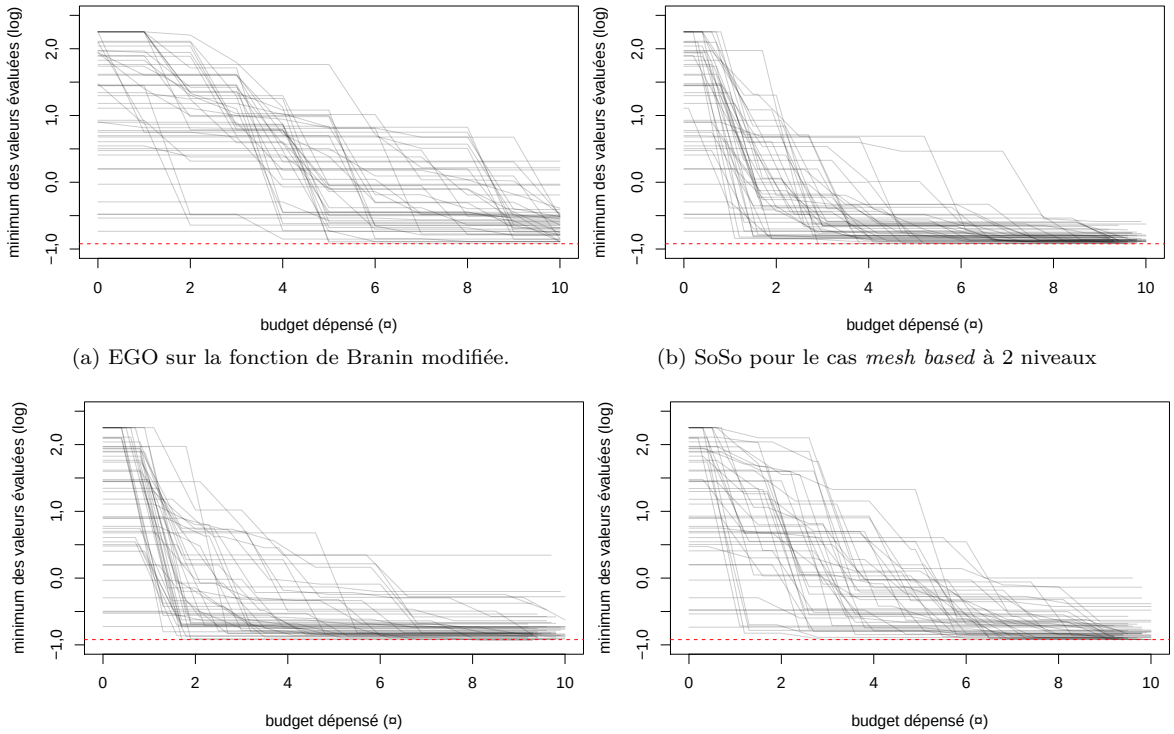
FIGURE 3.3 – Niveau des points calculés par l'algorithme SoSo aux cours des itérations. Les traits gris relient les points calculés en haute fidélité avec leur équivalent en plus basse fidélité.

initial sera différent pour chacun de ces 50 tests. Il s'agira à chaque fois de 10 points placés selon un *Latin Hypercube Sample* (optimisé par le critère *maximin*) et des 4 coins du domaine. Nous allons considérer les cas-tests utilisés à la section 2.4 : le cas *mesh based* qui simule les effets d'un raffinement spatial pour un solveur d'équations aux dérivées partielles, le cas Monte-Carlo qui représente les effets du changement d'échantillonnage dans une estimation Monte-Carlo, et le cas *time step* qui simule différentes étapes temporelles d'un simulateur. Dans tous ces cas, seulement deux sorties sont prises en compte ici, le niveau 1 et le niveau 3 qui ont été présentés en section 2.4. Ils ont tous pour niveau haute fidélité (grandeur d'intérêt) la fonction de Branin modifiée déjà utilisée (voir figure 2.18). On fixe ici que le calcul complet (niveau 3 plus niveau 1) coûte 1¤, tandis que le calcul du niveau 3 seul coûte 0,1¤. Le plan d'expérience initial est à chaque fois évalué sur tous les niveaux. Une implémentation standard à l'IRSN de l'algorithme EGO, basée sur DiceDesign et DiceKriging (Dupuy, Helbert, and Franco 2015 ; et Roustant, Ginsbourger, and Deville 2012), nous servira de référence.

Appliqué à la fonction de Branin modifiée, EGO converge tel que le montre la figure 3.4a. La convergence est étudiée ici sous la forme du meilleur minimum trouvé en fonction du budget dépensé. On prend le logarithme de la valeur afin d'accentuer visuellement les différences. Cette convergence de EGO est comparée à celles des algorithmes SoSo exploitant un niveau intermédiaire pour améliorer le processus d'optimisation. Les trois cas-tests sont utilisés séparément afin de voir les effets du type de niveau intermédiaire. Leurs résultats sont présentés sur les figures 3.4b (cas *mesh based*), 3.4c (cas Monte-Carlo) et 3.4d (cas *time step*). Pour les cas *mesh based* et Monte-Carlo, la version utilisée de SoSo s'appuie sur un modèle de cokrigeage de type LMC markovien (cf. 2.2.4), car celui-ci a le moins de paramètres. Dans le cas *time step* un modèle LMC symétrique a été utilisé car il est plus adapté à la structure du cas-test. En effet, pour ce cas-test, les niveaux intermédiaires ne sont pas des versions « dégradées » de la grandeur d'intérêt comme pour les autres cas-test. La logique de progression, plus complexe, est mieux appréhendée par un modèle symétrique, plus riche. On notera que dans le cas Monte-Carlo, les observations sont bruitées. Les observations sur le niveau 1 sont bruitées avec une variance à 10^{-4}. Le modèle utilisé considère ce bruit comme un bruit spatial (*nugget*) dont il estime sa valeur. En toute rigueur, il faudrait émuler ce bruit avec un *noise* fixé plutôt qu'un *nugget* estimé. Mais cela impliquerait de modifier la définition de l'EI (qui ne serait pas nulle aux points déjà calculés ; cf. (Picheny, Wagner, and Ginsbourger 2012)). Comme on impose de ne jamais revisiter un calcul déjà effectué, le modèle est incapable de voir la différence entre *noise* et *nugget*.

En comparant les courbes de convergence des figures 3.4b, 3.4c et 3.4d avec les courbes de convergences de EGO en figure 3.4a, on voit clairement que SoSo atteint les valeurs les plus faibles de la fonction-objectif plus rapidement et plus sûrement que EGO pour chacun des cas-tests. Ce résultat est dû à la capacité du cokrigeage à exploiter les informations provenant du niveau basse fidélité qui, bien que moins informatives, sont dix fois plus faciles à obtenir (dans notre exemple).

On peut également se demander si l'algorithme trouve le minimum global dans le bon bassin. Si on mesure la convergence des algorithmes par la distance du point le plus proche au vrai minimum global,



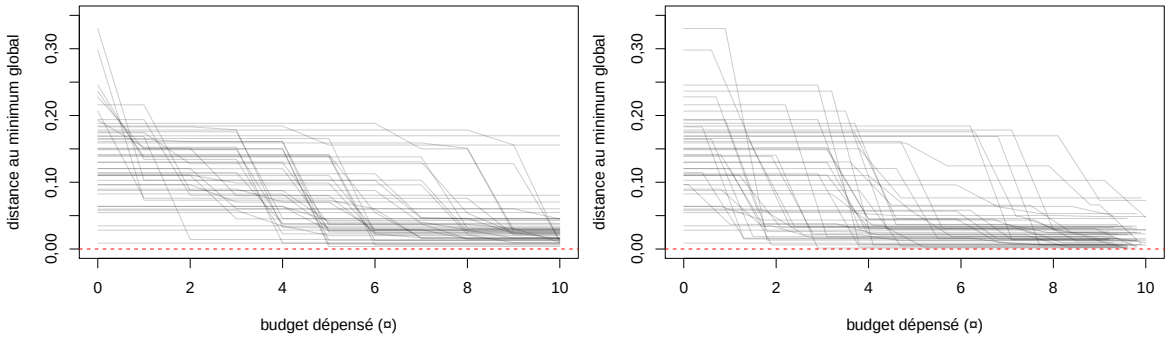
(a) EGO sur la fonction de Branin modifiée.

(b) SoSo pour le cas *mesh based* à 2 niveaux

(c) SoSo pour le cas Monte-Carlo à 2 niveaux

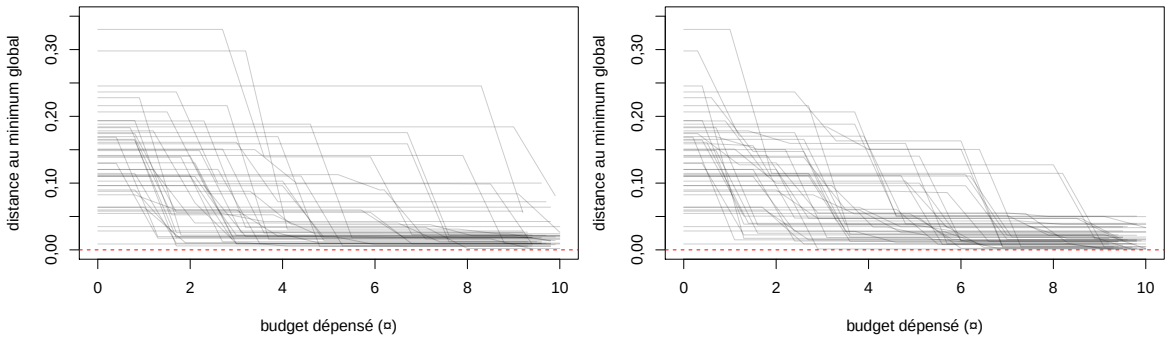
(d) SoSo pour le cas *time step* à 2 niveaux

FIGURE 3.4 – Convergence de EGO (a) et SoSo (b, c et d) : logarithme de la valeur minimale trouvée en fonction du budget dépensé, pour 50 situations initiales différentes. Le logarithme de la valeur minimale de la fonction de Branin modifiée est positionné en pointillé rouge.



(a) EGO sur la fonction de Branin modifiée.

(b) SoSo pour le cas *mesh based* à 2 niveaux



(c) SoSo pour le cas Monte-Carlo à 2 niveaux

(d) SoSo pour le cas *time step* à 2 niveaux

FIGURE 3.5 – Convergence de EGO (a) et SoSo (b, c et d) : plus petite distance au minimum global de la fonction de Branin modifiée, en fonction du budget dépensé, pour 50 situations initiales différentes. La distance nulle est positionnée en pointillé rouge.

on obtient la figure 3.5. Les courbes de convergence font moins ressortir les différences entre l'algorithme EGO et les autres, mais on distingue tout de même une amélioration dans les cas des algorithmes SoSo.

Il est également intéressant de comparer l'efficacité de EGO avec celle de SoSo pour un même plan d'expérience initial. C'est ce que résume la figure 3.6. Pour chacun des 50 plans d'expérience initiaux, on a comparé la convergence de EGO avec celle de SoSo dans chacun des cas-tests. On a calculé la différence entre le minimum trouvé par EGO et le minimum trouvé par SoSo pour différentes valeurs de budget (figures 3.6a, 3.6c et 3.6e). On a également calculé la différence entre la proximité au vrai minimum global trouvée par EGO et celle trouvée par SoSo (figures 3.6b, 3.6d et 3.6f). La synthèse des 50 situations initiales est donnée par des *boxplots*. On constate sur ces figures que l'algorithme SoSo permet très rapidement de trouver de meilleurs points à calculer. Bien que la différence en termes de proximité à la solution semble devenir rapidement négligeable pour les cas *mesh based* et Monte-Carlo (figures 3.6b et 3.6d), la valeur minimale trouvée reste meilleure, suggérant que les points à calculer sont tout de même mieux placés.

À travers ces exemples, on a couvert différentes possibilités de niveaux intermédiaires, pour des situations initiales randomisées. On a vu qu'à chaque fois l'algorithme SoSo faisait mieux que EGO. Il est vrai que ces résultats sont attendus sur un plan théorique puisque SoSo reprend le principe d'EGO en améliorant le processus gaussien et en ajoutant une possibilité supplémentaire d'explorer l'espace des entrées à faible coût de simulations. Il est néanmoins toujours possible de rencontrer des situations particulières, où les paramètres du modèles seraient mal estimés pendant quelques itérations par exemple ce qui pourrait faire perdre de l'efficacité à l'algorithme SoSo. Il est impossible de couvrir de manière exhaustive tous les cas de figure de fonction-objectif ou de niveau intermédiaire que peut rencontrer un ingénieur. Néanmoins l'algorithme SoSo représente tout de même une amélioration significative par rapport à EGO. La section suivante présente une application de SoSo à un problème d'ingénierie.

3.3.3 Application pratique en sûreté nucléaire

Cette application concerne un système de stockage de poudre de plutonium. Les phénomènes physiques sont simulés par une chaîne de codes combinant un premier niveau de prétraitement chimique (APOLLO) puis un calcul de criticité (MORET). Le schéma 3.7 résume la chaîne de calcul utilisée. La grandeur finale d'intérêt est le k-effectif, qui représente la capacité du système étudié à produire une réaction en chaîne incontrôlée. Fondamentalement, cette valeur doit être maintenue en deça de la valeur limite 1,0 qui, lorsqu'elle est dépassée, caractérise une population neutronique croissante, induite par les interactions de matières fissiles. La limite de sûreté usuellement utilisée peut ainsi être $k_{eff} < 0{,}95$, 0,97 ou 0,98, afin d'éviter tout risque d'emballement de la réaction en chaîne.

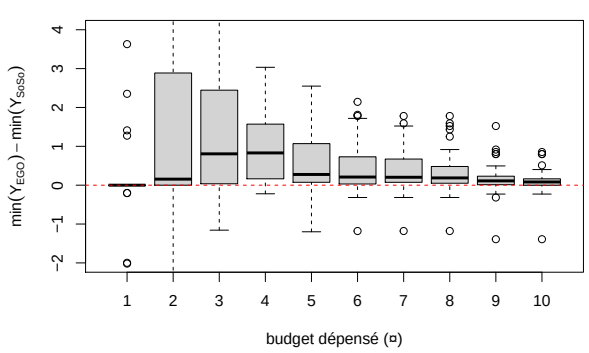
Le stockage de poudre de plutonium (PuO2) considéré est un ensemble infini de tubes intégrants deux conteneurs de poudre superposés (figure 3.8a) dont la conception est contrôlée par les paramètres suivants :

- la géométrie (des tubes, des conteneurs) ;
- l'interaction (espacement des tubes) ;
- la masse (de Pu par conteneur) ;
- la modération (teneur en eau à l'intérieur de la poudre) ;
- la densité (maximale de poudre) ;
- l'enrichissement (en Pu 239, Pu 240, Pu 241 du contenu).

