THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *jeudi 10 mars 2016* **par :**

Benoît PAUWELS

Optimisation sans dérivées sous incertitudes appliquée à des simulateurs coûteux

---

JURY

Grégoire ALLAIRE	Professeur	Examinateur
Frédéric DELBOS	Docteur	Promoteur de thèse
Laurent DUMAS	Professeur	Rapporteur
Serge GRATTON	Professeur	Directeur de thèse
Mohamed MASMOUDI	Professeur	Examinateur
Luís NUNES VICENTE	Professeur	Rapporteur

École doctorale et spécialité :
MITT : Domaine Mathématiques : Mathématiques appliquées
Unité de Recherche :
Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique (URA 1875)
Directeur(s) de Thèse :
Serge GRATTON et *Frédéric DELBOS*
Rapporteurs :
Laurent DUMAS et *Luís NUNES VICENTE*

Problématique et enjeux

De nombreux problèmes réels consistent à prendre une décision pour satisfaire au mieux un ou plusieurs critères de performance sous certaines contraintes. En ingénierie de réservoir par exemple, le problème du placement de puits consiste à choisir l'emplacement de nouveaux puits dans les limites d'un champ pétrolier de manière à obtenir une production d'huile sur dix ans la plus forte possible. Nous pouvons également citer d'autres exemples industriels tels que la planification d'opérations de maintenance sur des transformateurs électriques, le choix de paramètres de sécurité dans une centrale nucléaire, le calage de paramètres d'un modèle moléculaire, la conception mécanique d'éoliennes flottantes ou encore la calibration de moteurs. La modélisation des phénomènes complexes en jeu dans ces problèmes peut donner lieu à l'étude de codes de simulation numérique. Ces codes peuvent être très coûteux en ressources processeur, le temps d'exécution variant de quelques minutes à plusieurs heures. Les valeurs renvoyées par de tels codes sont fatalement inexactes du fait de la précision limitée des machines. Par ailleurs les modèles simulés par ces codes peuvent mettre en jeu des paramètres mal connus (*e.g.* données géologiques non directement mesurables dans le cas du placement de puits) ou même être intrinsèquement stochastiques (*e.g.* tirage aléatoire de la distribution spatiale d'un système de molécules).
Ces problèmes peuvent se modéliser mathématiquement sous la forme de problèmes d'optimisation. Dans ce mémoire nous considérerons que les décision peuvent être caractérisées par un nombre fini n de valeurs numériques de l'ensemble $\mathbb{R}$ des nombres réels. Elles seront donc représentées par des vecteurs de l'espace euclidien $\mathbb{R}^n$. Le critère de performance associée à une décision x de $\mathbb{R}^n$ sera un nombre réel $f(x)$. Nous considérons de ce fait une fonction — dite *fonction objectif* — sur l'espace $\mathbb{R}^n$ et à valeurs dans $\mathbb{R}$. Le cas de critères de performance multiples sera abordée au moment opportun. Nous notons $\mathcal{X}$ la partie de $\mathbb{R}^n$ modélisant les contraintes portant sur la décision à prendre. Dans le cas du placement de puits, l'ensemble $\mathcal{X}$ est une partie de $\mathbb{R}^2$ constituée des couple de coordonnées cartésiennes représentant les points de la surface du champ pétrolier, auquel doit naturellement appartenir la décision x pour avoir un sens pratique. Observons que, quitte à considérer l'opposée de la fonction objectif f, nous pourrons toujours considérer que nous cherchons des décisions x pour lesquelles les valeurs de f sont aussi faibles que possible, c'est-à-dire que nous chercherons à minimiser f. Résumons le problème d'optimisation considéré sous la forme (P) ci-dessous — où par "s. c." il faut comprendre "sous contrainte".

$$\begin{aligned} \text{Minimiser} \quad & f(x) \\ \text{s. c.} \quad & x \in \mathcal{X}. \end{aligned} \tag{P}$$

Dans le contexte mathématique de l'étude du problème (P) nous parlerons de *solution* plutôt que de décision et nous dirons d'une solution x de $\mathbb{R}^n$ qu'elle est *réalisable* si x est dans $\mathcal{X}$. Pour comparer les performances de deux solutions réalisables x et x' nous ordonnerons tout simplement les valeurs $f(x)$ et $f(x')$ avec l'ordre canonique de $\mathbb{R}$ — $f(x) \leq f(x')$ si et seulement si la différence $f(x') - f(x)$ est positive. Une solution réalisable x sera appelée *minimiseur global* de f si $f(x) \leq f(x')$ est vraie pour tout x' de $\mathcal{X}$, ou *minimiseur local* de f si cette inégalité n'est vérifiée que dans un voisinage de x. La valeur $f(x)$ est alors appelée *minimum global*, de f ou *minimum local* de f.
Lorsque la fonction objectif est différentiable, un approche classique consiste à chercher les solutions optimales du problème (P) parmi les points stationnaires de f, c'est-à-dire les points x de $\mathcal{X}$ qui annule son gradient : $\nabla f(x) = 0$. Cependant, dans le cas de codes de simulation complexes, les dérivées de la fonction objectif sont souvent inaccessibles, trop coûteuses à estimer par différences finies ou même inexistantes. En effet ces simulateurs sont généralement des *boîtes noires* : la relation en les entrée et les sorties n'est pas explicitement connue, et les évaluations de f ne sont obtenues que par des appels, souvent longs en temps de calcul, au simulateur. Nous sommes alors contraints de résoudre le problème (P) sans faire appel aux dérivées de f, c'est-à-dire de considérer des méthodes d'optimisation *sans dérivées*. De nombreux algorithmes d'optimisation sans dérivées existent dans la littérature et nous verrons qu'ils peuvent être répartis en deux catégories. Les premiers s'appuient directement sur un nombre fini d'évaluations de la fonction objectif pour définir le nouvel itéré. Les seconds affinent itérativement un modèle de substitution de f et en cherchent les minimiseurs.
L'incertitude portant sur les évaluations de la fonction objectif peuvent être de différentes natures. D'une part la précision des machines est bien sûr limitée, et les opérations à virgule flottante intervenant dans les méthodes numériques créent par ailleurs du bruit numérique altérant les simulations. Cette imprécision est en pratique toujours présente, même si les utilisateurs appliquent souvent les algorithmes d'optimisation sans la prendre en compte. D'autre part certains paramètres du modèle sous-jacent peuvent être incertains, ou le modèle lui-même peut reposer sur des tirages aléatoires. Dans ce mémoire nous considérerons souvent le cas particulier où l'incertitude portant sur les valeurs de f résulte de paramètres incertains figurant dans sa définition : nous supposerons qu'il existe un nombre fini m de paramètres réels, représentés par un vecteur u de l'espace euclidien $\mathbb{R}^m$, tels que la valeur de f en un point x est en fait de la forme $f(x; u)$. Le problème d'optimisation considéré prend alors la forme suivante.

$$\begin{aligned} \text{Minimiser} \quad & f(x; u) \\ \text{s. c.} \quad & x \in \mathcal{X}. \end{aligned} \tag{P'}$$

C'est habituellement la théorie des probabilités qui est employée pour modéliser l'incertitude, qu'elle relève de l'imprécision, de l'ignorance ou de l'aléa. La théorie des ensembles flous, et la théorie des possibilité dont elle est le socle, se veut pourtant une alternative aux probabilités lorsqu'il s'agit de modéliser l'imprécision de nature non stochastique. L'optimisation *floue* est particulièrement développée en ce qui concerne la programmation linéaire à coefficients incertains, et la programmation quadratique dans une certaine mesure. Cependant les problèmes non linéaires plus généraux ont été relativement peu étudiés jusqu'à présent.
Au Chapitre 1 nous étudions la littérature de l'optimisation sans dérivées et de la modélisation de l'incertitude. Nous passons en revue des méthodes de recherche directe et des méthodes

basées sur la construction de modèles de substitution. Nous insistons en particulier sur la recherche directe directionnelle dans le premier cas et l'algorithme *Efficient Global Optimization* (EGO) dans le second. Nous nous intéressons ensuite à la modélisation des incertitudes en optimisation. En premier lieu nous discutons des différentes natures de l'incertitude. Nous examinons ensuite sa modélisation par des intervalles et donnons un sens à leur minimisation via l'optimisation bicritère. Nous traitons également les méthodes d'optimisation *stochastique* et détaillons en particulier une extension de l'algorithme EGO aux fonctions présentant des paramètres aléatoires dans leur définition. Enfin nous présentons la théorie des ensembles flous, très liée à la théorie des intervalles, qui se donne pour but de modéliser l'imprécision.
Le Chapitre 2 est le fruit d'un projet, mené avec deux doctorants, au cours du Centre d'été de mathématiques et de recherche avancée en calcul scientifique (CEMRACS) 2013, consacré à l'émulation de codes de simulation stochastiques. L'élaboration d'émulateurs peu exigeants en ressources pour des simulateurs coûteux permet de mener des études préliminaires sur ces codes avec un temps de calcul moindre. Ce projet était motivé par trois applications industrielles du Commissariat à l'énergie atomique et aux énergies alternatives (CEA), de Électricité de France (EDF) et IFP Energies Nouvelles (IFPEN). Quatre méthodes de construction d'émulateurs pour des fonctions dont les valeurs sont des densités de probabilité ont été conçues. Ces méthodes ont été testées sur deux exemples-jouets et appliquées à des codes de simulation industriels concernés par trois phénomènes complexes : la distribution spatiale de molécules dans un système d'hydrocarbures (IFPEN), le cycle de vie de grands transformateurs électriques (EDF) et les répercussions d'un hypothétique accident dans une centrale nucléaire (CEA). Dans les deux premiers cas l'émulation est une étape préalable à la résolution d'un problème d'optimisation.
Au Chapitre 3 nous analysons l'influence de l'inexactitude des évaluations de la fonction objectif sur la recherche directe directionnelle. Dans les problèmes réels, l'imprécision est sans doute toujours présente. Pourtant les utilisateurs appliquent généralement les algorithmes de recherche directe sans prendre cette imprécision en compte. Nous posons trois questions. Quelle précision peut-on espérer obtenir, étant donnée l'inexactitude ? À quel prix cette précision peut-elle être atteinte ? Quels critères d'arrêt permettent de garantir cette précision ? Nous répondons à ces trois questions pour l'algorithme de recherche directe directionnelle appliqué à des fonctions dont l'imprécision sur les valeurs — stochastique ou non — est uniformément bornée. Nous déduisons de nos résultats un algorithme adaptatif pour utiliser efficacement des oracles de niveaux de précision distincts. Les résultats théoriques et l'algorithme sont validés avec des tests numériques et deux applications réelles : la minimisation de surface en conception mécanique et le placement de puits pétroliers en ingénierie de réservoir.
Le Chapitre 4 est dédié aux problèmes d'optimisation affectés par des paramètres imprécis, dont l'imprécision est modélisée grâce à la théorie des ensembles flous. Nous proposons un algorithme de résolution d'une large classe de problèmes par tri non-dominé itératif. Les distributions des paramètres flous sont seulement supposées partiellement connues. Nous définissons également un critère pour évaluer la précision des solutions obtenues et tirons des comparaisons avec d'autres algorithmes de la littérature. Nous démontrons également que notre algorithme garantit des solutions dont le niveau de précision est au moins égal à la précision des données disponibles.

Table des matières

Chapitre 1

Optimisation sans dérivées sous incertitudes

Dans ce chapitre nous nous intéressons au problème (P) de la minimisation de la fonction f à valeurs réelles incertaines sur le sous-ensemble $\mathcal{X}$ de l'espace euclidien $\mathbb{R}^n$ par des méthodes ne faisant pas appel aux valeurs des dérivées de f. Nous rappelons ci-dessous la forme prise par ce problème.

$$\begin{aligned} \text{Minimiser} \quad & f(x) \\ \text{s. c.} \quad & x \in \mathcal{X}. \end{aligned} \tag{P}$$

Dans la Section 1.1 nous passons en revue des méthodes d'optimisation dites *sans dérivées*. En pratique les dérivées de la fonction f peuvent être trop coûteuses à obtenir, peu fiables, inaccessibles, voire même inexistantes. C'est notamment le cas lorsque la fonction f est une *boîte noire* : sa définition analytique n'est pas connue et ses valeurs s'obtiennent par des appels à un oracle dont le temps de réponse est éventuellement élevé (de quelques minutes à plusieurs heures). Ces évaluations peuvent nécessiter l'exécution de codes de simulation numérique très complexes dont les sources sont éventuellement indisponibles. L'approximation du gradient de f par des méthodes de différences finies n'est alors pas envisageable. Nous sommes contraints de résoudre le problème (P) sans connaître le gradient de f ; nous parlons d'optimisation *sans dérivées*.
Dans la Section 1.2 nous nous intéressons à la modélisation des incertitudes portant sur les valeurs de la fonction objectif du problème (P). Cette incertitude peut être due à un manque de connaissance, à des informations imprécises ou à un manque de fiabilité dans les données du problème. Nous étudierons tout particulièrement le cas où l'incertitude sur les évaluations de f résulte de la présence de paramètres mal connus dans la définition de la fonction objectif.

1.1 Optimisation sans dérivées

Les premiers algorithmes d'optimisation sans dérivées ont été introduits dans les années 1960 [80] et se sont depuis perfectionnés et diversifiés [89]. Nous pouvons distinguer deux

types de méthodes. Les premières sont les méthodes par recherche directe qui s'appuient directement sur un nombre fini d'évaluations de la fonction objectif pour déterminer le prochain point à évaluer. Les secondes reposent sur un modèle de substitution — ou métamodèle — de la fonction f, moins coûteux à évaluer et plus simple à analyser, construit à partir de valeurs de la fonction objectif et raffiné itérativement.
Dans un premier temps nous passons en revue différentes méthodes d'optimisation sans dérivées par recherche directe et étudions en détails l'algorithme de recherche directe directionnelle à la Section 1.1.1, dont l'application à des fonctions imprécises sera analysée au Chapitre 3. Dans un second temps nous nous intéressons aux méthodes basées sur un métamodèle et détaillons l'algorithme *Efficient global optimization* [56], basé sur un métamodèle par krigeage, dans la Section 1.1.2.

1.1.1 Méthodes par recherche directe

Recherche directe déterministe Intéressons-nous d'abord aux algorithmes déterministes de recherche directe, c'est-à-dire ceux dont les itérations ne font pas appel à des tirages aléatoires. L'algorithme de Nelder–Mead [80] en est un exemple classique. Un simplexe de $\mathbb{R}^n$ aux sommets duquel la fonction objectif est évaluée est transformé itérativement : à chaque itération le sommet x^{n+1} auquel correspond la plus forte valeur de l'objectif est remplacé par un nouveau point de la droite passant par x^{n+1} et l'isobarycentre des n autres sommets, par une opération de réflexion, d'expansion, de contraction ou de réduction. L'algorithme de recherche directe directionnelle [25,61,109] consiste à évaluer l'objectif autour du point courant en le déplaçant d'un certain pas selon un ensemble de directions positivement générateur de $\mathbb{R}^n$. Nous étudions cet algorithme en détails plus loin dans cette section et nous analysons ses performances sur des fonctions imprécises au Chapitre 3. Lorsque la fonction objectif est supposée lipschitzienne et que le problème n'est soumis qu'à des contraintes de bornes — l'espace $\mathcal{X}$ des solutions admissibles est alors un hyper-rectangle — la méthode de Shubert [99] consiste à partitionner itérativement l'espace $\mathcal{X}$. La fonction f est d'abord évaluée en les points extrémaux de $\mathcal{X}$, puis des sous-estimateurs linéaires de f sont obtenus à partir de la condition de Lipschitz. Une partition de $\mathcal{X}$ en hyper-rectangles est alors construite à partir des points qui minimisent ces sous-estimateurs et les étapes précédentes sont appliquées à chacun des hyper-rectangles. Les algorithmes DIRECT (*Divide a hyper-rectangle*) et *Branch-and-bout* sont basés sur cette méthode. Une autre approche par partitionnement est appelée *Multilevel coordinate search* [48]. L'espace $\mathcal{X}$ est divisé en hyper-rectangles qui sont partitionnés itérativement. À chaque hyper-rectangle correspond un des ses éléments, en lequel la fonction objectif est évaluée, et un paramètre *level* qui dépend de manière croissante du nombre de fois que l'hyper-rectangle a déjà été subdivisé. À chaque itération, pour chaque valeur possible du paramètre *level*, l'algorithme subdivise les hyper-rectangles auxquels correspondent la valeur la plus faible de la fonction objectif, incrémente la valeur du paramètre *level*, et évalue f en un point de chaque nouvel hyper-rectangle.

Recherche directe stochastique Dans ce paragraphe nous considérons les méthodes aléatoires de recherche directe ; la progression des algorithmes est ici gouvernée par des tirages

de variables aléatoires. Le plus simple est l'algorithme *Hit-and-run* [6]. À chaque itération un vecteur u de la sphère unité et un pas s sont tirés aléatoirement de sorte que le point obtenu en translatant le point courant selon le vecteur su soit admissible. La fonction objectif est alors évaluée en ce nouveau point, qui devient ou non le point courant suivant que la valeur obtenue est inférieure ou non à celle du point courant. L'algorithme de recherche directe directionnelle peut être rendu stochastique en tirant aléatoirement les directions d'exploration. Les auteurs de [43] montrent qu'un tel algorithme ne nécessite qu'un nombre limité de directions pour assurer une convergence presque sûre. La méthode du *recuit simulé* [58] autorise, elle, un point en lequel la valeur de la fonction objectif est plus forte qu'au point courant à être conservé, avec une certaine probabilité. Ceci a pour but d'éviter que la trajectoire de l'algorithme ne soit piégée au voisinage d'un minimum local. Les algorithmes *génétiques*, comme *Covariance matrix adaptation – Evolution strategy* (CMA-ES) [45], s'inspirent de la théorie darwinienne de l'évolution pour faire évoluer une population de points selon des lois probabilistes vers des minimiseurs de la fonction objectif. Les algorithmes *par essaims particulaires* [57] consistent à faire se déplacer dans l'espace $\mathcal{X}$ un essaim de points, chacun affecté d'un vecteur vitesse, selon des règles probabilistes. Supposons que le problème (P) est non contraint, c'est-à-dire $\mathcal{X} = \mathbb{R}^n$. Celui-ci prend alors la forme ci-dessous.

$$\begin{aligned} \text{minimiser} \quad & f(x) \\ \text{s. c.} \quad & x \in \mathbb{R}^n. \end{aligned}$$

Faisons l'hypothèse suivante sur la fonction objectif qui permet, entre autres choses, d'assurer un résultat de convergence de l'algorithme de recherche directe directionnelle.

Hypothèse 1. La fonction f est minorée et continûment différentiable sur $\mathbb{R}^n$. Son gradient ∇f est lipschitzien sur $\mathbb{R}^n$. Notons ν la meilleure constante de Lipschitz du gradient de f, définie ci-dessous :

$$\nu = \sup\{\|\nabla f(x) - \nabla f(y)\|/\|x - y\| : x, y \in \mathbb{R}^n\}.$$

Nous supposons toujours que les valeurs du gradient de f sont inaccessibles et que les évaluations de f sont trop coûteuses pour permettre une approximation raisonnable. Sous l'Hypothèse 1, l'inégalité suivante est vérifiée pour tout x et y dans $\mathbb{R}^n$:

$$|f(y) - f(x) - \nabla f(x)^\top (y - x)| \leq \frac{\nu}{2}\|y - x\|^2. \tag{1.1}$$

Direction de descente L'algorithme de recherche directe directionnelle [25, 61, 109] est un algorithme classique à directions de descente : le point courant est déplacé itérativement le long d'une direction qui fait décroître la valeur de la fonction objectif. Plus précisément, nous appellerons *direction de descente* de f en un point x de $\mathbb{R}^n$ tout vecteur d de $\mathbb{R}^n$ tel que $f(x+\alpha d) < f(x)$ pour tout pas α strictement positif et suffisamment petit. Sous l'Hypothèse 1 il est connu qu'en chaque point x qui n'annule pas le gradient de f tout vecteur d de $\mathbb{R}^n$ tel que $-\nabla f(x)^\top d > 0$ est une direction de descente. En effet, en remplaçant y par $x + \alpha d$ — où d est un vecteur de $\mathbb{R}^n$ et α un nombre strictement positif — dans (1.1) nous obtenons

$$f(x + \alpha d) - f(x) \leq \alpha \nabla f(x)^\top d + \frac{\nu}{2}\alpha^2 \|d\|^2, \tag{1.2}$$

dont le membre droit est strictement négatif si α est assez petit. Autrement dit, tout vecteur non nul formant un angle aigu avec l'opposé du gradient (non nul) de f en x est une direction de descente. En particulier $-\nabla f(x)$ lui-même est une direction de descente en x. C'est d'ailleurs cette remarque qui est à la base de l'algorithme du gradient en optimisation différentiable.

Familles positivement génératrices Lorsque le gradient de la fonction f n'est pas accessible il n'est pas possible de trouver une direction de descente de manière aussi directe. Cependant nous pouvons assurer l'existence d'une direction de descente au sein de familles de vecteurs positivement génératrices. Une famille finie $\mathcal{D}$ de vecteurs $d_1, \ldots, d_r$ de $\mathbb{R}^n$ est dite *positivement génératrice* si tout vecteur v de $\mathbb{R}^n$ peut s'écrire comme une combinaison linéaire d'éléments de $\mathcal{D}$ dont les coefficients sont positifs : il existe des coefficients $\lambda_1, \ldots, \lambda_r$ dans $\mathbb{R}_+$ tels que $v = \lambda_1 d_1 + \lambda_2 d_2 + \cdots \lambda_r d_r$. Un exemple élémentaire de famille positivement génératrice de $\mathbb{R}^n$ est la réunion des vecteurs de la base canonique de $\mathbb{R}^n$ et de leurs opposés. Nous appellerons *directions* les éléments de $\mathcal{D}$. Il est connu [25, Theorem 2.3.] qu'une famille de vecteurs non nuls de $\mathbb{R}^n$ est positivement génératrice si et seulement si, pour tout vecteur non nul v de $\mathbb{R}^n$, il existe une direction d dans $\mathcal{D}$ telle que $v^\top d > 0$. Autrement dit, pour tout vecteur v non nul, il existe une direction d dans $\mathcal{D}$ formant un angle aigu avec v. Ainsi en parcourant les directions de $\mathcal{D}$ — selon un ordre arbitraire — nous sommes assurés d'en trouver une qui soit une direction de descente pour f au point courant. La *mesure cosinus* κ d'une famille positivement génératrice $\mathcal{D}$ est définie par

$$\kappa = \min_{\|v\|=1} \max_{d \in \mathcal{D}} v^\top d. \tag{1.3}$$

Si v est un vecteur de $\mathbb{R}^n$ alors, par définition de la mesure cosinus, il existe une direction d dans $\mathcal{D}$ telle que $\kappa\|v\| \leq -v^\top d$. Dans la suite de cette section nous ferons l'hypothèse suivante.

Hypothèse 2. L'ensemble $\mathcal{D}$ est une famille positivement génératrice de $\mathbb{R}^n$ dont les éléments d sont de norme 1.

Recherche directionnelle L'algorithme de recherche directe directionnelle découle naturellement des remarques précédentes. Donnons-nous une famille positivement génératrice $\mathcal{D}$, un point initial x_0 de $\mathbb{R}^n$ et un pas initial α_0 strictement positif. Au début de chaque itération k un point courant x_k et un pas α_k sont donnés. L'ensemble $\mathcal{D}$ est parcouru jusqu'à trouver une direction d telle que $f(x_k + \alpha_k d) < f(x_k)$ — nous savons qu'il en existe une si α est suffisamment petit. Si une telle direction est obtenue le nouvel itéré est défini par $x_{k+1} = x_k + \alpha_k d$ et le pas est maintenu tel quel, voire augmenté : $\alpha_{k+1} = \gamma\alpha_k$ où γ est un élément de $[1, \infty)$. L'itération k est alors qualifiée de *succès*. Si $\mathcal{D}$ a été parcouru entièrement sans qu'aucune direction satisfaisante n'ait été trouvée le nouvel itéré est identique au point courant et le pas est réduit : $x_{k+1} = x_k$ et $\alpha_{k+1} = \theta\alpha_k$, où θ est dans $(0, 1]$. L'itération k est alors qualifiée d'*échec*.

Condition de réduction suffisante Afin d'assurer la convergence de l'algorithme — en un sens que nous allons préciser — une diminution suffisante de la valeur de l'objectif au point

courant est exigée : donnons-nous une fonction ρ de $\mathbb{R}_{++}$ dans $\mathbb{R}_{++}$. Nous ferons le choix classique $\rho(\alpha) = c\alpha^2/2$ pour tout α dans $\mathbb{R}_{++}$. L'algorithme auquel nous nous intéressons est synthétisé dans le paragraphe suivant. D'après l'Équation (1.2) nous avons

$$f(x + \alpha d) - f(x) + \frac{c\alpha^2}{2} \leq \alpha \nabla f(x)^\top d + \frac{(c + \nu)\alpha^2}{2},$$

dont le membre de droite est strictement négatif si α est assez petit.

Algorithme 1.

1. Sélectionner θ dans $(0, 1)$, γ dans $[1, \infty)$, et $\mathcal{D} \subset \mathbb{R}^n$.
2. Choisir le point de départ x_0 dans $\mathbb{R}^n$ et le pas initial α_0 dans $(0, \infty)$.
3. Pour $k = 0, 1, 2, \ldots$
   - s'il existe une direction d dans $\mathcal{D}$ telle que $f(x_k + \alpha_k d) < f(x_k) - c\alpha_k^2/2$
     définir $\begin{cases} x_{k+1} = x_k + \alpha_k d \\ \alpha_{k+1} = \gamma \alpha_k \end{cases}$; l'itération k est qualifiée de *succès* ;
   - sinon
     définir $\begin{cases} x_{k+1} = x_k \\ \alpha_{k+1} = \theta \alpha_k \end{cases}$; l'itération k est qualifiée d'*échec*.

L'étape 3 peut éventuellement être précédée d'une phase de recherche où la fonction f est évaluée en des points choisis intuitivement, ou selon une heuristique, dans l'espoir de trouver directement un point diminuant la valeur de f.

Convergence Notons g_k le gradient de f en l'itéré x_k pour tout k de $\mathbb{N}$. Le résultat classique suivant assure la convergence de l'Algorithme 1 sous les hypothèses précédentes.

Théorème 1 (Torczon [109], Kolda, Lewis et Torczon [61])**.** *Supposons les Hypothèses 1 et 2 vérifiées. La suite générée $\{x_k\}_{k\in\mathbb{N}}$ par l'Algorithme 1 est telle que*

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{1.4}$$

Avant de proposer une brève présentation d'une preuve du Théorème 1 citons le résultat suivant, qui assure la convergence de la suite $\{\alpha_k\}_{k\in\mathbb{N}}$ vers 0.

Lemme 1 (Gratton, Royer, Vicente, Zhang [43])**.** *Supposons les Hypothèses 1 et 2 vérifiées. La suite de pas $\{\alpha_k\}_{k\in\mathbb{N}}$ générée par l'Algorithme 1 est telle que*

$$\sum_{k\in\mathbb{N}} \alpha_k^2 \leq \frac{2\gamma^2}{c(1 - \theta^2)} \left[\frac{c\alpha_0^2}{2\gamma^2} + f(x_0) - \inf_{x\in\mathbb{R}^n} f(x) \right].$$

Nous généraliserons ce résultat au Chapitre 3.

Démonstration du Théorème 1. Le Lemme 1 implique que la suite $\{\alpha_k\}_{k\in\mathbb{N}}$ converge vers 0 et qu'il y a une infinité d'itérations qui échouent, puisque c'est lors de ces itérations que la taille du pas est diminuée. Donnons-nous un entier k et notons g_k le gradient de f au point x_k. D'après la définition de la mesure cosinus (1.3), il existe une direction d de $\mathcal{D}$ telle que $\kappa\|g_k\| \leq -g_k^\top d$. Si l'itération k échoue alors

$$\begin{aligned} 0 &\leq f(x_{k+1}) - f(x_k) + \frac{c}{2}\alpha_k^2 \\ &\leq \alpha_k g_k^\top d + \frac{\nu}{2}\alpha_k^2 + \frac{c}{2}\alpha_k^2 \\ &\leq -\kappa\|g_k\|\alpha_k + \frac{c+\nu}{2}\alpha_k^2, \end{aligned}$$

d'où la majoration suivante de la norme du gradient de f en x_k :

$$\|g_k\| \leq \frac{c+\nu}{2\kappa}\alpha_k. \tag{1.5}$$

Puisque les échecs sont en nombre infini et que la suite des pas converge vers 0, l'Équation 1.4 est vérifiée. □

Complexité dans le pire des cas Le résultat suivant donne une borne sur le nombre d'itérations nécessaires pour conduire la norme du gradient sous un seuil ϵ donné. C'est également un critère d'arrêt lorsque la précision requise sur le gradient est connue. Nous étendrons ce résultat au Chapitre 3.

Théorème 2 (Vicente [110])**.** *Supposons les Hypothèses 1 et 2 vérifiées. Soit ϵ un réel dans $(0, 1]$. Notons k_ϵ est le premier indice tel que $\|g_{k_\epsilon}\| \leq \epsilon$. Alors*

$$k_\epsilon = \mathcal{O}(\epsilon^{-2}).$$

Démonstration. Présentons brièvement une preuve de ce résultat. D'après le Lemme 1 la suite des pas $\{\alpha_k\}_{k\in\mathbb{N}}$ converge vers 0. Il existe donc une infinité d'itérations qui échouent. Notons k_0 l'indice de la première d'entre elles — qui est indépendant de ϵ. Si $k_\epsilon \leq k_0$ alors $k_\epsilon \leq k_0\epsilon^{-2}$. Supposons $k_\epsilon > k_0$. D'après la définition de k_ϵ l'itération $k_\epsilon - 1$ est nécessairement un succès, donc $k_0 + 2 \leq k_\epsilon$. Pour tout entier k notons

$$y_k = \begin{cases} 1 & \text{si l'itération } k \text{ est un succès,} \\ 0 & \text{si l'itération } k \text{ est un échec.} \end{cases}$$

Nous pouvons alors écrire

$$\alpha_{k_\epsilon} = \alpha_{k_0}\gamma^{\sum_{k=k_0}^{k_\epsilon-1} y_k}\theta^{\sum_{k=k_0}^{k_\epsilon-1}(1-y_k)} = \alpha_{k_0}\exp\left[\ln\theta(k_\epsilon - k_0) + \ln(\theta^{-1}\gamma)\sum_{k=k_0}^{k_\epsilon-1} y_k\right].$$

Par conséquent

$$k_\epsilon = k_0 - \frac{\ln(\alpha_{k_\epsilon}^{-1}\alpha_{k_0})}{\ln\theta} - \frac{\ln(\theta^{-1}\gamma)}{\ln\theta}\sum_{k=k_0}^{k_\epsilon-1} y_k. \tag{1.6}$$

Comme k_0 est constant relativement à ϵ et que ϵ est inférieur à 1, on sait déjà que $k_0 \leq k_0\epsilon^{-2}$, d'où $k_0 = \mathcal{O}(\epsilon^{-2})$ quand ϵ tend vers 0. Notons k le plus grand entier strictement inférieur à k_ϵ tel que l'itération k échoue (il existe et est supérieur à k_0). Alors $\alpha_{k_\epsilon} = \gamma^{k_\epsilon - k - 1}\theta\alpha_k \geq \theta\alpha_k$. Il s'ensuit de $\|g_k\| > \epsilon$ et (1.5) que $\alpha_{k_\epsilon} > 2\epsilon\kappa\theta/(c+\nu)$. Par conséquent le second terme de (1.6) est majoré par $\ln(\epsilon^{-1})$, donc à ϵ^{-2}, à une constante multiplicative près. Intéressons-nous enfin au troisième terme du membre droit de (1.6). Comme l'itéré n'est pas modifié en cas d'échec, nous pouvons écrire ce qui suit :

$$\sum_{k=k_0}^{k_\epsilon-1} y_k \frac{c\alpha_k^2}{2} < \sum_{k=k_0}^{k_\epsilon-1} y_k[f(x_k) - f(x_{k+1})] = \sum_{k=k_0}^{k_\epsilon-1} [f(x_k) - f(x_{k+1})] = f(x_{k_0}) - f(x_{k_\epsilon}). \tag{1.7}$$

Soit k un entier compris entre k_0 et $k_\epsilon - 1$ tel que l'itération k soit un succès, alors, par un raisonnement déjà vu plus haut, on vérifie que $\alpha_k > 2\epsilon\kappa\theta/(c+\nu)$. Grâce à cette minoration nous déduisons de l'Équation (1.7) que $\sum_{k=k_0}^{k_\epsilon-1} y_k$ est inférieur à ϵ^{-2} à une constante multiplicative près. Ainsi chacun des termes du membre droit de (1.6) est inférieur à ϵ^{-2} à une constante multiplicative près et $k_\epsilon = \mathcal{O}(\epsilon^{-2})$ quand ϵ tend vers 0. □

Nous étendrons les résultats de convergence, de complexité et d'arrêt de l'algorithme de recherche directe directionnelle aux fonctions prenant des valeurs imprécises, et dont l'erreur est uniformément bornée, au Chapitre 3.