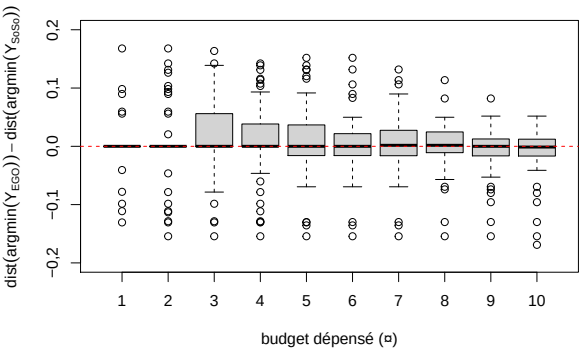
Ces paramètres sont fixés pour le problème donné ; la figure 3.8b en décrit les valeurs. Ils sont supposés garantir les conditions de sûreté, c'est-à-dire ici $k_{eff} < 0{,}98$, quels que soient les autres paramètres, non contrôlés, qui sont :

- la densité réelle de poudre de Pu, qui peut varier selon les conditions opérationnelles ;
- la modération par l'eau interstitielle entre les tubes, qui pourrait évoluer par exemple en cas d'aspersion du stockage.

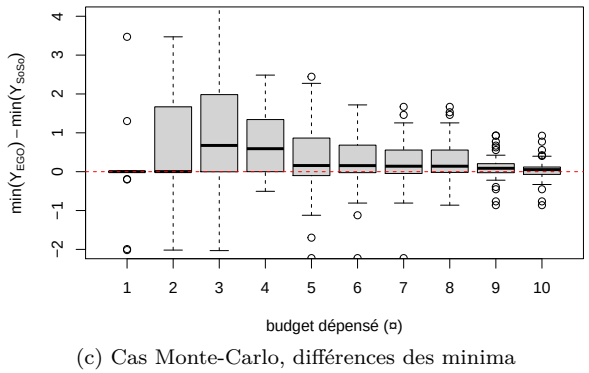
L'objectif de l'ingénieur en criticité est de vérifier si les paramètres contrôlés annoncés respectent les conditions de sûreté. Il doit donc faire varier les paramètres non contrôlés et rechercher le pire cas possible, celui qui maximise k_{eff}, pour savoir si le stockage de plutonium considéré, défini par les paramètres contrôlés, est sûr.



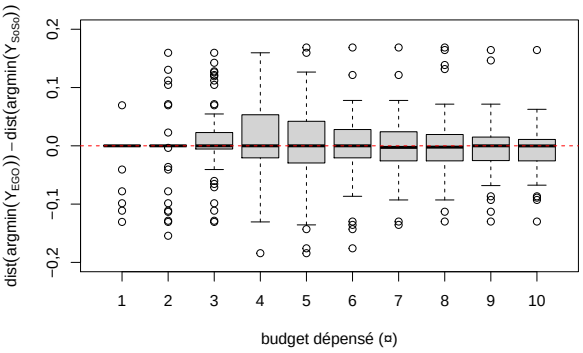
(a) Cas *mesh based*, différences des minima



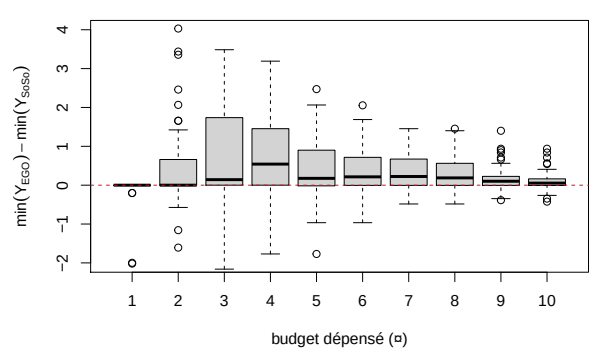
(b) Cas *mesh based*, différence des distances au minimum global



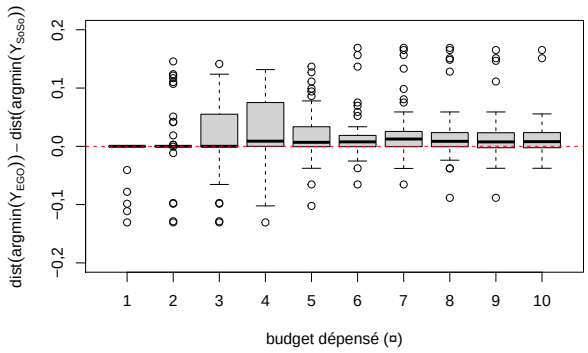
(c) Cas Monte-Carlo, différences des minima



(d) Cas Monte-Carlo, différence des distances au minimum global



(e) Cas *time step*, différences des minima



(f) Cas *time step*, différence des distances au minimum global

FIGURE 3.6 – Différence entre l'algorithme EGO et les algorithmes SoSo, pour chacun des plans d'expérience initiaux, en fonction du budget. À gauche, on calcule le minimum trouvé par EGO moins le minimum trouvé par SoSo. À droite, on calcule la plus petite distance au vrai minimum global trouvée par EGO moins celle trouvée par SoSo.

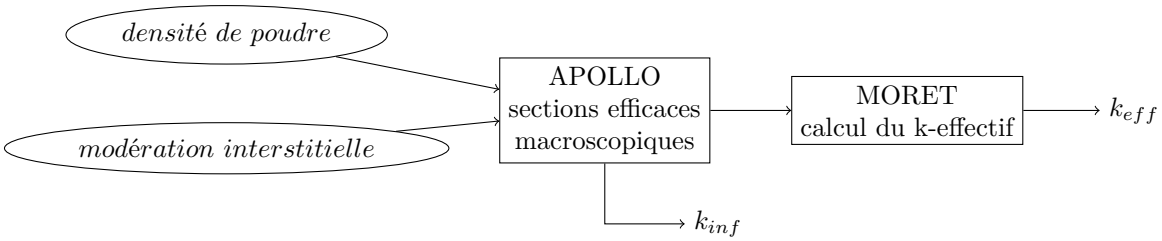


FIGURE 3.7 – Schéma de la chaîne de calcul Apollo-Moret.

Nous allons donc effectuer une étude d'optimisation. Les 2 variables d'entrée étudiées sont la densité de poudre de PuO2 (comprise entre 0,2 et 4,0 $g{\cdot}cm^{-3}$) et la concentration en eau de l'espace interstitiel autour des tubes (comprise entre 10^{-4} et 1,0 $g{\cdot}cm^{-3}$). La concentration en eau est transformée pour dilater les très faibles densités (on parle usuellement de « brouillard » d'eau), selon une échelle logarithmique, car la réactivité neutronique est connue pour être très sensible aux faibles modérations [2] dans ce type de configuration. La grandeur scalaire finale, sur laquelle porte l'optimisation, est le k-effectif du système complet (3.11), tandis que la sortie intermédiaire considérée par le cokrigeage est le k-infini du milieu fissile (3.10), calculé par APOLLO sur la base des mêmes données que celles transmises à MORET (les sections efficaces macroscopiques des milieux chimiques présents, voir 3.9). De manière simplifiée, ce k-infini revient au k-effectif du milieu fissile modéré (tube et modération interstitielle), pris dans un cylindre de longueur infinie, sans conteneur ni frontière.

Mathématiquement, le problème s'écrit ainsi :

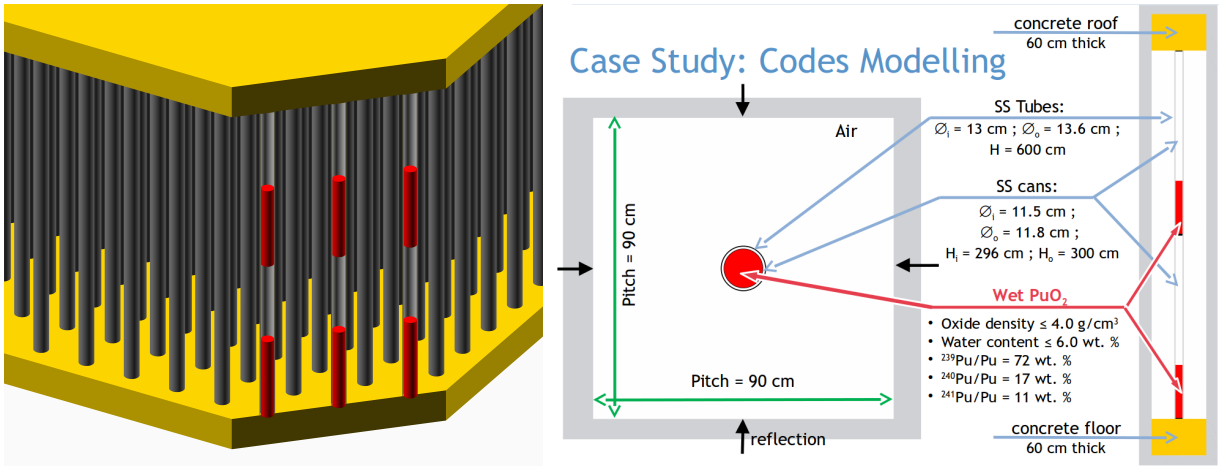
$$\max_{x_1\in[0,2;4],x_2\in[10^{-4};1]} k_{eff}(x_1,x_2)\ . \tag{3.13}$$

Les résultats des simulations sont métamodélisés avec un modèle LMC markovien (cf. section 2.2.4). La surface de réponse est théoriquement dérivable sur la majeure partie du domaine, sauf lorsque la densité de poudre franchit un seuil qui correspond à excéder le volume de son conteneur. Nous faisons alors un compromis en choisissant un noyau Matérn 3/2 (cf. section 1.2.1), une fois dérivable. Les noyaux Matérn 5/2 ou exponentiels carrés seraient trop lisses pour la région non-dérivable, le noyau exponentiel serait non différentiable donc particulièrement irrégulier dans tout le domaine. La recherche du maximum est effectuée à l'aide de l'algorithme SoSo (cf. section 3.3). L'initialisation se base sur un plan d'expériences de 10 points en LHS augmenté des 4 coins du domaine. Les figures 3.12 présentent cette initialisation, la position et la valeur des points y sont donnés, ainsi que la moyenne de prédiction du modèle à ce stade. Nous nous sommes donné un budget de 20 calculs complets (en plus de l'initialisation). Nous avons également considéré que le calcul du k-infini prend un dixième du temps (0,1 pour un calcul APOLLO et 0,9 pour un calcul MORET). Ceci est approximativement en accord avec ce que les experts observent, les durées relatives d'APOLLO et de MORET varient beaucoup selon la complexité du problème.

Les résultats de l'algorithme SoSo sont donnés sur la figure 3.13. En plus des points de l'initialisation, on a placé les points des différentes itérations (la progression est représentée par un dégradé de couleurs, du rouge au vert). On constate qu'après quelques itérations exploratoires, l'algorithme trouve un pic qu'il affine progressivement. Les figures 3.13c et 3.13d montrent des projections sur deux directions différentes d'un zoom sur le maximum. Ce pic est situé dans une petite portion du domaine et dépasse le seuil de criticité fixé (0,98).

La figure 3.14 montre les itérations successives de l'algorithme, avec le niveau et la position des points placés, et la valeur du k-effectif des calculs MORET. Un trait gris facilite le suivi des mêmes points évalués aux deux niveaux. La valeur critique est dépassée pour la première fois à la $23^{ème}$ itération, correspondant à un budget dépensé de 11,1. Les calculs suivants améliorent cette valeur maximale en recherchant dans la même zone. Au total, il y a 3 calculs APOLLO pour lesquels l'algorithme SoSo n'a pas donné suite. SoSo réalise ainsi une économie d'au moins 3 calculs complets, et probablement plus du fait du gain en précision du métamodèle à deux niveaux. Notons aussi que parmi les 19 calculs complets,

2. Un matériau « modérateur » est un matériau non-fissible, qui influence la neutronique du milieu dans un sens ou dans l'autre. L'eau est un modérateur très courant et très influent.



(a) Assemblage des tubes

(b) Schéma détaillé d'un tube

FIGURE 3.8 – Schéma de la configuration du stockage de poudre de plutonium considéré.

12 correspondent à un enchaînement direct d'APOLLO puis de MORET quand 7 ont un appel à MORET qui est différé.