1.1.2 Méthodes par métamodèle

Méthodes basées sur un métamodèle Ces méthodes consistent à construire un métamodèle $\hat{f}$ de la fonction objectif f à partir d'un échantillon de valeurs connues. C'est ce modèle qui est évalué au cours de la recherche de minimiseurs, tout en étant raffiné itérativement à mesure que de nouveaux points sont évalués. Les méthodes à région de confiance sont basées sur la construction d'un modèle — quadratique par exemple — dont la zone de validité est augmentée ou réduite suivant que la valeur minimale du modèle local est meilleure ou non que celle au point courant. L'algorithme *Efficient global optimization* (EGO) se base sur un modèle par krigeage, affiné itérativement par ajouts de points qui améliorent la valeur de l'estimateur en moyenne. Le krigeage et l'algorithme EGO sont détaillés à partir du prochain paragraphe ; une adaptation [55] de cet algorithme à l'optimisation sous incertitude sera également présentée à la Section 1.2.3. Une autre approche consiste à construire le métamodèle à partir des *Radial basis functions* introduites par Powell [83]. Lorsque le métamodèle est un champ aléatoire $\{\hat{f}(x)\}_{x\in\mathcal{X}}$ l'algorithme *Sequential design for optimization* se donne pour but de minimiser $x \mapsto \hat{f}(x) - \tau s(x)$, où s est l'estimateur de l'écart-type du métamodèle et τ une constante positive. La méthode *Surrogate management framework* se base également sur un modèle simplifié de la fonction objectif. Les itérations de l'algorithme alternent entre évaluation de la fonction objectif en des points choisis grâce au métamodèle et raffinement du modèle. L'optimisation *by branch-and-fit* repose sur des modèles quadratiques et linéaires, des points pouvant par ailleurs être tirés aléatoirement pour raffiner les modèles ou assurer l'exploration de tout l'espace $\mathcal{X}$.

L'algorithme EGO L'algorithme EGO [56] — pour *Efficient Global Optimization* — est un algorithme d'optimisation sans dérivées qui construit un métamodèle par krigeage de la fonction objectif à partir d'un plan d'expérience initial, puis l'affine itérativement en incorporant des points qui améliorent la valeur du modèle en moyenne.

Krigeage Le *krigeage*, ou modèle DACE — en référence au titre d'un article fondateur [95] —, est une méthode de construction de métamodèle. Les valeurs déterministes de la fonction f à modéliser sont considérées comme les réalisations d'un champ stochastique $\{Y_x\}_{x\in\mathbb{R}^n}$, c'est-à-dire d'une famille de variables aléatoires réelles, sur un espace probabilisé $(\Omega, \mathcal{A}, \mathrm{P})$, indexée par les points x de l'espace $\mathbb{R}^n$. Nous faisons ainsi l'hypothèse qu'il existe un état ω de l'espace Ω tel que

$$f(x) = Y_x(\omega), \quad x \in \mathbb{R}^n.$$

Nous supposerons que le champ stochastique $\{Y_x\}_{x\in\mathbb{R}^n}$ est stationnaire d'ordre 2, au sens où il satisfait les deux propriétés ci-dessous.

1. L'espérance du champ est une constante réelle μ :

$$\mathrm{E}[Y_x] = \mu, \quad x \in \mathbb{R}^n.$$

2. La covariance $\mathrm{Cov}\,[Y_x, Y_{x'}]$ ne dépend que de la différence $x - x'$ entre les points x et x' de $\mathbb{R}^n$. En particulier la variance du champ est une constante strictement positive (dont nous notons σ la racine carrée) :

$$\mathrm{Var}\,[Y_x] = \sigma^2, \quad x \in \mathbb{R}^n.$$

Fixons un vecteur de paramètres x. L'idée essentielle du krigeage est de définir un estimateur linéaire $\widehat{Y}_x$ de Y_x, de biais $\mathrm{E}[\widehat{Y}_x - Y_x]$ nul, et optimal au sens où son erreur quadratique — *mean squared error*, en anglais — $\mathrm{MSE}[\widehat{Y}_x] = \mathrm{Var}[\widehat{Y}_x - Y_x]$ est minimale. Supposons que l'on dispose d'un échantillon de N jeux de paramètres $x^1, \ldots, x^N$ et des valeurs $y_1, \ldots, y_N$ associées de la fonction objectif :

$$y_i = f(x^i), \quad i = 1, \ldots, N. \tag{1.8}$$

Pour tout x de $\mathbb{R}^n$ définissons un estimateur de Y_x de la façon suivante :

$$\widehat{Y}_x = \sum_{i=1}^{N} \lambda_i(x) Y_{x^i} = \lambda(x)^\top \mathrm{Y}, \tag{1.9}$$

où le vecteur des poids de krigeage $\lambda(x) = (\lambda_1(x), \ldots, \lambda_N(x))$ est un élément de $\mathbb{R}^N$ dont la somme des coefficients vaut 1 et $\mathrm{Y} = (Y_{x^1}, \ldots, Y_{x^N})$ est le vecteur (aléatoire) des simulations aux points de l'échantillon.

Krigeage simple Le krigeage est dit *simple* lorsque la valeur μ de l'espérance est supposée connue. Supposons dans un premier temps que μ est nul. L'estimateur est alors sans biais :

$$\mathrm{E}[\widehat{Y}_x - Y_x] = \sum_{i=1}^{N} \lambda_i(x)\mu - \mu = 0, \quad x \in \mathbb{R}^n.$$

Nous supposons également la variance σ^2 et les corrélations du champ connues. Notons R la matrice symétrique positive de taille $N \times N$ des corrélations de l'échantillon, et $r(x)$ le vecteur de taille N des corrélations entre un point x et l'échantillon, dont les coefficients respectifs sont donnés par :

$$\begin{aligned} R_{ij} &= \mathrm{Corr}\left[Y_{x^i}, Y_{x^j}\right], \\ r_j(x) &= \mathrm{Corr}\left[Y_x, Y_{x^j}\right], \end{aligned} \qquad i, j = 1, \ldots, N. \tag{1.10}$$

L'erreur quadratique s'écrit alors :

$$\mathrm{MSE}[\widehat{Y}_x] = \sigma^2 \left[\lambda(x)^\top R \lambda(x) - 2\lambda(x)^\top r(x) + 1\right], \quad x \in \mathbb{R}^n. \tag{1.11}$$

Sous réserve que la matrice R soit inversible, le vecteur $\lambda(x)$ est l'unique solution du problème strictement convexe suivant :

$$\begin{aligned} \text{minimiser} \quad & \lambda^\top R \lambda - 2\lambda^\top r(x) \\ \text{s. c.} \quad & \lambda \in \mathbb{R}^N. \end{aligned}$$

Ainsi, pour tout x de $\mathbb{R}^n$, nous obtenons $\lambda(x) = R^{-1} r(x)$ et, par remplacement dans les équations (1.9) et (1.11),

$$\widehat{Y}_x = r(x)^\top R^{-1} \mathrm{Y}, \tag{1.12}$$

$$\mathrm{MSE}[\widehat{Y}_x] = \sigma^2 \left[1 - r(x)^\top R^{-1} r(x)\right]. \tag{1.13}$$

Si μ n'est pas nul il suffit maintenant de considérer le champ centré $\{Y_x - \mu\}_{x \in \mathbb{R}^n}$: nous obtenons

$$\widehat{Y}_x = \mu + r(x)^\top R^{-1} (\mathrm{Y} - \mu e), \quad x \in \mathbb{R}^n, \tag{1.14}$$

où e est le vecteur de $\mathbb{R}^N$ de coordonnées toutes égales à 1, et l'erreur quadratique (1.13) est inchangée. L'estimateur $\widehat{Y}_x$ donné par (1.14) est alors le meilleur estimateur affine sans biais de Y_x, pour tout x de $\mathbb{R}^n$. Dans la suite nous faisons l'hypothèse que le champ stochastique $\{Y_x\}_{x \in \mathbb{R}^n}$ est gaussien. Il alors est connu que, pour tout x dans $\mathbb{R}^n$, la loi de $\widehat{Y}_x$ est la loi conditionnelle de Y_x sachant les simulations aléatoires :

$$\widehat{Y}_x = \mathrm{E}[Y_x | Y_{x^1}, \ldots, Y_{x^N}], \quad x \in \mathbb{R}^n. \tag{1.15}$$

Nous définissons à présent l'estimateur par krigeage de la fonction f comme l'estimateur (1.15) conditionné par les évaluations (1.8) des points de l'échantillon par la fonction f :

$$\widehat{f(x)} = \mathrm{E}[Y_x | Y_{x^1} = y_1, \ldots, Y_{x^N} = y_N], \quad x \in \mathbb{R}^n. \tag{1.16}$$

L'estimateur $\widehat{f(x)}$ suit une loi normale dont l'espérance est donnée par

$$\mathrm{E}[\widehat{f(x)}] = \mu + r(x)^\top R^{-1} (y - \mu e), \quad x \in \mathbb{R}^n, \tag{1.17}$$

où y est le vecteur $(y_1, \ldots, y_N)$ de $\mathbb{R}^N$, et la variance par

$$\mathrm{Var}[\widehat{f(x)}] = \mathrm{MSE}[\widehat{Y}_x] = \sigma^2 \left[1 - r(x)^\top R^{-1} r(x)\right], \quad x \in \mathbb{R}^n. \tag{1.18}$$

Vérifions que l'estimateur est interpolant (P-presque sûrement). Soit i dans $\{1,\dots,N\}$. Le vecteur $r(x^i)^\top$ étant la i-ème ligne de la matrice R, il s'ensuit d'une part que $r(x^i)^\top R^{-1}$ est le i-ème vecteur de la base canonique de $\mathbb{R}^N$, et d'autre part que $r(x^i)^\top R^{-1} r(x^i) = \mathrm{Corr}\,[Y_{x^i}, Y_{x^i}] = 1$. Nous déduisons alors de (1.17) et (1.18) que

$$\mathrm{E}[\widehat{f(x^i)}] = \mu + (y^i - \mu) = y^i = f(x^i) \text{ et } \mathrm{Var}[\widehat{f(x^i)}] = \sigma^2(1-1) = 0.$$

Ainsi $\widehat{f(x^i)}$ est égale à $f(x^i)$ P-presque sûrement.

Estimation des hyperparamètres Dans la pratique l'espérance μ, la variance σ^2 et les corrélations ne sont pas connues. Supposons ques les corrélations sont paramétrées par des réels positifs $\theta_1,\dots,\theta_n$ comme suit :

$$\mathrm{Corr}\,[Y_x, Y_{x'}] = \exp\left[-\sum_{j=1}^{n} \theta_j |x_j - x'_j|^2\right],$$

pour tout $x = (x_1,\dots,x_n)$ et tout $x' = (x'_1,\dots,x'_n)$ dans $\mathbb{R}^n$. Observons que les corrélations ainsi définie sont bien stationnaires. Cette hypothèse suppose que des variables Y_x et $Y_{x'}$ sont d'autant plus corrélées que les points correspondants x et x' sont voisins. Dans la suite on notera R_θ au lieu de R pour insister sur la dépendance des corrélations en $\theta = (\theta_1,\dots,\theta_n)$. Les paramètres — ou *hyperparamètres* — du modèle à estimer sont donc μ, σ et θ. Nous procédons par maximisation de vraisemblance. Définissons une fonction de *vraisemblance* de la façon suivante : pour des valeurs fixées de μ dans $\mathbb{R}$, σ dans $\mathbb{R}_+^*$ et θ dans $\mathbb{R}_+^n$, notons $\mathcal{L}(\mu,\sigma,\theta)$ la valeur de la densité de la loi jointe normale des variables $Y_{x^1},\dots,Y_{x^N}$ au point y :

$$\mathcal{L}(\mu,\sigma,\theta) = \frac{1}{\sqrt{(2\pi\sigma^2)^N \det(R_\theta)}} \exp\left[-\frac{(y-\mu e)^\top R_\theta^{-1}(y-\mu e)}{2\sigma^2}\right]. \tag{1.19}$$

Nous estimons alors les hyperparamètres du modèle par les valeurs $\hat\mu$, $\hat\sigma$ et $\hat\theta$ qui maximisent la valeur de vraisemblance (1.19). Remarquons que

$$\sup_{(\mu,\sigma,\theta)\in\mathbb{R}\times\mathbb{R}_+^*\times\mathbb{R}_+^n} \mathcal{L}(\mu,\sigma,\theta) = \sup_{\theta\in\mathbb{R}_+^n} \sup_{\sigma\in\mathbb{R}_+^*} \sup_{\mu\in\mathbb{R}} \mathcal{L}(\mu,\sigma,\theta).$$

Soit σ dans $\mathbb{R}_+^*$ et θ dans $\mathbb{R}^n$. Par monotonie nous obtenons

$$\sup_{\mu\in\mathbb{R}} \mathcal{L}(\mu,\sigma,\theta) = \frac{1}{\sqrt{(2\pi\sigma^2)^N \det(R_\theta)}} \exp\left[-\frac{1}{2\sigma^2} \inf_{\mu\in\mathbb{R}} (y-\mu e)^\top R_\theta^{-1}(y-\mu e)\right].$$

Sous réserve que la matrice R_θ est inversible, nous vérifions que l'infimum en μ est atteint pour

$$\hat\mu = \frac{e^\top R_\theta^{-1} y}{e^\top R_\theta^{-1} e}. \tag{1.20}$$

Par conséquent

$$\sup_{\sigma\in\mathbb{R}_+^*} \sup_{\mu\in\mathbb{R}} \mathcal{L}(\mu,\sigma,\theta) = \frac{1}{\sqrt{(2\pi)^N \det(R_\theta)}} \sup_{\sigma\in\mathbb{R}_+^*} \frac{1}{\sigma^N} \exp\left[-\frac{(y-\hat\mu e)^\top R_\theta^{-1}(y-\hat\mu e)}{2\sigma^2}\right].$$

Nous pouvons vérifier que le supremum porte sur une fonction concave de σ admettant un unique maximum en $\hat{\sigma}$ donné par

$$\hat{\sigma}^2 = \frac{1}{N}(y - \hat{\mu}e)^\top R_\theta^{-1}(y - \hat{\mu}e). \tag{1.21}$$

Le calcul de $\hat{\theta}$, plus complexe, est effectué numériquement. En conclusion les valeurs des hyperparamètres μ, σ et θ sont estimées respectivement par (1.20), (1.21) et une valeur $\hat{\theta}$ maximisant $\mathcal{L}(\hat{\mu}, \hat{\sigma}, \cdot)$.

Présentons à présent la méthode *Efficient global optimization* [56] à proprement parler.

Initialisation de l'algorithme Un plan d'expérience initial de N points est construit $x_1, \ldots, x_N$ — par exemple avec un échantillonnage par hypercube latin [75]. Les valeurs (1.8) de f en les points de l'échantillon sont évaluées et la valeur $f_{\min}$ est initialisée avec la plus petite des évaluations :

$$f_{\min} = \min\{f(x^1), \ldots, f(x^N)\}. \tag{1.22}$$

L'estimateur initial $\widehat{f(x)}$ de la fonction objectif est alors obtenu avec la formule (1.16). Rappelons qu'il s'agit d'une variable gaussienne dont l'espérance (1.17) et la variance (1.18) se calculent simplement.
Il s'agit maintenant d'utiliser ce métamodèle pour sélectionner des points dont les valeurs par f sont inférieures à $f_{\min}$ et qui, intégrés au plan d'expérience, permettent de raffiner l'estimateur. Il existe différents types de critère de sélection — *Infill Sampling Criterion* [98]. Le choix classique est l'*Expected Improvement*, employé dans la définition originale de l'algorithme EGO [56].

Expected Improvement L'*Improvement* est le champ stochastique défini par

$$I_x = \max\{f_{\min} - \widehat{f(x)}, 0\}, \quad x \in \mathbb{R}^n.$$

Observons que, pour un état ω de Ω fixé, $I_x(\omega)$ mesure le gain sur le minimum courant $f_{\min}$ si $\widehat{f(x)}(\omega)$ est inférieur à $f_{\min}$, et vaut zéro si ce n'est pas le cas. L'*Expected Improvement* (EI) est tout simplement l'espérance de l'*Improvement* :

$$\mathrm{EI}(x) = \mathrm{E}[I_x], \quad x \in \mathbb{R}^n. \tag{1.23}$$

Quelques lignes de calcul intégral permettent d'écriture l'EI sous la forme suivante :

$$\begin{aligned} \mathrm{EI}(x) &= s[t\Phi(t) + \varphi(t)], \\ \text{avec } s &= \mathrm{MSE}[\widehat{f(x)}]^{1/2} \text{ et } t = (f_{\min} - \mathrm{E}[\widehat{f(x)}])/s, \end{aligned} \quad x \in \mathbb{R}^n, \tag{1.24}$$

où φ et Φ sont respectivement la densité de probabilité et la fonction de répartition de la loi normale standard – souvent notée $\mathcal{N}(0,1)$. L'EI (1.23) peut s'interpréter comme le gain moyen que le point x apporte par rapport au minimiseur courant — qui atteint la valeur $f_{\min}$ (1.22). La sélection du point à ajouter au plan d'expérience consiste ainsi à maximiser l'EI. Malgré la formule explicite (1.24) la maximisation de l'EI peut s'avérer complexe ; ce critère est en effet souvent très multimodal.

L'algorithme EGO L'étape de sélection d'un nouveau point est répétée jusqu'à ce qu'un critère d'arrêt soit atteint : un seuil sur le nombre d'évaluations de f, ou sur la valeur minimale atteinte par les évaluations de f. Résumons l'algorithme EGO de la manière suivante.

Algorithme 2 (EGO)**.**

1. Construire un plan d'expérience $x^1, \ldots, x^N$ initial.
2. Calculer les valeurs $f(x^1), \ldots, f(x^N)$ et initialiser $f_{\min}$:

$$f_{\min} \leftarrow \min\{f(x^1), \ldots, f(x^N)\}.$$

3. Tant que le critère d'arrêt n'est pas atteint :
   (a) calculer les hyperparamètres du métamodèle par krigeage,
   (b) trouver un maximiseur x^* de l'EI et l'ajouter au plan d'expérience,
   (c) calculer $f(x^*)$ et mettre $f_{\min}$ à jour :

$$f_{\min} \leftarrow \min\{f(x^*), f_{\min}\}.$$

La Figure 1.1 présente l'allure de la fonction objectif (inconnue, en trait discontinu), de l'espérance du métamodèle interpolant (dans la partie supérieure, en trait plein) et de l'EI (dans la partie inférieure, en trait plein), à l'initialisation de l'algorithme ainsi qu'aux itérations 2, 4 et 6 (de haut en bas et de gauche à droite). Nous observons qu'après l'itération 2 l'EI présente un maximum local au voisinage d'un minimum local de f. Les points interpolants ajoutés aux itérations 3 et 4 sont situés dans cet intervalle. Après l'itération 4 l'EI admet un maximum local prononcé au voisinage d'un autre minimum local de f ne contenant pas de point d'interpolation : le modèle y admet localement une erreur quadratique moyenne élevée. Les points ajoutés aux itérations 5 et 6 sont placés dans cette zone.

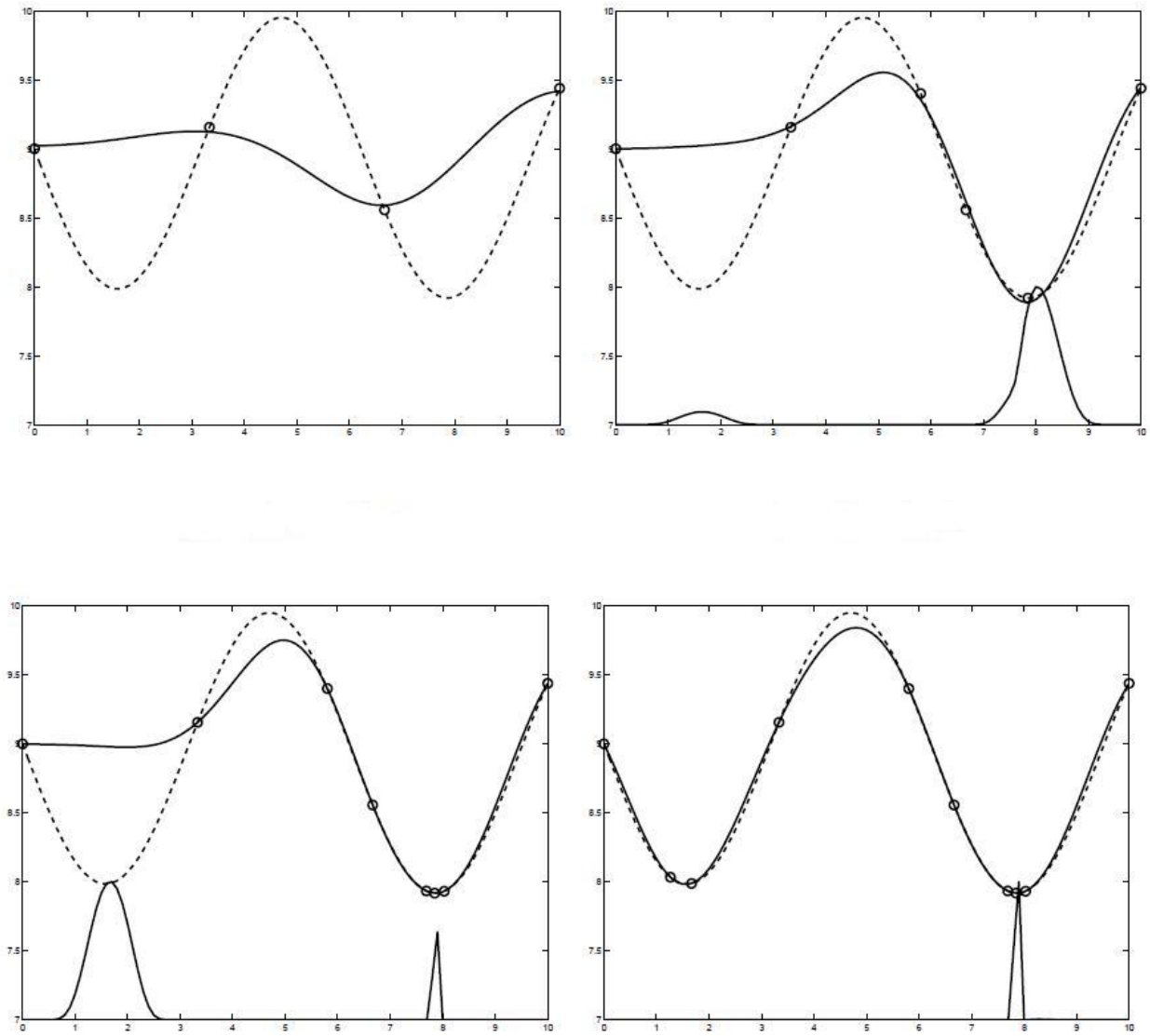

Figure 1.1 – Graphes de la fonction objectif (inconnue) en pointillés, de l'espérance du métamodèle interpolant en trait plein, et de l'*Expected Improvement* en trait plein au bas des graphes. De haut en bas et de gauche à droite : initialisation, itération 2, itération 4 et itération 6 de l'algorithme.

1.2 Modélisation des incertitudes et optimisation

Dans cette section nous nous intéressons à la modélisation de l'incertitude portant sur les valeurs de la fonction objectif du problème (P). Nous examinons les différentes sources de l'incertitude, puis nous présentons différentes théories pour la modéliser : les probabilités, les intervalles et les possibilités. Nous nous intéressons en particulier au cas où l'incertitude se manifeste par la présence de paramètres incertains dans la définition de f.

1.2.1 L'incertitude et sa modélisation

Nature de l'incertitude L'incertitude peut résulter d'un manque de connaissance ou de savoir-faire ; l'incertitude est alors dite *épistémique*. Une compréhension limitée ou une incapacité technique empêche de réunir l'information nécessaire pour lever l'incertitude. Si, au contraire, les moyens d'acquisition de mesures ou de données sont disponibles, celles-ci peuvent se révéler inexactes. L'information est alors accessible, mais avec une précision limitée. Ceci peut être dû à des mesures physiques inexactes ou en faible quantité, à des erreurs d'arrondi ou de troncature apparaissant au cours de simulations numériques. L'imprécision peut éventuellement être réduite grâce à un plus grand nombre de mesures ou avec plus de ressources de calcul. L'incertitude peut également être due à un doute sur la fiabilité des sources d'informations : appareils de mesures mal calibrés, données obtenues via un code numérique inaccessible pour des raisons juridiques, manque de confiance dans la méthodologie d'acquisition des données, hypothèses invérifiables, *etc*. En particulier l'incertitude peut résulter de données subjectives : avis d'experts, estimations intuitives.

Paramètres incertains Dans la suite de ce mémoire nous considérerons généralement que l'incertitude portant sur le problème de minimisation (P) se manifeste sous la forme de paramètres incertains dans la définition des valeurs de la fonction objectif. Nous représentons ces paramètres par m valeurs réelles $u_1, \ldots, u_m$ incertaines et nous notons u le vecteur de $\mathbb{R}^m$ défini par ces valeurs. Les évaluations de l'objectif prendront alors la forme $f(x; u)$ pour tout x dans $\mathbb{R}^n$. Réécrivons alors le problème (P) sous la forme ci-dessous.

$$\begin{aligned} \text{Minimiser} \quad & f(x; u) \\ \text{s. c.} \quad & x \in \mathcal{X}. \end{aligned} \tag{P'}$$

Nous étudierons plus loin différentes approches pour modéliser l'incertitude portant sur les paramètres.

Placement de puits Un problème classique permettant d'illustrer l'optimisation d'une fonction objectif soumise à des paramètres incertains est le choix d'emplacement de puits en ingénierie de réservoir. Il s'agit de déterminer les meilleurs endroits où ajouter des puits producteurs ou des puits injecteurs sur un champ pétrolier de manière à maximiser la quantité d'huile extraite du réservoir sur une période donnée, dix ans par exemple. Un cas test connu est le modèle de réservoir PUNQ [37], sur lequel existent initialement 6 puits producteurs et 5 puits injecteurs. Chaque nouveau puits à situer est caractérisé par deux coordonnées horizontales : ce sont les paramètres contrôlables x. Le modèle comporte une vingtaine de paramètres incertains u liés à la perméabilité du réservoir et la saturation en huile, dont la connaissance est imprécise. Pour un choix x de coordonnées de nouveaux puits, la réponse $f(x; u)$ du simulateur d'écoulements dans le réservoir est la quantité d'huile extraite sur la durée considérée en prenant en compte l'ajout des nouveaux puits. La Figure 1.2 montre trois scénarios de production cumulée d'huile, correspondant chacun à un vecteur u différent de paramètres incertains.

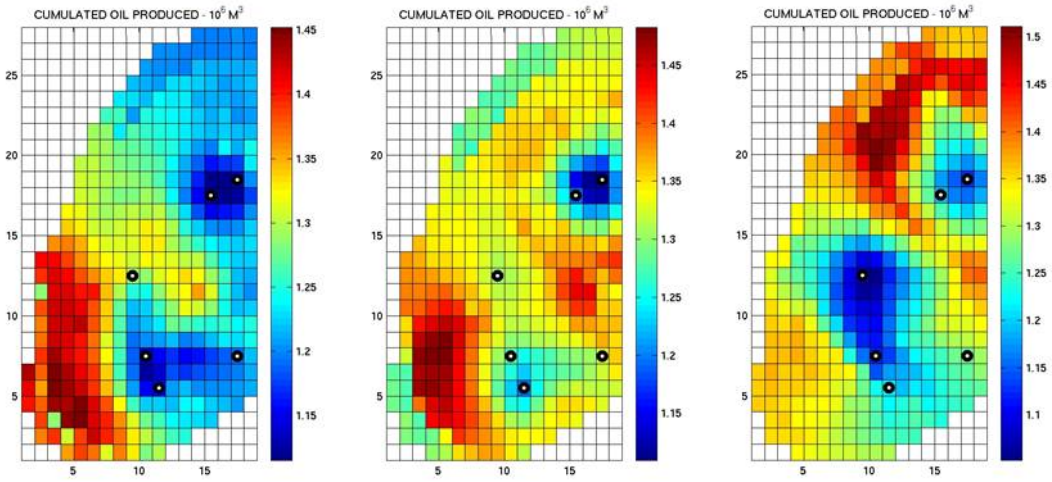

Figure 1.2 – Trois scénarios possibles de production cumulée d'huile sur le cas test PUNQ. Les carreaux représentent la surface des mailles et les point noirs sont les six puits producteurs.

Nous observons que le choix de ces paramètres a une grande influence sur la production d'huile.

Modélisation de l'incertitude Différentes approches existent pour modéliser l'incertitude dans le cadre d'un problème d'optimisation tel que (P'). L'optimisation sous incertitude est parfois aussi appelée optimisation *robuste* [8, 39] lorsque l'incertitude porte sur les valeurs de la fonction objectif, et *fiable* si l'incertitude porte sur les contraintes. Rappelons que dans ce mémoire nous ne nous intéresserons qu'aux incertitudes portant sur la fonction objectif.
Les toutes premières méthodes d'optimisation robuste, apparues dans les années 1980 [8], consistaient à discrétiser l'espace $\mathcal{X}$ des paramètres et une partie $\mathcal{U}$ de $\mathbb{R}^m$ contenant le vecteur $\bar{u}$ des véritables valeurs des paramètres inconnus. Les valeurs $f(x;u)$ sont alors calculées pour chaque couple (x,u) du produit cartésien des deux discrétisations, puis les paramètres donnant une valeur déviant le moins possible de l'objectif recherché sont sélectionnés. Relativement récemment les auteurs de [8] ont relevé les approches suivantes pour traiter les valeurs incertaines de la fonction objectif.
Une première approche consiste à remplacer les valeurs incertaines de la fonction objectif par les pires possibles : le problème d'optimisation considéré est alors

$$\begin{aligned} &\text{minimiser} \quad \sup f(x;\mathcal{U}) \\ &\text{s. c.} \quad x \in \mathcal{X}. \end{aligned} \tag{1.25}$$

Cette approche est très robuste dans le sens où elle revient à considérer le pire des cas. Cependant les véritables valeurs de la fonction objectif risquent d'être fortement dégradées par ce procédé, notamment si l'ensemble $\mathcal{U}$ est grand. Les auteurs de [26] font remarquer que

le problème (1.25) est en fait équivalent à un programme d'optimisation biniveaux :

$$\begin{aligned} \text{minimiser} \quad & f(x;u) \\ \text{s. c.} \quad & x \in \mathcal{X}, \\ & u \in \operatorname{Argmax} f(x;\mathcal{U}). \end{aligned}$$

Observons que si $\mathcal{U}$ est un singleton réduit au véritable vecteur de paramètres nous retrouvons le cas sans incertitude.
Une seconde approche consiste à substituer aux valeurs de l'objectif des intervalles qui les contiennent. Si par exemple, pour tout x de $\mathbb{R}^n$, il existe des réels $a(x)$ et $b(x)$ tels que $a(x) \leq f(x;\bar{u}) \leq b(x)$, où $\bar{u}$ est le véritable vecteur des paramètres, nous pouvons considérer le programme de minimisation d'intervalle suivant :

$$\begin{aligned} \text{minimiser} \quad & [a(x), b(x)] \\ \text{s. c.} \quad & x \in \mathcal{X}. \end{aligned}$$

Nous donnons un sens à ce type de problèmes à la Section 1.2.2.
L'approche la plus répandue dans la littérature est peut-être celle qui consiste à se ramener à un problème d'optimisation *stochastique*. En supposant que le vecteur de paramètres incertains est aléatoire — notons-le U — l'évaluation de la fonction objectif en un point x de $\mathbb{R}^n$ est alors interprétée comme une réalisation de la variable aléatoire $f(x;U)$. Nous pouvons substituer aux valeurs de la fonction objectif les espérances relativement à la loi de U — sous réserve d'intégrabilité — :

$$\begin{aligned} \text{minimiser} \quad & \mathrm{E}[f(x;U)] \\ \text{s. c.} \quad & x \in \mathcal{X}. \end{aligned}$$

Ainsi, plutôt que d'étudier le pire cas (1.25), c'est le cas moyen qui est traité. Observons que si U est constante presque sûrement nous retrouvons le problème sans incertitude. Nous étudions l'approche stochastique un peu plus en détails à la Section 1.2.3, où nous exposons une version stochastique de l'algorithme EGO [55].
La quatrième approche que nous considérons consiste à modéliser l'imprécision des paramètres incertains avec la théorie des ensembles flous [118], que nous présentons à la Section 1.2.4. Le vecteur incertain est ici remplacé par un vecteur flou $\tilde{u}$:

$$\begin{aligned} \text{minimiser} \quad & f(x;\tilde{u}) \\ \text{s. c.} \quad & x \in \mathcal{X}. \end{aligned} \tag{PF}$$

Nous explorons l'optimisation de fonctions présentant des paramètres flous dans leur définition au Chapitre 4. Nous y étudions la littérature donnant un sens au problème (PF) et nous proposons également un algorithme de résolution.

1.2.2 Intervalles et optimisation bicritère

Les valeurs incertaines de la fonction objectif f peuvent être modélisées par des intervalles de $\mathbb{R}$. Pour tout x de $\mathbb{R}^n$ un intervalle $A(x)$ est substitué à la valeur $f(x)$ de l'objectif. Cet intervalle est supposé contenir la véritable valeur. Voyons comment donner un sens à la minimisation d'une fonction dont les valeurs sont des intervalles.

Intervalle Nous désignerons par *intervalle* toute partie connexe de $\mathbb{R}$. Nous nous préoccuperons plus particulièrement des intervalles fermés et bornés de $\mathbb{R}$ ou, de manière équivalente, des parties convexes compactes de $\mathbb{R}$. Notons $\mathcal{I}_c$ l'ensemble des intervalles compacts de $\mathbb{R}$. Pour tout A dans $\mathcal{I}_c$ notons $A^L = \min A$ (pour *lower*) sa borne inférieure, $A^U = \max A$ (pour *upper*) sa borne supérieure, et écrivons $A = [A^L, A^U]$. Dans la suite de ce mémoire nous utiliserons des notations similaires dès que nous aurons à considérer des intervalles compacts de $\mathbb{R}$ — en particulier les coupes de nombres flous compacts dans la Section 1.2.4.

Arithmétique d'intervalles Avant d'aborder la question de la minimisation d'intervalle nous donnons quelques éléments d'arithmétique d'intervalles [76]. Ces notions permettront de faire un parallèle éclairant avec l'arithmétique des nombres flous à la Section 1.2.4. Soit A et B deux intervalles fermés et bornés. La somme de A et B est classiquement définie comme l'image du produit cartésien $A \times B$ par l'addition des nombres réels :

$$A + B = \{a + b : a \in A, b \in B\}.$$

Le produit externe λA par un réel λ et la différence $A - B$ sont définis selon le même principe :

$$\lambda A = \{\lambda a : a \in A\},$$
$$A - B = \{a - b : a \in A, b \in B\}.$$

Observons que la soustraction d'intervalles n'est pas une opération algébrique inverse de l'addition.

Ordre sur les intervalles Différents ordres partiels — relations binaires réfléxives, antisymétriques et transitives — sur l'ensemble $\mathcal{I}_c$ des intervalles compacts de $\mathbb{R}$ sont considérés dans la littérature [9, 21, 54, 71, 76]. Donnons-nous deux intervalles compacts A et B. Rappelons qu'avec les notations précédentes nous avons $A = [A^L, A^U]$ et $B = [B^L, B^U]$. Notons par ailleurs $A^C = A^L/2 + A^U/2$ (pour *center*) le milieu de A et $A^W = A^U - A^L$ (pour *width*) sa demi-longueur. Des nombres B^C et B^W sont définis de façon similaire. Dans le contexte de la minimisation d'intervalle, où le critère d'intérêt est une valeur dans un intervalle objectif A, la borne A^L peut s'interpréter comme le meilleur cas, la borne A^U comme le pire cas, le centre A^C comme le cas médian et A^W comme une mesure de la dispersion des valeurs du critère d'intérêt. Selon les sources l'intervalle A est dit inférieur ou égal à B si :

1. $A^L \leq B^L$ et $A^U \leq B^U$,
2. $A^C \leq B^C$ et $A^W \leq B^W$,
3. $A^U \leq B^U$ et $A^C \leq B^C$,
4. $(1 - \lambda)A^L + \lambda A^U \leq (1 - \lambda)B^L + \lambda B^U$ pour tout λ dans $[0, 1]$,
5. $A^U \leq B^L$.

Il est aisé de vérifier que la troisième définition généralise chacune des deux premières — plus précisément 3 est équivalent à la disjonction "1 ou 2" [54]. Les approches selon les définitions 1, 2 et 3 sont unifiées dans [21]. Au contraire la quatrième définition est plus forte que 1 et 3 — prendre λ égal à 0, 0.5 et 1 — et la dernière, qui réclame que les intervalles

soient disjoints, est plus restrictive encore que la quatrième. C'est la première définition que nous utiliserons dans la suite ; c'est un choix classique [30, 71, 91]. Nous noterons $A \preceq B$ si et seulement si les deux inégalités $A^L \leq B^L$ et $A^U \leq B^U$ sont vérifiées. L'odre $\preceq$ n'est pas un ordre total : par exemple ni $[0,3] \preceq [1,2]$ ni $[1,2] \preceq [0,3]$ n'est vérifié ; les intervalles $[0,3]$ et $[1,2]$ sont dits *incomparables* pour $\preceq$.

Optimisation d'intervalle Intéressons-nous maintenant à la minimisation d'une fonction g, définie sur $\mathbb{R}^n$ et à valeurs dans $\mathcal{I}_c$, relativement à l'ordre sur $\mathcal{I}_c$ défini au paragraphe précédent. Pour tout x de $\mathbb{R}^n$ notons $g^L(x) = [g(x)]^L$ et $g^U(x) = [g(x)]^U$. L'ordre $\preceq$ n'étant pas total, l'espace objectif $\mathcal{Y} = \{g(x) : x \in \mathcal{X}\}$ n'admet pas nécessairement de plus petit élement ou même de borne inférieure dans $\mathcal{I}_c$. En revanche, sous certaines hypothèses sur $\mathcal{X}$, $\mathcal{Y}$ admet un ou plusieurs éléments minimaux. Rappelons qu'un élément A de $\mathcal{Y}$ est minimal relativement à $\preceq$ si et seulement si tout élément B de $\mathcal{Y}$ tel que $B \preceq A$ lui est égal. La contraposée de cette définition stipule qu'il n'existe pas d'intervalle B dans $\mathcal{Y}$ distinct de A tel que $B \preceq A$.
Explicitons la paramétrisation de $\mathcal{Y}$ par $\mathcal{X}$ dans la condition de minimalité. Soit x^* un point de $\mathcal{X}$. L'intervalle $g(x^*)$ est minimal dans $\mathcal{Y}$ relativement à $\preceq$ si et seulement si il n'existe pas de point x dans $\mathcal{X}$ tel que $g^L(x) \leq g^L(x^*)$ et $g^U(x) \leq g^U(x^*)$ soient vérifiées simultanément avec au moins une inégalité stricte. Ceci est exactement la définition de la Pareto-optimalité [32] de x^* pour le problème de minimisation bicritère de (g^L, g^U) sur l'ensemble $\mathcal{X}$.

Pareto-optimalité Considérons une fonction F définie sur $\mathbb{R}^n$ et à valeurs dans $\mathbb{R}^2$ — la généralisation pour des dimensions de l'espace d'arrivée supérieures à 2 est limpide. Notons F_1 et F_2 les composantes de F, de sorte que $F(x) = (F_1(x), F_2(x))$ pour tout x dans $\mathbb{R}^n$. En général il n'existe pas de point x dans $\mathcal{X}$ qui minimise à la fois F_1 et F_2. La notion de Pareto-optimalité définit un compromis entre la minimisation de chacun des objectifs. Un point x^* de $\mathcal{X}$ est dit *Pareto-optimal*, ou *Pareto-efficace*, si et seulement si il n'existe pas d'autre point x de $\mathcal{X}$ tel que $F_1(x) \leq F_1(x^*)$ et $F_2(x) \leq F_2(x^*)$ soient vérifiées avec au moins une inégalité stricte. Dans ce cas le point $F(x^*)$ de $\mathbb{R}^2$ est dit *non-dominé*. Nous noterons $\mathrm{Eff}_{F_1,F_2}(\mathcal{X})$ (pour *efficient*) l'ensemble des points Pareto-optimaux pour un ensemble réalisable donné $\mathcal{X}$. L'image de l'ensemble efficace $\mathrm{Eff}_{F_1,F_2}(\mathcal{X})$ par F est appelé ensemble *non-dominé* ou *front de Pareto*. Lorsque l'ensemble des solutions réalisables $\mathcal{X}$ est convexe et que les critères F_1 et F_2 sont convexes sur $\mathcal{X}$ un fort lien entre optimisation multicritère et optimisation monocritère existe [32]. Si une solution x minimise $(1-\lambda)F_1 + \lambda F_2$ sur $\mathcal{X}$, avec λ dans $(0,1)$, alors x est une solution Pareto-optimale. Lorsque le problème est linéaire — $\mathcal{X}$ est un polyèdre et F_1 et F_2 sont affines — il y a même équivalence. À l'inverse, si x est une solution Pareto-optimale alors x minimise $(1-\lambda)F_1 + \lambda F_2$ sur $\mathcal{X}$ pour une valeur de λ dans $[0,1]$ — observons qu'ici les valeurs 0 et 1 sont permises pour λ.

Exemple bicritère Étudions l'exemple suivant à titre illustratif. Supposons $\mathcal{X} = \mathbb{R}^2$ et définissons $F_1(x)$ et $F_2(x)$ pour tout $x = (x_1, x_2)$ dans $\mathbb{R}^2$ de la manière suivante :

$$\mathrm{F}_1(x) = \frac{x_1^2}{2} - x_2 \ \text{ et } \ \mathrm{F}_2(x) = \frac{x_2^2}{2} - x_1.$$

L'ensemble réalisable et les objectifs étant convexes il est possible de résoudre la minimisation bicritère de (F_1, F_2) sur $\mathcal{X}$ par minimisation monocritère. Notons $f_\lambda = (1-\lambda)\mathrm{F}_1 + \lambda \mathrm{F}_2$ pour tout λ dans $[0,1]$. Un simple calcul différentiel permet de vérifier que, étant donnés λ dans $[0,1]$ et x dans $\mathcal{X}$, le gradient de f_λ s'annule seulement pour λ dans $(0,1)$ en $x_\lambda = (\lambda/(1-\lambda), (1-\lambda)/\lambda)$. Il s'ensuit des remarques du paragraphe précédent que l'ensemble efficace est donné par

$$\mathrm{Eff}_{F_1,F_2}(\mathcal{X}) = \left\{ \left(\frac{\lambda}{1-\lambda}, \frac{1-\lambda}{\lambda} \right) : \lambda \in (0,1) \right\}.$$

L'ensemble efficace est partiellement représenté en trait plein dans la Figure 1.3.

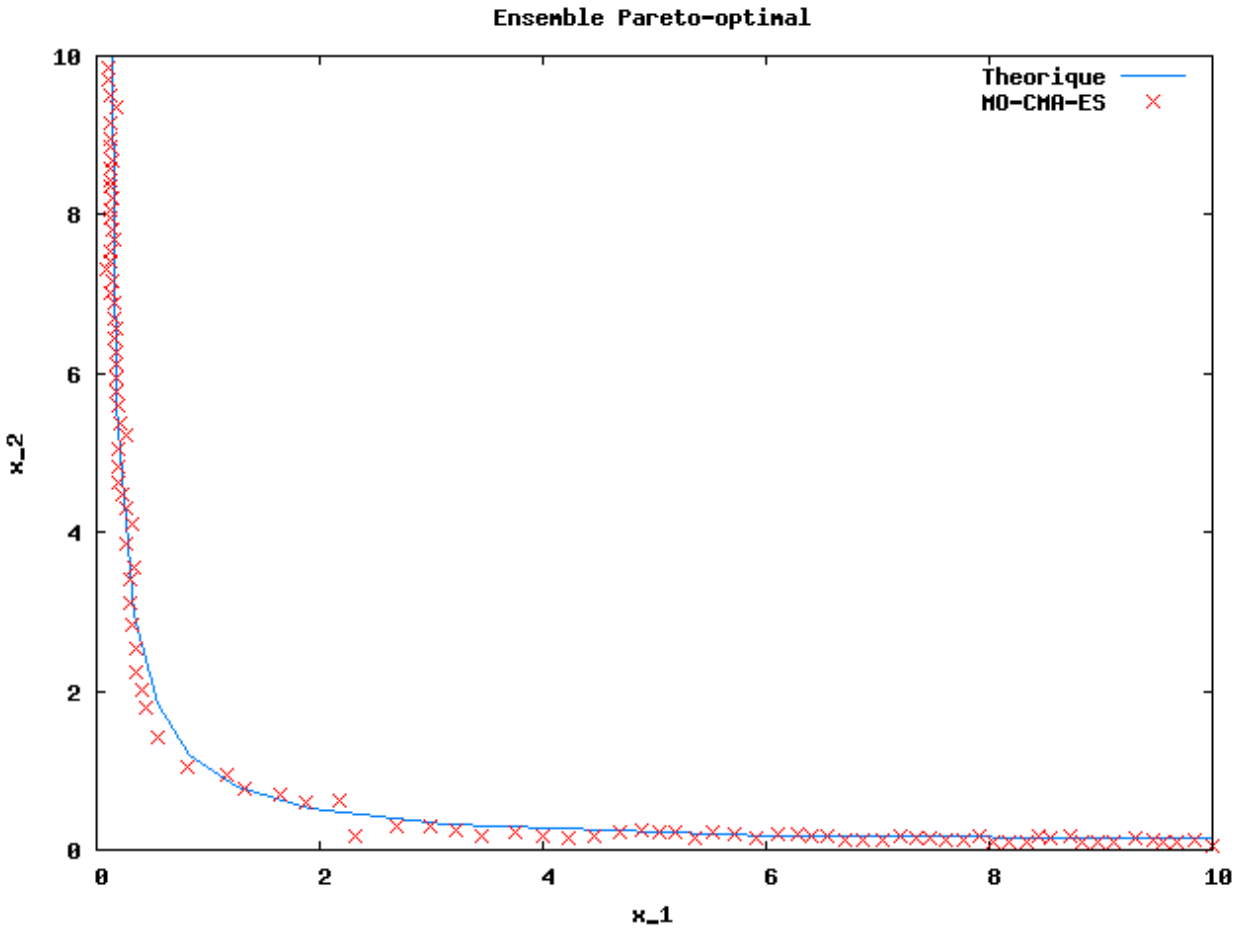

Figure 1.3 – Ensemble efficace pour l'exemple bicritère quadratique — théorique (trait plein) et obtenu numériquement avec l'algorithme MO-CMA-ES (croix).

Pour chaque λ dans $(0,1)$ le calcul de l'image de x_λ donne $F_1(x_\lambda) = [\lambda^3 - 2(1-\lambda)^3]/[2\lambda(1-\lambda)^2]$ et $F_2(x_\lambda) = \mathrm{F}(x_{1-\lambda})$ par symétrie. Ainsi l'ensemble non-dominé est

$$\left\{ \left(\frac{\lambda^3 - 2(1-\lambda)^3}{2\lambda(1-\lambda)^2}, \frac{(1-\lambda)^3 - 2\lambda^3}{2(1-\lambda)\lambda^2} \right) : \lambda \in (0,1) \right\}.$$

L'ensemble non-dominé est en partie représenté en trait plein dans la Figure 1.4.

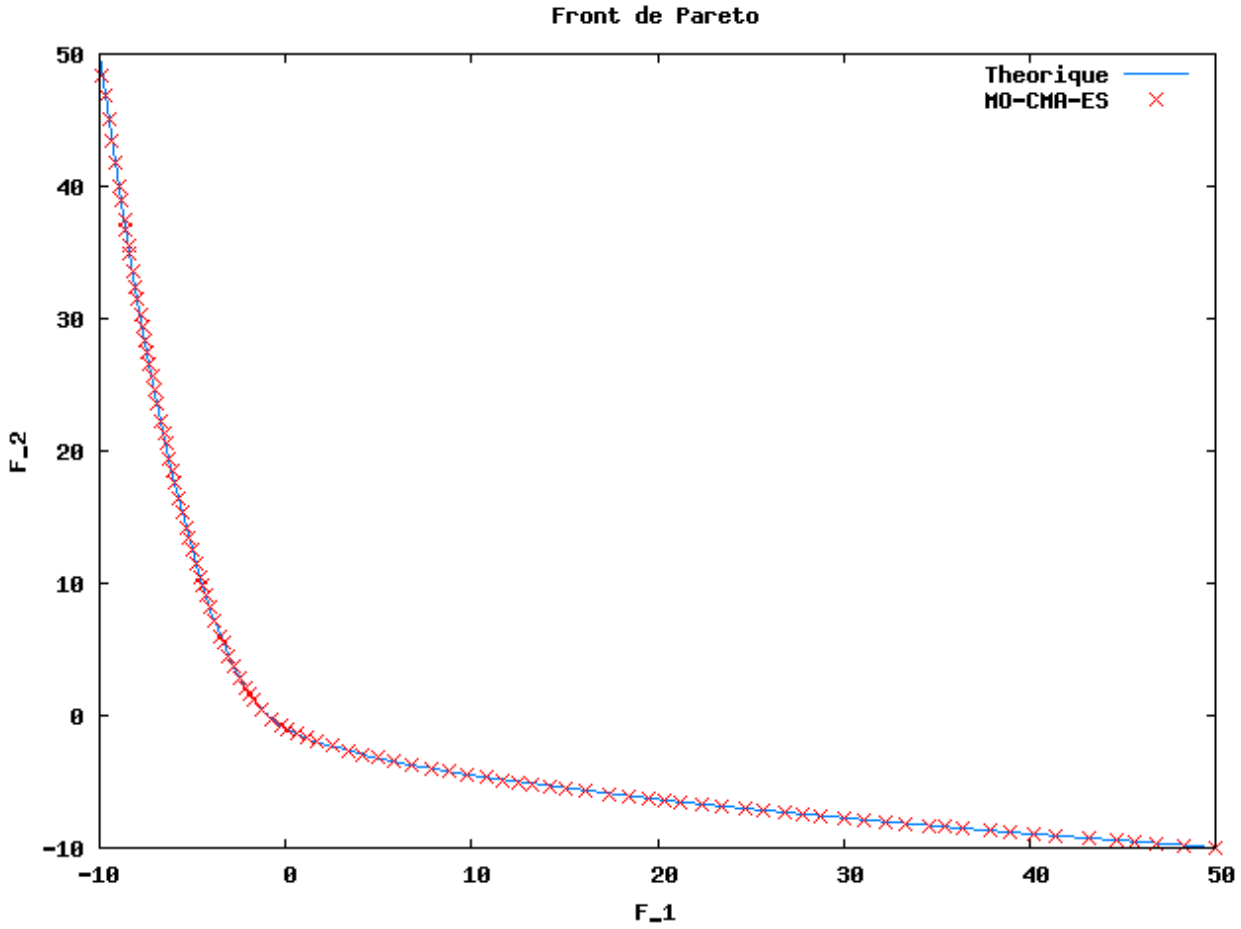

Figure 1.4 – Ensemble non-dominé pour l'exemple bicritère quadratique — théorique (trait plein) et obtenu numériquement avec l'algorithme MO-CMA-ES (croix).

Algorithmes bicritères Nous avons maintenant ramené le problème de la minimisation d'intervalle à un problème de minimisation bicritère. Divers algorithmes ont été introduits pour résoudre ce type de problème, les plus connus étant peut être MO-CMA-ES [49] (*Multi-objective – covariance matrix adaptation – evolution strategy* et NSGAII [28] (*Nondominated sorting genetic algorithm-II*). Nous avons mis en œuvre l'algorithme MO-CMA-ES pour résoudre numériquement l'exemple précédent. Les résultats sont représentés sur les Figures 1.3 et 1.4.

1.2.3 Probabilités et optimisation stochastique

Optimisation stochastique L'optimisation dite *stochastique* regroupe aussi bien les problèmes d'optimisation dans la définition desquels apparaissent des paramètres aléatoires ou des contraintes de probabilité que les méthodes de résolution de problèmes d'optimisation

— éventuellement définis de façon parfaitement déterministe — faisant intervenir des tirages aléatoires, comme les algorithmes génétiques ou évolutionnaires. Dans cette section nous nous préoccupons de la première catégorie de méthodes.
Supposons que le problème (P') soit paramétré par un vecteur aléatoire U de $\mathbb{R}^m$. Les valeurs $f(x;U)$ de la fonction objectif sont alors des variables aléatoire réelles, pour tout x de $\mathbb{R}^n$. Si les lois de ces variables admettent un moment d'ordre 1 une approche classique [8] consiste à substituer l'espérance $\mathrm{E}[f(x;U)]$ aux valeurs de la fonction objectif pour chaque x dans $\mathbb{R}^n$. Le problème (P') devient alors :

$$\begin{aligned} \text{minimiser} \quad & \mathrm{E}[f(x;U)] \\ \text{s. c.} \quad & x \in \mathcal{X}. \end{aligned}$$

Par ce procédé l'espace objectif — l'espace des variables aléatoires réelles — est ramené à l'ensemble ordonné $\mathbb{R}$ et la minimisation a un sens. Si les variables aléatoires $f(x;U)$ admettent un moment d'ordre 2 nous pouvons également remplacer le problème (P') par le programme de minimisation bicritère suivant — auquel nous avons donné un sens à la Section 1.2.2 — :

$$\begin{aligned} \text{minimiser} \quad & (\mathrm{E}[f(x;U)], \sqrt{\mathrm{Var}[f(x;U)]}) \\ \text{s. c.} \quad & x \in \mathcal{X}. \end{aligned}$$

Ce problème consiste à chercher un compromis Pareto-optimal entre une valeur moyenne faible de la fonction objective et une bonne précision, au sens d'une faible dispersion de sa distribution de probabilité. Pour éviter d'avoir à résoudre ce problème bicritère complexe nous pouvons choisir de minimiser une moyenne pondérée de l'espérance et de l'écart-type : pour une valeur positive λ fixée on résout

$$\begin{aligned} \text{minimiser} \quad & \mathrm{E}[f(x;U)] + \lambda\sqrt{\mathrm{Var}[f(x;U)]} \\ \text{s. c.} \quad & x \in \mathcal{X}. \end{aligned}$$

Un quantile pourrait également être choisi comme critère probabiliste.

EGO en présence de paramètres aléatoires L'algorithme EGO présenté à la Section 1.1.2 a été adapté aux fonctions objectif présentant des paramètres aléatoires dans leur définition dans [55]. Nous considérons ici une fonction f définie sur $\mathbb{R}^n \times \mathbb{R}^m$ et à valeurs dans $\mathbb{R}$ et un vecteur aléatoire U de $\mathbb{R}^m$. Supposons que les valeurs (aléatoires) du critère d'intérêt sont données par

$$f(x;U), \qquad x \in \mathbb{R}^n.$$

Dans le même esprit que l'algorithme original [56], nous commençons par construire un estimateur par krigeage de la fonction f relativement au point x et au paramètre u à partir d'un plan d'expérience initial $x^1, \ldots, x^N$, de tirages des paramètres $u^1, \ldots, u^N$ selon la loi de U, et des valeurs de f associées :

$$y_i = f(x^i; u^i), \qquad i = 1, \ldots, N.$$

Notons $\widehat{f(x;u)}$ l'estimateur associée au couple (x,u) de $\mathbb{R}^n \times \mathbb{R}^m$. De même que dans le cas sans paramètres l'estimateur (1.16) suivait une loi normale d'espérance (1.17) et de variance (1.18), l'estimateur défini ici suit une loi normale d'espérance et de variance données par :

$$\begin{aligned} \mathrm{E}[\widehat{f(x;u)}] &= \mu + r(x,u)^\top R^{-1}\,(y - \mu e)\,, \\ \mathrm{Var}[\widehat{f(x;u)}] &= \sigma^2[1 - r(x,u)^\top R^{-1} r(x,u)], \end{aligned} \qquad x \in \mathbb{R}^n,\ u \in \mathbb{R}^m$$

où μ et σ^2 sont l'espérance et la variance du processus (supposés constants), $r(x,u)$ et R sont les vecteurs et matrice des corrélations définis similairement à (1.10), y est le vecteur des évaluations $(y_1,\dots,y_N)$ et e est le vecteur de $\mathbb{R}^N$ de coordonnées toutes égales à 1.
C'est l'*Expected Improvement* (EI) du processus défini ci-dessous — dit *processus projeté* — qui va servir de critère pour sélectionner le point (x^{N+1},u^{N+1}) à ajouter au plan d'expérience :

$$\widehat{f(x)} = \int_{\mathbb{R}^m} \widehat{f(x;u)}\mathrm{d}\,\mathrm{P}_U(u), \qquad x \in \mathbb{R}^n. \tag{1.26}$$

Sous réserve que la condition de Fubini $\mathrm{E}[\int_{\mathbb{R}^m} |\widehat{f(x;u)}|\mathrm{d}\,\mathrm{P}_U(u)] < \infty$ soit vérifiée pour tout x dans $\mathbb{R}^n$, l'égalité suivante est vérifiée

$$\mathrm{E}[\widehat{f(x)}] = \int_{\mathcal{U}} \mathrm{E}[\widehat{f(x;u)}]\mathrm{d}\,\mathrm{P}_U(u), \qquad x \in \mathbb{R}^n.$$

De même si la condition de Fubini $\mathrm{E}[\int_{\mathbb{R}^m}\int_{\mathbb{R}^m} |\widehat{f(x;u)}\widehat{f(x;u')}|\mathrm{d}\,\mathrm{P}_U(u)\mathrm{d}\,\mathrm{P}_U(u')] < \infty$ est également vérifiée pour tout x dans $\mathbb{R}^n$, nous obtenons

$$\mathrm{Var}[\widehat{f(x)}] = \int_{\mathbb{R}^m}\int_{\mathbb{R}^m} \mathrm{Cov}[\widehat{f(x;u)},\widehat{f(x;u')}]\mathrm{d}\,\mathrm{P}_U(u)\mathrm{d}\,\mathrm{P}_U(u'), \qquad x \in \mathbb{R}^n.$$

De même que dans l'Algorithme 2 un nouveau point x^* est choisi parmi les maximiseurs de l'EI — toujours définie par (1.23). Nous savons maintenant que le point x^* est intéressant pour réduire la valeur du critère d'intérêt. Voyons comment définir un couple (x^{N+1},u^{N+1}) à ajouter au plan d'expérience en prenant cet information en compte. Plutôt que de définir $x^{N+1} = x^*$ et de tirer aléatoirement u^{N+1}, les auteurs de [55] suggèrent de choisir ce couple de façon à ce que le nouvel estimateur

$$\widehat{f(x)}^{N+1}, \qquad x \in \mathbb{R}^n,$$

qu'il engendre ait une variance aussi faible que possible au point x^* :

$$(x^{N+1},u^{N+1}) \in \operatorname{Argmin} \mathrm{Var}\,\widehat{f(x^*)}^{N+1}.$$

Autrement dit, nous sélectionnons un point (x^{N+1},u^{N+1}) permettant d'affiner l'estimateur au point x^* qui maximise l'EI actuelle.

Finalement l'Algorithme 2 est modifié de la manière suivante.

Algorithme 3.

1. Construire un plan d'expérience $x^1, \dots, x^N$ initial.
2. Tirer des paramètres $u^1, \dots, u^N$ selon la loi de U.
3. Calculer les valeurs $f(x^1; u^1), \dots, f(x^N; u^N)$ et initialiser $f_{\min}$:
$$f_{\min} \leftarrow \min \left\{ f(x^1; u^1), \dots, f(x^N; u^N) \right\}.$$
4. Tant que le critère d'arrêt n'est pas atteint :
   (a) calculer les hyperparamètres du métamodèle par krigeage,
   (b) trouver un maximiseur x^* de l'EI du processus projeté,
   (c) calculer $\mathrm{E}[\widehat{f(x^*)}]$ et mettre $f_{\min}$ à jour :
   $$f_{\min} \leftarrow \min \left\{ \mathrm{E}[\widehat{f(x^*)}], f_{\min} \right\},$$
   (d) ajouter un couple (x^{N+1}, u^{N+1}) au plan d'expérience qui minimise $\mathrm{Var}\, \widehat{f\,(x^*)}^{N+1}$.

L'Algorithme 3 a déjà été appliqué au problème du placement de puits dans le cadre de deux stages à IFPEN [38,40]. Au Chapitre 3 nous résolvons ce problème par recherche directe directionnelle.

1.2.4 Ensembles flous et théorie des possibilités

Dans cette section nous exposons des notions de base de la théorie des ensembles flous, qui permettent de modéliser l'imprécision. Les ensembles flous peuvent également être vus comme des éléments de base de la théorie des possibilités que nous présentons brièvement.