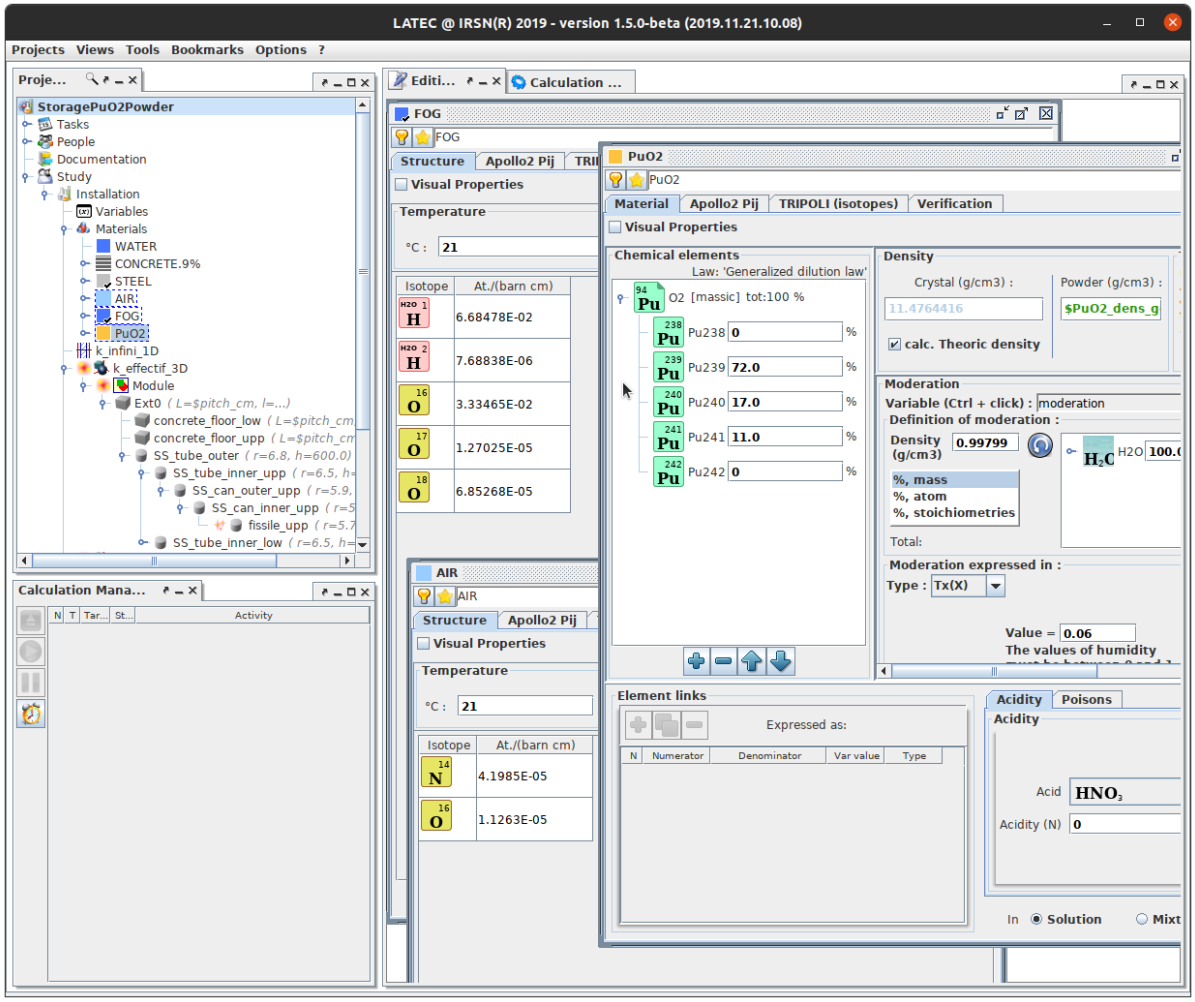


FIGURE 3.9 – Visualisation des modèles chimiques de stockage de poudre de plutonium dans le logiciel LATEC de l'IRSN, qui permet le calcul des sections efficaces macroscopiques transmises à MORET.

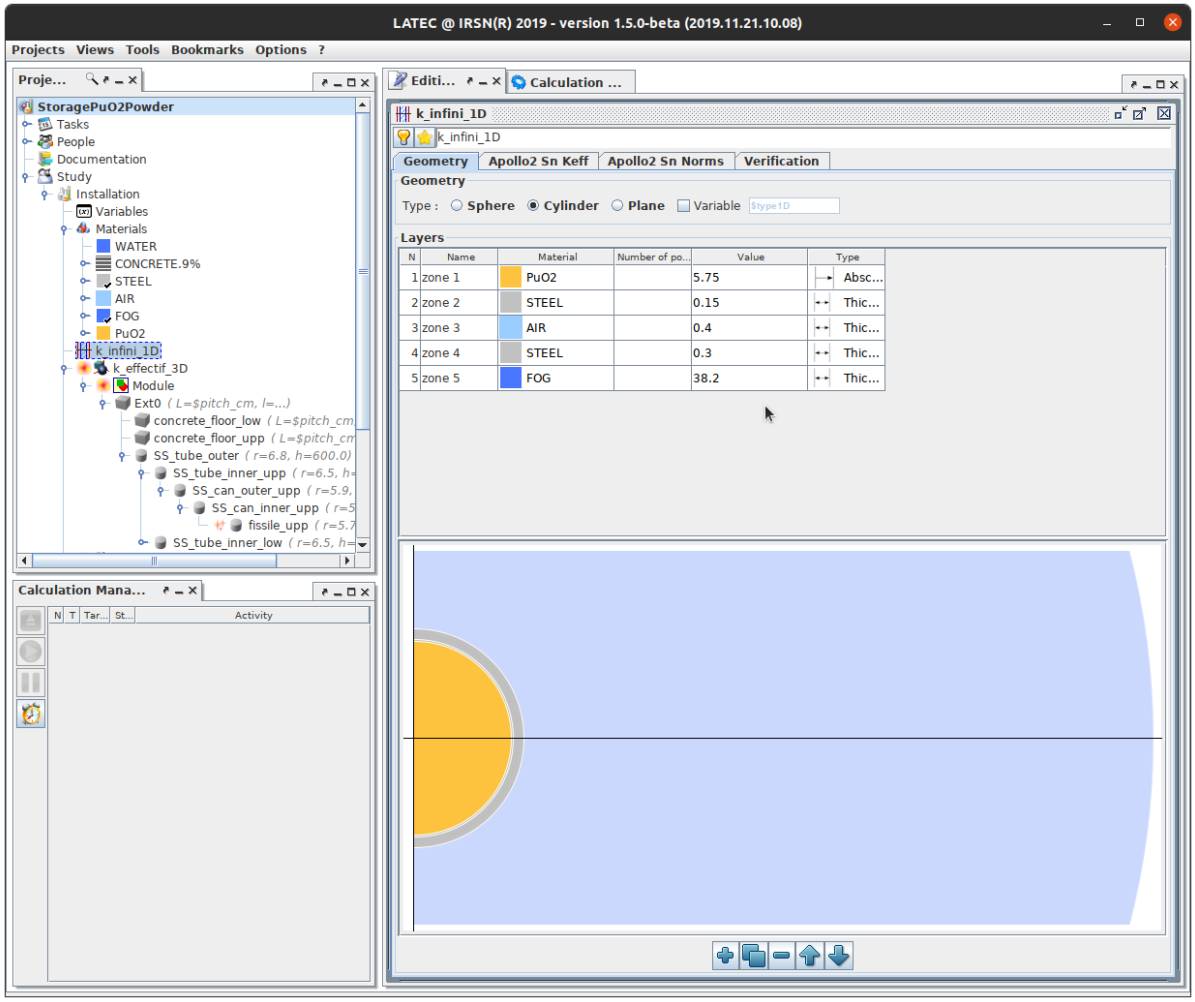


FIGURE 3.10 – Modèle 1D de stockage de poudre de plutonium considéré, qui permet le calcul du k-infini.

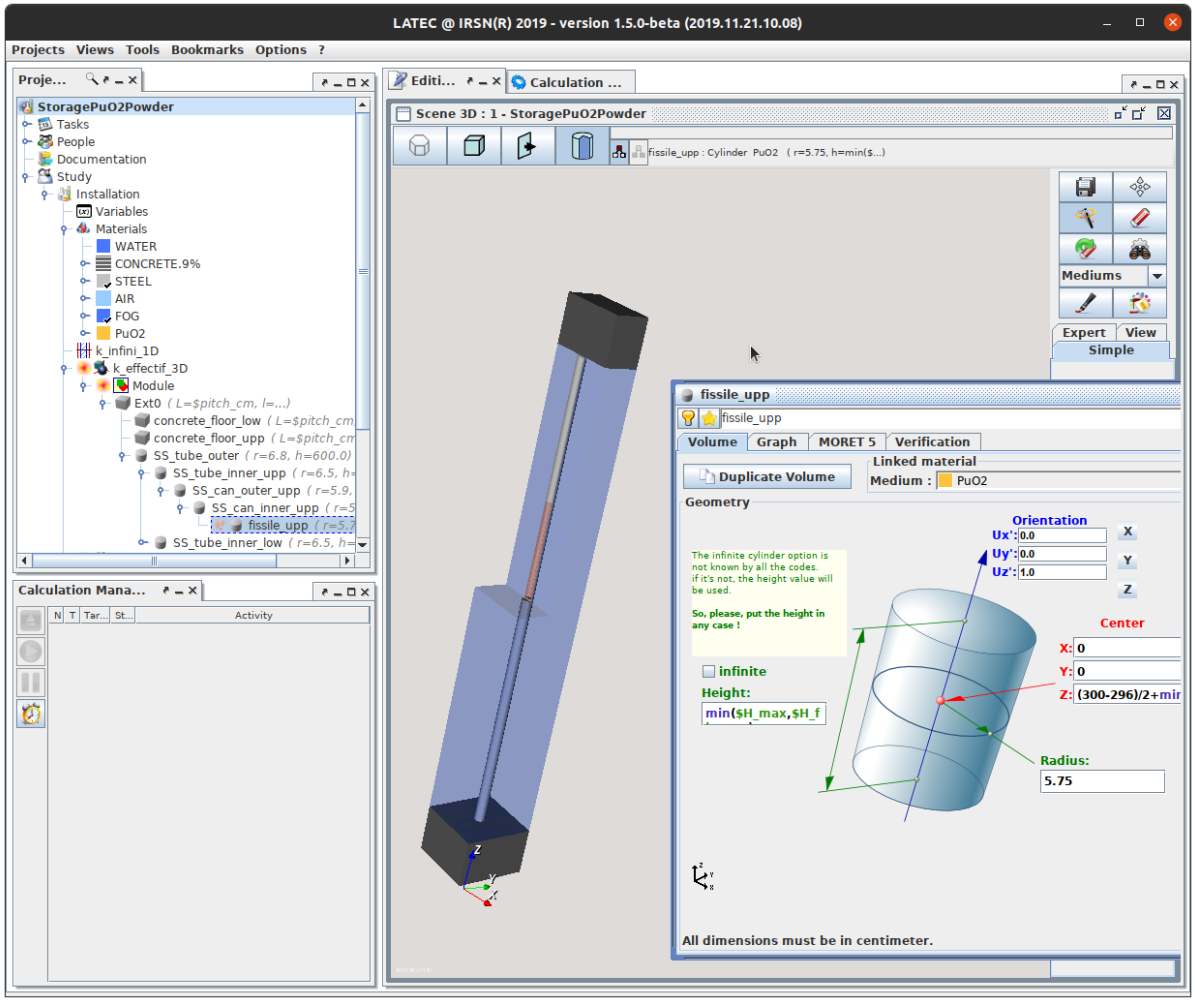
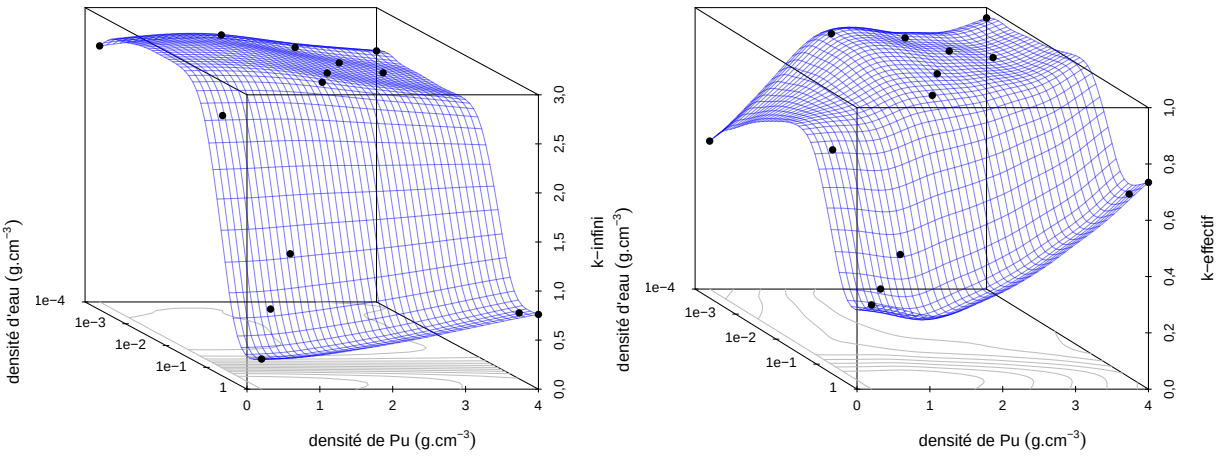


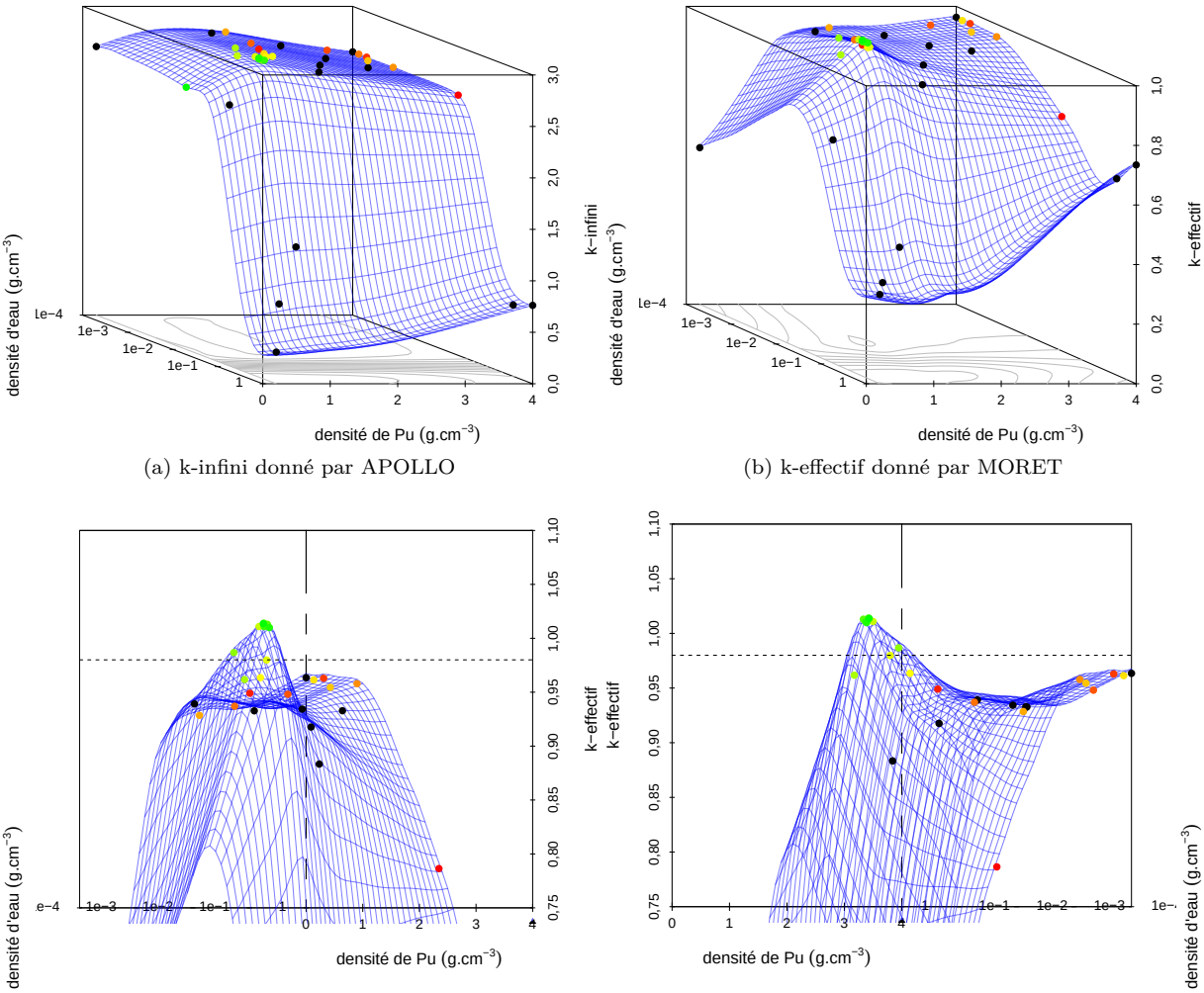
FIGURE 3.11 – Modèle 3D de stockage de poudre de plutonium considéré, qui permet le calcul du k-effectif.