Ensembles flous Un *ensemble flou* [118] est un ensemble d'objets affectés d'un degré d'appartenance. Dans toute la suite de ce document les degrés d'appartenance seront des éléments de l'intervalle $[0, 1]$. L'appartenance à un ensemble flou est ainsi graduelle, et non absolue. De la même façon qu'une partie classique A d'un ensemble X est caractérisée par sa fonction caractéristique χ_A, donnée par

$$\chi_A : X \to \{0, 1\} : x \mapsto \begin{cases} 1 & \text{si } x \in A \\ 0 & \text{si } x \notin A, \end{cases} \tag{1.27}$$

une partie floue $\tilde{A}$ de X est définie par sa *fonction d'appartenance* $\mu_{\tilde{A}}$, qui associe à chaque élément x de l'espace de référence X son degré d'appartenance $\mu_{\tilde{A}}(x)$:

$$\mu_{\tilde{A}} : X \to [0, 1] : x \mapsto \mu_{\tilde{A}}(x). \tag{1.28}$$

Le signe diacritique "tilde" ($\tilde{}$) est traditionnellement utilisé pour distinguer ensembles flous $\tilde{A}$ et ensembles classiques A, et de même pour tous les objets dérivés de la notion d'ensemble (relations binaires, fonctions, *etc.*) Observons que toute fonction caractéristique (1.27) est

de la forme (1.28), de sorte que la notion d'ensemble flou prolonge celle d'ensemble au sens classique. Ce concept a initialement été introduit pour concevoir des ensembles aux contours mal définis [118], comme celui représenté graphiquement sur la Figure 1.5.

Figure 1.5 – Illustration du concept d'ensemble flou : la coloration est d'autant plus foncée que le degré d'appartenance est élevé.

Le degré d'appartenance s'interprète ainsi comme un niveau de précision.
Les relations d'inclusion et d'égalité entre ensembles s'étendent aux ensembles flous de la façon suivante. Soit $\tilde{A}$ et $\tilde{B}$ des parties floues de X. L'ensemble flou $\tilde{A}$ est dit *inclus* dans $\tilde{B}$ si $\mu_{\tilde{A}} \leq \mu_{\tilde{B}}$. Les parties floues $\tilde{A}$ et $\tilde{B}$ sont dites égales si $\mu_{\tilde{A}} = \mu_{\tilde{B}}$, c'est-à-dire si chacune d'elle est incluse dans l'autre. Il est aisé de vérifier que ces définitions sont bien des extensions de l'inclusion et de l'égalité pour les ensembles classiques en considérant les fonctions caractéristiques. Les opérations ensemblistes telles que l'intersection, l'union ou le passage au complémentaire ainsi que leur propriétés algébriques peuvent également être étendues aux ensembles flous [118] mais nous n'emploierons pas ces notions dans la suite de ce mémoire.

Coupes La *coupe* de niveau α, ou α-*coupe*, d'une partie floue $\tilde{A}$ de X est l'ensemble défini comme suit pour tout α dans $(0, 1]$:

$$[\tilde{A}]_\alpha = \{x \in X : \mu_{\tilde{A}}(x) \geq \alpha\}\,. \tag{1.29}$$

Les coupes de $\tilde{A}$ sont des parties de X au sens classique qui caractérisent l'ensemble flou $\tilde{A}$. Nous reviendrons sur ce point au paragraphe consacré au Théorème de décomposition. La coupe de niveau 1 est parfois appelée *coeur* de $\tilde{A}$ [46]. La notion de α-coupe pourrait être étendue à la valeur 0 selon la formule (1.29) ; cet ensemble serait trivialement égal à l'espace de référence X quelle que soit la partie floue $\tilde{A}$ et n'apporterait pas d'information. Dans un souci de simplification de la suite de l'exposé nous réserverons la notation $[\tilde{A}]_0$ à l'adhérence de l'ensemble des éléments de X dont le degré d'appartenance à $\tilde{A}$ est non nul — X est alors supposé muni d'une topologie :

$$[\tilde{A}]_0 = \overline{\{x \in X : \mu_{\tilde{A}}(x) > 0\}}. \tag{1.30}$$

Nous appellerons *support* de $\tilde{A}$ l'ensemble $[\tilde{A}]_0$. Précisons que le support d'un ensemble flou est parfois défini dans la littérature comme l'ensemble des éléments dont le degré d'appartenance est non nul [69], parfois comme l'adhérence de cet ensemble [91].

Théorème de décomposition Tout ensemble flou $\tilde{A}$ est caractérisé par ses coupes, au sens où elles suffisent à définir sa fonction d'appartenance. Ce résultat, appelé *Théorème de décomposition* [46], prend la forme suivante : étant donnée la famille $\{[\tilde{A}]_\alpha : \alpha \in (0,1]\}$ des coupes d'une partie floue $\tilde{A}$ de X, le degré d'appartenance d'un élément x de X à $\tilde{A}$ vérifie

$$\begin{aligned}\mu_{\tilde{A}}(x) &= \sup\{\alpha \in (0,1] : \alpha \leq \mu_{\tilde{A}}(x)\} \\ &= \sup\{\alpha \in (0,1] : x \in [\tilde{A}]_\alpha\}. \end{aligned} \tag{1.31}$$

Dans l'esprit du Théorème de décomposition, de nombreuses propriétés sur les ensembles flous écrites avec les fonctions d'appartenance se formulent également en termes de coupes — nous ramenant ainsi à des assertions sur des ensembles au sens classique du terme. À titre d'exemple l'inclusion d'un ensemble flou $\tilde{A}$ dans un autre ensemble flou $\tilde{B}$, définie plus haut par l'inégalité $\mu_{\tilde{A}} \leq \mu_{\tilde{B}}$, est équivalente à l'inclusion, pour chaque α dans $(0,1]$, de la coupe $[\tilde{A}]_\alpha$ dans la coupe $[\tilde{B}]_\alpha$. De la même façon l'égalité entre deux ensembles flous est équivalente à l'égalité des coupes niveau par niveau :

$$\tilde{A} = \tilde{B} \iff (\forall \alpha \in (0,1]) \;\; [\tilde{A}]_\alpha = [\tilde{B}]_\alpha \tag{1.32}$$

Principe d'extension Les applications définies entre ensembles classiques peuvent être étendues aux ensembles flous selon une formule appelée *Principe d'extension* [7, 119–121]. Soit X et Y deux ensembles non flous et g une application définie sur X à valeurs dans Y. L'image de toute partie floue $\tilde{A}$ de X par g est définie comme le sous-ensemble flou de Y, noté $g(\tilde{A})$, donné par la fonction d'appartenance suivante :

$$\mu_{g(\tilde{A})}(y) = \sup\{\mu_{\tilde{A}}(x) : x \in X, g(x) = y\}, \quad y \in Y. \tag{1.33}$$

Observons que si A est une partie classique X, qui est aussi la partie floue de fonction d'appartenance χ_A, alors le Principe d'extension (1.33) définit son image par g comme l'ensemble flou donné par $\chi_{g(A)}$, c'est-à-dire l'ensemble classique $g(A)$. Ce principe permet effectivement d'étendre les applications définies entre ensembles classiques aux ensembles flous et porte donc bien son nom.
Dans l'esprit du Théorème de décomposition (1.31) nous pouvons nous demander si le Principe d'extension, exprimé en termes de fonction d'appartenance, peut s'écrire avec les coupes des parties floues en question. Pour répondre à cette question nous allons nous restreindre aux parties floues de l'espace eucliden $\mathbb{R}^n$, qui sera l'espace dans lequel nous nous placerons pour l'étude de l'optimisation floue au Chapitre 4.

Parties floues de $\mathbb{R}^n$ Dans la suite nous prendrons toujours $\mathbb{R}^n$ — en particulier $\mathbb{R}$ — comme espace de référence. Considérons une partie floue $\tilde{A}$ de $\mathbb{R}^n$. Les coupes $[\tilde{A}]_\alpha$, pour α dans $(0,1]$, sont alors des parties classiques de $\mathbb{R}^n$ qui suffisent à caractériser la fonction d'appartenance $\mu_{\tilde{A}}$, comme nous l'avons vu plus haut (1.31). Dans cet esprit, diverses propriétés sur les parties floues de $\mathbb{R}^n$ peuvent s'exprimer tant en termes de fonction d'appartenance qu'en termes de coupes ; nous choisirons l'approche la plus commode suivant les situations. La partie floue $\tilde{A}$ de $\mathbb{R}^n$ sera dite *normale* [69] ou *normalisée* [31] si toutes ses coupes sont non-vides ou, de manière équivalente, si sa fonction d'appartenance atteint la valeur $\max \mu_{\tilde{A}}(\mathbb{R}^n) = 1$ sur

l'espace $\mathbb{R}^n$. L'ensemble flou $\tilde{A}$ est dit *convexe* [118] si ses coupes sont convexes ou, de manière équivalente, si $\mu_{\tilde{A}}$ est quasi-concave sur $\mathbb{R}^n$. Un ensemble flou $\tilde{A}$ est dit *fermé* si ses coupes sont fermées ou, ce qui est équivalent, si sa fonction d'appartenance $\mu_{\tilde{A}}$ est semi-continue supérieurement sur $\mathbb{R}^n$. Il nous sera utile pour la suite de rappeler la caractérisation suivante : $\mu_{\tilde{A}}$ est semi-continue supérieurement sur $\mathbb{R}^n$ si et seulement si, pour tout x de $\mathbb{R}^n$ et toute suite $\{x^k\}_{k\in\mathbb{N}}$ de $\mathbb{R}^n$ convergente vers x, la suite $\{\mu_{\tilde{A}}(x^k)\}_{k\in\mathbb{N}}$ de $\mathbb{R}$ vérifie

$$\limsup_{k\to\infty} \mu_{\tilde{A}}(x^k) \leq \mu_{\tilde{A}}(x). \tag{1.34}$$

Une partie floue $\tilde{A}$ de $\mathbb{R}^n$ est dite *bornée* [118] si ses coupes sont bornées ou, en termes de fonction d'appartenance, si $\mu_{\tilde{A}}(x)$ tend vers 0 quand $\|x\|$ tend vers l'infini. Enfin nous dirons que $\tilde{A}$ est *compact* si $\tilde{A}$ est normale, convexe, fermée, bornée et que son support est borné.

Démontrons à présent un résultat traduisant le Principe d'extension (1.33) en termes de coupes. Nous nous restreignons aux applications g de $\mathbb{R}^n$ dans $\mathbb{R}$ car ce sont celles qui nous intéresseront au Chapitre 4 dans le cadre de l'optimisation floue. Nous n'avons pas rencontré ce résultat sous la forme générale ci-dessous dans littérature de l'optimisation floue, bien qu'il soit très souvent utilisé dans le cas d'applications linéaires. Sans doute est-il déjà démontré dans la littérature plus vaste consacrée aux ensembles flous.

Lemme 2. *Soit g une application définie sur $\mathbb{R}^n$ à valeurs dans $\mathbb{R}$ et $\tilde{A}$ une partie floue de $\mathbb{R}^n$. Si g est continue et $\tilde{A}$ est compact alors $g(\tilde{A})$ est une partie floue compacte de $\mathbb{R}$, et ses coupes sont données par*

$$[g(\tilde{A})]_\alpha = g([\tilde{A}]_\alpha), \quad \alpha \in [0,1]. \tag{1.35}$$

Démonstration. Soit α dans $[0,1]$. Étudions dans un premier temps l'inclusion $[g(\tilde{A})]_\alpha \subset g([\tilde{A}]_\alpha)$. Soit t dans $[g(\tilde{A})]_\alpha$.
Si α est nul alors la définition du support (1.30) induit l'existence d'une suite $\{t^k\}_{k\in\mathbb{N}}$ de $\mathbb{R}$ convergente vers t et telle que $\mu_{g(\tilde{A})}(t^k) > 0$ pour tout k dans $\mathbb{N}$. Pour chaque k dans $\mathbb{N}$, le Principe d'extension stipule que

$$\mu_{g(\tilde{A})}(t^k) = \sup\{\mu_{\tilde{A}}(x) : x \in \mathbb{R}^n,\ g(x) = t^k\}. \tag{1.36}$$

En conséquence, pour tout k dans $\mathbb{N}$, il existe un vecteur x^k de $\mathbb{R}^n$ vérifiant $g(x^k) = t^k$ et $\mu_{\tilde{A}}(x^k) > 0$. Ainsi $\{x^k\}_{k\in\mathbb{N}}$ est une suite du compact $[\tilde{A}]_0$ et admet donc une sous-suite convergente vers un élément x de $[\tilde{A}]_0$. La continuité de g assure l'égalité $g(x) = t$ par passage à la limite ; le réel t appartient donc à l'ensemble $g([\tilde{A}]_0)$.
Supposons à présent que α est strictement positif. L'inégalité $\mu_{g(\tilde{A})}(t) \geq \alpha$ assure conjointement avec le Principe d'extension (1.36) que, pour tout k dans $\mathbb{N}\setminus\{0\}$, il existe un vecteur x^k de $\mathbb{R}^n$ tel que $g(x^k) = t$ et $\mu_{\tilde{A}}(x^k) \geq (1-2^{-k})\alpha$. Puisque $\alpha > 0$ la suite $\{x^k\}_{k\in\mathbb{N}\setminus\{0\}}$ est incluse dans le compact $[\tilde{A}]_0$ et admet donc une sous-suite convergente vers un élément x de $[\tilde{A}]_0$. Comme $\tilde{A}$ est une partie floue fermée de $\mathbb{R}^n$, sa fonction d'appartenance $\mu_{\tilde{A}}$ est semi-continue supérieurement. D'où

$$\mu_{\tilde{A}}(x) \geq \limsup_{k\to\infty} \mu_{\tilde{A}}(x^k) \geq \limsup_{k\to\infty} (1-2^{-k})\alpha = \alpha.$$

Conséquemment le vecteur x appartient à $[\tilde{A}]_\alpha$. Par ailleurs la continuité de g assure l'égalité $g(x) = t$ par passage à la limite — sur la sous-suite de $\{x^k\}_{k\in\mathbb{N}\setminus\{0\}}$. Le réel t est donc élément de $g([\tilde{A}]_\alpha)$, ce qui achève de démontrer l'inclusion directe.
Intéressons-nous à présent à l'inclusion réciproque, d'abord pour α strictement positif. Soit x un vecteur de $[\tilde{A}]_\alpha$ et $t = g(x)$ son image par g. Il s'ensuit de (1.36) que $\mu_{g(\tilde{A})}(t) \geq \mu_{\tilde{A}}(x) \geq \alpha$, le réel t appartient donc à $[g(\tilde{A})]_\alpha$. Observons qu'à ce stade de la démonstration l'égalité (1.35) est prouvée pour tout α strictement positif.
Nous traitons le dernier cas grâce à l'enchaînement suivant — où l'inclusion résulte de la continuité de g [23] :

$$g([\tilde{A}]_0) = g\left(\overline{\cup_{\alpha\in(0,1]}[\tilde{A}]_\alpha}\right) \subset \overline{g\left(\cup_{\alpha\in(0,1]}[\tilde{A}]_\alpha\right)} = \overline{\cup_{\alpha\in(0,1]} g([\tilde{A}]_\alpha)} = \overline{\cup_{\alpha\in(0,1]}[g(\tilde{A})]_\alpha} = [g(\tilde{A})]_0.$$

Nous pouvons maintenant conclure en remarquant que l'Equation (1.35) assure que, pour tout α dans $[0,1]$, la coupe de niveau α de $g(\tilde{A})$ est l'image du compact convexe (donc connexe) non vide $[\tilde{A}]_\alpha$ par la fonction continue g. De ce fait $[g(\tilde{A})]_\alpha$ est une partie compacte connexe non vide de $\mathbb{R}$ — c'est-à-dire un intervalle fermé borné non vide — pour tout α dans $[0,1]$. Par conséquent $g(\tilde{A})$ est une partie floue compacte de $\mathbb{R}$. □

Nombres et vecteurs flous Le Principe d'extension permet d'étendre les opérations algébriques sur les nombres réels, telles que l'addition et le produit, aux parties floues de $\mathbb{R}$. Donnons-nous deux parties floues $\tilde{A}$ et $\tilde{B}$ de $\mathbb{R}$. Admettons que l'on définisse le produit cartésien flou $\tilde{A}\times\tilde{B}$ par $\mu_{\tilde{A}\times\tilde{B}}(t,t') = \min\{\mu_{\tilde{A}}(t), \mu_{\tilde{B}}(t')\}$ pour tout (t,t') dans $\mathbb{R}^2$. La somme de deux parties floues $\tilde{A}$ et $\tilde{B}$ de $\mathbb{R}$ est le sous-ensemble flou $\tilde{A}+\tilde{B}$ de $\mathbb{R}$ donné par

$$\begin{aligned}\mu_{\tilde{A}+\tilde{B}}(s) &= \sup\{\mu_{\tilde{A}\times\tilde{B}}(t,t') : (t,t')\in\mathbb{R}^2,\, t+t' = s\}\\ &= \sup\{\min\{\mu_{\tilde{A}}(t), \mu_{\tilde{B}}(t')\} : (t,t')\in\mathbb{R}^2,\, t+t' = s\}\end{aligned}$$

pour tout s dans $\mathbb{R}$. Le produit et la soustraction des nombres réels s'étendent de même aux parties floues de $\mathbb{R}$ — cependant la soustraction n'est alors plus une opération inverse de l'addition.
Dans le contexte de l'optimisation nous nous intéresserons plus particulièrement à certaines parties floues de $\mathbb{R}$ appelées "nombres flous". Les définitions rencontrées dans la littérature pour la notion de nombre flou varient légèrement d'un auteur à l'autre ; nous adopterons la plus générale. Nous appelons *nombre flou* toute partie floue de $\mathbb{R}$ normale et convexe [69]. Comme nous venons de le vérifier les opérations sur les nombres réels s'étendent aux nombres flous grâce au Principe d'extension.
Nous dirons qu'un nombre flou $\tilde{a}$ est *compact* si $\tilde{a}$ est une partie floue compacte de $\mathbb{R}$ — certains auteurs incluent cette propriété dans la définition de nombre flou. Les coupes d'un nombre flou compact $\tilde{a}$ sont des intervalles compacts de $\mathbb{R}$. Notons $\tilde{a}_\alpha^L$ et $\tilde{a}_\alpha^U$ — pour *lower* et *upper* — les bornes inférieure et supérieure respectivement de $[\tilde{a}]_\alpha$, pour tout α dans $[0,1]$:

$$\tilde{a}_\alpha^L = \min[\tilde{a}]_\alpha \text{ et } \tilde{a}_\alpha^U = \max[\tilde{a}]_\alpha.$$

Ainsi les coupes d'un nombre flou compact $\tilde{a}$ sont données par

$$[\tilde{a}]_\alpha = \left[\tilde{a}_\alpha^L, \tilde{a}_\alpha^U\right], \quad \alpha\in[0,1].$$

Appliquons à présent le Lemme 2 à l'addition des nombres flous compacts. Soit $\tilde{u}_1$ et $\tilde{u}_2$ deux nombres flous compacts. Définissons un *vecteur flou* $\tilde{u} = (\tilde{u}_1, \tilde{u}_2)$ de la façon suivante :

$$\mu_{\tilde{u}}(u) = \min\{\mu_{\tilde{u}_1}(u_1),\, \mu_{\tilde{u}_2}(u_2)\}, \tag{1.37}$$

pour tout vecteur $u = (u_1, u_2)$ de $\mathbb{R}^2$. D'après la définition (1.37) les coupes de $\tilde{u}$ sont données par

$$[\tilde{u}]_\alpha = [\tilde{u}_1]_\alpha \times [\tilde{u}_2]_\alpha, \quad \alpha \in (0, 1]. \tag{1.38}$$

Au Chapitre 4 il nous arrivera régulièrement de considérer des vecteurs flous de dimension n éventuellement supérieure à 3. Leurs fonctions d'appartenances et leurs coupes seront données par des généralisations immédiates des formules (1.37) et (1.38). Puisque les coupes de $\tilde{u}_1$ et $\tilde{u}_2$ sont non vides convexes fermées et bornées, il en est de même pour les coupes (1.38) de $\tilde{u}$. L'ensemble $\mathbb{R}$ étant un groupe topologique pour l'addition nous pouvons appliquer le Lemme 2 à l'addition de $\tilde{u}_1$ et $\tilde{u}_2$. Notons g l'opération d'addition de $\mathbb{R}^2$ dans $\mathbb{R}$. Alors

$$\begin{aligned} [\tilde{u}_1 + \tilde{u}_2]_\alpha &= [g(\tilde{u}_1, \tilde{u}_2)]_\alpha = g([\tilde{u}]_\alpha) = g([\tilde{u}_1]_\alpha \times [\tilde{u}_2]_\alpha) \\ &= \{t_1 + t_2 : t_1 \in [\tilde{u}_1]_\alpha,\, t_2 \in [\tilde{u}_2]_\alpha\} \qquad (\forall \alpha \in (0, 1]). \\ &= [\tilde{u}_1]_\alpha + [\tilde{u}_2]_\alpha. \end{aligned} \tag{1.39}$$

Ce résultat est à mettre en parallèle avec l'addition des intervalles compacts présentée à la Section 1.2.2. Effectivement nous remarquons que, pour tout α dans $(0, 1]$, la coupe de niveau α de la somme de deux nombres flous compacts $\tilde{u}_1$ et $\tilde{u}_2$ est un intervalle compact de $\mathbb{R}$ égal à la somme des coupes de niveau α de $\tilde{u}_1$ et $\tilde{u}_2$ respectivement, qui sont également des intervalles compacts de $\mathbb{R}$.

Nombres flous triangulaires Un façon classique de définir des nombres flous est de se donner deux fonctions L et R — comme *left* et *right* — de $\mathbb{R}$ dans $\mathbb{R}$, paires, décroissantes sur $[0, \infty)$ et valant 1 en 0. De telles fonctions sont appelées *fonctions de référence* [31]. Un nombre flou $\tilde{u}$ peut alors être paramétré par un réel m et deux réels strictement positifs a et b, appelés respectivement *mode*, *étendue gauche* et *étendue droite*, de la façon suivante :

$$\mu_{\tilde{u}}(t) = \begin{cases} L\left(\dfrac{t-m}{a}\right) & \text{si } t \leq m \\ R\left(\dfrac{m-t}{b}\right) & \text{si } t \geq m, \end{cases} \qquad t \in \mathbb{R}.$$

On parle de *représentation L-R* d'un nombre flou, ou de *nombre flou de type L-R*. Des exemples typiques de fonctions de références sont

1. $L = \chi_{\{0\}}$ qui permet de retrouver le nombre classique m (égal au mode),
2. $L : t \mapsto \max\{0, 1 - |t|\}$ qui permet de définir les nombres flous dits *triangulaires*,
3. ou encore $L = t \mapsto \exp(t^2/2)$ qui permet de paramétrer les nombres dits *gaussiens*.

Les nombres flous triangulaires sont très souvent utilisés dans les applications numériques pour leur simplicité. Nous noterons $\langle m; a, b\rangle$ le nombre flou triangulaire de mode m et d'étendues

a et b. La fonction d'appartenance de $\langle m; a, b\rangle$ est donnée par : pour tout t dans $\mathbb{R}$,

$$\mu_{\langle m;a,b\rangle}(t) = \begin{cases} 1 - \dfrac{m-t}{a} & \text{si } m-a \leq t \leq m \\ 1 - \dfrac{t-m}{b} & \text{si } m \leq t \leq m+b \\ 0 & \text{sinon.} \end{cases}$$

La représentation graphique de $\mu_{\langle m;a,b\rangle}$ évoque un triangle, comme l'illustre la Figure 1.6.

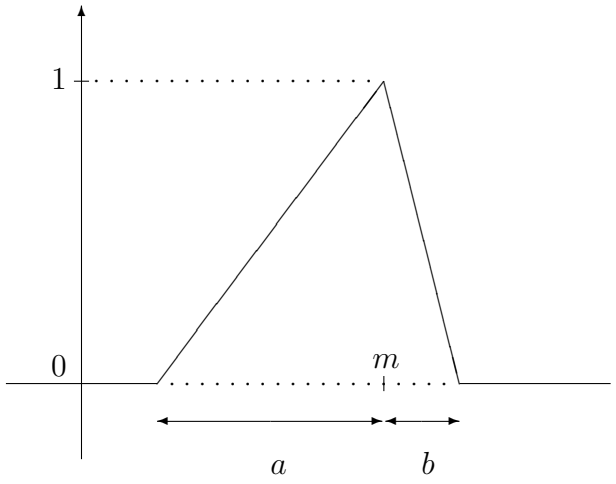

Figure 1.6 – Nombre flou triangulaire.

Il est aisé de vérifier que les coupes d'un nombre flou triangulaire $\langle m; a, b\rangle$ sont des intervalles compacts de $\mathbb{R}$ dont les bornes sont données ci-dessous :

$$\begin{aligned} \langle m; a, b\rangle_\alpha^L &= m - (1-\alpha)a, \\ \langle m; a, b\rangle_\alpha^U &= m + (1-\alpha)b, \end{aligned} \quad \alpha \in [0,1]. \tag{1.40}$$

Ainsi les nombres flous triangulaires sont des cas particuliers de nombres flous compacts. Grâce au Principe d'extension, au Théorème de décomposition et à l'Equation (1.40) nous pouvons facilement étudier les propriétés arithmétiques des nombres flous triangulaires. Donnons-nous deux réel m_1 et m_2 et quatre nombres strictement positifs a_1, b_1, a_2 et b_2. Alors (1.39) donne, pour tout α dans $(0,1]$,

$$\begin{aligned} [\langle m_1; a_1, b_1\rangle + \langle m_2; a_2, b_2\rangle]_\alpha &= [\langle m_1; a_1, b_1\rangle]_\alpha + [\langle m_2; a_2, b_2\rangle]_\alpha \\ &= [m_1 - (1-\alpha)a_1,\, m_1 + (1-\alpha)b_1] \\ &\quad + [m_2 - (1-\alpha)a_2, m_2 + (1-\alpha)b_2] \\ &= [(m_1+m_2) - (1-\alpha)(a_1+a_2),\, (m_1+m_2) + (1-\alpha)(b_1+b_2)] \\ &= [\langle m_1+m_2; a_1+a_2, b_1+b_2\rangle]_\alpha. \end{aligned}$$

Nous savons d'après (1.32) — ou tout simplement par le Théorème de décomposition — que deux ensembles flous ayant les mêmes coupes sont égaux. Par conséquent nous obtenons la relation suivante pour l'addition des nombres flous triangulaires :

$$\langle m_1; a_1, b_1\rangle + \langle m_2; a_2, b_2\rangle = \langle m_1 + m_2; a_1 + a_2, b_1 + b_2\rangle\,.$$

Nous pouvons également vérifier la relation suivante sur le produit d'un nombre flou triangulaire $\langle m; a, b\rangle$ par un nombre réel λ avec un raisonnement similaire :

$$\lambda\,\langle m; a, b\rangle = \begin{cases} \langle \lambda m; \lambda a, \lambda b\rangle & \text{si } \lambda > 0, \\ \langle \lambda m; \lambda b, \lambda a\rangle & \text{si } \lambda < 0, \\ 0 & \text{si } \lambda = 0. \end{cases}$$

Optimisation floue L'optimisation floue est une branche de l'optimisation sous incertitude qui concerne les problèmes définis de façon imprécise. Cette imprécision est modélisée à l'aide de la théorie des ensembles flous présentée plus haut. L'optimisation floue a commencé à se développer dans les années 1970 [7] et s'est depuis largment ramifiée [19, 29, 51, 52, 67, 70, 92]. Elle comprend plusieurs champs d'étude : l'optimisation flexible [4, 7, 51, 84, 105–108, 111, 122], où les valeurs de la fonction objectif et la satisfaction des contraintes sont exprimées en des termes vagues et modélisées rigoureusement par des ensembles flous et des relations floues ; l'optimisation robuste floue [50, 53, 101] ; l'optimisation avec paramètres flous [31, 85, 102] et l'optimisation possibiliste (terme préféré lorsqu'une interprétation possibiliste existe) ; l'optimisation stochastique floue, qui prend en compte aussi bien l'imprécision que les incertitudes aléatoires [68, 103, 112] ; l'optimisation multiobjectifs floue [71, 97, 114] ; la programmation dynamique floue [33, 34] ; l'optimisation multiniveaux floue [94, 96].

Nous traiterons l'optimisation de fonctions objectif avec paramètres flous au Chapitre 4. Nous terminons cette section avec quelques éléments de la théorie des possibilité, qui permet de donner une interprétation possibiliste des problèmes comportant des paramètres flous.

Théorie des possibilités La théorie des possibilités était initialement motivée par la modélisation de l'imprécision intrinsèque des langages humains [117]. De la même manière que l'imprécision sur un nombre peut tenir au simple fait qu'il est représenté par un nombre fini de chiffres, l'ambiguïté d'une phrase peut être dûe à la nécessité de l'exprimer avec un nombre limité de mots.
Une *mesure de possibilité* [31, 117] sur un ensemble X est une application — notons-là Poss — de l'ensemble $\mathcal{P}(X)$ des parties de X et à valeurs dans $\mathbb{R}$ vérifiant les deux axiomes suivants :

1. $\mathrm{Poss}(X) = 1$;
2. pour toute famille $\{A_i\}_{i\in I}$ de parties de X,

$$\mathrm{Poss}(\cup_{i\in I} A_i) = \sup_{i\in I} \mathrm{Poss}(A_i).$$

Nous déduisons aisément de l'Axiome 1 que $\mathrm{Poss}(\varnothing) = 0$ — il suffit de considérer une famille I vide — et que l'application Poss est croissante sur l'ensemble $\mathcal{P}(X)$ ordonné par l'inclusion :

pour toutes parties A et B de X telles que $A \subset B$ nous avons $\mathrm{Poss}(A) \leq \mathrm{Poss}(B)$. En combinant ces remarques avec l'Axiome 2 nous déduisons que Poss est à valeurs dans $[0, 1]$. Des exemples simples de mesures de possibilités sur X sont

1. la mesure maximale, qui à toute partie non vide de X associe la valeur 1,
2. ou les mesures de Dirac relatives à un élément x donné de X, qui attribuent la valeur 1 aux parties de X contenant x et 0 aux autres.

Étant donné une partie A de X, le nombre $\mathrm{Poss}(A)$ peut s'interpréter comme la *possibilité* qu'un élément de X soit dans A, la facilité qu'a un élément de X à être dans A. La citation suivante permettra peut-être de se faire une meilleure idée de ce concept :

> 'Intuitively, possibility relates to our perception of the degree of feasability or ease of attainment whereas probability is associated with a degree of likelihood, belief, frequency or proportion.' [116]

La possibilité renvoie ainsi à un niveau de réalisabilité ou de facilité d'atteinte, la probabilité ayant plutôt à voir avec la vraisemblance ou la fréquence.

Possibilité et ensembles flous Les ensembles flous, introduits initialement pour modéliser l'imprécision, sont aussi les éléments de base de la théorie des possibilités. Ils jouent un rôle similaire aux densités de probabilité pour les mesures de probabilité.
Une *fonction de distribution de possibilité* [117] sur X est une application π de X dans $[0, 1]$ telle que $\sup \pi(X) = 1$. Les fonctions d'appartenance des parties floues normales — c'est-à-dire qui atteignent la valeur maximale 1 — sont ainsi des fonctions de distribution de possibilité sur X. L'application définie sur l'ensemble $\mathcal{P}(X)$ qui à toute partie A de X associe le nombre $\sup \pi(A)$ est une mesure de possibilité sur X. Réciproquement, si Poss est une mesure de possibilité sur X alors l'application qui à tout élément x de X associe $\mathrm{Poss}(\{x\})$ est une distribution de possibilité sur X.