(a) k-infini donné par APOLLO

(b) k-effectif donné par MORET

FIGURE 3.12 – Plan d'expériences pour initialiser l'algorithme itératif. Le plan est composé de 10 points en LHS et des 4 coins du domaine (points noirs). La moyenne de krigeage du modèle construit sur ce plan initial est représentée en bleu.



(a) k-infini donné par APOLLO

(b) k-effectif donné par MORET

(c) k-effectif, vu en projection sur la diagonale.

(d) k-effectif, vu en projection sur l'autre diagonale.

FIGURE 3.13 – Résultat de l'algorithme itératif SoSo. Le plan d'expériences initial est composé des points noirs. Les points de couleur rouge à verte sont les points ajoutés au cours des itérations, les points les plus rouges étant les premiers, les points les plus verts étant les derniers. La moyenne de krigeage du modèle de la dernière itération est dessinée en bleu. En pointillé, sur les deux graphes du bas, se trouve la limite de sûreté (0,98).

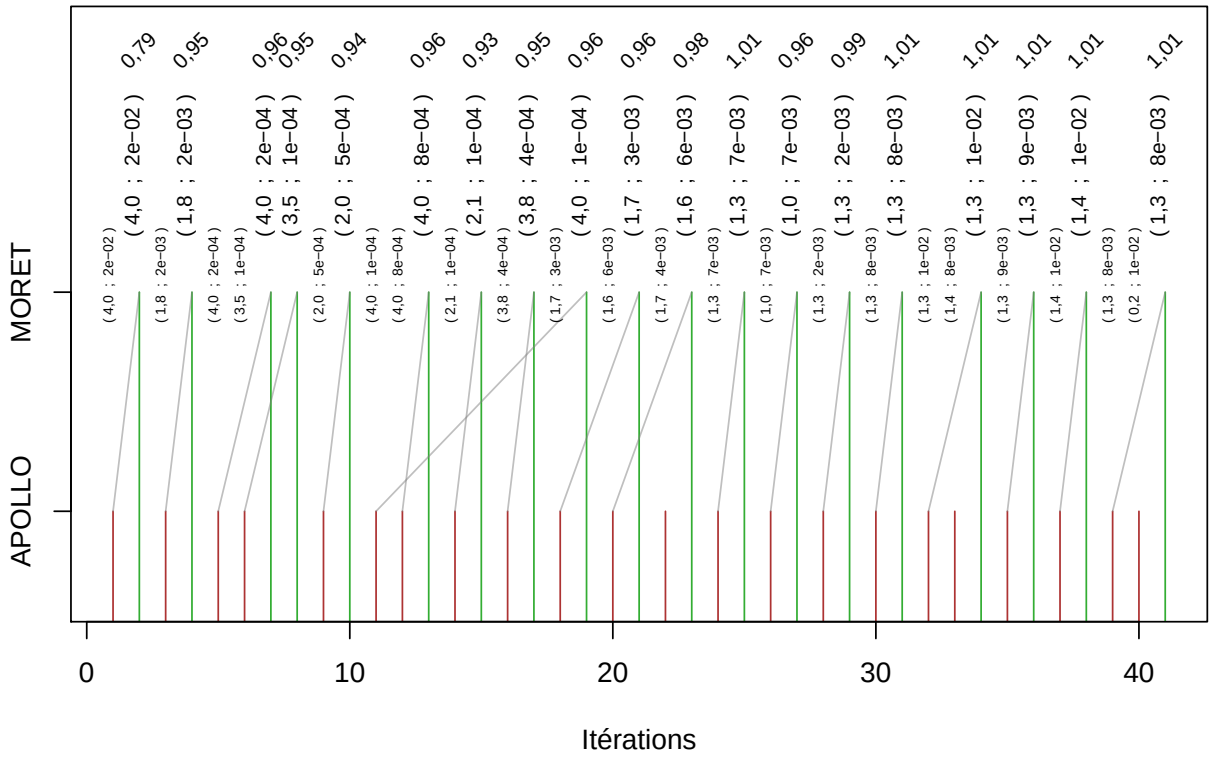


FIGURE 3.14 – Nouveaux points calculés par l'algorithme SoSo au cours des itérations. Les traits gris relient les points calculés par MORET avec leur équivalent calculé par APOLLO. Les positions des points sont également indiquées, ainsi que la valeur du k-effectif le cas échéant.

3.4 Stratégie *Step or Stop* avec d'autres critères

Dans la section précédente nous avons vu comment adapter l'algorithme EGO au cas multisortie. D'autres algorithmes itératifs, construits sur le même modèle que EGO, peuvent également bénéficier de la même adaptation, et un exemple générique de l'algorithme SoS a été décrit en algorithme 3.1.

Pour un problème d'inversion par exemple, les critères de Ranjan et Bichon (cf. section 1.3.2), définis par une espérance, peuvent être transformés en critères à ratio temporel pour les mêmes raisons que pour l'EI :

$$crit_{RB}(x) \;=\; \mathbb{E}\left[\max(0, \alpha^{\delta}\sigma^{(1)}(x)^{\delta} - |T - \mu^{(1)}(x)|^{\delta}\right]$$

$$\rightarrow RB_{SoS}(x) \;=\; \frac{crit_{RB}(x)}{T_{g(x)-1}} \;. \tag{3.14}$$

Le critère ainsi créé permet de construire un algorithme *Step or Stop inversion* (SoSi) exploitant la stratégie SoS (algorithme 3.1).

Les algorithmes de type SUR (*Stepwise Uncertainty Reduction*) fonctionnent également de manière itérative en recherchant le maximum d'un critère basé sur le métamodèle (cf. 1.3). Les critères de type SUR sont construits en définissant une mesure de l'incertitude sur la réponse à l'objectif d'ingénierie, puis en recherchant le point qui va réduire au mieux cette incertitude. Il est possible de considérer la réduction espérée par unité de temps et de construire ainsi un critère $SURR$ (*Stepwise Uncertainty Reduction Rate*), de la même manière qu'on a construit l'EIR. Toutefois un calcul partiel peut déjà apporter une réduction d'incertitude sur le niveau d'intérêt. On aurait donc un critère un peu différent, où le taux d'amélioration est calculé avec le temps nécessaire à la prochaine étape mais pas le temps total nécessaire à une exécution complète de la chaîne de calcul :

$$SUR(x) \;=\; \underset{y}{\mathbb{E}}\left(-uncertainty(M^{(1)}_{+(x,y)})\right) \;,$$

avec $M^{(1)}_{+(x,y)}$ le métamodèle pour le niveau 1 sur les données $(\boldsymbol{X}, \boldsymbol{Y}) \cup (x, y)$, et $y \sim Z^{(g(x)-1)}(x)$,

$$\Rightarrow \quad SURR(x) \;=\; \frac{SUR(x)}{t_{g(x)-1}} \tag{3.15}$$

Afin d'éviter une myopie préjudiciable, il est intéressant d'estimer la réduction d'incertitude apportée par toutes les étapes jusqu'au niveau 1. On construit ainsi un critère $SCURR$ (*Stepwise Complete Uncertainty Reduction Rate*) :

$$SCURR(x) \;=\; \underset{\boldsymbol{y}}{\mathbb{E}}\left(-uncertainty\left(M^{(1)}_{+(\boldsymbol{x},\boldsymbol{y})}\right)\right) \Big/ \; T_{g(x)-1} \;, \tag{3.16}$$

avec $M^{(1)}_{+(\boldsymbol{x},\boldsymbol{y})}$ le métamodèle pour le niveau 1 sur les données $(\underline{\boldsymbol{X}}, \boldsymbol{G}, \boldsymbol{Y}) \cup \left(x, g(x)-1, y^{(g(x)-1)}\right) \cup \ldots \cup \left(x, 1, y^{(1)}\right)$, et $\boldsymbol{y} \sim \boldsymbol{Z}^{(g(x)-1,\ \ldots,\ 1)}(x)$.

$SCURR(x)$ peut être mis en compétition avec $SURR(x)$. Comme l'espérance porte sur plusieurs variables ($\boldsymbol{y}$ suit une loi normale multivariée), et en l'absence de formule analytique (qui dépend de la mesure d'incertitude que l'on s'est donnée), le critère $SCURR$ devient difficilement calculable. L'algorithme qui en découle est de plus légèrement différent de SoS et nous n'avons pas mis en œuvre ce critère au cours de cette thèse.

Conclusions et perspectives

L'objectif de cette thèse est d'apporter des améliorations à des algorithmes utilisés pour piloter des simulations physiques. Les simulations physiques considérées pour ces travaux sont constituées de plusieurs codes de calcul qui se succèdent, et certaines grandeurs produites par les premiers codes sont corrélées avec la grandeur finale. Il s'agit d'un cas très répandu dans les milieux industriels, mais les informations intermédiaires ne sont pas toujours prises en compte. Ce sont ces informations supplémentaires qui font l'objet des améliorations étudiées ici.

Indépendamment de cela, on considère les simulations comme coûteuses et leurs résultats sont valorisés à travers un métamodèle capable d'émettre des prédictions sur de futurs simulations. Les métamodèles de krigeage sont présentés en section 1.1. Différents choix peuvent être fait lors de la modélisation statistique qui conduisent à autant de métamodèles différents, comme discuté en section 1.2. Le métamodèle retenu est exploité par un algorithme afin de déterminer itérativement les prochaines simulations physiques à effectuer. Ces algorithmes, notamment pour l'optimisation, sont présentés en section 1.3.

Les améliorations recherchées concernent l'intégration, dans les métamodèles et les algorithmes itératifs, d'informations supplémentaires disponibles issues des premières étapes de calcul des simulations physiques. Le chapitre 2 se concentre sur les métamodèles introspectifs, c'est-à-dire capables d'exploiter ces informations supplémentaires pour affiner leurs prédictions. Nous étudions une famille de métamodèle, le cokrigeage, en section 2.2. En particulier nous nous intéressons aux cokrigeages de type LMC (section 2.2.2) formés comme une combinaison linéaire de processus indépendants, les variables latentes. Nous apportons une explicitation des variables latentes après conditionnement en section 2.2.2. Ces expressions pourraient permettre une approche systémique de l'interprétation du métamodèle, mais davantage de travaux seraient nécessaires pour en valider la pertinence. Les modèles LMC sont très souples et nous avons établi qu'il est possible de construire un modèle LMC en adéquation avec la structure de la chaîne de codes en section 2.2.5. De futurs travaux pourraient étudier l'implémentation d'un LMC générique, adaptable à une structure quelconque, et les gains qui en résultent. Nous avons également découvert en 2.2.7 une formule quasi-analytique pour l'estimation par maximum de vraisemblance de certains paramètres des modèles de cokrigeage. Il reste cependant à étudier comment l'incertitude de ces paramètres peut être répercutée sur le modèle. En section 2.3, nous développons un métamodèle introspectif différent que nous avons nommé hyperkrigeage. Celui-ci présente des difficultés pour sa mise en pratique, partiellement surmontées en section 2.3.1 à l'aide d'approximations. Une implémentation propre et une analyse poussée de l'hyperkrigeage approximé pourraient améliorer la compétitivité de ce métamodèle. L'idée de l'hyperkrigeage croisé, introduite en section 2.3.2, pourrait alors être développée. Nous n'avons fait qu'effleurer cette idée. Il faudrait s'interroger sur l'efficacité de ce nouveau modèle, plus flexible mais plus compliqué à appréhender, selon les cas d'application. Une comparaison avec les autres modèles de cokrigeage est présentée en section 2.4. Le métamodèle d'hyperkrigeage mériterait qu'on s'y attarde car dans nos expériences ses prédictions sont souvent très précises.

Le chapitre 3 se concentre sur les algorithmes introspectifs, qui exploitent les métamodèles introspectifs, mais qui eux-même utilisent la structure nivelée des codes de calcul pour améliorer leur efficacité. Nous avons mis au point une stratégie générique, nommée « SoS », qui tire parti de la possibilité de ne faire certains calculs que partiellement, expliquée en section 3.2. Nous avons ensuite décliné cette stratégie en un algorithme d'optimisation « SoSo » à l'aide d'un critère EIR développé pour l'occasion (cf. section 3.3). SoSo a été appliqué à un cas industriel fourni par l'IRSN (section 3.3.3). La stratégie SoS peut également être appliquée à d'autres problématiques, comme par exemple les problèmes d'inversion (section 3.4). L'application aux problèmes d'inversion semble prometteuse et sa mise en œuvre pourrait faire l'objet de prochains travaux.

De manière générale, il existe beaucoup de travaux sur le krigeage permettant d'adapter le principe de la métamodélisation et du critère d'acquisition pour prendre en compte des connaissances supplémentaires, des expertises, propres au problème considéré. Les travaux présentés dans ce manuscrit entrent dans cette catégorie pour le cas fréquent des chaînes de calcul. Les modèles et algorithmes développés peuvent s'utiliser en conjonction avec la plupart des autres améliorations du krigeage et de EGO, comme par exemple le krigeage à grand nombre de points (Rullière et al. 2018) ou les algorithmes EGO avec recherches locales (Eriksson et al. 2019).

Annexe A

Vraisemblance concentrée (cas classique)

Pour trouver les paramètres du métamodèle, il est courant de chercher à maximiser la vraisemblance du métamodèle par rapport aux données. Certains de ces paramètres ont un maximum qui peut être écrit de façon analytique. Pour les autres paramètres, une optimisation par un algorithme adapté est nécessaire.

Nous reprenons les notations du chapitre I. Pour un processus gaussien univarié, la vraisemblance des données qui le conditionnent s'écrit (équation (1.14)) :

$$L(\beta, \sigma_0^2, \Theta) = \frac{1}{(2\pi)^{N/2} \det(K)^{1/2}} \exp\left(-\frac{1}{2}\ {}^t(Y - m(X))\, K^{-1}\, (Y - m(X))\right) \ . \tag{A.1}$$

$m(\cdot)$ est la moyenne du processus gaussien avant conditionnement, qu'on appelle aussi la tendance. On la considère comme une combinaison linéaire de fonctions définies par l'utilisateur $f_1(\cdot),\ f_2(\cdot),\ \ldots,\ f_p(\cdot)$. Les paramètres de cette combinaison linéaire sont à estimer et forment le vecteur β. Appliquée aux données X, les fonctions de la tendance forment la matrice $F = [f_1(X)\ f_2(X)\ \ldots\ f_p(X)]$, de taille $N \times p$. On a alors :

$$m(X) = F\beta \tag{A.2}$$

Par ailleurs, σ_0 est un paramètre d'échelle qui vient en facteur de la fonction de covariance $k(\cdot, \cdot)$. Il peut donc être factorisé hors de la matrice $K = k(X, X)$ et on a la relation : $K = \sigma_0^2 R$. Puisque K est de taille $N \times N$ on a la relation entre les déterminants : $\det(K) = \sigma_0^{2N} \det(R)$. On s'intéresse à présent au logarithme népérien de la vraisemblance, la log-vraisemblance, dont le maximum coïncide avec celui de la vraisemblance :

$$\begin{aligned} \mathcal{L}(\beta, \sigma_0^2, \Theta) &= \ln(L(\beta, \sigma_0^2, \Theta)) \\ &= -\tfrac{N}{2}\ln(2\pi) - \tfrac{1}{2}\ln(\det(R)) - \tfrac{N}{2}\ln(\sigma_0^2)) - \tfrac{1}{2\sigma_0^2}\ {}^t(Y - F\beta)\, R^{-1}\, (Y - F\beta) \ , \end{aligned} \tag{A.3}$$

Paramètres de tendance maximisants la vraisemblance, $\hat{\beta}$

On cherche d'abord les paramètres $\hat{\beta}$ qui maximisent la log-vraisemblance. On commence par chercher les valeurs pour lesquelles les dérivées s'annule et on doit résoudre :

$$\left\{ \frac{\partial \mathcal{L}}{\partial \beta_l}(\beta = \hat{\beta}) = 0 \right\}_{l \in [1 \ldots p]} \ , \tag{A.4}$$

ce qui nous amène à :

$$\left\{ -\frac{1}{\sigma_0^2} \, {}^t(-Fe_l) R^{-1} \left(Y - F\hat{\beta} \right) = 0 \right\}_{l \in [1 \dots p]} , \tag{A.5}$$

avec e_i le vecteur ayant un 1 à la l-ième position et 0 partout ailleurs. On peut alors écrire le système

$$\left\{ {}^t e_l \, {}^t F R^{-1} \left(Y - F\hat{\beta} \right) = 0 \right\}_{l \in [1 \dots p]} , \tag{A.6}$$

sous une forme matricielle :

$$ {}^t F R^{-1} \left(Y - F\hat{\beta} \right) = 0 . \tag{A.7}$$

Cela nous conduit à :

$$ {}^t F R^{-1} F \hat{\beta} = {}^t F R^{-1} Y . \tag{A.8}$$

Et on en déduit :

$$\hat{\beta} = \left({}^t F R^{-1} F \right)^{-1} \, {}^t F R^{-1} Y . \tag{A.9}$$

Les dérivées de $\mathcal{L}$ par rapport à β ne s'annulent qu'en un seul endroit. Les dérivées secondes valent :

$$\left\{ \frac{\partial^2 \mathcal{L}}{\partial \beta_k \partial \beta_l} = -\frac{1}{\sigma_0^2} \, {}^t(-Fe_l) R^{-1} (-Fe_k) \right\}_{k,l \in [1 \dots p]} , \tag{A.10}$$

$$\left\{ \frac{\partial^2 \mathcal{L}}{\partial \beta_k \partial \beta_l} = -\frac{1}{\sigma_0^2} \left\{ (\, {}^t F R^{-1} F) \right\}_{l,k} \right\}_{k,l \in [1 \dots p]} , \tag{A.11}$$

La hessienne $(-\frac{1}{\sigma_0^2} \, {}^t F R^{-1} F)$ est négative, puisque R et R^{-1} sont positives, et donc $\hat{\beta}$ correspond bien à un maximum.

Paramètre d'échelle maximisant la vraisemblance, $\hat{\sigma}_0^2$

Pour trouver la valeur de σ_0^2 qui maximise la log-vraisemblance, nous résolvons :

$$\frac{\partial \mathcal{L}}{\partial (\sigma_0^2)} (\sigma_0^2 = \hat{\sigma_0}^2) = 0 . \tag{A.12}$$

Ce qui se traduit par :

$$-\frac{N}{2\hat{\sigma_0}^2} + \frac{1}{2\hat{\sigma_0}^4} \, {}^t(Y - F\beta) \, R^{-1} \, (Y - F\beta) = 0 . \tag{A.13}$$

Cette équation donne :

$$\frac{N}{2} \hat{\sigma_0}^2 - \frac{1}{2} \, {}^t(Y - F\beta) \, R^{-1} \, (Y - F\beta) = 0 . \tag{A.14}$$

Et on en déduit :

$$\hat{\sigma_0}^2 = \frac{1}{N} \, {}^t(Y - F\beta) \, R^{-1} \, (Y - F\beta) \ . \tag{A.15}$$

On peut vérifier que la dérivée seconde en $\hat{\sigma_0}$ est bien négative :

$$\begin{array}{ll} \frac{\partial^2 \mathcal{L}}{\partial (\sigma_0^2)^2}(\sigma_0^2 = \hat{\sigma_0}^2) & = \frac{N}{2\hat{\sigma_0}^4} - \frac{1}{\hat{\sigma_0}^6}\ {}^t(Y - F\beta)\, R^{-1}\, (Y - F\beta) \\ & = \frac{N}{\hat{\sigma_0}^6}\left(\frac{1}{2}\hat{\sigma_0}^2 - \frac{1}{N}\ {}^t(Y - F\beta)\, R^{-1}\, (Y - F\beta)\right) \\ & = \frac{N}{\hat{\sigma_0}^6}\left(\frac{1}{2}\hat{\sigma_0}^2 - \hat{\sigma_0}^2\right) \\ & = -\frac{N}{2\hat{\sigma_0}^4} \\ & < 0\ . \end{array} \tag{A.16}$$

Log-vraisemblance concentrée

La log-vraisemblance concentrée s'écrit en reprenant la formule de la log-vraisemblance (A.3) et en y substituant les paramètres $\hat{\beta}$ et $\hat{\sigma_0}$ qui la maximisent :

$$\begin{array}{rcl} \mathcal{L}_c(\Theta) & = & \mathcal{L}(\hat{\beta}, \hat{\sigma_0}^2, \Theta) \\ & = & -\frac{N}{2}\ln(2\pi) - \frac{1}{2}\ln(\det(R)) - \frac{N}{2}\ln(\hat{\sigma_0}^2) - \frac{1}{2\hat{\sigma_0}^2}\ {}^t\left(Y - F\hat{\beta}\right) R^{-1}\left(Y - F\hat{\beta}\right) \\ & = & -\frac{N}{2}\ln(2\pi) - \frac{1}{2}\ln(\det(R)) - \frac{N}{2}\ln(\hat{\sigma_0}^2) - \frac{1}{2\sigma_0^2} N\hat{\sigma_0}^2 \\ & = & -\frac{N}{2}\ln(2\pi) - \frac{1}{2}\ln(\det(R)) - \frac{N}{2}\ln(\hat{\sigma_0}^2) - \frac{N}{2}\ . \end{array} \tag{A.17}$$

Le recherche des paramètres qui maximisent la vraisemblance se réduit alors à la recherche des Θ qui maximisent la log-vraisemblance concentrée.

Annexe B

Vraisemblance concentrée (cas multisortie)

Nous employons ici les notations du chapitre II. Nous avons établi à la section 2.2.7 que la log-vraisemblance maximisée au regard des $\boldsymbol{\beta}$ s'écrit :

$$\mathcal{L}(\boldsymbol{\beta}=\hat{\boldsymbol{\beta}},\boldsymbol{\sigma},\boldsymbol{\Theta}) = -\frac{N}{2}\ln(2\pi) - \frac{1}{2}\ln(|\boldsymbol{R}|) - \sum_{a=1}^{n_g}(N_a\ln(\sigma_a)) - \frac{1}{2}\sum_{a=1}^{n_g}\sum_{b=1}^{n_g}\frac{1}{\sigma_a}\frac{1}{\sigma_b}w_{ab}\ , \tag{B.1}$$

avec w_{ab} les éléments de la matrice W, symétrique et définie positive, et définie par l'équation (2.50). Nous avons également montré que les dérivées par rapport aux σ_a de la log-vraisemblance, maximisée pour les $\boldsymbol{\beta}$, valent :

$$\left\{\frac{\partial\mathcal{L}}{\partial\sigma_a} = -\frac{N_a}{\sigma_a} + \frac{1}{\sigma_a^2}\sum_{b=1}^{n_g}\frac{1}{\sigma_b}w_{ab}\right\}_{a\in[1\ldots n_g]}\ , \tag{B.2}$$

ce qui nous permet d'affirmer que les $\hat{\sigma}_a$ qui maximisent la log-vraisemblance obéissent au système d'équations (2.54) :

$$\left\{N_a\sigma_a - \sum_{b=1}^{n_g}\frac{1}{\sigma_b}w_{ab} = 0\right\}_{a\in[1\ldots n_g]}\ . \tag{B.3}$$

Sans résoudre cette équation, On peut s'en servir pour vérifier les dérivées secondes de la log-vraisemblance :

$$\left\{\frac{\partial^2\mathcal{L}}{\partial\sigma_a\partial\sigma_b} = \frac{N_a}{\sigma_a^2}\delta_{ab} + \frac{-2\delta_{ab}}{\sigma_a^3}\sum_{c=1}^{n_g}\frac{1}{\sigma_c}w_{ac} - \frac{1}{\sigma_a^2}\frac{1}{\sigma_b^2}w_{ab}\right\}_{a,b\in[1\ldots n_g]}\ , \tag{B.4}$$

avec δ_{ab} le symbole de Kronecker, valant 1 si et seulement si $a=b$. En substituant l'expression de $\hat{\boldsymbol{\sigma}}$, il vient, pour un couple d'indices a et b :

$$\begin{aligned}\tfrac{\partial^2\mathcal{L}}{\partial\sigma_a\partial\sigma_b}(\boldsymbol{\sigma}=\hat{\boldsymbol{\sigma}}) &= \tfrac{\delta_{ab}}{\hat{\sigma_a}^3}\left(N_a\hat{\sigma_a} - 2\sum_{c=1}^{n_g}\tfrac{1}{\sigma_c}w_{ac}\right) - \tfrac{1}{\hat{\sigma_a}^2}\tfrac{1}{\hat{\sigma_b}^2}w_{ab}\\ &= -\delta_{ab}\tfrac{N_a}{\hat{\sigma_a}^2} - \tfrac{1}{\sigma_a^2}\tfrac{1}{\sigma_b^2}w_{ab}\\ &= \left\{-D_N D_\sigma^{-2} - D_\sigma^{-2}WD_\sigma^{-2}\right\}_{a,b}\ ,\end{aligned} \tag{B.5}$$

avec D_N et D_σ les matrices diagonales de taille $n_g\times n_g$ contenant les N_a et les $\boldsymbol{\sigma}_a$, respectivement. La hessienne $-D_N D_\sigma^{-2} - D_\sigma^{-2}WD_\sigma^{-2}$ est bien négative, puisque W est positive, ainsi que D_N et D_σ^{-2}.