Possibilité et probabilité Observons que mesures de possibilités et mesures de probabilité — sur une tribu d'un espace donné — partagent certaines propriétés : valeur nulle sur l'ensemble vide, valeur 1 sur l'espace entier, croissance relativement à l'inclusion. Leurs axiomes respectifs concernant les unions d'événements sont cependant bien distincts. La possibilité de la réunion d'une famille dénombrable d'événements $\{A_k\}_{k\in\mathbb{N}}$ disjoints deux-à-deux n'est *a priori* pas égale à la somme des possibilités des événements :

$$\mathrm{Poss}\left(\bigcup_{k\in\mathbb{N}} A_k\right) = \sup_{k\in\mathbb{N}} \mathrm{Poss}(A_k) \leq \sum_{k\in\mathbb{N}} \mathrm{Poss}(A_k).$$

En particulier, si l'on considère une paire $\{A, \complement A\}$, nous obtenons $\mathrm{Poss}(A) + \mathrm{Poss}(\complement A) \geq 1$. Plus précisément, pour toute partie A de X, la possibilité de A vaut 1 ou la possibilité de son complémentaire vaut 1 :

$$\max\{\mathrm{Poss}(A),\, \mathrm{Poss}(\complement A)\} = \mathrm{Poss}(A \cup \complement A) = \mathrm{Poss}(X) = 1. \tag{1.41}$$

La définition de la *mesure de nécessité* associée à Poss est donnée par

$$\mathrm{Nec}(A) = 1 - \mathrm{Poss}(\complement A),$$

pour toute partie A de X. Un événement est ainsi d'autant plus nécessaire qu'il ne peut être impossible. L'Equation (1.41) peut alors s'interpréter comme suit : des deux ensembles A et $\complement A$, l'un est totalement possible et l'autre n'est pas du tout nécessaire.
Il arrive qu'en pratique on cherche à définir une mesure de possibilité qui soit en accord — en un sens à préciser — avec une mesure de probabilité connue. Le *Principe de cohérence probabilité-possibilité* repose sur l'idée qu'un événement est d'autant plus possible qu'il est probable. Considérons une mesure de probabilité P définie sur une tribu $\mathcal{A}$ de l'ensemble X. Une mesure de possibilité Poss sur X est dite *cohérente* avec la mesure de probabilité P si et seulement si $\mathrm{P}(A) \leq \mathrm{Poss}(A)$ pour tout élément A de $\mathcal{A}$. En particulier un événement A presque sûr ($\mathrm{P}(A) = 1$) est totalement possible ($\mathrm{Poss}(A) = 1$), et un événement impossible ($\mathrm{Poss}(A) = 0$) est négligeable ($\mathrm{P}(A) = 0$). Comme la probabilité du complémentaire d'un événement A est $\mathrm{P}(\complement A) = 1 - \mathrm{P}(A)$, nous vérifions aisément que

$$\mathrm{Nec}(A) \leq \mathrm{P}(A) \leq \mathrm{Poss}(A).$$

Une mesure de possibilité cohérente avec une mesure de probabilité, et sa mesure de nécessité associée, procurent ainsi un majorant et un minorant sur les probabilités des événements.

Chapitre 2

Émulation de codes numériques aléatoires

Divers phénomènes complexes rencontrés dans des problématiques industrielles, tels que la distribution spatiale des molécules d'un système d'hydrocarbures, le cycle de vie de transformateurs électriques ou les répercussions d'un accident rare sur les composants d'une centrale nucléaire, donnent lieu à des codes de simulation intrinsèquement aléatoires. Non seulement les réponses d'un tel code ne sont pas déterministes mais le temps d'exécution requis compromet généralement l'efficacité de toute analyse de sensibilité ou méthode d'optimisation. L'émulation de tels codes se révèle ainsi une question cruciale pour mener à bien des projets industriels variés. Tel était l'objet du projet CODESTOCH réalisé au Centre d'été de mathématiques et recherche avancée en calcul scientifique (CEMRACS) au cours de l'été 2013.

2.1 Le projet CODESTOCH

CEMRACS 2013 Le Centre international de rencontres mathématiques (CIRM) accueillait l'édition 2013 du CEMRACS, du 22 juillet au 31 août 2013 à Marseille. Cet événement international réunissait des étudiants ainsi que des chercheurs académiques et industriels sur le thème "Modélisation et simulation des systèmes complexes : approches stochastiques et déterministes". La première semaine était consacrée à une école d'été portant sur des méthodes déterministes et probabilistes appliquées à des problèmes de propagation d'incertitudes, des techniques de simulation moléculaire et de dynamique des populations. Les cinq semaines suivantes étaient dédiées à une session de recherche intensive : seize groupes de un à trois doctorants travaillaient chacun sur un projet proposé et encadré par des partenaires académiques et industriels.

Métamodèles Dans les études de traitement des incertitudes et d'analyse de sensibilité ou dans les problèmes d'optimisation, lorsque le modèle numérique sous-jacent est coûteux en temps de calcul (une évaluation pouvant requérir plusieurs heures de calculs par exemple), l'approche adoptée consiste à construire une fonction mathématique simple — un métamodèle — à partir de quelques évaluations bien choisies du modèle [35]. Ce métamodèle (polynôme,

krigeage, chaos polynomial, *etc*) est capable d'approximer le code numérique et de prédire (avec un certain niveau de confiance) de nouveaux calculs avec un temps processeur négligeable. Il peut alors être utilisé pour réaliser des études d'incertitudes et de sensibilité ou pour résoudre des problèmes d'optimisation.
Les métamodèles avaient jusqu'alors été principalement développés pour des codes de calcul déterministes, c'est-à-dire pour lesquels deux calculs avec un même jeu de paramètres d'entrée procurent les mêmes réponses. Dans certaines applications, les codes de calcul sont eux-mêmes de nature stochastique, c'est-à-dire qu'ils intègrent un aléa intrinsèque au modèle. De tels codes sont par exemple :

- des codes de calcul basés sur la méthode de Monte Carlo pour calculer l'irradiation de matériaux dans un cœur de réacteur nucléaire en y simulant les trajectoires de neutrons,
- des codes de calcul d'impact dans l'environnement modélisant la diffusion et la dispersion de particules par des équations stochastiques (approche lagrangienne),
- des codes de calcul dont certains paramètres d'entrée, trop complexes pour être pris en compte explicitement, sont jugés incontrôlables. Ces paramètres peuvent être des fonctions temporelles ou des champs aléatoires [73],
- des modèles de files d'attente (illustrés par exemple dans [59] et de manière plus générale les modèles dits *agent-based simulation models*.

Trois porteurs de projet Le Département de management des risques industriels (MRI) d'EDF R&D développe des modèles technico-économiques (outil VME) permettant de calculer, à partir de dates d'événements aléatoires, des options constituant une stratégie de maintenance exceptionnelle (d'une installation de production d'électricité) à valoriser, ainsi que des indicateurs technico-économiques de la stratégie étudiée — valeur actuelle nette (VAN) et coefficients d'indisponibilité, principalement. Les indicateurs technico-économiques étant des variables aléatoires, leurs distributions sont estimées par tirages Monte Carlo via l'outil VME [36]. Pour un jeu de paramètres d'entrée fixé, la réponse d'un calcul VME fournit donc en sortie une densité de probabilité avec un temps processeur typique de plusieurs dizaines de minutes à quelques heures.
Le Département d'étude des réacteurs du CEA Cadarache réalise des études sur la tenue mécanique des cuves des réacteurs, notamment lors d'accident de perte de refroidissement primaire pouvant initier un choc thermique pressurisé et des études de calcul d'impact dans l'environnement avec des modèles hydrogéologiques complexes. Concernant le premier thème évoqué, l'un des points critiques de ces calculs concerne le chaînage des codes de calcul thermohydrauliques très coûteux en temps processeur (*e.g.* CATHARE) et des codes de calcul mécanique (*e.g.* CASTEM). Dans l'évaluation probabiliste de l'intégrité des cuves, des transitoires de référence (profils de pression, température et flux échangés avec la paroi de la cuve) sont donnés par les thermohydrauliciens et pris en entrée des codes de calcul mécanique. En considérant ces transitoires thermohydrauliques comme des entrées incontrôlables du code mécanique, le modèle devient un code stochastique. Les autres entrées du code mécanique, de nature scalaire, peuvent être prises en compte explicitement et sont considérées comme "contrôlables" (par opposition aux entrées temporelles considérées comme incontrôlables [73]. Une première étape consiste alors à estimer la densité de probabilité de la sortie du code

conditionnellement aux entrées contrôlables du code mécanique. Le second thème d'étude est relatif au calcul d'impact dans l'environnement.
Le Département thermodynamique et modélisation moléculaire d'IFPEN développe des méthodes pour caractériser les propriétés thermodynamiques de systèmes d'intérêt pétrolier. L'utilisation des méthodes de modélisation moléculaire requiert la connaissance d'énergies d'interaction potentielles intermoléculaires et intramoléculaires. Des modèles mathématiques permettent de relier ces énergies à des paramètres internes aux molécules (dont les charges et certaines propriétés géométriques). Le problème de l'optimisation des champs de forces est un problème inverse qui consiste à estimer ces paramètres à partir de données macroscopiques expérimentales telles que la densité, la pression ou l'énergie. Le logiciel de modélisation moléculaire GIBBS permet de calculer numériquement ces propriétés macroscopiques. Il utilise une méthode statistique fondée sur la distribution de Boltzmann des énergies à l'équilibre thermodynamique. A partir d'une configuration initiale, la méthode de Monte Carlo génère un ensemble de configurations possibles (entre 10^6 et 10^8) qui permettent d'estimer une densité de probabilité pour chacune des propriétés macroscopiques à caler. Construire des métamodèles de ces densités de probabilité en étape préalable à l'optimisation permettrait d'éliminer des paramètres non influents ou de mieux déterminer leurs bornes de variation. De plus, remplacer, dans l'optimisation, le modèle par le métamodèle permettrait aussi d'obtenir une première valeur sous-optimale de ces paramètres.
De manière générale, dans ce type de modèles stochastiques, on ne cherche plus à récupérer la valeur d'une variable scalaire en sortie d'un calcul mais sa distribution de probabilité. Cette distribution de probabilité dépend des variables d'entrée du modèle, dont certaines sont elles-mêmes incertaines mais contrôlables. Ces modèles numériques sont le plus souvent extrêmement coûteux ce qui motive fortement le développement d'approches de type "métamodèle" afin de déployer des analyses d'incertitudes et de sensibilité ou de résoudre des problèmes d'optimisation.

État de l'art Des travaux récents proposaient de premières pistes de travail pour la construction de métamodèles adaptés aux codes stochastiques. Par exemple, les auteurs de [73] proposent de modéliser la moyenne et la variance du résultat d'un code stochastique, conditionnellement aux variables contrôlables. Pour cela, ils utilisent un modèle joint de type processus gaussien (krigeage) : un premier métamodèle modélise la moyenne conditionnelle et un second la variance. Les auteurs de [59] et [3] proposent de traiter ce problème à l'aide de métamodèles hétéroscédastiques basés sur le modèle du krigeage. Ces travaux, ne s'intéressant qu'aux deux premiers moments statistiques de la variable de sortie, des méthodologies doivent être mises en œuvre pour calculer les quantiles conditionnels (voir [88] dans le cas gaussien et [115] dans le cas non gaussien). Récemment les auteurs de [87] ont proposé une méthode pour émuler la distribution conditionnelle de la sortie d'un code de calcul en se basant sur un modèle de mélange de lois gaussiennes. Une piste prometteuse consistait à construire un métamodèle à sortie fonctionnelle [5] sous la contrainte d'être une densité de probabilité (fonction positive d'intégrale égale à un). L'idée serait d'utiliser une méthode de réduction de dimension (par exemple l'analyse en composantes principales) et de modéliser les coefficients conservés dans la décomposition par des métamodèles. Le calcul des coefficients de la projection conduira à la formulation d'un problème d'optimisation sous contraintes (positivité de la

fonction reconstruite et fonction d'intégrale égale à un).

Projet CODESTOCH Le projet "Développement d'émulateurs pour les codes de simulation stochastiques", ou plus simplement CODESTOCH, était proposé par le Commissariat à l'énergie atomique et aux énergies alternatives (CEA) de Cadarache, Électricité de France (EDF) et IFP Energies Nouvelles (IFPEN). L'objectif était de proposer des métamodèles permettant la reconstruction complète de la distribution de probabilité conditionnelle et d'estimer des critères à partir de cette densité (par exemple une probabilité de défaillance ou de dépassement d'un seuil donné) en intégrant toutes les sources d'incertitudes. Les problèmes à régler dépendaient des trois applications traitées.

1. Pour l'application VME, une exécution du code était déjà un ensemble de plusieurs milliers de tirages élémentaires Monte Carlo, de telle sorte que la sortie (une densité de probabilité) pouvait être plus ou moins précise suivant le nombre de tirages élémentaires effectués (le nombre de tirages pouvait être choisi). La distribution de probabilité en sortie du code était très éloignée d'une gaussienne, typiquement multimodale, avec certains modes de type Dirac. Les queues de distribution devaient être bien reproduites car l'utilisateur peut vouloir s'intéresser de près à des quantités dans ces zones — par exemple $\mathrm{P}(\mathrm{VAN} < 0)$.

2. Pour l'application CATHARE-CASTEM, le coût d'un calcul mécanique n'était pas négligeable, ce qui limitait le nombre de simulations possibles. De plus, la sortie du code dépendant de manière monotone de certaines des entrées, il pouvait être intéressant d'intégrer cette information en utilisant, par exemple, les approches proposées par [12] ou [27].

3. Pour l'application GIBBS (comme pour l'application VME) les sorties provenaient d'une méthode Monte Carlo. Le coût de calcul d'une exécution du code est très prohibitif. Bien que les distributions de probabilité en sortie du code n'avaient pas été étudiées, nous nous attendions à ce quelles soient généralement unimodales et lisses. Pour cette application, la moyenne et la variance devaient être bien reproduites car elles pourraient être utilisées ultérieurement dans l'optimisation des champs de force.

J'ai mené ce projet avec Vincent Moutoussamy (EDF) et Simon Nanty (CEA), alors également doctorants, sous l'encadrement de Amandine Marrel, Nadia Pérot (CEA), Nicolas Bousquet, Bertrand Ioos, Jérôme Lonchampt (EDF), Frédéric Delbos, Rafael Lugo (IFPEN) et les conseils avisés de Sébastien Da Veiga (ex-IFPEN), Delphine Sinoquet (IFPEN) et Anthony Nouy (École centrale de Nantes).

Formalisation du problème Considérons un code de simulation stochastique prenant d paramètres scalaires en entrée et dont la sortie sera supposée scalaire. Notons $\mathcal{X}$ la partie de $\mathbb{R}^d$ correspondant aux vecteurs de paramètres $x = (x_1, \ldots, x_d)$ avec lesquels le code peut être exécuté. Pour tous paramètres x dans $\mathcal{X}$ nous notons $G(x, \cdot)$ la densité de probabilité sur $\mathbb{R}$ de la sortie du code de simulation. Supposons que nous disposions d'un échantillon de N densités de probablité $f_1, \ldots f_N$ sur $\mathbb{R}$ associées à N vecteurs de paramètres $x_1, \ldots x_N$ dans

$\mathcal{X}$, connus également, de sorte que $G(x_i, \cdot) = f_i$ pour $i = 1, \ldots, N$. Dans la pratique le code de simulation est exécuté un grand nombre de fois pour chaque vecteur x_i de l'échantillon et la densité f_i est approchée à partir des valeurs obtenues avec une méthode d'estimation par noyau [82, 93]. Donnons-nous à présent un vecteur de paramètres quelconque de $\mathcal{X}$. Le problème consiste à approcher la densité de probabilité $f_0 = G(x_0, \cdot)$ de la sortie du code associée aux paramètres d'entrée x_0 à partir des données de l'échantillon.

Méthodes développées Nous proposons quatre méthodes de résolution, dont le point commun est d'écrire l'estimateur $\hat{f}_0$ de la densité f_0 sous la forme d'une combinaison convexe de q densités de probabilité $w_1, \ldots, w_q$ bien choisies :

$$\hat{f}_0 = \sum_{k=1}^{q} \psi_k(x_0) w_k, \tag{2.1}$$

avec $\psi_1(x_0), \ldots, \psi_q(x_0) \geq 0$ et $\sum_{k=1}^{q} \psi_k(x_0) = 1$. Observons que l'estimateur $\hat{f}_0$ défini par (2.1) est lui-même une densité de probabilité.

1. La première approche, baptisée *Kernel Regression*, est une adaptation de la régression non paramétrique à noyau [78,113]. Ici la taille de la base est $q = N$ et les densités de base $w_1, \ldots, w_q$ sont les densités $f_1, \ldots, f_N$ de l'échantillon. Donnons-nous un noyau gaussien K_H défini à partir d'une matrice diagonale H de taille d, à coefficients strictement positifs, de la façon suivante :

$$K_H(x, y) = \frac{1}{\sqrt{2\pi \, \text{dét}(H)}} \exp[-(x-y)^\top H^{-1}(x-y)], \quad x, y \in \mathcal{X}.$$

   Les coefficients de la combinaison convexe (2.1) sont alors obtenus à partir des points $x_, \ldots, x_N$ de l'échantillon :

$$\psi_k(x_0) = \frac{K_H(x_k, x_0)}{\sum_{j=1}^{N} K_H(x_j, x_0)}, \quad k = 1, \ldots, N.$$

   Les valeurs de la diagonale de H caractérisent l'importance donnée à chaque densité f_k suivant la distance euclidienne entre x_k et x_0. Elles sont choisies de façon à minimiser la distance entre l'estimateur $\hat{f}_0$ et la densité à estimer f_0. Nous avons testé cet estimateur pour la distance L^2 et la distance de Hellinger (distance L^2 entre les racines carrées des fonctions).

2. La seconde approche consiste à construire une base de densités $w_1, \ldots, w_q$ de taille q inférieure à N de façon à réduire la dimension du problème — en termes des coefficients $\psi_1(x_0), \ldots, \psi_q(x_0)$ à calculer. Les trois méthodes ci-après sont trois constructions différentes d'une telle base. Les valeurs $\psi_{ik} = \psi_k(x_i)$, $i = 1, \ldots, N$, sont également calculées au cours de la construction. Cependant les valeurs des coefficients en x_0 ne sont pas directement connues — à moins bien sûr que x_0 ne soit un point de l'échantillon. Notre idée était de construire les fonctions coefficients ψ_k par interpolation du système

$$\psi_k(x_i) = \psi_{ik}, \quad (i = 1, \ldots, N)(k = 1, \ldots, q),$$

mais les délais imposés pour la finalisation des projets du CEMRACS (pré-publications pour la fin 2013) ne nous ont pas permis de parvenir à des résultats satisfaisants ni de traiter cette question en profondeur. C'est pourquoi dans les trois paragraphes suivants nous nous intéresserons seulement aux estimateurs des N densités connues :

$$\hat{f}_i = \sum_{k=1}^{q} \psi_{ik} w_k, \quad i = 1, \dots, N. \tag{2.2}$$

(a) Notre deuxième méthode est une adaptation de l'analyse en composantes principales fonctionnelle [60, 86] qui satisfait la contrainte de positivité des densités de probabilité. En conséquence nous l'avons nommée *Constrained Principal Components Analysis*. Notons $\bar{f} = \sum_{i=1}^{N} f_i/N$ l'estimateur moyen de notre échantillon et $f_i^c = f_i - \bar{f}$ les densités centrées pour $i = 1, \dots, N$. La méthode consiste à construire une famille orthonormée de q fonctions $w_1, \dots, w_q$ de L^2 qui permettent d'approcher au mieux les densités centrées, au sens où les fonctions de cette famille minimisent le critère suivant :

$$\sum_{i=1}^{N} \left\| f_i^c - \sum_{k=1}^{q} \langle f_i^c, w_k \rangle_{\mathrm{L}^2} \, w_k \right\|_{\mathrm{L}^2}^2 .$$

Les parties positives des fonctions obtenues sont alors normalisées — au sens de la norme L^1 —, et sont encore notées $w_1, \dots, w_q$. L'estimateur de chaque densité de l'échantillon est alors donné par

$$\hat{f}_i = \bar{f} + \sum_{k=1}^{q} \langle f_i, w_k \rangle_{\mathrm{L}^2} \, w_k, \quad i = 1, \dots, N,$$

les coefficients $\langle f_i, w_k \rangle_{\mathrm{L}^2}$, $k = 1, \dots, q$, étant calculés au cours de l'algorithme.

(b) Notre troisième méthode, *Modified Magic Points*, est inspirée d'une publication récente [72] en théorie de l'interpolation. Les densités $w_1, \dots, w_q$ de base qui servent à définir l'estimateur (2.2) sont choisies itérativement parmi les densités $f_1, \dots, f_N$ de l'échantillon. À chaque itération les coefficients ψ_{ik} de l'estimateur $\hat{f}_i$ dans la base courante sont calculés comme solutions du programme suivant :

$$\begin{aligned} \underset{\psi_{ik}}{\text{minimiser}} \quad & \sum_{i=1}^{N} \left\| f_i - \sum_{k=1}^{q} \psi_{ik} w_k \right\|_{\mathrm{L}^2}^2 \\ \text{s. c.} \quad & \begin{cases} \psi_{ik} \geq 0 & (i = 1, \dots, N)(k = 1, \dots, q) \\ \displaystyle\sum_{k=1}^{q} \psi_{ik} = 1 & (i = 1, \dots, N). \end{cases} \end{aligned} \tag{2.3}$$

La densité f_i la plus éloignée de son estimateur courant $\hat{f}_i$ (au sens de la norme L^2) est ajoutée à la base de densités et q est incrémenté. Le processus se poursuit jusqu'à ce que l'erreur maximale $\max_{i=1,\dots,N} \|\hat{f}_i - f_i\|_{\mathrm{L}^2}$ de l'estimation passe sous un seuil fixé empiriquement. Nous construisons également des estimateurs similaires en remplaçant la distance L^2 par la distance de Hellinger.

(c) Les densités $w_1, \dots, w_q$ de la quatrième méthode ne sont pas nécessairement choisies parmi les densités $f_1, \dots, f_N$ de l'échantillon. Elles sont en effet directement calculées, avec les coefficients ψ_{ik}, comme solutions du programme suivant :

$$\begin{array}{ll} \underset{w_k, \psi_{ik}}{\text{minimiser}} & \dfrac{1}{2} \displaystyle\sum_{i=1}^{N} \left\| f_i - \sum_{k=1}^{q} \psi_{ik} w_k \right\|_{\mathrm{L}^2}^2 \\ \text{s. c.} & \begin{cases} w_k \geq 0 \text{ et } \displaystyle\int w_k = 1 & (k = 1, \dots, q) \\ \psi_{ik} \geq 0 & (i = 1, \dots, N)(k = 1, \dots, q) \\ \displaystyle\sum_{k=1}^{q} \psi_{ik} = 1 & (i = 1, \dots, N). \end{cases} \end{array} \tag{2.4}$$

Le problème de minimisation (2.4), plus complexe que (2.3), est résolu en agissant alternativement sur les densités w_k et les coefficients ψ_{ik} de façon à se ramener à des problèmes quadratiques, d'où le nom de la méthode : *Alternate Quadratic Minimizations*.

Les quatres méthodes développées ont été testées sur deux fonctions que nous avons définies analytiquement et appliquées aux trois codes industriels fournis par le CEA, EDF et IFPEN.

2.2 Émulateurs pour codes de simulation stochastiques

Nous reproduisons ici l'article [77] publié dans *ESAIM : Proceedings* suite au projet CODESTOCH.

ESAIM: PROCEEDINGS AND SURVEYS, January 2015, Vol. 48, p. 116-155
N. Champagnat, T. Lelièvre, A. Nouy, Editors

EMULATORS FOR STOCHASTIC SIMULATION CODES

Vincent Moutoussamy[1, 2], Simon Nanty[3, 4] and Benoît Pauwels[2, 5]

Abstract. Numerical simulation codes are very common tools to study complex phenomena, but they are often time-consuming and considered as *black boxes*. For some statistical studies (*e.g.* asset management, sensitivity analysis) or optimization problems (*e.g.* tuning of a molecular model), a high number of runs of such codes is needed. Therefore it is more convenient to build a fast-running approximation – or metamodel – of this code based on a design of experiments. The topic of this paper is the definition of metamodels for *stochastic* codes. Contrary to deterministic codes, stochastic codes can give different results when they are called several times with the same input. In this paper, two approaches are proposed to build a metamodel of the probability density function of a stochastic code output. The first one is based on *kernel regression* and the second one consists in decomposing the output density on a basis of well-chosen probability density functions, with a metamodel linking the coefficients and the input parameters. For the second approach, two types of decomposition are proposed, but no metamodel has been designed for the coefficients yet. This is a topic of future research. These methods are applied to two analytical models and three industrial cases.

Résumé. Les codes de simulation numérique sont couramment utilisés pour étudier des phénomènes complexes. Ils sont cependant souvent coûteux en temps de calcul. Pour certaines études statistiques (*e.g.* gestion d'actifs, analyse de sensibilité) ou problèmes d'optimisation (*e.g.* réglage des paramètres d'un modèle moléculaire), un grand nombre d'appels à de tels codes est nécessaire. Il est alors plus approprié d'élaborer une approximation – ou *métamodèle* – peu coûteuse en temps de calcul à partir d'un plan d'expérience. Le sujet de cet article est la définition de métamodèles pour des codes *stochastiques*. Contrairement aux codes *déterministes*, les codes stochastiques peuvent renvoyer des résultats différents lorsqu'ils sont appelés plusieurs fois avec le même jeu de paramètres. Dans cet article deux approches sont proposées pour la construction d'un métamodèle de la densité de probabilité de la sortie d'un code stochastique. La première repose sur la *régression à noyau* et la seconde consiste à décomposer la densité de la sortie du code dans une base de densités de probabilité bien choisies, avec un métamodèle exprimant les coefficients en fonction des paramètres d'entrée. Pour la seconde approche, deux types de décomposition sont proposés, mais aucun métamodèle n'a encore été élaboré pour les coefficients: c'est un sujet de recherches futures. Ces méthodes sont appliquées à deux cas analytiques et trois cas industriels.

[1] EDF R&D, MRI, Chatou, France; e-mail: vincent.moutoussamy@edf.fr
[2] Université Paul Sabatier, Toulouse, France
[3] CEA, DEN, DER, F-13108 Saint-Paul-Lez-Durance, France; e-mail: simon.nanty@cea.fr
[4] Université Joseph Fourier, Grenoble, France
[5] IFPEN, DTIMA, Rueil-Malmaison, France; e-mail: benoit.pauwels@ifpen.fr

Article published online by EDP Sciences and available at http://www.esaim-proc.org or http://dx.doi.org/10.1051/proc/201448005

Introduction

Numerical simulation codes are very common tools to study complex phenomena. However these computer codes can be intricate and time-consuming: a single run may take from a few hours to a few days. Therefore, when carrying out statistical studies (*e.g.* sensitivity analysis, reliability methods) or optimization processes (*e.g.* tuning of a molecular model), it is more convenient to build an approximation – or *metamodel* – of the computer code. Since this metamodel is supposed to take less time to run than the actual computer code, it is substituted for the numerical code. A numerical code is said *deterministic* if it gives the same output each time it is called with a given set of input parameters. For this case, the design of metamodels has already been widely studied. The present work deals with the formulation of a metamodel for a numerical code which is not deterministic, but *stochastic*. Two types of numerical codes can be covered by this framework. First, the code under consideration may be *intrinsically* stochastic: if the code is called with the same input several times, the output may differ. Thus, the output conditionally to the input is itself a random variable. Second, the computer code may be deterministic with complex inputs. For example, these inputs may be functions or spatiotemporal fields. These complex inputs are not treated explicitly and are called *uncontrollable parameters*, while the other parameters are considered as *controllable parameters*. When the code is run several times with the same set of controllable parameters, the uncontrollable parameters may change, and thus different output values may be obtained: the output is a random variable conditionally to the controllable parameters.

Two types of approaches can be found in the literature for stochastic code metamodeling. In the first one, strong hypotheses are made on the shape of the conditional distribution of the output. In [24] the authors consider that the output is normally distributed and estimate its mean and variance, in [23] the authors assume that the distribution of the output of the code is a mixture of normal distributions and estimate its mean. In the second approach, the mean and variance of the output are fitted jointly. In [13, 31] the authors use joint generalized linear models and joint generalized additive models respectively to fit these quantities. Joint metamodels based on Gaussian processes have been developed in [3, 17]. As we can observe, so far in the literature the metamodels proposed only provide an estimate for one or two of the output distribution moments, sometimes under strong hypotheses on the distribution of the output.

Our approach does not require such a priori hypotheses. Furthermore we study the whole distribution of the output, allowing to derive estimations for the mean, variance and other moments, and quantiles as well. The metamodeling is not carried out on the stochastic computer code under consideration itself, but rather on the code G that has same input and that returns the probability density function of the computer code output rather than just a single random value:

$$\begin{aligned} G : \boldsymbol{\mathcal{X}} \subset \mathbb{R}^d &\to \mathcal{F} \\ \mathbf{x} &\mapsto G(\mathbf{x}, \bullet), \end{aligned}$$

where $\mathcal{F}$ is the space of probability density functions with support in an interval I of $\mathbb{R}$:

$$\mathcal{F} = \left\{ f : I \to \mathbb{R}_+ : \int_I f = 1 \right\}.$$

Let $\mathbf{X}_N = \{(\mathbf{x}_i, f_i) : i = 1, \dots, N\} \subset (\boldsymbol{\mathcal{X}} \times \mathcal{F})^N$ be a *training set* ($N \in \mathbb{N} \setminus \{0\}$): we suppose that we have access to the probability density functions f_i of the outputs associated with N given sets of input parameters $\mathbf{x}_i$, *i.e.*

$$f_i = G(\mathbf{x}_i, \bullet) \qquad (i = 1, \dots, N).$$

In practice, for a given i in $\{1, \dots, N\}$, the output of a stochastic computer code is not a probability density function f_i. We only have access to a finite sample of output values $y_{i1}, \dots, y_{iM}$ which can be written as a vector $\begin{bmatrix} y_{i1} & \cdots & y_{iM} \end{bmatrix}$ representing the function. In real applications (*cf.* Sections 3.1, 3.2 and 3.3) we chose to

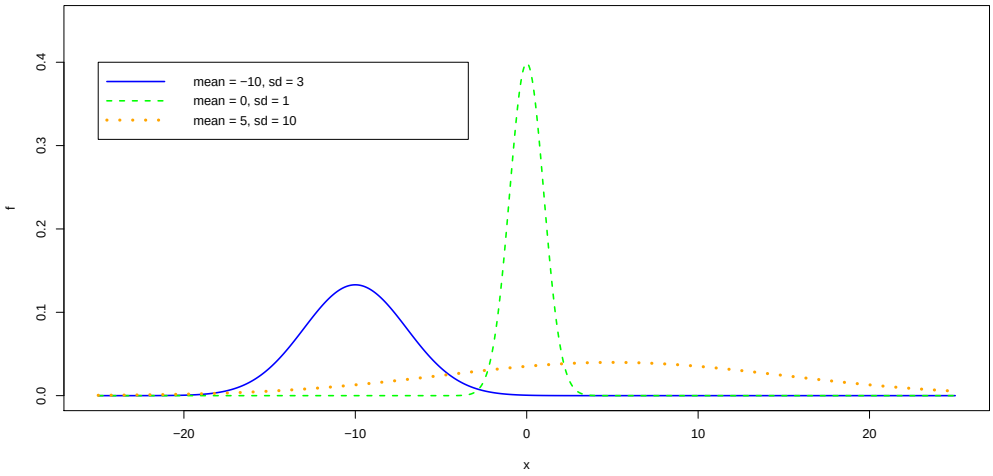

FIGURE 1. Probability density function given by the numerical code G for different values of the input parameters.

derive each probability density function f_i thanks to kernel density estimation. In this non-parametric method developed by Rosenblatt [25] and Parzen [21], the estimator of the probability density function f_i, $i = 1, \dots, N$ at a point y in I is defined as follows:

$$\hat{f}_i(y) = \frac{1}{M} \sum_{j=1}^{M} K_H \left(y_{ij} - y \right),$$

where $H \in \mathbb{R}$ is the bandwidth and K_H is the kernel. Let $\mathbf{x}_0 \in \boldsymbol{\mathcal{X}}$ be a new set of input parameters, we want to estimate the probability density function f_0 of the corresponding output:

$$\begin{aligned} f_0 = G(\mathbf{x}_0, \bullet) : I \subset \mathbb{R} &\to \mathbb{R}_+ \\ t &\mapsto f_0(t) = G(\mathbf{x}_0, t). \end{aligned}$$

For example, let us suppose that $\boldsymbol{\mathcal{X}} = \mathbb{R} \times \mathbb{R}_+^*$, $\mathbf{x}_0 = (\mu_0, \sigma_0)$ and $G((\mu_0, \sigma_0), \bullet) = \dfrac{e^{-\frac{1}{2}\left(\frac{\bullet - \mu_0}{\sigma_0}\right)^2}}{\sqrt{2\pi\sigma_0^2}}$. Figure 1 shows the output of G for different values of μ_0 and σ_0.

Section 1 expounds two types of metamodeling for stochastic computer codes: the first one provides a metamodel with functional outputs based on *kernel regression*, the second one consists in choosing relevant probability density functions to express the probability density of the output as a convex combination of them, the coefficients being linked to the input parameters by another metamodel. In Section 2, the previous methods are applied to two analytical test cases. Finally, the same methods are carried out on three industrial applications in Section 3.

1. METAMODELS FOR PROBABILITY DENSITY FUNCTIONS

In this section we expound our two approaches of formulating a metamodel for G. The first relies on *kernel regression* (*cf.* Section 1.1). The second consists in writing f_0 as a convex combination of well-chosen probability

density functions (*cf.* Section 1.2). In the following we denote $\|\bullet\|_{\mathrm{L}^2}$ and $\langle \bullet, \bullet \rangle_{\mathrm{L}^2}$ the norm and inner product of $\mathrm{L}^2(I)$ respectively: for all u and v in $\mathrm{L}^2(I)$,

$$\|u\|_{\mathrm{L}^2} = \left(\int_I u^2(t)\,\mathrm{d}t \right)^{1/2} \quad \text{and} \quad \langle u, v \rangle_{\mathrm{L}^2} = \int_I u(t)v(t)\,\mathrm{d}t.$$

1.1. **Kernel regression-based metamodel**

In this subsection, first we apply the classical kernel regression method to the problem under consideration, then a new kernel regression estimator involving the Hellinger distance is introduced and, finally, some perspectives of improvement are suggested.

1.1.1. *Classical kernel regression*

Let $\mathbf{x}_0 \in \boldsymbol{\mathcal{X}}$ and $f_0 = G(\mathbf{x}_0, \bullet)$. We suppose that the sample set $\mathbf{X}_N = \{(\mathbf{x}_i, f_i) : i = 1, \ldots, N\} \subset (\boldsymbol{\mathcal{X}} \times \mathcal{F})^N$ is available. We want to estimate f_0 by $\hat{f}_0 \in \mathcal{F}$, knowing $\mathbf{X}_N$. However, forcing $\hat{f}_0$ to be in $\mathcal{F}$ can be difficult in practice. We propose a first estimator given by

$$\hat{f}_0 = \sum_{i=1}^{N} \alpha_i f_i$$

where $\alpha_i \in \mathbb{R}$ $(i = 1, \ldots, N)$. In order to have $\hat{f}_0$ in $\mathcal{F}$, we impose

$$\begin{aligned} \alpha_i &\geq 0 \quad (i = 1, \ldots, N)\,, \\ \sum_{i=1}^{N} \alpha_i &= 1. \end{aligned}$$

The *non-parametric kernel regression* verifies the previous constraints. This estimator, introduced in [20, 30], can be written as follows:

$$\hat{f}_0 = \sum_{i=1}^{N} \frac{K_H(\mathbf{x}_i, \mathbf{x}_0)}{\sum_{j=1}^{N} K_H(\mathbf{x}_j, \mathbf{x}_0)} f_i, \tag{1}$$

where $K_H : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a *kernel function*. In the following, the *Gaussian kernel* is used:

$$K_H(\mathbf{x}, \mathbf{y}) = \frac{1}{\sqrt{2\pi \det(H)}} e^{-(\mathbf{x}-\mathbf{y})^{\mathsf{T}} H^{-1} (\mathbf{x}-\mathbf{y})} \quad \left(\mathbf{x}, \mathbf{y} \in \mathbb{R}^d\right),$$

where

$$H = \mathrm{diag}(h_1, \ldots, h_d)$$

is the (diagonal) *bandwidth matrix*, with $h_1, \ldots, h_d \in \mathbb{R}_+$. The higher h_j is, the more points $\mathbf{x}_i$ are taken into account in the regression in the direction j. The method relies on the intuition that if $\mathbf{x}_0$ is close to $\mathbf{x}_i$ then f_0 will be more influenced by the probability density function f_i (see Figure 2). We want that the bandwidth matrix H^* minimizes the global error of estimation. That means finding $H^* = \mathrm{diag}(h_1^*, \ldots, h_d^*)$ such that

$$\mathrm{diag}(h_1^*, \ldots, h_d^*) \in \underset{h_1, \ldots, h_d \in \mathbb{R}_+}{\arg\min} \int_{\boldsymbol{\mathcal{X}}} \left\| \hat{f}_0 - f_0 \right\|_{\mathrm{L}^2}^2 \mathrm{d}\mathbf{x}_0.$$

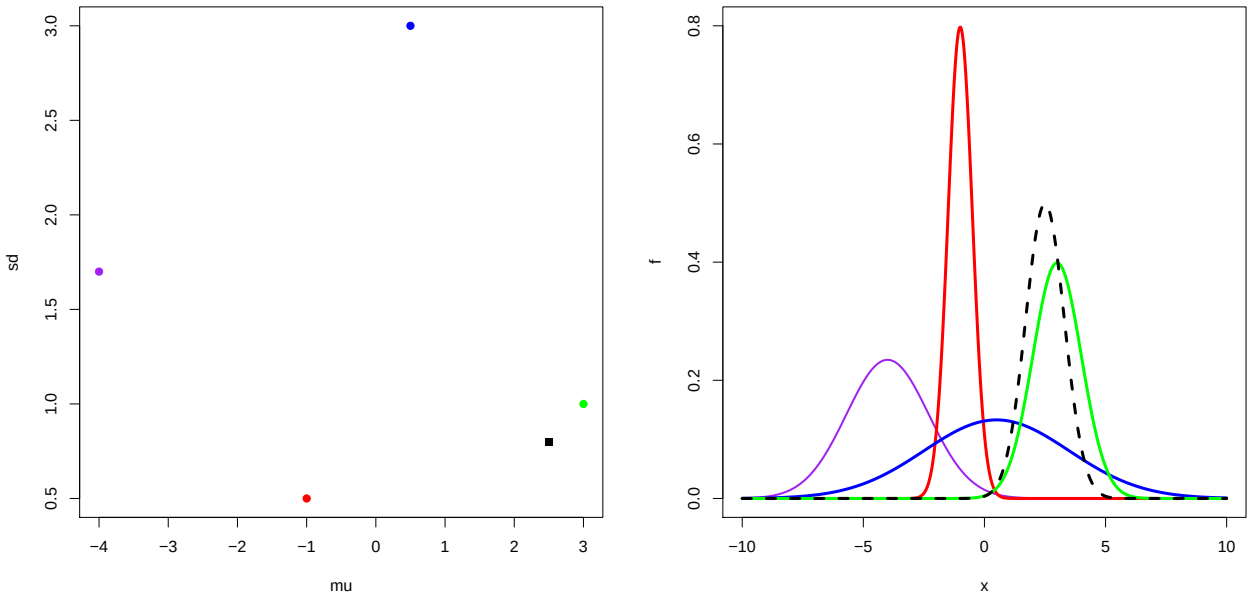

FIGURE 2. Left: the spots correspond to design points in $\boldsymbol{\mathcal{X}}$. Right: the associated probability density functions given by the code G defined in the introduction. The probability density function associated to the black square (dashed line) in left seems to be closer to the probability density function represented in green than to the others.

In order to get an approximation of H^*, one uses the *leave-one-out cross-validation*, as proposed in Hardle and Marron [11]. Then

$$H^* \in \underset{h_1,\dots,h_d\in\mathbb{R}_+}{\arg\min} \sum_{i=1}^{N} \left\| \hat{f}_{-i} - f_i \right\|_{\mathrm{L}^2}^2,$$

with

$$\hat{f}_{-i} = \sum_{\substack{j=1 \\ j\neq i}}^{N} \alpha_j^{-i} f_j$$

$$\alpha_j^{-i} = \frac{e^{-(\mathbf{x}_i-\mathbf{x}_j)^t H^{-1}(\mathbf{x}_i-\mathbf{x}_j)}}{\sum_{\substack{l=1 \\ l\neq i}}^{N} e^{-(\mathbf{x}_i-\mathbf{x}_l)^t H^{-1}(\mathbf{x}_i-\mathbf{x}_l)}}.$$

This optimization problem is solved with the L-BFGS-B algorithm developed by Byrd et al. [5]. Finally, the kernel regression metamodel is given by

$$\hat{f}_0 = \sum_{i=1}^{N} \alpha_{i,H^*} f_i, \tag{2}$$

with

$$\alpha_{i,H^*} = \frac{e^{-(\mathbf{x}_0-\mathbf{x}_i)^T(H^*)^{-1}(\mathbf{x}_0-\mathbf{x}_i)}}{\sum_{j=1}^{N} e^{-(\mathbf{x}_0-\mathbf{x}_j)^T(H^*)^{-1}(\mathbf{x}_0-\mathbf{x}_j)}}.$$

1.1.2. *Kernel regression with the Hellinger distance*

The kernel regression introduced in equation (1) can be considered as a minimization problem. The estimator of the function f_0 corresponding to the point $\mathbf{x}_0$ is as follows:

$$\hat{f}_0 = \underset{f\in\mathcal{F}}{\arg\min} \sum_{i=1}^{N} K_H(\mathbf{x}_i, \mathbf{x}_0) \int_I (f_i - f)^2 . \tag{3}$$

The kernel regression is therefore a locally weighted constant regression, which is fitted by minimizing weighted least squares. The equivalence between estimators (1) and (3) is proven in Appendix A.1. We observe that the L^2 distances between the sample functions and the unknown function appear in the objective function of the minimization problem (3).

Another classical example of distance between probability density functions is the Hellinger distance. We derived a new kernel regression estimator with respect to this distance replacing the L^2 distance by the Hellinger distance in equation (3). The kernel estimator thus takes the following form:

$$\hat{f}_0 = \underset{f\in\mathcal{F}}{\arg\min} \sum_{i=1}^{N} K_H(\mathbf{x}_i, \mathbf{x}_0) \int_I \left(\sqrt{f_i(t)} - \sqrt{f(t)}\right)^2 \mathrm{d}t. \tag{4}$$

The following analytical expression can be derived for this Hellinger distance-based estimator:

$$\hat{f}_0 = \frac{\left(\sum_{i=1}^{N} K_H(\mathbf{x}_i, \mathbf{x}_0)\sqrt{f_i}\right)^2}{\int_I \left(\sum_{i=1}^{N} K_H(\mathbf{x}_i, \mathbf{x}_0)\sqrt{f_i}\right)^2}. \tag{5}$$

The calculation that leads to (5) is provided in Appendix A.2.

From now on, we will denote $\hat{f}_{0,\mathrm{L}^2}$ and $\hat{f}_{0,\mathrm{He}}$ the estimators given by equations (2) and (5) respectively. Let us illustrate the Hellinger kernel regression on the four gaussian density functions given on Figure 2. On Figure 3 the same $N = 4$ learning curves (in dark blue, red, light blue and green) are represented, along with an unknown output f_0 (in dashed black) and its associated Hellinger kernel estimator $\hat{f}_{0,\mathrm{He}}$ (in orange). We can observe that, although the output is not very well estimated, most of the weight is given to the curve corresponding to the closest vector of parameters (in green).

1.1.3. *Perspectives*

One of the well-known major drawbacks of the kernel regression is due to the curse of dimensionality. The bandwidth estimation quality is deteriorating as the dimension d of the problem increases for a constant sample size. First, as the sample points are sparsely distributed in high dimension, the local averaging performs poorly. Second, the optimization problem to find the optimal bandwidth is more complex in higher dimension. In the presented work, kernel regression has been used with a fixed bandwidth for all sample points. The adaptive or "variable-bandwidth" kernel estimation techniques have been proposed in the literature to use

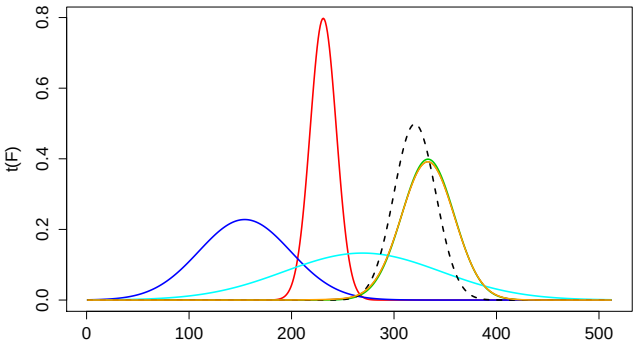

FIGURE 3. The orange curve is the Hellinger kernel prediction for the (unknown) output in black, built from the dark blue, red, light blue and green learning probability density functions ($N = 4$).

varying bandwidth over the domain of the parameters $\mathcal{X}$. We refer the reader to Muller and Stadtmuller [19] and Terrell and Scott [27] for more details on these methods. Terrell and Scott [27] review several adaptive methods and compare their asymptotic mean squared error, while Muller and Stadtmuller [19] propose a varying bandwidth selection method. In numerous cases, using non-constant bandwidths leads to better estimates as it enables to better capture the different features of the curves. Hence, the quality of the proposed methods could be improved by using adaptive kernel regression. However, the use of kernel regression is not advised in high dimension.

1.2. **Functional decomposition-based metamodel**

In this section, we aim at building a set of *basis functions* $w_1, \ldots, w_q$ in $\mathcal{F}$ to approximate f_0 by an estimator of the form

$$\hat{f}_0 = \sum_{k=1}^{q} \psi_k(\mathbf{x}_0) w_k, \tag{6}$$

where ψ_k are functions from $\boldsymbol{\mathcal{X}}$ to $\mathbb{R}$ $(k = 1, \ldots, q)$, called *coefficient functions*, satisfying

$$\begin{cases} \psi_k(\mathbf{x}) \geq 0 \\ \displaystyle\sum_{k=1}^{q} \psi_k(\mathbf{x}) = 1 \end{cases} \qquad (\mathbf{x} \in \boldsymbol{\mathcal{X}}) \,.$$

Kernel regression and functional decomposition have different goals but have similar forms. The former provides an estimator which is a convex combination of all the N sampled probability density functions $f_1, \ldots, f_N$. The latter gives a combination of fewer density functions built from the experimental data, thus lying in a smaller space, in order to *reduce the dimension* of the problem. We observe that the estimator provided by classical kernel regression (1) is of the form (6) with $q = N$ (no dimension reduction), $w_k = f_k$ for $k = 1, \ldots, q$ and

$$\psi_k(\mathbf{x}_0) = \frac{K_H(\mathbf{x}_k, \mathbf{x}_0)}{\displaystyle\sum_{j=1}^{N} K_H(\mathbf{x}_j, \mathbf{x}_0)} \qquad (k = 1, \ldots, q) \,.$$

We will build the basis functions and the coefficient functions so that they fit the available data $\mathbf{X}_N = \{(\mathbf{x}_i, f_i) : i = 1, \dots, N\}$. Thus the functions $w_1, \dots, w_q$ will be designed so that, for all i in $\{1, \dots, N\}$, f_i is approximated by

$$\hat{f}_i = \sum_{k=1}^{q} \psi_{ik} w_k$$
$$\text{where} \quad \begin{cases} \psi_{ik} = \psi_k(\mathbf{x}_i) & (k = 1, \dots, q) \\ \psi_{ik} \geq 0 & (k = 1, \dots, q) \\ \displaystyle\sum_{k=1}^{q} \psi_{ik} = 1. \end{cases} \tag{7}$$

For all i in $\{1, \dots, N\}$, we require the approximation $\hat{f}_i$ of f_i to be a probability density function as well; therefore $\hat{f}_i$ must be non-negative and have its integral equal to 1.

We study three ways of choosing the basis functions. In Section 1.2.1 we discuss the adaptation of *Functional Principal Components Analysis* (FPCA) to compute the basis functions: $w_1, \dots, w_q$ are then orthonormal with integral equal to 1 and their negative values are interpreted as 0, hence the predictions on the experimental design are probability density functions. In the following two subsections we propose two new approaches to probability density function decomposition. The non-negativity constraint is taken into account directly rather than through a post-treatment of FPCA results (setting negative values to zero and renormalizing). In Section 1.2.2, we propose to adapt the *Empirical Interpolation Method* or *Magic Points method* [16] to our problem: we obtain an algorithm to select the basis functions (not necessarily orthogonal) among the sample distributions $f_1, \dots, f_N$ (while losing the interpolation property). It follows that the predictions on the experimental design are probability density functions. In Section 1.2.3 a third approach is proposed, consisting in computing directly the basis functions that minimize the L^2-approximation error on the experimental design (without imposing orthogonality):

$$\sum_{i=1}^{N} \left\| f_i - \sum_{k=1}^{q} \psi_{ik} w_k \right\|_{\mathrm{L}^2}^2. \tag{8}$$

We introduce an iterative procedure to ease the numerical computation. The basis functions obtained are probability density functions and so are the predictions on the experimental design. Finally, in Section 1.2.4 we discuss the formulation of the coefficient functions.

1.2.1. *Constrained Principal Component Analysis (CPCA)*

Among all the *dimension reduction* techniques, one of the best known is the *Functional Principal Component Analysis* (FPCA), developed by Ramsay and Silverman [22], which is based on *Principal Component Analysis* (PCA). Let us denote $\bar{f}$ the *mean estimator* of $f_1, \dots, f_N$:

$$\bar{f} = \frac{1}{N} \sum_{i=1}^{N} f_i.$$

The goal of FPCA is to build an orthonormal basis $w_1, \dots, w_q$ for the *centered functions* $f_i^c = f_i - \bar{f}$ $(i = 1, \dots, N)$ so that the projected variance of every centered function onto the functions $w_1, \dots, w_q$ is maximized. The following maximization problem is thus considered:

$$\begin{aligned} \underset{w_1, \dots, w_q : I \to \mathbb{R}}{\text{Minimize}} \quad & \sum_{i=1}^{N} \left\| f_i^c - \sum_{k=1}^{q} \langle f_i^c, w_k \rangle_{\mathrm{L}^2} \, w_k \right\|_{\mathrm{L}^2}^2 \\ \text{subject to} \quad & \langle w_k, w_l \rangle_{\mathrm{L}^2} = \delta_{kl} \qquad (k, l = 1, \dots, q), \end{aligned}$$

where δ_{kl} is 1 when $k = l$ and 0 otherwise. Therefore the estimator of f_i $(i = 1, \dots, N)$ is

$$\hat{f}_i = \bar{f} + \sum_{k=1}^{q} \psi_{ik} w_k,$$

with $\psi_{ik} = \langle f_i^c, w_k \rangle_{\mathrm{L}^2}$ $(k = 1, \dots, q)$. To solve this minimization problem, Ramsay and Silverman [22] propose to express the sample functions on a spline basis. Then PCA can be applied to the coefficients of the functions on the spline basis. They also propose to apply PCA directly to the discretized functions. The FPCA decompostion ensures that $\int_I \hat{f}_i = \int_I f_i$ for $i = 1, \dots, N$ so that the integrals of the sample probability density functions approximations $\hat{f}_i$ are equal to one. However, the FPCA decomposition does not ensure that the estimators are non-negative.

Delicado [8] proposes to apply FPCA to the logarithm transform $g_i = \log f_i$ to ensure the positivity of the prediction $\hat{f}_i = \exp \hat{g}_i$ $(i = 1, \dots, N)$. However this approach does not ensure that the approximations are normalized.

Affleck-Graves et al. [1] have proposed to put a non-negativity constraint on the first basis function w_1 and let the other basis functions free. Indeed they claim that in general it is not possible to put such a constraint on the basis functions $w_2, \dots, w_q$. Hence the first basis function is forced to be non-negative but not the other ones. Therefore, it cannot be guaranteed that the approximations are non-negative.

Another method has been proposed by Kneip and Utikal [14] to ensure that the FPCA approximations are non-negative. The FPCA method is applied to the sample functions. Then the negative values of the probability density functions approximations are interpreted as 0 and they are normalized to one. In the numerical studies of Sections 2 and 3, we will refer to this method as CPCA, for Constrained Principal Component Analysis.

1.2.2. *Modified Magic Points (MMP)*

The *Empirical Interpolation Method* (EIM) or *Magic points* (MP) [16] is a *greedy* algorithm that builds interpolators for a set of functions. Here the set of functions under consideration is $G(\boldsymbol{\mathcal{X}}, \bullet) = \{G(\mathbf{x}, \bullet) : \mathbf{x} \in \boldsymbol{\mathcal{X}}\}$. The algorithm iteratively picks a set of basis functions in $G(\boldsymbol{\mathcal{X}}, \bullet)$ and a set of *interpolation points* in I to build an interpolator with respect to these points. Let $\mathbf{x}$ be in $\boldsymbol{\mathcal{X}}$ and $f_{\mathbf{x}} = G(\mathbf{x}, \bullet)$. At step $q-1$, the current basis functions and interpolation points are denoted $w_1, \dots, w_{q-1} : I \to \mathbb{R}$ and $t_{i_1}, \dots, t_{i_{q-1}} \in I$ respectively. The interpolator $I_{q-1}[f_{\mathbf{x}}]$ of $f_{\mathbf{x}}$ is defined as a linear combination of the basis functions:

$$I_{q-1}[f_{\mathbf{x}}] = \sum_{k=1}^{q-1} \psi_k(\mathbf{x}) w_k,$$

where the values of the coefficient functions $\psi_1, \dots, \psi_{q-1} : \boldsymbol{\mathcal{X}} \to \mathbb{R}$ at $\mathbf{x}$ are uniquely defined by the following *interpolation equations* (see [16] for existence and uniqueness):

$$I_{q-1}[f_{\mathbf{x}}](t_{i_k}) = f_{\mathbf{x}}(t_{i_k}) \qquad (k = 1, \dots, q-1).$$

At step q, one picks the parameter $\mathbf{x}_{i_q}$ such that $f_{\mathbf{x}_{i_q}} \left(= G\left(\mathbf{x}_{i_q}, \bullet\right)\right)$ is the element of $G(\boldsymbol{\mathcal{X}}, \bullet)$ whose L^2-distance from its own current interpolator $I_{q-1}[f_{\mathbf{x}_{i_q}}]$ is the highest. The next interpolation point t_{i_q} is the one that maximizes the gap between $f_{\mathbf{x}_{i_q}}$ and $I_{q-1}[f_{\mathbf{x}_{i_q}}]$. Then the new basis function w_q is defined as a particular linear combination of $f_{\mathbf{x}_{i_q}}$ and $I_{q-1}[f_{\mathbf{x}_{i_q}}]$. Let us be more specific by summarizing the algorithm [4] hereafter.

Algorithm 1.1 (MP)**.**

- *Set* $q \leftarrow 1$ *and* $I_0 \leftarrow 0$.
- *While* $\varepsilon > tol$ *do*
  (a) choose $\mathbf{x}_{i_q}$ *in* $\boldsymbol{\mathcal{X}}$*:*

$$\mathbf{x}_{i_q} \leftarrow \arg\sup_{\mathbf{x} \in \boldsymbol{\mathcal{X}}} \left\| f_{\mathbf{x}} - I_{q-1}[f_{\mathbf{x}}] \right\|_{\mathrm{L}^2}$$

and the associated interpolation point t_{i_q}*:*

$$t_{i_q} \leftarrow \arg\sup_{t\in I} \left| f_{\mathbf{x}_{i_q}}(t) - I_{q-1}[f_{\mathbf{x}_{i_q}}](t) \right|,$$

(b) define the next element of the basis:

$$w_q \leftarrow \frac{f_{\mathbf{x}_{i_q}}(\bullet) - I_{q-1}[f_{\mathbf{x}_{i_q}}](\bullet)}{f_{\mathbf{x}_{i_q}}(t_{i_q}) - I_{q-1}[f_{\mathbf{x}_{i_q}}](t_{i_q})},$$

(c) compute the interpolation error*:*

$$\varepsilon \leftarrow \|err_q\|_{\mathrm{L}^\infty(\boldsymbol{\mathcal{X}})}, \quad \textit{where } err_q : \mathbf{x} \mapsto \|f_{\mathbf{x}} - I_q[f_{\mathbf{x}}]\|_{\mathrm{L}^2},$$

(d) set $q \leftarrow q+1$*.*

This method has been successfully applied *e.g.* to a heat conduction problem, crack detection and damage assessment of flawed materials, inverse scattering analysis [6], parameter-dependent convection-diffusion problems around rigid bodies [28], in biomechanics to describe a stenosed channel and a bypass configuration [26].

In general, the interpolators provided by the MP algorithm are not probability density functions. Therefore we modified the MP method so that the approximations are non-negative with integral equal to one. In the derived new method the interpolation is lost, but the basis functions are picked in a similar greedy way. In the following we denote $A_{q-1}[f_{\mathbf{x}}]$ the interpolator at step $q-1$ of the function $f_{\mathbf{x}}$ associated with a parameter $\mathbf{x}$ in $\boldsymbol{\mathcal{X}}$:

$$A_{q-1}[f_{\mathbf{x}}] = \sum_{k=1}^{q-1} \psi_k(\mathbf{x}) w_k,$$

where the values of the coefficient functions at $\mathbf{x}$ are now defined as solutions of the following convex quadratic program:

$$\begin{aligned} \underset{\psi_1(\mathbf{x}),\ldots,\psi_{q-1}(\mathbf{x})\in\mathbb{R}}{\text{Minimize}} \quad & \left\| f_{\mathbf{x}} - \sum_{k=1}^{q-1} \psi_k(\mathbf{x}) w_k \right\|^2_{\mathrm{L}^2} \\ \text{subject to} \quad & \begin{cases} \psi_k(\mathbf{x}) \geq 0 & (k=1,\ldots,q-1) \\ \displaystyle\sum_{k=1}^{q-1} \psi_k(\mathbf{x}) = 1. \end{cases} \end{aligned} \tag{9}$$

At step q, the parameter $\mathbf{x}_{i_q}$ is chosen in the same way as in the MP algorithm. The new basis function w_q is then $f_{\mathbf{x}_{i_q}}$. The *Modified Magic Points* (MMP) algorithm is detailed below.

Algorithm 1.2 (MMP)**.**

- *Set* $q \leftarrow 1$ *and* $I_0 \leftarrow 0$*.*
- *While* $\varepsilon > tol$ *do*

  (a) choose $\mathbf{x}_{i_q}$ *in* $\boldsymbol{\mathcal{X}}$*:*

  $$\mathbf{x}_{i_q} \leftarrow \arg\sup_{\mathbf{x}\in\boldsymbol{\mathcal{X}}} \|f_{\mathbf{x}} - A_{q-1}[f_{\mathbf{x}}]\|_{\mathrm{L}^2},$$

  (b) define the next element of the basis:

  $$w_q \leftarrow f_{\mathbf{x}_{i_q}},$$

(c) compute the estimation error*:*

$$\varepsilon \leftarrow \|err_q\|_{\mathrm{L}^\infty(\boldsymbol{\mathcal{X}})}\,, \quad \textit{where } err_q : \mathbf{x} \mapsto \|f_{\mathbf{x}} - A_q[f_{\mathbf{x}}]\|_{\mathrm{L}^2}\,,$$

(d) set $q \leftarrow q+1$.

Remark 1.3. In step (a) of the MMP algorithm, the L^2 norm can be replaced by the Hellinger distance. This yields a new algorithm which is tested along with the previous one in Section 2.

In practice, we do not apply the MMP algorithm to the whole set $G(\boldsymbol{\mathcal{X}}, \bullet)$ since the computation of the functions is time-consuming. It is rather applied to the available sample set $\{f_1, \ldots, f_N\}$. Thus the MMP algorithm is a way to select the most relevant functions among the sample $f_1, \ldots, f_N$. Then the step (a) of the algorithm is only a finite maximization. Furthermore we build a regular grid of M points $t_1, \ldots, t_M$ in the interval I:

$$\begin{array}{l} t_j = t_1 + (j-1)\Delta t \qquad (j = 1, \ldots, M) \\ \text{with } \Delta t \in \mathbb{R}_+, \end{array} \tag{10}$$

and we discretize the functions f_i $(i = 1, \ldots, N)$ and w_k $(k = 1, \ldots, q)$ on this grid:

$$\begin{array}{c} f_{ij} = f_i(t_j) \\ \mathrm{w}_{kj} = w_k(t_j) \end{array} \qquad (j = 1, \ldots, M).$$

Thus we can rewrite the problem (9) in the following way for all $i \in \{1, \ldots, N\}$:

$$\begin{array}{ll} \underset{\psi_{i1}, \ldots, \psi_{iq}}{\text{Minimize}} & \displaystyle\sum_{j=1}^{M} \left[f_{ij} - \sum_{k=1}^{q} \psi_{ik} w_{kj} \right]^2 \Delta t \\ \text{subject to} & \begin{cases} \psi_{i1}, \ldots, \psi_{iq} \geq 0 \\ \displaystyle\sum_{k=1}^{q} \psi_{ik} = 1. \end{cases} \end{array} \tag{11}$$

These N problems are convex quadratic programs with q unknowns and $q+1$ linear constraints, they are easily solvable for basis sizes smaller than 10^4.

Remark 1.4. Let us suppose $q = 1$, *i.e.* the basis contains only one function. The equality constraint in the minimization (11) reduces to $\psi_{1i} = 1$: the approximation is the same for each function f_i in the sample.