B.1 Solution explicite pour 2 groupes

Pour 2 groupes, le système d'équations (2.54) devient :

$$\left\{ \begin{array}{rcl} N_1\sigma_1 - \frac{1}{\sigma_1}w_{11} - \frac{1}{\sigma_2}w_{12} & = & 0 \\ N_2\sigma_2 - \frac{1}{\sigma_1}w_{21} + \frac{1}{\sigma_2}w_{22} & = & 0 \ . \end{array} \right. \tag{B.6}$$

En multipliant par $\sigma_1\sigma_2$, on obtient :

$$\left\{ \begin{array}{l} -N_1\sigma_1^2\sigma_2 + \sigma_2 w_{11} + \sigma_1 w_{12} = 0 \\ -N_2\sigma_1\sigma_2^2 + \sigma_2 w_{12} + \sigma_1 w_{22} = 0 \ . \end{array} \right. \tag{B.7}$$

On factorise σ_2 dans la première ligne et on multiplie la seconde par $(N_1\sigma_1^2 - w_{11})^2$:

$$\left\{ \begin{array}{l} \sigma_2(N_1\sigma_1^2 - w_{11}) = \sigma_1 w_{12} \\ -N_2\sigma_1\sigma_2^2(N_1\sigma_1^2 - w_{11})^2 + \sigma_2 w_{12}(N_1\sigma_1^2 - w_{11})^2 + \sigma_1 w_{22}(N_1\sigma_1^2 - w_{11})^2 = 0 \ . \end{array} \right. \tag{B.8}$$

En injectant la première ligne dans la seconde, on obtient :

$$-N_2\sigma_1(\sigma_1 w_{12})^2 + w_{12}\sigma_1 w_{12}(N_1\sigma_1^2 - w_{11}) + \sigma_1 w_{22}(N_1\sigma_1^2 - w_{11})^2 = 0 \ . \tag{B.9}$$

On simplifie par σ_1, on développe et on ordonne l'équation par puissance décroissante de σ_1 :

$$N_1^2 w_{22}\sigma_1^4 + \left((N_1 - N_2)w_{12}^2 - 2N_1 w_{11} w_{22}\right)\sigma_1^2 - w_{11}w_{12}^2 + w_{11}^2 w_{22} = 0 \ . \tag{B.10}$$

Cette équation, du second degré par rapport à σ_1^2, et son équation symétrique par interversion de 1 et 2, conduisent aux solutions potentielles :

$$\left\{ \begin{array}{rcl} \sigma_1^2 & = & \frac{(N_2-N_1)w_{12}^2 + 2N_1 w_{11} w_{22} \pm |w_{12}|\sqrt{\Delta}}{2N_1^2 w_{22}} \\ \sigma_2^2 & = & \frac{(N_1-N_2)w_{12}^2 + 2N_2 w_{11} w_{22} \pm |w_{12}|\sqrt{\Delta}}{2N_2^2 w_{11}} \ , \end{array} \right. \tag{B.11}$$

avec pour Δ :

$$\begin{array}{rcl} \Delta & = & (N_1 - N_2)^2 w_{12}^2 - 4N_1(N_1 - N_2)w_{11}w_{22} + 4N_1^2 w_{11}^2 w_{22}^2/w_{12}^2 + 4N_1^2 w_{11}w_{22} - 4N_1^2 w_{11}^2 w_{22}^2/w_{12}^2 \\ & = & (N_1 - N_2)^2 w_{12}^2 + 4N_1 N_2 w_{11} w_{22} \ . \end{array} \tag{B.12}$$

Δ est bien positif, car on sait que W est définie positive et donc que $det(W) = w_{11}w_{22} - w_{12}^2 \geq 0$, et on peut alors en déduire :

$$\begin{array}{rcl} \Delta & = & (N_1 - N_2)^2 w_{12}^2 + 4N_1 N_2 w_{11} w_{22} \\ & \geq & (N_1^2 + N_2^2 - 2N_1 N_2)w_{12}^2 + 4N_1 N_2 w_{12}^2 \\ & \geq & (N_1 + N_2)^2 w_{12}^2 \geq 0 \ , \end{array} \tag{B.13}$$

et par conséquent :

$$\sqrt{\Delta} \ \geq \ (N_1 + N_2)\,|w_{12}| \ . \tag{B.14}$$

Vérifions à présent quelles sont les solutions positives parmi les solutions (B.11). Revenons au système (B.7). En multipliant la première ligne par σ_2 cela nous donne :

$$\sigma_1\sigma_2 w_{12} = N_1\sigma_1^2\sigma_2^2 - \sigma_2^2 w_{11} \ . \tag{B.15}$$

En remplaçant σ_1^2 par sa solution (B.11), on obtient :

$$\begin{array}{rcl} \sigma_1\sigma_2 w_{12} & = & \left(N_1 \frac{(N_2-N_1)w_{12}^2 + 2N_1 w_{11} w_{22} \pm |w_{12}|\sqrt{\Delta}}{2N_1^2 w_{22}} - w_{11}\right)\sigma_2^2 \\ & = & \left(\frac{(N_2-N_1)w_{12}^2 + 2N_1 w_{11} w_{22} \pm |w_{12}|\sqrt{\Delta}}{2N_1 w_{22}} - \frac{2N_1 w_{11} w_{22}}{2N_1 w_{22}}\right)\sigma_2^2 \ . \end{array} \tag{B.16}$$

On peut simplifier et remplacer σ_2^2 par son expression (B.11) :

$$\sigma_1\sigma_2 w_{12} = \left(\frac{-(N_1-N_2)w_{12}^2 \pm |w_{12}|\sqrt{\Delta}}{2N_1 w_{22}}\right)\left(\frac{(N_1-N_2)w_{12}^2 + 2N_2 w_{11}w_{22} \pm |w_{12}|\sqrt{\Delta}}{2N_2^2 w_{11}}\right) . \quad \text{(B.17)}$$

On développe et on simplifie progressivement :

$$\sigma_1\sigma_2 w_{12} = \frac{-(N_1-N_2)^2 w_{12}^4 - 2N_2(N_1-N_2)w_{11}w_{22}w_{12}^2 + w_{12}^2\Delta}{4N_1N_2^2 w_{11}w_{22}} \pm \frac{|w_{12}|\sqrt{\Delta}\left((N_1-N_2)w_{12}^2 + 2N_2w_{11}w_{22} - (N_1-N_2)w_{12}^2\right)}{4N_1N_2^2 w_{11}w_{22}} , \quad \text{(B.18)}$$

$$\sigma_1\sigma_2 w_{12} = \frac{-(N_1-N_2)^2 w_{12}^4 - 2N_2(N_1-N_2)w_{11}w_{22}w_{12}^2 + w_{12}^2\left((N_1-N_2)^2 w_{12}^2 + 4N_1N_2w_{11}w_{22}\right)}{4N_1N_2^2 w_{11}w_{22}} \pm \frac{\pm 2N_2 w_{11}w_{22}|w_{12}|\sqrt{\Delta}}{4N_1N_2^2 w_{11}w_{22}} , \quad \text{(B.19)}$$

$$\sigma_1\sigma_2 w_{12} = \frac{2N_2(N_1+N_2)w_{11}w_{22}w_{12}^2 \pm 2N_2 w_{11}w_{22}|w_{12}|\sqrt{\Delta}}{4N_1N_2^2 w_{11}w_{22}} , \quad \text{(B.20)}$$

$$\sigma_1\sigma_2 w_{12} = \frac{(N_1+N_2)w_{12}^2 \pm |w_{12}|\sqrt{\Delta}}{2N_1N_2} , \quad \text{(B.21)}$$

et enfin :

$$\sigma_1\sigma_2 = \frac{(N_1+N_2)w_{12} \pm \frac{w_{12}}{|w_{12}|}\sqrt{\Delta}}{2N_1N_2} . \quad \text{(B.22)}$$

Le produit $\sigma_1\sigma_2$ doit être positif. L'inégalité (B.14) impose alors que $\pm\frac{w_{12}}{|w_{12}|} = +1$, ce qui conduit à :

$$\left\{\begin{array}{lcl} \sigma_1\sigma_2 & = & \frac{(N_1+N_2)w_{12}+\sqrt{\Delta}}{2N_1N_2} \\ \sigma_1^2 & = & \frac{(N_2-N_1)w_{12}^2+2N_1w_{11}w_{22}+w_{12}\sqrt{\Delta}}{2N_1^2w_{22}} \\ \sigma_2^2 & = & \frac{(N_1-N_2)w_{12}^2+2N_2w_{11}w_{22}+w_{12}\sqrt{\Delta}}{2N_2^2w_{11}} . \end{array}\right. \quad \text{(B.23)}$$

On peut alors vérifier avec l'inégalité (B.14) que les solutions pour σ_1^2 et σ_2^2 sont toutes les deux positives.

Lorsque $w_{12} = 0$, il n'y a pas d'interaction entre les groupes, et la solution devient :

$$\left\{\begin{array}{lcl} \sigma_1^2 & = & \frac{w_{11}}{N_1} \\ \sigma_2^2 & = & \frac{w_{22}}{N_2} , \end{array}\right. \quad \text{(B.24)}$$

ce qui correspond aux solutions classiques pour 2 processus gaussiens indépendants.

B.2 Solution numérique pour un nombre quelconque de groupes

On repart du système d'équations (2.54), que l'on veut résoudre cette fois pour un nombre quelconque de groupes :

$$\left\{N_a\sigma_a - \sum_{b=1}^{n_g}\frac{1}{\sigma_b}w_{ab} = 0\right\}_{a\in[1\ldots n_g]} . \quad \text{(B.25)}$$

On écrit ce système de n_g équations sous une forme matricielle :

$$D_N D_\sigma \mathbb{1}_{n_g} - W D_\sigma^{-1} \mathbb{1}_{n_g} = \mathbb{0}_{n_g} \ . \tag{B.26}$$

Pour rappel, D_N et D_σ sont des matrices diagonales positive contenant respectivement les $N.$ et les $\sigma.$ de chaque groupe, W est une matrice symétrique définie positive (définie par l'équation (2.50)), et $\mathbb{1}_{n_g}$ et $\mathbb{0}_{n_g}$ sont les vecteurs de taille n_g ne contenant que des 1 et des 0 respectivement. Nous n'avons pas de solution explicite qui résolve cette équation, mais nous pouvons utiliser une méthode itérative de Newton-Raphson pour approcher rapidement le zéro de la fonction vectorielle $\mathbf{f}(\cdot)$ suivante :

$$\mathbf{f} \ : \ S \in {\mathbb{R}_+^*}^{n_g} \ \rightarrow \ \mathbf{f}(S) = D_N S - W S^{(-1)} \in \mathbb{R}^{n_g} \ , \tag{B.27}$$

avec la convention $S^{(-1)} = D_S^{-1} \mathbb{1}_{n_g}$ et D_S la matrice diagonale des S_a. Cette fonction n'a aucun rapport avec les fonctions $f_l^{(a)}(\cdot)$ qui constituent la tendance.

La matrice jacobienne de $\mathbf{f}$ vaut :

$$\begin{array}{rcl} J_f(S) & = & D_N + W D_S^{-2} \\ & = & \left(D_N D_S^2 + W\right) D_s^{-2} \end{array} \tag{B.28}$$

$\left(D_N D_S^2 + W\right)$ est strictement définie positive, en tant que somme de matrices strictement définies positives, et inversible. La jacobienne $J_f(S)$ est donc elle aussi inversible.

On sait que le problème a au moins une solution par construction, puisqu'il provient de la dérivation de la log-vraisemblance qui a au moins un maximum (elle est continue et va vers moins l'infini quand un σ_a va vers 0 ou vers l'infini). On va à présent démontrer l'unicité de la solution.

Supposons deux vecteurs S_1 et S_2 qui sont tous les deux solution de notre problème. On a $\{S_1\}_a > 0$ et $\{S_2\}_a > 0$ pour n'importe quel $a \in [1 \dots n_g]$, et $D_N S_1 = W S_1^{(-1)}$, et $D_N S_2 = W S_2^{(-1)}$. Notons D_t la matrice diagonale telle que $\{D_t\}_{aa} = t_a = \frac{\{S_1\}_a}{\{S_2\}_a}$; D_t est strictement positive.

$$S_1 = D_t S_2 \tag{B.29}$$

On sait que :

$$ {}^t\left(S_1^{(-1)} - S_2^{(-1)}\right) W \left(S_1^{(-1)} - S_2^{(-1)}\right) \geq 0 \ , \tag{B.30}$$

par la positivité de W.

Par ailleurs, on peut calculer :

$$\begin{array}{rcl} {}^t\left(S_1^{(-1)} - S_2^{(-1)}\right) W \left(S_1^{(-1)} - S_2^{(-1)}\right) & = & {}^t S_1^{(-1)} W S_1^{(-1)} - {}^t S_1^{(-1)} W S_2^{(-1)} - {}^t S_2^{(-1)} W S_1^{(-1)} + {}^t S_2^{(-1)} W S_2^{(-1)} \\ & = & {}^t S_1^{(-1)} D_N S_1 - {}^t S_1^{(-1)} D_N S_2 - {}^t S_2^{(-1)} D_N S_1 + {}^t S_2^{(-1)} D_N S_2 \\ & = & {}^t \mathbb{1}_{n_g} D_N \mathbb{1}_{n_g} - {}^t \mathbb{1}_{n_g} D_t^{-1} D_N \mathbb{1}_{n_g} - {}^t \mathbb{1}_{n_g} D_t D_N \mathbb{1}_{n_g} + {}^t \mathbb{1}_{n_g} D_N \mathbb{1}_{n_g} \\ & = & {}^t \mathbb{1}_{n_g} D_N \left(2I - D_t^{-1} - D_t\right) \mathbb{1}_{n_g} \\ & = & {}^t \mathbb{1}_{n_g} D_N \left(-D_t^{-1}\right) \left(-2D_t + I + D_t^2\right) \mathbb{1}_{n_g} \\ & = & {}^t \mathbb{1}_{n_g} D_N \left(-D_t^{-1}\right) \left(D_t - I\right)^2 \mathbb{1}_{n_g} \\ & = & -\sum_{i=1}^m n_i \ t_i^{-1} (t_i - 1)^2 \\ & \leq & 0 \ . \end{array} \tag{B.31}$$

Puisque la quantité ${}^t\left(S_1^{(-1)} - S_2^{(-1)}\right) W \left(S_1^{(-1)} - S_2^{(-1)}\right)$ est à la fois positive et négative, elle doit être égale à 0. Comme W est définie positive strictement, elle définie une norme et on a donc $\left\| S_1^{(-1)} - S_2^{(-1)} \right\|_W = 0$. Ce qui implique :

$$S_1 = S_2 \ , \tag{B.32}$$

et démontre alors l'unicité de la solution.