1.2.3. *Alternate Quadratic Minimizations (AQM)*

Our third functional decomposition approach consists in tackling directly the problem consisting in minimizing the L^2 norm of the approximation error (8) with basis functions $w_1, \ldots, w_q$ in $\mathcal{F}$ and coefficients ψ_{ik} satisfying the constraints (7). It differs from PCA because no orthonormality condition is imposed, and the decomposition is carried out directly on the raw data, *i.e.* without centering. This functional minimization program can be written as follows.

$$\begin{array}{ll} \underset{\substack{w_k, \psi_{ik} \\ k=1,\ldots,q, i=1,\ldots,N}}{\text{Minimize}} & \displaystyle\frac{1}{2} \sum_{i=1}^{N} \left\| f_i - \sum_{k=1}^{q} \psi_{ik} w_k \right\|_{\mathrm{L}^2}^2 \\ \text{subject to} & \begin{cases} w_k \in \mathcal{F} & (k = 1, \ldots, q) \\ \psi_{ik} \geq 0 & (i = 1, \ldots, N)(k = 1, \ldots, q) \\ \displaystyle\sum_{k=1}^{q} \psi_{ik} = 1 & (i = 1, \ldots, N). \end{cases} \end{array} \tag{12}$$

Just as before, instead of working on actual functions, we consider discretizations of them. We discretize the interval I in the same way (10) and we set

$$\begin{aligned}\mathbf{f}_i &= \begin{bmatrix} f_i(t_1) & \cdots & f_i(t_M)\end{bmatrix} \qquad (i=1,\dots,N)\\ &= \begin{bmatrix} \mathrm{f}_{i1} & \cdots & \mathrm{f}_{iM}\end{bmatrix}\\ \mathbf{w}_k &= \begin{bmatrix} w_k(t_1) & \cdots & w_k(t_M)\end{bmatrix} \qquad (k=1,\dots,q)\\ &= \begin{bmatrix} \mathrm{w}_{k1} & \cdots & \mathrm{w}_{kM}\end{bmatrix}.\end{aligned}$$

The functional minimization program (12) is then replaced by the following vectorial minimization program (in this section $\|\bullet\|_{\mathbb{R}^M}$ designates the euclidean norm on $\mathbb{R}^M$).

$$\begin{aligned}\underset{\substack{\mathbf{w}_k,\psi_{ik}\\ k=1,\dots,q,i=1,\dots,N}}{\text{Minimize}} \quad & \frac{1}{2}\sum_{i=1}^{N}\left\|\mathbf{f}_i-\sum_{k=1}^{q}\psi_{ik}\mathbf{w}_k\right\|_{\mathbb{R}^M}^2\\ \text{subject to} \quad & \begin{cases} \mathrm{w}_{kj}\geq 0 & (k=1,\dots,q)(j=1,\dots,M)\\ \displaystyle\sum_{j=1}^{M}\mathrm{w}_{kj}\Delta t=1 & (k=1,\dots,q)\\ \psi_{ik}\geq 0 & (i=1,\dots,N)(k=1,\dots,q)\\ \displaystyle\sum_{k=1}^{q}\psi_{ik}=1 & (i=1,\dots,N).\end{cases}\end{aligned} \tag{13}$$

Let us introduce additional notations to reformulate the objective function and the variables in a more compact way.

$$\mathbf{\Psi}=[\psi_{ik}]_{i=1,\dots,N;k=1,\dots,q}=\begin{bmatrix}\boldsymbol{\psi}_1\\ \vdots\\ \boldsymbol{\psi}_N\end{bmatrix} \qquad \mathbf{W}=[\mathrm{w}_{kj}]_{k=1,\dots,q;\ j=1,\dots,M}=\begin{bmatrix}\mathbf{w}_1\\ \vdots\\ \mathbf{w}_q\end{bmatrix}$$

We recall that the Frobenius norm of a matrix $\mathbf{A}=[a_{ij}]_{i=1,\dots,N;j=1,\dots,M}\in\mathbb{R}^{N\times M}$ is defined as

$$\|\mathbf{A}\|_{\mathrm{F}}=\sqrt{\sum_{i=1}^{N}\sum_{j=1}^{M}a_{ij}^2}.$$

This being said, the vectorial minimization program (13) can be written in matrix form.

$$\begin{aligned}\underset{\mathbf{\Psi}\in\mathbb{R}_+^{N\times q},\mathbf{W}\in\mathbb{R}_+^{q\times M}}{\text{Minimize}} \quad & \mathrm{O}\left(\mathbf{\Psi},\mathbf{W}\right)=\frac{1}{2}\left\|\mathbf{F}-\mathbf{\Psi}\mathbf{W}\right\|_{\mathrm{F}}^2\\ \text{subject to} \quad & \begin{cases}\boldsymbol{\psi}_i\mathbf{1}=1 & (i=1,\dots,N)\\ \mathbf{w}_k\mathbf{1}=1/\Delta t & (k=1,\dots,q).\end{cases}\end{aligned} \tag{14}$$

The objective function of the program (13) is second-order polynomial, hence twice continuously differentiable (but not necessarily convex). Its first-order derivatives are expressed as

$$\begin{aligned}\partial_{\psi_{ik}}\mathrm{O}\left(\mathbf{\Psi},\mathbf{W}\right)&=-\mathbf{w}_k\left(\mathbf{f}_i-\boldsymbol{\psi}_i\mathbf{W}\right)^{\mathsf{T}} && (i=1,\dots,N)(k=1,\dots,q),\\ \partial_{w_{kj}}\mathrm{O}\left(\mathbf{\Psi},\mathbf{W}\right)&=-\mathbf{\Psi}_{\bullet,k}^{\mathsf{T}}\left(\mathbf{F}_{\bullet,j}-\mathbf{\Psi}\mathbf{W}_{\bullet,j}\right) && (k=1,\dots,q)\,(j=1,\dots,M),\end{aligned}$$

where $\mathbf{\Psi}_{\bullet,k}$ is the k^{th} column of $\mathbf{\Psi}$, $\mathbf{F}_{\bullet,j}$ and $\mathbf{W}_{\bullet,j}$ are the j^{th} columns of $\mathbf{F}$ and $\mathbf{W}$ respectively, and here are the second-order derivatives which will be useful further down:

$$\begin{aligned}
\partial^2_{\psi_{ik}\psi_{ik'}} \mathrm{O}(\mathbf{\Psi},\mathbf{W}) &= \mathbf{w}_k\mathbf{w}_{k'}^{\mathsf{T}} && (i=1,\dots,N)(k,k'=1,\dots,q),\\
\partial^2_{w_{kj}w_{kj'}} \mathrm{O}(\mathbf{\Psi},\mathbf{W}) &= \delta_{jj'}\,\|\mathbf{\Psi}_{\bullet,k}\|^2_{\mathbb{R}^N} && (k=1,\dots,q)\,(j,j'=1,\dots,M)\,.
\end{aligned}$$

The program (14) has $q(M+N)\approx 10^4$ design variables, $q(M+N)+q+N\approx 10^4$ constraints and may not be convex. Hence its resolution through the use of a numerical solver may be very time-consuming. In order to circumvent this issue we chose to implement (14) as successive convex quadratic minimization programs. The value of the program can be written as follows.

$$\inf_{\substack{(\mathbf{\Psi},\mathbf{W})\in\left(\mathbb{R}_+^{N\times M}\right)^2\\ \boldsymbol{\psi}_i\mathbf{1}=1\ (i=1,\dots,N)\\ \mathbf{w}_k\mathbf{1}=1/\Delta t\ (k=1,\dots,q)}} \mathrm{O}(\mathbf{\Psi},\mathbf{W}) = \inf_{\substack{\mathbf{w}_1\in\mathbb{R}_+^M\\ \mathbf{w}_1\mathbf{1}=1/\Delta t}} \cdots \inf_{\substack{\mathbf{w}_q\in\mathbb{R}_+^M\\ \mathbf{w}_q\mathbf{1}=1/\Delta t}} \inf_{\substack{\boldsymbol{\psi}_1\in\mathbb{R}_+^q\\ \boldsymbol{\psi}_1\mathbf{1}=1}} \cdots \inf_{\substack{\boldsymbol{\psi}_N\in\mathbb{R}_+^q\\ \boldsymbol{\psi}_N\mathbf{1}=1}} \mathrm{O}(\mathbf{\Psi},\mathbf{W})\,.$$

The idea is to minimize the criterion $\mathrm{O}(\mathbf{\Psi},\mathbf{W})$ working on one line-vector at a time: $\boldsymbol{\psi}_1$, then $\boldsymbol{\psi}_2$, and so on until $\boldsymbol{\psi}_N$, and then $\mathbf{w}_1$, $\mathbf{w}_2$ and so forth until $\mathbf{w}_q$. This process being repeated as many times as necessary – and affordable – to get some kind of convergence.

Let us make one more remark before giving out the algorithm. Let $i\in\{1,\dots,N\}$. The two following programs are equivalent – in the sense that their feasible and optimal solutions are the same (we just reduced the criterion to minimize).

$$\begin{array}{lll}
\underset{\boldsymbol{\psi}_i}{\text{Minimize}} & \mathrm{O}(\mathbf{\Psi},\mathbf{W}) & \\
\text{subject to} & \begin{cases}\boldsymbol{\psi}_i\in\mathbb{R}_+^q\\ \boldsymbol{\psi}_i\mathbf{1}=1\end{cases} & \iff\\
\end{array}
\qquad
\begin{array}{lll}
\underset{\boldsymbol{\psi}_i}{\text{Minimize}} & \frac{1}{2}\,\|\mathbf{f}_i-\boldsymbol{\psi}_i\mathbf{W}\|^2_{\mathbb{R}^M} & \\
\text{subject to} & \begin{cases}\boldsymbol{\psi}_i\in\mathbb{R}_+^q\\ \boldsymbol{\psi}_i\mathbf{1}=1.\end{cases} &
\end{array}$$

The algorithm we implemented to numerically solve (13) is the following.

Algorithm 1.5 (AQM)**.**

- *Initialize* $\mathbf{\Psi}$ *and* $\mathbf{W}$ *with uniform values.*

$$\begin{aligned}
\psi_{ik} &= 1/q && (i=1,\dots,N)(k=1,\dots,q)\\
\mathrm{w}_{kj} &= 1/(M\Delta t) && (k=1,\dots,q)(j=1,\dots,M).
\end{aligned}$$

- *While some stopping criterion is not reached (i.e. a maximum number* $iter_{max}\approx 20$ *of iterations),*
  (a) do, for $i=1,\dots,N$*,*

$$\begin{array}{lll}
\underset{\boldsymbol{\psi}_i}{\textit{Minimize}} & \frac{1}{2}\,\|\mathbf{f}_i-\boldsymbol{\psi}_i\mathbf{W}\|^2_{\mathbb{R}^M} & \\
\textit{subject to} & \begin{cases}\boldsymbol{\psi}_i\in\mathbb{R}_+^q\\ \boldsymbol{\psi}_i\mathbf{1}=1.\end{cases} &
\end{array}$$

  (b) do, for $k=1,\dots,q$*,*

$$\begin{array}{lll}
\underset{\mathbf{w}_k}{\textit{Minimize}} & \mathrm{O}(\mathbf{\Psi},\mathbf{W}) & \\
\textit{subject to} & \begin{cases}\mathbf{w}_k\in\mathbb{R}_+^M\\ \mathbf{w}_k\mathbf{1}=1/\Delta t.\end{cases} &
\end{array}$$

Each of the minimization problems addressed in this algorithm has a convex quadratic criterion depending on at most $M \approx 2000$ variables constrained by the same amount of lower bounds plus one linear equality. These optimization problems were solved with a dual method of the *active set* type, detailed in Goldfarb and Idnani [10], specifically designed to address strictly convex quadratic programs. For a given quadratic program P, the *active set* of a point $\mathbf{x}$ is the set of (linear) inequality constraints satisfied with equality (*active constraints*). The algorithm starts from a point $\mathbf{x}$ such that the set A of its active constraints (possibly empty) is linearly independent, and the following steps are repeated until all constraints are satisfied. First, a violated constraint $\mathbf{c}$ is picked. Second, the feasibility of the subproblem $P(A \cup \{\mathbf{c}\})$ of P constrained only by the constraint $\mathbf{c}$ and those of the current active set A is tested. If $P(A \cup \{\mathbf{c}\})$ is infeasible then P is infeasible as well. Otherwise $\mathbf{x}$ is updated with a point whose active set is a linearly independent subset of $A \cup \{\mathbf{c}\}$ including $\mathbf{c}$. The loop terminates with an optimal solution.

1.2.4. *Metamodelling of functional decomposition coefficients*

In Sections 1.2.1, 1.2.2 and 1.2.3, three methods have been studied to approximate a sample of probability density functions in a basis such that:

$$\hat{f}_i = \sum_{k=1}^{q} \psi_{ik} w_k.$$

The functions are characterized by their coefficients. The dimension of the problem is therefore reduced to the basis size q. For MMP and AQM, the coefficient functions must respect the following constraints:

$$\begin{cases} \displaystyle\sum_{k=1}^{q} \psi_{ik} = 1 \\ \quad \psi_{ik} \geq 0 \qquad (k = 1, \ldots, q). \end{cases} \tag{15}$$

To provide a link between the input parameters and the coefficients, a metamodel can be designed as a function $\hat{\boldsymbol{\psi}} : \boldsymbol{\mathcal{X}} \to \mathbb{R}^q$, where $\hat{\boldsymbol{\psi}} = (\hat{\psi}_1, \ldots, \hat{\psi}_q)$. The metamodel is built from the known values of ψ_k in the learning sample: $\psi_{ik} = \psi_k(\mathbf{x}_i)$ $(i = 1, \ldots, N, k = 1, \ldots, q)$. The outputs of the metamodel to be built must respect the constraints in equation (15), *i.e.* be nonnegative and have their sum equal to one. For a new $\mathbf{x} \in \boldsymbol{\mathcal{X}}$, we look for the estimation $\hat{f}_{\mathbf{x}}$ of the output $f_{\mathbf{x}}$ of G:

$$\hat{f}_{\mathbf{x}} = \sum_{k=1}^{q} \hat{\psi}_k(\mathbf{x}) w_k.$$

So far, the main approach investigated has been to build independently one metamodel for each coefficient without constraining the output of the metamodel. When the metamodel is used to make predictions on new points in the input space, the predicted coefficients which are lower than zero are put equal to zero, and the coefficients are then renormalized such that their sum is equal to one. Any type of metamodel can be used with this approach. Polynomial regression, generalized additive models [12] and Gaussian process metamodels have been tested, but the obtained results on different test cases are not convincing. The metamodel error is bigger than the error due to the decomposition on a functional basis. This is the reason why the results are not reported here.

In the two developed decomposition methods, MMP and AQM, the studied functions are discretized on a grid. This discretization approach may lead to a less robust estimation of the coefficient functions. In particular, the estimation of the discretized coefficient functions could be too noisy, and these functions could become too irregular. These problems especially impact the metamodelling step. Because of the noise added to the coefficient functions to be approximated, the accuracy of the metamodel could be reduced. As proposed by Ramsay and Silverman [22], a possible way to overcome this problem and to improve the estimation stability is to search for the coefficient functions ψ_k in a functional basis with regularity constraints like spline or wavelet basis.

Future research should be dedicated to the construction of more efficient metamodels taking into account these constraints. Metamodels which ensure the nonnegativity of the output exist in the literature (see for example [7] for Gaussian process with inequality constraints). The main difficulty is to ensure that the sum of the coefficients is equal to one. A solution can be to consider the coefficients as realizations of a random variable on a simplex space. Aitchison [2] proposes to apply a bijective transformation from this simplex space to $\mathbb{R}^q$. Then, an unconstrained metamodel could be built to model the transformed coefficients.

2. Numerical tests on toy functions

In this section, we apply the methods presented on two toy examples respectively for $d = 1$ in Section 2.1 and $d = 5$ in Section 2.2. We compare the relative error of the estimation of different quantities of interest: L^1, L^2 and the Hellinger distances between two densities, mean, variance and the (1%, 25%, 75%, 99%)-quantiles. We recall that the Hellinger distance between f and g is the L^2 distance between $\sqrt{f}$ and $\sqrt{g}$. The relative error is defined below for the three norms and the other scalar quantities of interest:

$$
\begin{aligned}
E_{rel,\mathrm{L}^1} &= 100\frac{\int_I \left|f(t) - \hat{f}(t)\right| \,\mathrm{d}t}{\int_I |f(t)| \,\mathrm{d}t} \\
E_{rel,\mathrm{L}^2} &= 100\frac{\int_I \left|f(t) - \hat{f}(t)\right|^2 \,\mathrm{d}t}{\int_I f(t)^2 \,\mathrm{d}t} \\
E_{rel,He} &= 100\frac{\int_I \left|\sqrt{f(t)} - \sqrt{\hat{f}(t)}\right|^2 \,\mathrm{d}t}{\int_I \sqrt{f(t)}^2 \,\mathrm{d}t} \\
E_{rel,u} &= 100\frac{|u - \hat{u}|}{|u|},
\end{aligned}
$$

where u is the mean, variance and the studied quantiles which are defined as $\inf\{q \in \mathbb{R},\ \hat{F}(q) > p\}$ with $p \in \mathbb{R}$ and $\hat{F}(q) = \int_{-\infty}^{q} \hat{f}(t)dt$. The methods developed in this article may not be suited to extreme quantiles and probabilities estimation as they are designed to approximate the global shape of the probability density functions. A sequential strategy of simulation on the space of input parameters is more adapted to estimate such quantities.

This study is separated in two independent parts for the two toy examples.

First, we compare the error obtained from the two estimators $\hat{f}_{\mathrm{L}^2}$ and $\hat{f}_{\mathrm{He}}$ based on kernel regression. We show the relative error in function of different sizes $N_1 < \cdots < N_k$ of the design of experiments such that a design with N_i points is included in the designs with N_j points ($i < j$). For each N_i, relative error for different quantities of interest are averaged on the same 1000 test points chosen uniformly in $\boldsymbol{\mathcal{X}}$. These computations are repeated 25 times with different designs of experiments.

In the second part, we compare the relative error obtained from the construction of basis obtained in Sections 1.2.1, 1.2.2 and 1.2.3 to reconstruct the N density probability functions of the design of experiments. These two methods decompose the known probability density functions on a functional basis and approximate them. However, no metamodel has been designed between the coefficients of the probability density functions on the basis and the parameters of the computer code, so that no estimation of an unknown probability density function can be done. Therefore, both methods cannot be compared to the kernel regression.

Each probability density function is discretized on 512 points.

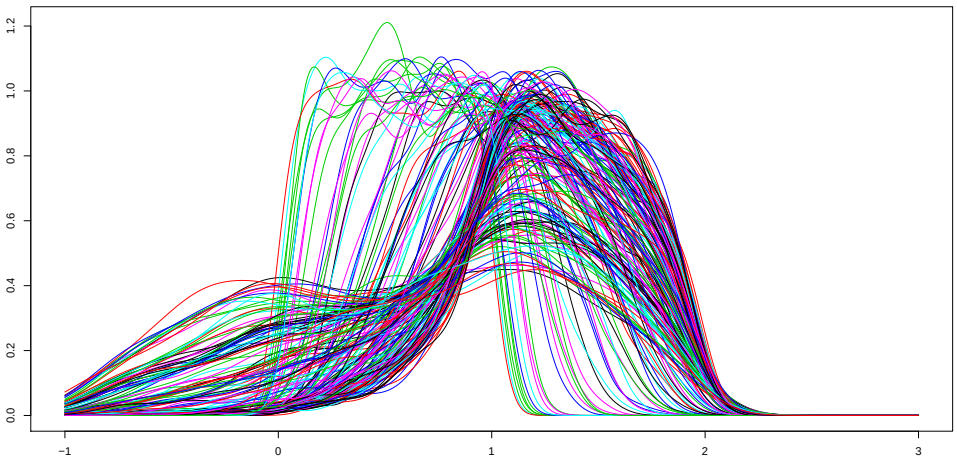

FIGURE 4. Toy example 1: representations of outputs for $N = 100$ different parameters.

2.1. **A one-dimensional example (Toy example 1)**

The first toy example is defined as follows:

$$G(x, \xi_1, \xi_2, U) = (\sin(x(\xi_1 + \xi_2)) + U)\mathbb{1}_{\{\sin(x(\xi_1+\xi_2))+U \geq -1\}}$$

where

$$\begin{aligned} x &\in \boldsymbol{\mathcal{X}} = [0, 1] \\ \xi_1 &\sim \mathcal{N}(1, 1) \\ \xi_2 &\sim \mathcal{N}(2, 1) \\ U &\sim \mathcal{U}([0, 1]). \end{aligned}$$

Let $x_0 \in [0, 1]$, we want to estimate the probability density function f_0 of the random variable $G(x_0, \xi_1, \xi_2, U)$.

2.1.1. *Kernel regression*

In this section, one compares the estimators based on Hellinger distance and L^2 norm given respectively by equations (2) and (5). Figure 4 represents the output for $N = 100$ points simulated uniformly and independently in $\boldsymbol{\mathcal{X}}$. Figure 5 represents for different values of x_0 the mean (in plain line) and the standard deviation (in dashed line) of the estimation obtained from the two estimators of the kernel regression in red for L^2 norms and in blue for the Hellinger distance. For these four parameters, the two estimators give approximately the same estimations. Figure 6 shows the boxplot of the relative error for L^2 norms and the Hellinger distance. In this first example the estimator based on the Hellinger distance seems slightly better than the one based on the L^2 norm. The errors are very close for the norms and the mean but are more different for other quantities of interest. The variance of the error decreases very quickly for norms. Moreover, the errors are low for most quantities of interest except for 1% and 25% quantiles.

2.1.2. *Functional decomposition methods*

In this section, the four functional decompositions presented in 1.2, MMP with L^2 and Hellinger distances, AQM and CPCA are compared. The decompositions are applied to learning samples whose sizes are $N = 50$

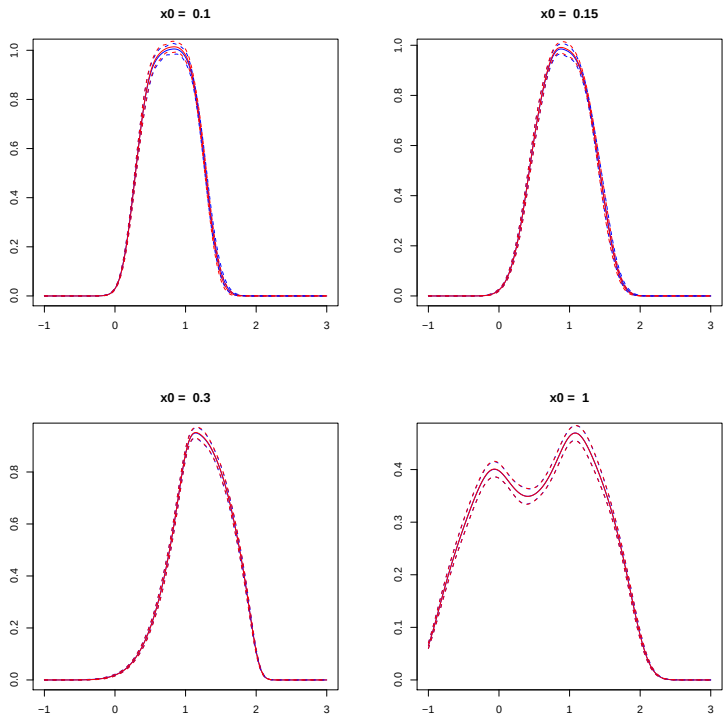

FIGURE 5. Toy example 1: curves in red (resp. blue) plain line represents the estimation of a density for a fixed x_0 from the first (resp. second) method and dashed red (resp. blue) line represent the standard deviation of the estimation obtained from 25 independent design of experiments. The blue and the red lines are superposed.

and 100 and for decomposition basis sizes ranging from 1 to 20. The relative errors between the learning sample functions and their approximations are computed for the 9 quantities of interest. Figure 7 represents, in logarithmic scale, these relative errors in function of the basis size for the four decompositions. First, the relative errors are low for all quantities of interest and especially for the norms, the mean and the higher quantiles. For the four decompositions, the relative error for norm decreases very evenly. The decrease is quite regular too for the mean, variance, 75% and 99% quantiles. The quality of approximation of the lower quantiles behaves more irregularly, as it sometimes increases with the basis size. CPCA method outperforms the three others for the three distances. This method seems better for all other quantities of interest except the 25% quantile, but the gap between the decompositions is smaller. The relative error for the learning sample of 100 densities is for most of the quantities higher than the relative error for the sample of 50 densities. For the same number of components q, it is indeed more difficult to approximate a higher number N of functions, so that the relative error increases with the size of the design of experiments for a constant number of components.

Figure 8 represents the L^2 relative error of MMP (in red) with $N = 200$ and the relative error of twenty independent basis (in black) containing probability functions chosen randomly in the learning sample $f_1, \ldots, f_N$. The MMP outperforms the random strategies. This result validates the way to choose a new curve among the sample of probability density functions, as this choice is better than randomly picking functions in the sample.

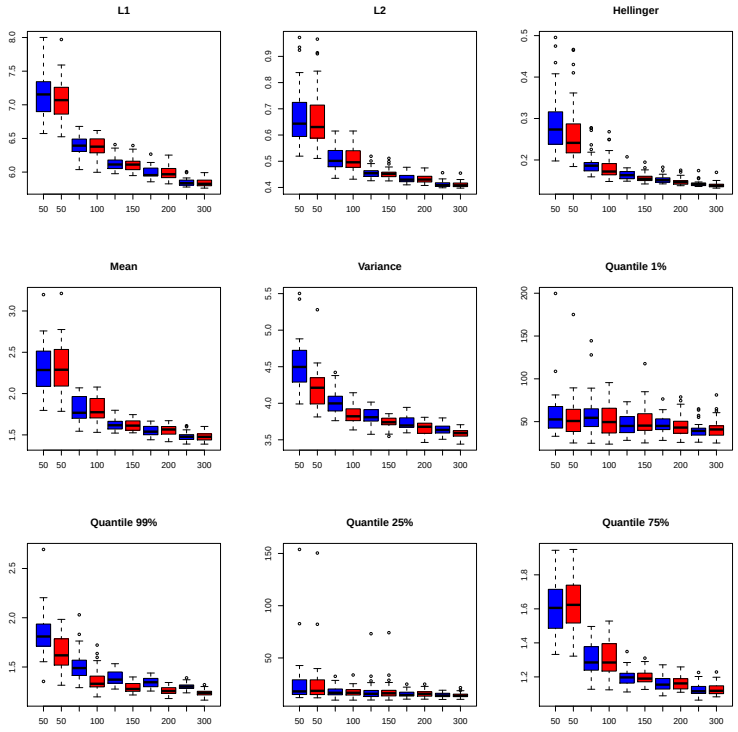

FIGURE 6. Toy example 1: boxplot of the errors for different sizes of N. Blue: estimator given by L^2 norm. Red: estimator given by the Hellinger distance. Results have been averaged with 25 independent experiments. Blue: estimator given by L^2 norm. Red: estimator given by the Hellinger distance.

2.2. **A five-dimensional example (Toy example 2)**

Let $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5) \in \boldsymbol{\mathcal{X}} = [0,1]^5$. A second toy example is defined as follows:

$$G(\mathbf{x}, N, U_1, U_2, B) = (x_1 + 2x_2 + U_1)\sin(3x_3 - 4x_4 + N) + U_2 + 10x_5 B + \sum_{i=1}^{5} i x_i$$

where

$$\begin{aligned} N &\sim \mathcal{N}(0,1) \\ U_1 &\sim \mathcal{U}([0,1]) \\ U_2 &\sim \mathcal{U}([1,2]) \\ B &\sim \mathcal{B}ern(1/2). \end{aligned}$$

Figure 9 represents the output for $N = 100$ points simulated uniformly and independently in $\boldsymbol{\mathcal{X}}$.

2.2.1. *Kernel regression*

In this section, the kernel regression is tested on this second toy example with isotropic (Figure 11) and anisotropic bandwidth (Figure 12). As in the first example, Figure 10 represents for different values of x_0 the

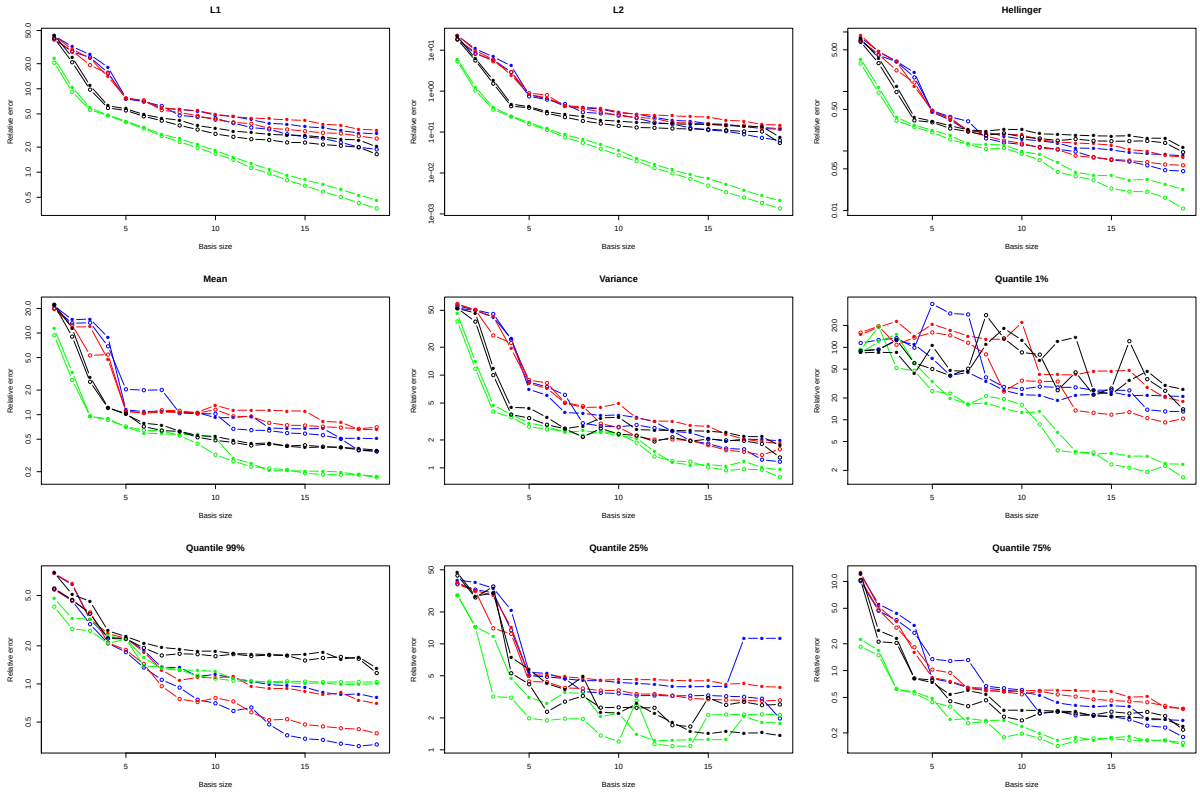

FIGURE 7. Toy example 1: comparison of the relative error for different quantities in function of the size of the basis q with a design of experiments of size $N = 50$ (circles) and 100 (filled circles). Blue: MMP decomposition given by L^2 norm. Red: MMP decomposition given by the Hellinger distance. Black: AQM decomposition. Green: CPCA decomposition.

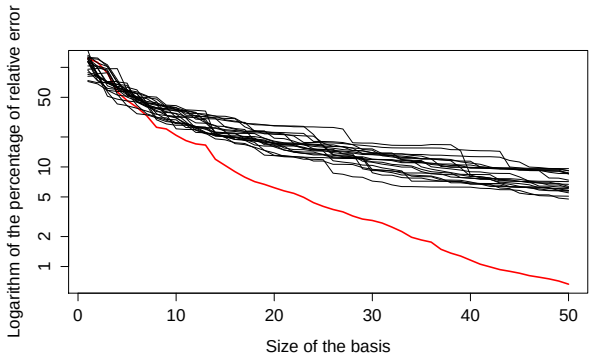

FIGURE 8. Toy example 1: comparison of the L^2 relative error obtained with MMP (in red) and a random choice in the design of experiments (in black).