Cet unique minimum peut être rapidement approché par une méthode de Newton-Raphson. On construit la suite de vecteurs définie par :

$$\begin{aligned} S_{n+1} &= S_n - J_f(S_n)^{-1}\mathbf{f}(S_n) \\ &= S_n - \left(D_N + WD_{S_n}^{-2}\right)^{-1}\left(D_N D_{S_n} - WD_{S_n}^{-1}\right)\mathbb{1}_{n_g} \\ &= D_{S_n}\mathbb{1}_{n_g} - \left(D_N + WD_{S_n}^{-2}\right)^{-1}\left(\left(D_N + WD_{S_n}^{-2}\right)D_{S_n} - WD_{S_n}^{-2}D_{S_n} - WD_{S_n}^{-1}\right)\mathbb{1}_{n_g} \\ &= D_{S_n}\mathbb{1}_{n_g} - D_{S_n}\mathbb{1}_{n_g} + 2\left(D_N + WD_{S_n}^{-2}\right)^{-1}WD_{S_n}^{-1}\mathbb{1}_{n_g} \\ &= 2\left(D_{S_n}W^{-1}D_N + D_{S_n}^{-1}\right)^{-1}\mathbb{1}_{n_g}\ . \end{aligned} \tag{B.33}$$

Comme il y a une discontinuité au niveau des axes en 0, on n'a pas la garantie de ne pas tomber sur une solution négative (ou partiellement négative). Pour pallier à ça, on vérifie à chaque itération de l'algorithme si il y a une valeur négative dans le vecteur S_{n+1}, et si c'est le cas on le remplace par la moyenne entre S_{n+1} et S_n (et on revérifie la positivité). Cela revient à diminuer la distance parcourue dans la direction donnée par la jacobienne. On obtient la relation algorithmique suivante :

$$\begin{gathered} S_{n+1} = 2\left(\mathrm{diag}(S_n)W^{-1}D_N + \mathrm{diag}(S_n)^{-1}\right)^{-1}\mathbb{1}_{n_g}\ , \\ \textit{tant que}\ :\ (\min(S_{n+1}) \leq 0)\ ,\ \textit{faire}\ :\ \left(S_{n+1} \leftarrow \tfrac{1}{2}\left(S_n + S_{n+1}\right)\right)\ . \end{gathered} \tag{B.34}$$

Bien sur, il faut initier la séquence avec un vecteur positif, par exemple :

$$S_0 = \mathbb{1}_{n_g}\sqrt{\frac{1}{n_g}\ {}^t\mathbb{1}_{n_g}D_N^{-1}W\mathbb{1}_{n_g}}\ . \tag{B.35}$$

Une autre manière d'éviter la discontinuité en 0 est de travailler sur une transformation logarithmique des σ_a. Appelons $\lambda = log(S)$ le vecteur des logarithmes de S. Nous allons chercher le zéro de la fonction vectorielle suivante :

$$\mathbf{f}_\lambda\ :\ \lambda \in \mathbb{R}^{n_g}\ \rightarrow\ \mathbf{f}_\lambda(\lambda) = D_N\exp(\lambda) - W\exp(-\lambda) \in \mathbb{R}^{n_g}\ , \tag{B.36}$$

dont la jacobienne vaut :

$$J_{f\lambda}(\lambda) = D_N D_{e\lambda} + WD_{e\lambda}^{-1}\ , \tag{B.37}$$

avec $D_{e\lambda} = D_S$, la matrice diagonale telle que $\{D_{e\lambda}\}_{a,a} = \exp(\lambda_a)$.

On construit alors la suite de vecteurs définie par :

$$\begin{aligned} \lambda_{n+1} &= \lambda_n - J_{f\lambda}(\lambda_n)^{-1}\mathbf{f}_\lambda(\lambda_n) \\ &= \lambda_n - \left(D_N D_{e\lambda_n} + WD_{e\lambda_n}^{-1}\right)^{-1}\left(D_N D_{e\lambda_n} - WD_{e\lambda_n}^{-1}\right)\mathbb{1}_{n_g} \\ &= \lambda_n - \left(D_N D_{e\lambda_n} + WD_{e\lambda_n}^{-1}\right)^{-1}\left(D_N D_{e\lambda_n} + WD_{e\lambda_n}^{-1} - 2WD_{e\lambda_n}^{-1}\right)\mathbb{1}_{n_g} \\ &= \lambda_n - \mathbb{1}_{n_g} + 2\left(D_N D_{e\lambda_n} + WD_{e\lambda_n}^{-1}\right)^{-1}WD_{e\lambda_n}^{-1}\mathbb{1}_{n_g} \\ &= \lambda_n - \mathbb{1}_{n_g} + 2\left(D_{e\lambda_n}W^{-1}D_N D_{e\lambda_n} + I_{n_g}\right)^{-1}\mathbb{1}_{n_g}\ , \end{aligned} \tag{B.38}$$

Ce qui nous amène à la relation algorithmique suivante :

$$\log(S_{n+1}) = \log(S_n) - \mathbb{1}_{n_g} + 2\left(\mathrm{diag}(S_n)W^{-1}D_N\,\mathrm{diag}(S_n) + I_{n_g}\right)^{-1}\mathbb{1}_{n_g}\ . \tag{B.39}$$

Bibliographie

Allaire, Douglas, Karen Willcox, and Olivier Toupet. 2010. "A Bayesian-Based Approach to Multifidelity Multidisciplinary Design Optimization." In *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, 9183.

Álvarez, Mauricio A., and Neil D. Lawrence. 2011. "Computationally Efficient Convolved Multiple Output Gaussian Processes." *J. Mach. Learn. Res.* 12 (July) : 1459–1500.

Álvarez, Mauricio A., Lorenzo Rosasco, and Neil D. Lawrence. 2012. "Kernels for Vector-Valued Functions : A Review." *Foundations and Trends® in Machine Learning* 4 (3) : 195–266.

Bachoc, François. 2013. "Cross Validation and Maximum Likelihood Estimation of Hyper-Parameters of Gaussian Processes with Model Misspecification." *Computational Statistics and Data Analysis* 66 : 55–69.

Bect, Julien, François Bachoc, and David Ginsbourger. 2019. "A Supermartingale Approach to Gaussian Process Based Sequential Design of Experiments." *Bernoulli* 25 (4A) : 2883–2919.

Bect, Julien, David Ginsbourger, Ling Li, Victor Picheny, and Emmanuel Vazquez. 2012. "Sequential Design of Computer Experiments for the Estimation of a Probability of Failure." *Statistics and Computing* 22 (3) : 773–93.

Bichon, Barron J., Michael S. Eldred, Laura Painton Swiler, Sandaran Mahadevan, and John M. McFarland. 2008. "Efficient Global Reliability Analysis for Nonlinear Implicit Performance Functions." *AIAA Journal* 46 (10) : 2459–68.

Bouhlel, Mohamed Amine, Nathalie Bartoli, Abdelkader Otsmane, and Joseph Morlier. 2016. "Improving Kriging Surrogates of High-Dimensional Design Models by Partial Least Squares Dimension Reduction." *Structural and Multidisciplinary Optimization* 53 (5) : 935–52.

Byrd, Richard H., Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. 1995. "A Limited Memory Algorithm for Bound Constrained Optimization." *SIAM J. Scientific Computing* 16 : 1190–1208.

Chevalier, Clément, Julien Bect, David Ginsbourger, Emmanuel Vazquez, Victor Picheny, and Yann Richet. 2014. "Fast Parallel Kriging-Based Stepwise Uncertainty Reduction with Application to the Identification of an Excursion Set." *Technometrics* 56 (4) : 455–65.

Chevalier, Clément, and David Ginsbourger. 2013. "Fast Computation of the Multi-Points Expected Improvement with Applications in Batch Selection." In *International Conference on Learning and Intelligent Optimization*, 59–69. Springer.

Chevalier, Clément, Victor Picheny, and David Ginsbourger. 2014. "Kriginv : An Efficient and User-Friendly Implementation of Batch-Sequential Inversion Strategies Based on Kriging." *Computational Statistics & Data Analysis* 71 : 1021–34.

Chevalier, Clément, Victor Picheny, David Ginsbourger, and Dario Azzimonti. 2011. "R Package KrigInv : Kriging-Based Inversion for Deterministic and Noisy Computer Experiments." https://cran.r-project.org/packages=KrigInv.

Christensen, Daniel Erik. 2012. "Multifidelity Methods for Multidisciplinary Design Under Uncertainty." PhD thesis, Massachusetts Institute of Technology.

Conti, Stefano, and Anthony O'Hagan. 2010. "Bayesian Emulation of Complex Multi-Output and Dynamic Computer Models." *Journal of Statistical Planning and Inference* 140 (3) : 640–51.

Cressie, Noel. 1992. *Statistics for Spatial Data. Terra Nova.* Vol. 4. 5. https://doi.org/10.1111/j.1365-3121.1992.tb00605.x.

Damianou, Andreas C., and Neil D. Lawrence. 2013. "Deep Gaussian Processes." In *Artificial Intelligence and Statistics*, 207–15.

Deville, Yves, David Ginsbourger, and Olivier Roustant. 2015. "R Package Kergp : Gaussian Process Laboratorys." https://CRAN.R-project.org/package=kergp.

DICE. 2006. "Consortium : Deep Inside Computer Experiments." http://dice.emse.fr/.

Dupuy, Delphine, Céline Helbert, and Jessica Franco. 2015. "DiceDesign and DiceEval : Two R Packages for Design and Analysis of Computer Experiments." *Journal of Statistical Software* 65 (11) : 1–38. http://www.jstatsoft.org/v65/i11/.

Durrande, Nicolas, David Ginsbourger, Olivier Roustant, and Laurent Carraro. 2013. "ANOVA Kernels and RKHS of Zero Mean Functions for Model-Based Sensitivity Analysis." *Journal of Multivariate Analysis* 115 : 57–67.

Dutordoir, Vincent. 2017. "Non-Stationary Surrogate Modeling with Deep Gaussian."

Eriksson, David, Michael Pearce, Jacob Gardner, Ryan D. Turner, and Matthias Poloczek. 2019. "Scalable Global Optimization via Local Bayesian Optimization." In *Advances in Neural Information Processing Systems*, 5496–5507.

Fernández-Godino, Maria Giselle, Chanyoung Park, Nam-Ho Kim, and Raphael T. Haftka. 2016. "Review of Multi-Fidelity Models." *arXiv Preprint arXiv :1609.07196.*

Forrester, Alexander I. J., András Sóbester, and Andy J. Keane. 2007. "Multi-Fidelity Optimization via Surrogate Modelling." *Proceedings of the Royal Society of London A : Mathematical, Physical and Engineering Sciences* 463 (2088) : 3251–69. https://doi.org/10.1098/rspa.2007.1900.

Forrester, Alexander IJ, and Andy J Keane. 2009. "Recent Advances in Surrogate-Based Optimization." *Progress in Aerospace Sciences* 45 (1-3) : 50–79.

Frazier, Peter I. 2018. "A Tutorial on Bayesian Optimization." *arXiv Preprint arXiv :1807.02811.*

Fricker, Thomas E., Jeremy E. Oakley, and Nathan M. Urban. 2013. "Multivariate Gaussian Process Emulators with Nonseparable Covariance Structures." *Technometrics* 55 (1) : 47–56. https://doi.org/10.1080/00401706.2012.715835.

Garland, Nicolas. 2017. "Algorithme PEGO, Code Source." Lien github : https://github.com/Funz/algorithm-EGO/blob/master/src/main/doe/PEGO.R.

Garland, Nicolas, Rodolphe Le Riche, Yann Richet, and Nicolas Durrande. 2020. "Multi-Fidelity for MDO Using Gaussian Processes." In *Aerospace System Analysis and Optimization in Uncertainty*, edited by Loïc Brevault, Balesdent Mathieu, and Jérôme Morio, 295–320. Springer International Publishing.

Genz, Alan, and Frank Bretz. 2009. *Computation of Multivariate Normal and t Probabilities.* Vol. 195. Springer Science & Business Media.

Ginsbourger, David, Jean Baccou, Clément Chevalier, Frédéric Perales, Nicolas Garland, and Yann Monerie. 2014. "Bayesian Adaptive Reconstruction of Profile Optima and Optimizers." *SIAM/ASA Journal on Uncertainty Quantification* 2 (1) : 490–510.

Ginsbourger, David, Rodolphe Le Riche, and Laurent Carraro. 2010. "Kriging Is Well-Suited to Parallelize Optimization." In *Computational Intelligence in Expensive Optimization Problems*, 131–62. Springer.

Gneiting, Tilmann, William Kleiber, and Martin Schlather. 2010. "Matérn Cross-Covariance Functions for Multivariate Random Fields." *Journal of the American Statistical Association* 105 (491) : 1167–77.

Gramacy, Robert B., and Herbert K. H. Lee. 2010. "Optimization Under Unknown Constraints." *arXiv Preprint arXiv :1004.4027.*

Hastie, Trevor J. 2017. "Generalized Additive Models." In *Statistical Models in s*, 249–307. Routledge.

Helbert, Céline, Delphine Dupuy, and Laurent Carraro. 2009. "Assessment of Uncertainty in Computer Experiments from Universal to Bayesian Kriging." *Applied Stochastic Models in Business and Industry* 25 (2) : 99–113.

Hensman, James, Nicolo Fusi, and Neil D. Lawrence. 2013. "Gaussian Processes for Big Data." *arXiv Preprint arXiv :1309.6835*.

Huang, Deng, Theodore T. Allen, William I. Notz, and R. Allen Miller. 2006. "Sequential Kriging Optimization Using Multiple-Fidelity Evaluations." *Structural and Multidisciplinary Optimization* 32 (5) : 369–82. https://doi.org/10.1007/s00158-005-0587-0.

Jones, Donald R. 2001. "A Taxonomy of Global Optimization Methods Based on Response Surfaces." *Journal of Global Optimization* 21 (4) : 345–83. https://doi.org/10.1023/A:1012771025575.

Jones, Donald R., Matthias Schonlau, and William J. Welch. 1998. "Efficient Global Optimization of Expensive Black-Box Functions." *Journal of Global Optimization* 13 : 455–92.

Kandasamy, Kirthevasan, Gautam Dasarathy, Junier Oliva, Jeff Schneider, and Barnabas Poczos. 2019. "Multi-Fidelity Gaussian Process Bandit Optimisation." *Journal of Artificial Intelligence Research* 66 : 151–96.

Kennedy, Marc C., and Anthony O'Hagan. 2000. "Predicting the Output from a Complex Computer Code When Fast Approximations Are Available." *Biometrika* 87 (1) : 1–13.

Kordonowy, David, Nathan Fitzgerald, Justin McClellan, and Daniel Christensen. 2012. "Multifidelity Modeling Framework for Bayesian-Based Multidisciplinary Aircraft Design Optimization." In *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 5691.

Krige, Danie G. 1951. "A Statistical Approach to Some Basic Mine Valuation Problems on the Witwatersrand." *Journal of the Chemical, Metallurgical and Mining Society of South Africa* 52 : 119–39.

Laurent, Luc, Rodolphe Le Riche, Bruno Soulier, and Pierre-Alain Boucard. 2017. "An Overview of Gradient-Enhanced Metamodels with Applications." *Archives of Computational Methods in Engineering*, July. https://doi.org/10.1007/s11831-017-9226-3.

Le Gratiet, Loïc. 2013. "Multi-Fidelity Gaussian Process Regression for Computer Experiments." PhD thesis, Université de Paris-Diderot.

Le Gratiet, Loïc, and Claire Cannamela. 2015. "Cokriging-Based Sequential Design Strategies Using Fast Cross-Validation Techniques for Multi-Fidelity Computer Codes." *Technometrics* 57 (3) : 418–27.

Le Riche, Rodolphe, R. Girdziušas, and Janis Janusevskis. 2012. "A Study of Asynchronous Budgeted Optimization." In *NIPS Workshop on Bayesian Optimization and Decision Making (Lake Tahoe)*.

Loshchilov, Ilya. 2013. "Surrogate-Assisted Evolutionary Algorithms." PhD thesis.

López-Lopera, Andrés F, François Bachoc, Nicolas Durrande, and Olivier Roustant. 2018. "Finite-Dimensional Gaussian Approximation with Linear Inequality Constraints." *SIAM/ASA Journal on Uncertainty Quantification* 6 (3) : 1224–55.

Maatouk, Hassan. 2015. "Correspondance Entre régression Par Processus Gaussien Et Splines d'interpolation Sous Contraintes Linéaires de Type inégalité. Théorie Et Applications." PhD thesis, École Nationale Supérieure des Mines de Saint-Étienne.

March, Andrew, and Karen Willcox. 2012. "Multifidelity Approaches for Parallel Multidisciplinary Optimization." In *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 5688.

Marque-Pucheu, Sophie, Guillaume Perrin, and Josselin Garnier. 2017. "Efficient Sequential Experimental Design for Surrogate Modeling of Nested Codes." *arXiv Preprint arXiv :1712.01620*.

Matheron, Georges. 1963. "Principles of Geostatistics." *Economic Geology* 58 : 1246–66.

Maurya, Ashwini. 2016. "A Well-Conditioned and Sparse Estimation of Covariance and Inverse Covariance Matrices Using a Joint Penalty." *The Journal of Machine Learning Research* 17 (1) : 4457–84.

Mehmani, Ali, Souma Chowdhury, Weiyang Tong, and Achille Messac. 2015. "Adaptive Switching of Variable-Fidelity Models in Population-Based Optimization." In *Engineering and Applied Sciences Optimization*, 175–205. Springer.

Micchelli, Charles A., and Massimiliano Pontil. 2005. "Kernels for Multi–Task Learning." In *Advances in Neural Information Processing Systems 17*, edited by L. K. Saul, Y. Weiss, and L. Bottou, 921–28. MIT Press.

Mohammadi, Hossein, Rodolphe Le Riche, Xavier Bay, and Eric Touboul. 2018. "An Analysis of Covariance Parameters in Gaussian Process-Based Optimization." *Croatian Operational Research Review*, 1–10.

Mohammadi, Hossein, Rodolphe Le Riche, Nicolas Durrande, Eric Touboul, and Xavier Bay. 2016. "An Analytic Comparison of Regularization Methods for Gaussian Processes." *arXiv Preprint arXiv :1602.00853*.

Morris, Max D., Toby J. Mitchell, and Donald Ylvisaker. 1993. "Bayesian Design and Analysis of Computer Experiments : Use of Derivatives in Surface Prediction." *Technometrics* 35 (3) : 243–55.

Myers, Donald E. 1982. "Matrix Formulation of Co-Kriging." *Journal of the International Association for Mathematical Geology* 14 (3) : 249–57.

OQUAIDO. 2016. "Chaire En Mathématiques Appliquées : Optimisation et QUAntification d'Incertitude pour les Données Onéreuses." https://oquaido.emse.fr/.

Paiva, Ricardo M., André R. D. Carvalho, Curran Crawford, and Afzal Suleman. 2010. "Comparison of Surrogate Models in a Multidisciplinary Optimization Framework for Wing Design." *AIAA Journal* 48 (5) : 995–1006.

Park, Jeong-Soo, and Jangsun Baek. 2001. "Efficient Computation of Maximum Likelihood Estimators in a Spatial Linear Model with Power Exponential Covariogram." *Computer Geosciences* 27 : 1–7.

Peherstorfer, Benjamin, Karen Willcox, and Max Gunzburger. 2018. "Survey of Multifidelity Methods in Uncertainty Propagation, Inference, and Optimization." *SIAM Review* 60 (3) : 550–91.

Perdikaris, Paris, Daniele Venturi, Johannes O. Royset, and George Em Karniadakis. 2015. "Multi-Fidelity Modelling via Recursive Co-Kriging and Gaussian–Markov Random Fields." *Proc. R. Soc. A* 471 (2179) : 20150018.

Phillip, Boyle, and Frean Marcus. 2005. "Dependent Gaussian Processes." In *In Advances in Neural Information Processing Systems 17*, 217–24. MIT Press.

Picheny, Victor, David Ginsbourger, Yann Richet, and Gregory Caplin. 2013. "Quantile-Based Optimization of Noisy Computer Experiments with Tunable Precision." *Technometrics* 55 (1) : 2–13.

Picheny, Victor, David Ginsbourger, Olivier Roustant, Raphael T. Haftka, and Nam-Ho Kim. 2010. "Adaptive Designs of Experiments for Accurate Approximation of a Target Region." *Journal of Mechanical Design* 132 (7).

Picheny, Victor, Tobias Wagner, and David Ginsbourger. 2012. "A Benchmark of Kriging-Based Infill Criteria for Noisy Optimisation." *HAL-00658212*.

Potvin, Catherine, Martin J. Lechowicz, and Serge Tardif. 1990. "The Statistical Analysis of Ecophysiological Response Curves Obtained from Experiments Involving Repeated Measures." *Ecology* 71 : 1389–1400.

Ranjan, Pritam, Derek Bingham, and George Michailidis. 2008. "Sequential Experiment Design for Contour Estimation from Complex Computer Codes." *Technometrics* 50 (4) : 527–41.

Rasmussen, Carl Edward, and Christopher K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, MA, USA.

ReDICE. 2011. "Consortium ReDICE : Suite Du Consortium DICE." http://redice.emse.fr/.

Richet, Yann. 2015. "Critère IECI, Code Source." Lien github : https://github.com/IRSN/IECI.

Richet, Yann, Grégory Caplin, Jérôme Crevel, David Ginsbourger, and Victor Picheny. 2013. "Using the Efficient Global Optimization Algorithm to Assist Nuclear Criticality Safety Assessment." *Nuclear Science and Engineering* 175 (1) : 1–18.

Roustant, Olivier, David Ginsbourger, and Yves Deville. 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. http://www.jstatsoft.org/v51/i01/.

Roustant, Olivier, Espéran Padonou, Yves Deville, Aloïs Clément, Guillame Perrin, Jean Giorla, and Henry Wynn. 2018. "Group Kernels for Gaussian Process Metamodels with Categorical Inputs." *arXiv :1802.02368v1*.

Rullière, Didier, Nicolas Durrande, François Bachoc, and Clément Chevalier. 2018. "Nested Kriging Predictions for Datasets with a Large Number of Observations." *Statistics and Computing* 28 (4) : 849–67.

Sacher, Matthieu. 2018. "Méthodes Avancées d'optimisation Par Méta-Modèles – Application à La Performance Des Voiliers de Compétition." PhD thesis, Arts et Métiers ParisTech.

Sacher, Matthieu, Olivier Le Maître, Régis Duvigneau, Frédéric Hauville, Mathieu Durand, and Corenthin Lothodé. 2020. "A Non-Nested Infilling Strategy for Multi-Fidelity Based Efficient Global Optimization." *International Journal for Uncertainty Quantification*.

Salimbeni, Hugh, and Marc Deisenroth. 2017. "Doubly Stochastic Variational Inference for Deep Gaussian Processes." In *Advances in Neural Information Processing Systems*, 4588–99.

Sanson, François, Olivier Le Maître, and Pietro Marco Congedo. 2019. "Systems of Gaussian Process Models for Directed Chains of Solvers." *Computer Methods in Applied Mechanics and Engineering* 352 : 32–55.

Santner, Thomas J., Brian J. Williams, and William I. Notz. 2003. *The Design and Analysis of Computer Experiments*. Springer-Verlag New York.

Schonlau, Matthias, William J. Welch, and Donald R. Jones. 1998. "Global Versus Local Search in Constrained Optimization of Computer Models." *Lecture Notes-Monograph Series*, 11–25.

Seeger, Matthias, Yee-Whye Teh, and Michael Jordan. 2004. "Semiparametric Latent Factor Models." Workshop on Artificial Intelligence ; Statistics 10.

Sellar, R. S., Stephen M. Batill, and J. E. Renaud. 1996. "Response Surface Based, Concurrent Subspace Optimization for Multidisciplinary System Design." In *34th Aerospace Sciences Meeting and Exhibit*, 714.

Simpson, Timothy W., Timothy M. Mauery, John J. Korte, and Farrokh Mistree. 2001. "Kriging Models for Global Approximation in Simulation-Based Multidisciplinary Design Optimization." *AIAA Journal* 39 (12) : 2233–41.

Sobieski, Ian P., and Ilan M. Kroo. 2000. "Collaborative Optimization Using Response Surface Estimation." *AIAA Journal* 38 (10) : 1931–38. https://doi.org/10.2514/2.847.

Song, Hyeongjin, Kyung K. Choi, and David Lamb. 2013. "A Study on Improving the Accuracy of Kriging Models by Using Correlation Model/Mean Structure Selection and Penalized Log-Likelihood Function." In. Orlando, Florida, USA : 10th World Congress on Structural ; Multidisciplinary Optimization.

Stein, Michael. 1987. "Large Sample Properties of Simulations Using Latin Hypercube Sampling." *Technometrics* 29 (2) : 143–51.

Sundararajan, Sellamanickam, and S. Sathiya Keerthi. 2001. "Predictive Approaches for Choosing Hyperparameters in Gaussian Processes." *Neural Computation* 13 (5) : 1103–18. https://doi.org/10.1162/08997660151134343.

Teckentrup, Aretha L., Peter Jantsch, Clayton G. Webster, and Max Gunzburger. 2015. "A Multilevel Stochastic Collocation Method for Partial Differential Equations with Random Input Data." *SIAM/ASA Journal on Uncertainty Quantification* 3 (1) : 1046–74.

Tripathy, Rohit, Ilias Bilionis, and Marcial Gonzalez. 2016. "Gaussian Processes with Built-in Dimensionality Reduction : Applications to High-Dimensional Uncertainty Propagation." *Journal of Computational Physics* 321 : 191–223.

Villemonteix, Julien, Emmanuel Vazquez, and Eric Walter. 2009. "An Informational Approach to the Global Optimization of Expensive-to-Evaluate Functions." *Journal of Global Optimization* 44 (4) : 509.

Wackernagel, Hans. 1998. "Multivariate Geostatistics." *Springer*.

Wang, Xiaobang, Yuanzhi Liu, Wei Sun, Xueguan Song, and Jie Zhang. 2018. "Multidisciplinary and Multifidelity Design Optimization of Electric Vehicle Battery Thermal Management System." *Journal of Mechanical Design* 140 (9) : 094501.

Zadeh, Parviz M., and Vassili Toropov. 2002. "Multi-Fidelity Multidisciplinary Design Optimization Based on Collaborative Optimization Framework." In *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 5504.

Zhang, Boya, D. Austin Cole, and Robert B. Gramacy. 2019. "Distance-Distributed Design for Gaussian Process Surrogates." *Technometrics*, 1–13.

École Nationale Supérieure des Mines
de Saint-Étienne

NNT : *2020LYSEM017*

Nicolas GARLAND

Introspective Metamodeling Analysis of Physical Phenomena. Formalization with Kriging and Algorithmic Use for Optimization and Inversion.

Speciality: Applied Mathematics

Keywords: Kriging, Co-kriging, Gaussian Process, Optimization, Computer Simulations, Multi-fidelity, Sequential Codes, Sequential Design

Abstract :

In this thesis, we aim to improve the algorithms used by safety experts to analyze the outputs of their computer simulations of physical phenomena. Since simulations are expensive, the related algorithms build the design of experiments sequentially, using metamodels to predict the outcome of the simulations. We pay special attention to chain-produced simulations (where outputs of one code are used as inputs in other numerical simulators). The information given by the first codes allows a so-called "introspective" analysis of the final quantity. Firstly, we review different possible introspective metamodels, which treat the results as a multi-outputs function. We thus study the different forms of co-kriging and propose an improvement for the optimization of its parameters. We also develop another introspective metamodel, the hyper-kriging. We then make a comparison between the predictive performance of these metamodels. Secondly, we work on the algorithms. An optimization algorithm, called "Step or Stop Optimization", is developed, taking into account the specificities of the introspective case. It allows to stop the computation if the first steps are disengaging. Comparisons with currently used algorithm tend to confirm a significant saving in computing resources thanks to a better consideration of the intermediate physics of the simulation. The strategy used in this algorithm can be generalized, and extended to be used beyond the context of optimization problems.

École Nationale Supérieure des Mines
de Saint-Étienne

NNT : *2020LYSEM017*

Nicolas GARLAND

Métamodélisation introspective pour l'analyse des phénomènes physiques simulés. Formalisation dans le cadre du krigeage et intégration algorithmique en optimisation et inversion

Spécialité : Mathématiques appliquées

Mots clefs : krigeage, cokrigeage, processus gaussien, optimisation, simulations numériques, multifidélité, chaîne de calcul, plan séquentiel

Résumé :

L'objectif de la thèse est d'améliorer des algorithmes actuellement utilisés en assistance aux expertises de sûreté pour analyser la réponse d'un code de calcul simulant un phénomène physique. Les simulations étant coûteuses, les algorithmes utilisés construisent des plans d'expériences de manière itérative, grâce à des métamodèles capables de prédire les résultats des simulations. On s'intéresse en particulier aux simulations produites par des chaînes de calcul (enchaînement séquentiel de plusieurs codes de calculs distincts). Les informations produites par les premiers codes permettent une analyse dite « introspective » de la grandeur finale. Dans un premier temps, nous travaillons sur différents métamodèles introspectifs admissibles. Nous étudions ainsi les différentes formes de cokrigeage et proposons une amélioration au problème de l'optimisation de ses paramètres. Nous développons également un autre métamodèle introspectif, l'hyperkrigeage. Nous faisons ensuite une comparaison des performances prédictives de ces métamodèles. Dans un second temps, nous travaillons sur les algorithmes. Un premier algorithme d'optimisation, appelé « Step or Stop Optimization », prenant en compte les spécificités du cas introspectif a été développé. Il permet de ne pas poursuivre des calculs si les premières étapes n'y encouragent pas. Des comparaisons avec l'algorithme actuel tendent à confirmer une économie importante des ressources de calcul grâce à la meilleure prise en compte de la physique intermédiaire de la simulation. La stratégie utilisée par cet algorithme est généralisée, et étendue à des problématiques autres que l'optimisation.