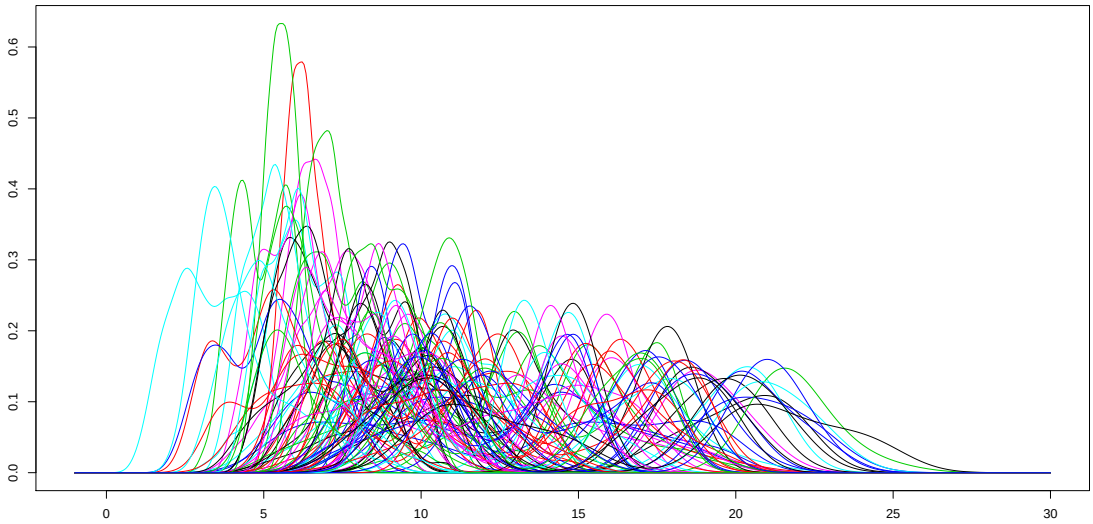

FIGURE 9. Toy example 2: representations of outputs for $N = 100$ different parameters.

true probability density function (blue dashed line) and the estimation obtained from the two estimators of the kernel regression (red plain line and orange dots line). In this example, the two estimators give different results. In red is represented the relative error for the estimator based on Hellinger distance, in blue for the estimator based on L^2 norm. The Hellinger estimator gives a better approximation of the quantities presented in Figure 11, except for the mean. Contrary to the case of the first toy example, the errors are quite high. The variance of the error on the norms, presented in Figure 11, is low but does not seem to decrease steadily. The use of an anisotropic bandwidth does not seem to improve the quality of the estimation, while it is much longer to compute. The errors are approximately the same with isotropic or anisotropic bandwidth. The greater precision brought by the anisotropic bandwidth is compensated by the difficulty to estimate it.

2.2.2. *Functional decomposition methods*

In this section, the four functional decompositions MMP with L^2 and Hellinger distances, AQM and CPCA are compared. The decompositions are applied to learning samples whose sizes are $N = 50$ and 100 and for decomposition basis sizes ranging from 1 to 20. The relative errors between the learning sample functions and their approximations are computed for the 9 quantities of interest. Figure 13 represents, in logarithmic scale, these relative errors in function of the basis size for the three decompositions. First, the relative errors are low for all quantities of interest, as in the previous example. The relative errors decrease is less even than in the one-dimensional case. The behavior of the errors on 1% and 99% quantiles is particularly unsteady. CPCA method gives better results than the other methods for L^1, L^2 norms, 25% and 75% quantiles. However, for all studied quantities, the AQM and CPCA method seem to give quite equivalent results. MMP method errors are slightly higher than AQM errors for most of the quantities.

3. INDUSTRIAL APPLICATIONS

In this section we apply the methods we developed in Section 1 to three industrial numerical codes of CEA, EDF, and IFPEN. We compute the same relatives errors as for the toy examples of Section 2.

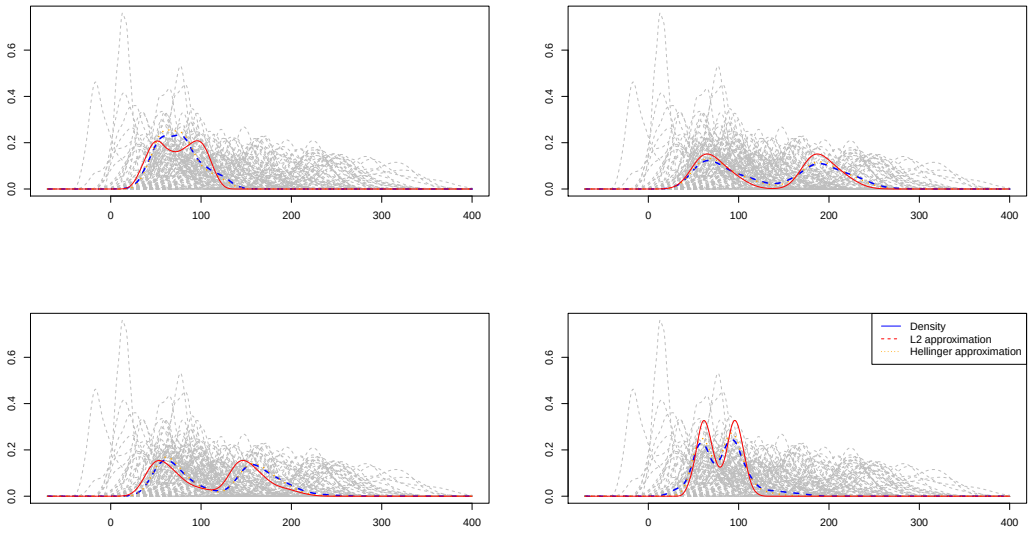

FIGURE 10. Toy example 2: curves in dashed gray lines are probability density functions, red plain line (resp. orange dots line) represents the estimation of a density for a fixed $\mathbf{x}_0$ from the first (resp. second) method and dashed blue line is the true probability density function. These estimations were made with h isotropic. The orange dotted line and the red line are superposed.

3.1. **CEA application: CASTEM test case**

3.1.1. *Code description*

In the framework of nuclear plant risk assessment studies, the evaluation of component reliability during accidental conditions is a major issue required for the safety case. Thermal-hydraulic (T-H) and thermal-mechanic (T-M) system codes model the behaviour of the considered component subjected to highly hypothetic accidental conditions. In the study that we consider here, the T-M code CASTEM takes as input 13 uncertain parameters, related to the initial plant conditions or to the safety system characteristics. Three of them are functional T-H parameters which depend on time: fluid temperature, flow rate and pressure. The other ten parameters are T-M scalar variables. For each set of parameters, CASTEM calculates the absolute mechanical strength of the component and the thermo-mechanical actual applied load. From these two elements, a safety margin (SM) is deduced.

The objective is to assess how these uncertain parameters can affect the code forecasts and more specifically the predicted safety margin. However, CASTEM code is too time expensive to be directly used to conduct uncertainty propagation studies or global sensitivity analysis based on sampling methods. To avoid the problem of huge calculation time, it can be useful to replace CASTEM code by a metamodel. One way to fit a metamodel on CASTEM could be to discretize the functional inputs and to consider the values of the discretization as scalar inputs of CASTEM code. Nevertheless, this solution is often intractable due to the high number of points in the discretization. To cope with this problem, in [13, 17] a method was proposed to treat implicitly these "uncontrollable" parameters functional parameters, while the other ten scalar parameters are considered as "controllable". CASTEM output is then a random variable conditionally to "controllable" parameters.

A latin hypercube sampling method [18] is used to build a learning sample of 500 points in dimension 10. For each set of controllable parameters, CASTEM has been run 400 times with different uncontrollable functional

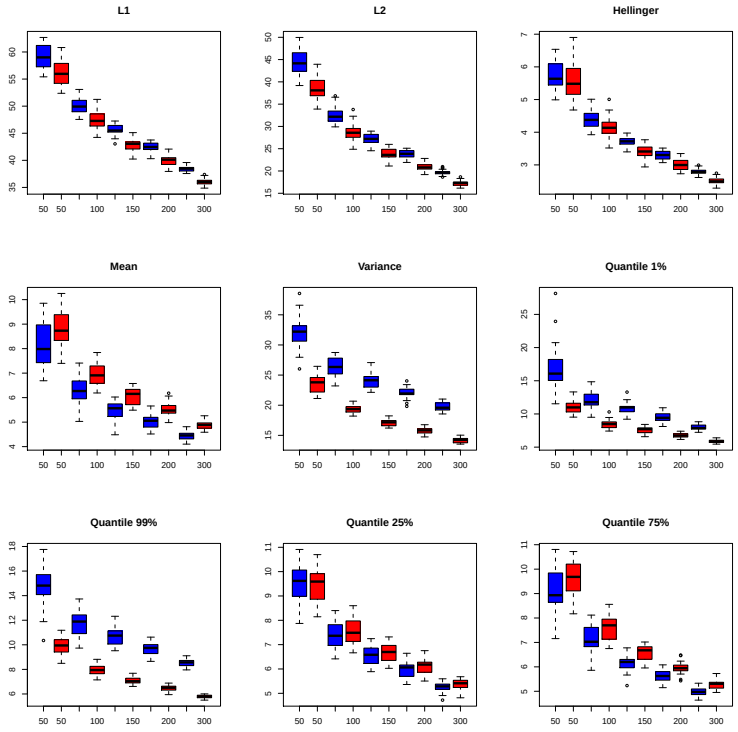

FIGURE 11. Toy example 2: boxplot of the errors for different sizes of N. Left: estimator given by L^2 norm. Right: estimator given by the Hellinger distance. Results have been averaged with 25 independent experiments. These estimations were made with h isotropic. Blue: estimator given by L^2 norm. Red: estimator given by the Hellinger distance.

parameters, randomly chosen in an available database. The probability density function f_i $(i = 1, .., 500)$ of the safety margin is computed by kernel estimation with the 400 outputs of CASTEM for each set of parameters. A few examples of the obtained probability density functions are represented on Figure 14. In the following, the two kernel regression metamodels, then MMP, AQM and CPCA decomposition methods are applied on CASTEM test case.

3.1.2. *Kernel regression*

In this section, the kernel regression method is applied with isotropic and anisotropic bandwidths. Figure 15 represents for four different values of x_0 the true probability density function (blue dashed line) and the estimation obtained from the two estimators of the kernel regression (L^2 estimator in red plain line and Hellinger estimator in orange dotted line), with isotropic bandwidth. One can see that the estimation by the L^2 estimator of the four probability density functions is very far from the real function. In particular, at the bottom left of the figure, the mean of the predicted probability function in red is very far from the mean of the real one in dashed blue. On these four probability density functions, the Hellinger estimator gives much better results than the L^2 one.

To verify this first graphical analysis, the Leave-One-Out method has been used to assess the efficiency of the kernel regression-based metamodel. The bandwidth of the kernel regression is estimated thanks to all probability density functions except the i^{th} one. The function f_i is estimated with the corresponding metamodel and the relative errors on the quantities of interest are computed between f_i and its estimation. This process is

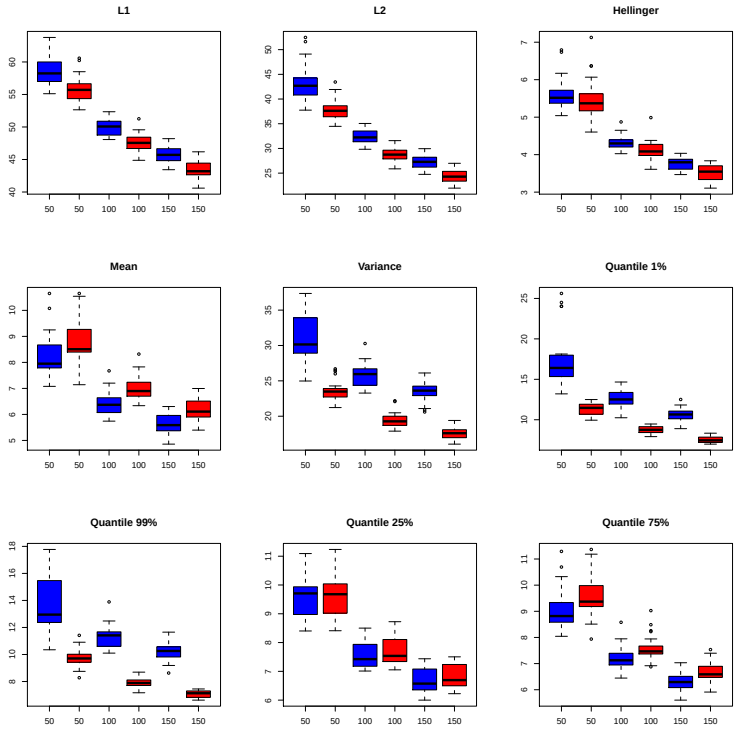

FIGURE 12. Toy example 2: boxplot of the errors for different sizes of N. Left: estimator given by L^2 norm. Right: estimator given by the Hellinger distance. Results have been averaged with 25 independent experiments. These estimations were made with h anisotropic. Blue: estimator given by L^2 norm. Red: estimator given by the Hellinger distance.

repeated for each probability density function in the dataset and all the computed relative errors are averaged. The mean relative errors of the estimators based on Hellinger distance and L^2 norm are given in Table 1 and Table 2 for the different quantities of interest respectively for an isotropic and anisotropic bandwidths. The two estimators perform poorly on this dataset. The errors for L^1 and L^2 norms are particularly high. The Leave-One-Out validation confirms that the kernel regression method is not adapted to CASTEM test case. This can be explained by the important influence of the controllable T-M parameters compared to the one of the uncontrollable T-H parameters.

3.1.3. *Functional decomposition methods*

In this section, the four functional decompositions MMP with L^2 and Hellinger distances, AQM and CPCA are compared. Figure 16 represents, in logarithmic scale, the relative errors on the different norms, modes and quantiles versus the basis size (from 1 to 20). First, the relative errors are low for all quantities of interest, except for the variance. For the variance, the errors are over 10% with a decomposition basis with 20 functions. The errors for small bases are high but decrease very quickly. The decrease is especially quick and even for errors on L^1, L^2 and Hellinger distances. For modes and quantiles, the errors of the four methods do not decrease steadily for all quantities of interest, and especially for small basis sizes. CPCA clearly outperforms other methods for the L^1, L^2 and Hellinger distances. For other quantities of interest, it gives good results. For the variance, 1% and 99% quantiles, AQM method performs poorly compared to the three others. The errors

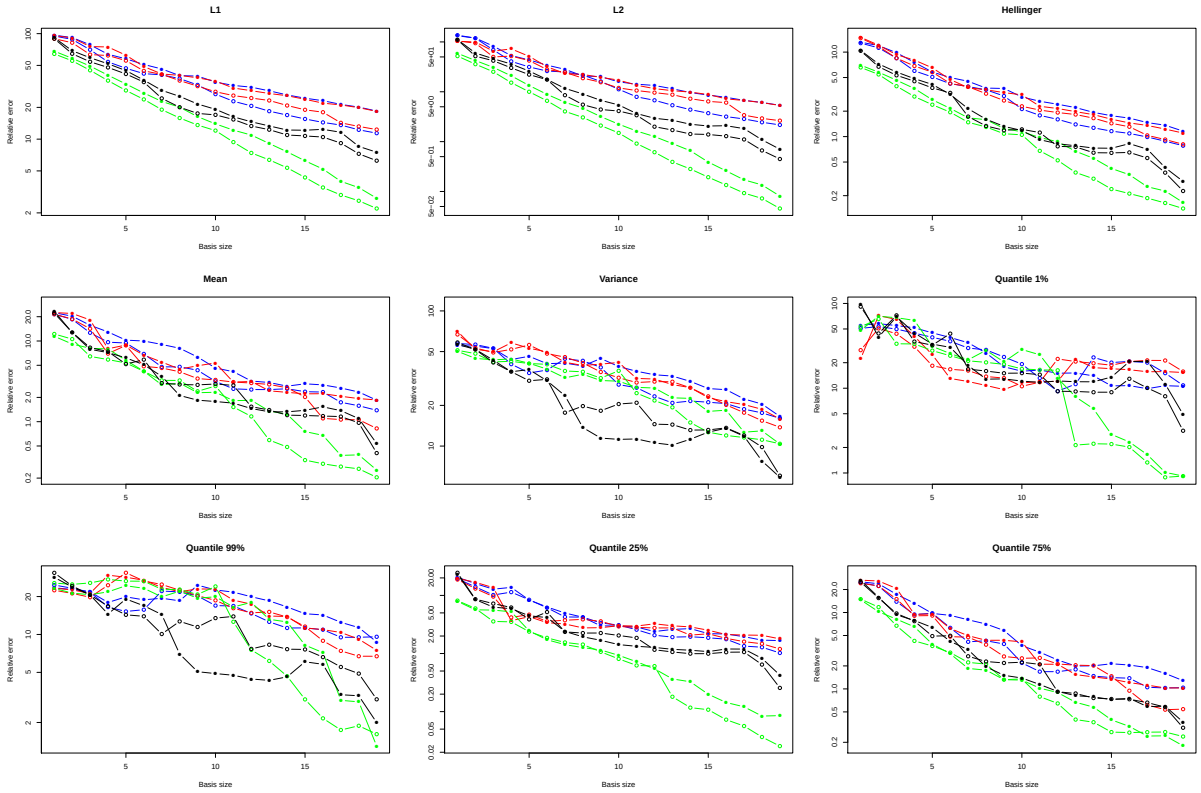

FIGURE 13. Toy example 2: comparison of the relative error for different quantities in function of the size of the basis q with a design of experiments of size $N = 50$ (circles) and 100 (filled circles). Blue: MMP decomposition given by L^2 norm. Red: MMP decomposition given by the Hellinger distance. Black: AQM decomposition. Green: CPCA decomposition.

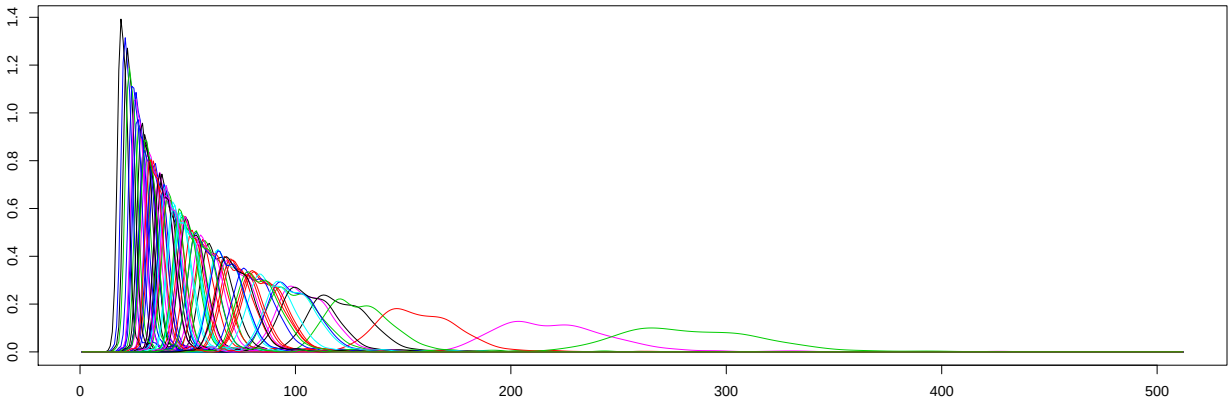

FIGURE 14. CASTEM test case: representations of outputs for $N = 75$ different parameters.

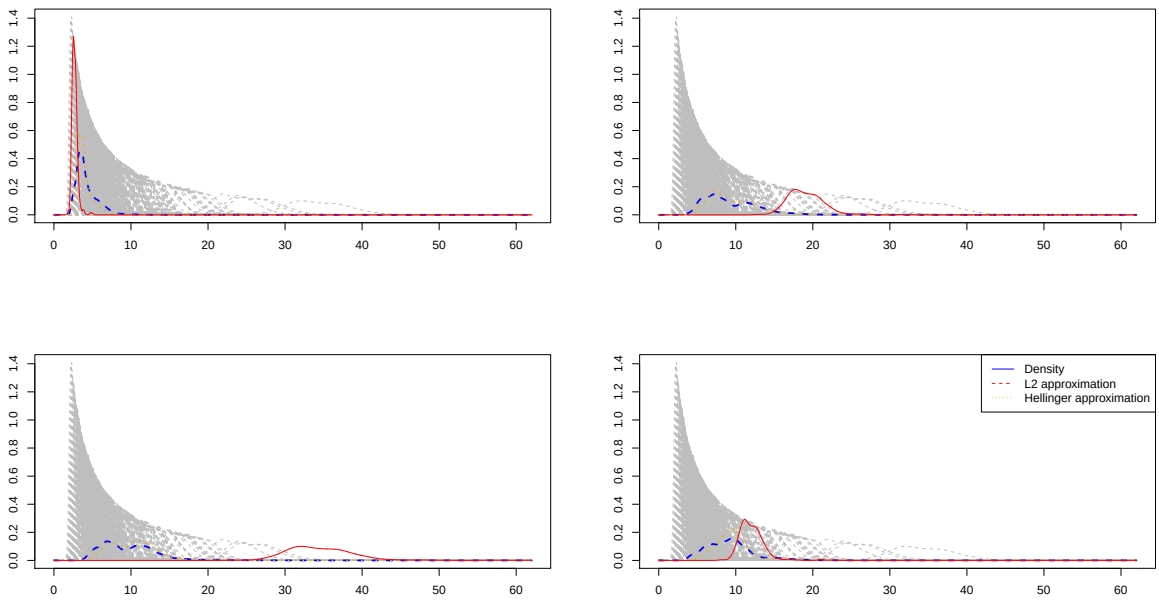

FIGURE 15. CASTEM test case: curves in dashed gray lines are probability density functions, red plain line represents the estimation of a density for a fixed x_0 and dashed blue line is the true probability density function.

	Hellinger distance estimator	L^2 norm estimator
L^1 norm	114.2%	114.9%
L^2 norm	167.7%	171.9%
Hellinger distance	17.0%	17.0%
Mean	22.2%	22.4%
Variance	86.1%	86.9%
1% quantile	72.0%	74.3%
99% quantile	39.7%	40.9%
25% quantile	30.9%	31.5%
75% quantile	23.7%	23.9%

TABLE 1. CASTEM test case: the mean relative errors on the quantities of interest computed by the Leave-One-Out method for the kernel regression estimators based on the Hellinger distance and L^2 norm with an isotropic bandwidth.

of AQM and CPCA methods seem more stable than for MMP method for most of the quantities of interest, especially for modes and quantiles. Overall, the results are quite good for the four methods.

3.2. **EDF application: VME test case**

3.2.1. *Code description*

To optimize the whole life cost of its nuclear fleet, EDF has developed an asset management methodology [9]. A part of this methodology deals with exceptional maintenance tasks strategies. To help the decision maker

	Hellinger distance estimator	L^2 norm estimator
L^1 norm	94.3%	94.5%
L^2 norm	108.0%	108.6%
Hellinger distance	14.3%	14.3%
Mean	15.4%	15.6%
Variance	71.5%	71.9%
1% quantile	42.7%	43.3%
99% quantile	23.0%	23.8%
25% quantile	20.4%	20.4%
75% quantile	15.9%	16.3%

TABLE 2. CASTEM test case: the mean relative errors on the quantities of interest computed by the Leave-One-Out method for the kernel regression estimators based on the Hellinger distance and L^2 norm with an anisotropic bandwidth.

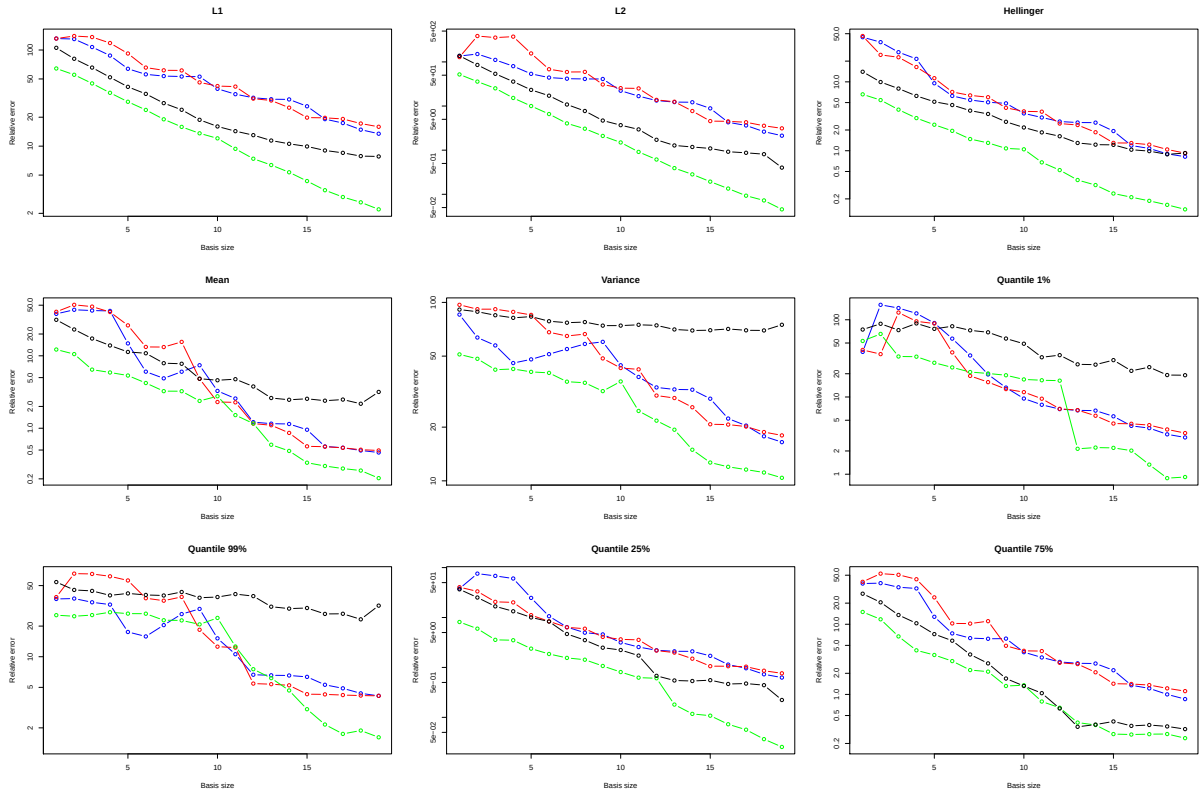

FIGURE 16. CASTEM test case: comparison of the relative error for the 9 quantities in function of the size of the basis q. Blue: MMP decomposition given by L^2 norm. Red: MMP decomposition given by the Hellinger distance. Black: AQM decomposition. Green: CPCA decomposition.

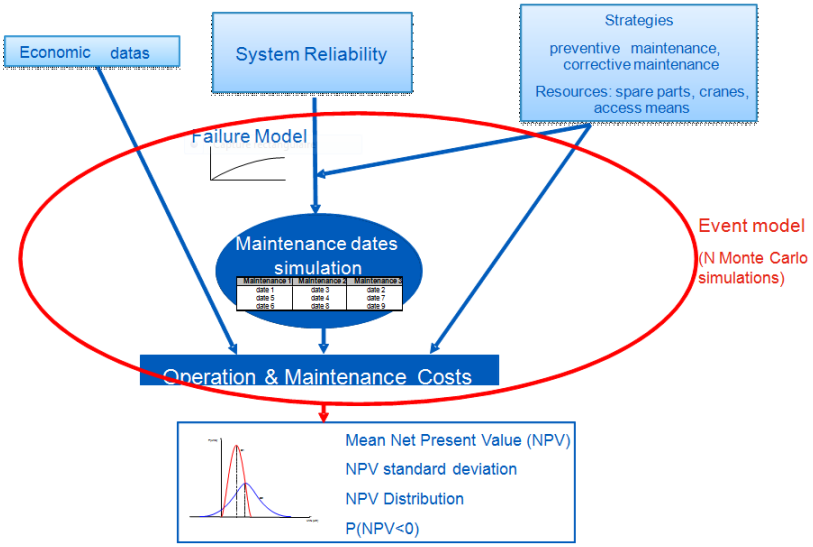

FIGURE 17. VME: EDF valorization of maintenance strategies tool description.

choose the best strategy (how many times do we need to carry out exceptional tasks, when,...?), EDF has developed a dedicated tool called VME (described in Figure 17) based on Monte-Carlo simulation to compute many technical economic indicators among which the density function of the Net Present Value (called VAN for "Valeur Actuelle Nette") is the most relevant.

This tool leads to an important simulation time and requires an important amount of input data that are surrounded with uncertainties:

- Reliability data: generally there is not enough (or sometimes not any) feedback data to precisely evaluate reliability model parameters;
- Economic data: economic indicators and duration of maintenance tasks remain on several hypothesis that can be modified;
- Other data: uncertainty on operating times of power plants, maintenance tasks dates, etc.

In our work, we study a particular scenario simulation with VME concerning the life cycle management of four large transformers, adapted from the EPRI study made for CENG [15]. The uncertainty on the inputs are represented by four Weibull distributions which are summarized in Table 3. Define for $i = 1, \dots, 4$,

$$D_i = \{\mathbf{x} = (x_1, x_2, x_3) \in \mathbb{R}^3 : \ \mathbb{E}[\mathcal{W}eibull(x_1, x_2, x_3)] \in [\mathbb{E}[W_i] \pm 0.2\mathbb{E}[W_i]]\},$$

and denote sh_i, sc_i, loc_i respectively the shape, the scale and the location of W_i. For $i = 1, \dots, 4$ let $x_{i,1}, x_{i,2}, x_{i,3}, x_{i,4}$ be parameters simulated uniformly on D_i. Note that W_2 and W_4 have the same probability distribution but sh_2, sc_2, loc_2 and sh_4, sc_4, loc_4 are considered as different parameters. That means there are three parameters for each Weibull distribution. Moreover, there are four other parameters which represent some costs and two other parameters which represent durations of services. All these 18 parameters are summarized in Table 4.

In this case, we are particulary interested by the VAN indicator. It is defined by

$$VAN \quad = \quad \mathbf{P}(G(\mathbf{x}, \mathbf{W}) \leq 0),$$

where $\mathbf{W} = (W_1, \dots, W_4)$ and $\mathbf{x}$ the eighteen fixed parameters.

Uncertainty inputs	distribution	scale	shape	location
W_1	$\mathcal{W}eibull$	1/0.021	2.6	1
W_2	$\mathcal{W}eibull$	100	3.8	20
W_3	$\mathcal{W}eibull$	100	4	24
W_4	$\mathcal{W}eibull$	100	3.8	20

TABLE 3. VME: uncertainty inputs used in the model. Each distribution depends of three parameters: the shape, the scale and the location.

Uncertainty inputs	distribution	support
(sh_1, sc_1, loc_1)	$\mathcal{U}niform$	D_1
(sh_2, sc_2, loc_2)	$\mathcal{U}niform$	D_2
(sh_3, sc_3, loc_3)	$\mathcal{U}niform$	D_3
(sh_4, sc_4, loc_4)	$\mathcal{U}niform$	D_4
Cost1	$\mathcal{U}niform$	[796 , 1194]
Cost2	$\mathcal{U}niform$	[49.6 , 74.4]
Cost3	$\mathcal{U}niform$	[5.6 , 8.4]
Cost4	$\mathcal{U}niform$	[4 , 8]
Duration1	$\mathcal{U}niform$	[0.8 , 1.2]
Duration2	$\mathcal{U}niform$	[0.8 , 1.2]

TABLE 4. VME: distribution of the eighteen parameters.

Figure 18 represents 75 outputs from the VME model. The kernel regression is applied to designs of experiments of increasing sizes such that smaller designs are included in larger ones. A test sample of 500 points is used to assess the efficiency of the regression.

3.2.2. *Kernel regression*

Figure 19 represents for different values of x_0 the true probability density function (blue dashed line) and the estimation obtained from the two estimators of the kernel regression (red plain line and orange dots line). In this section, the kernel regression is tested on this real case study with isotropic and anisotropic bandwidth (Figure 20).

In red is represented the relative error for the estimator based on the Hellinger distance, in blue for the estimator based on the L^2 norm. The estimation seems equivalent for the two estimators. The results are equivalent for anisotropic and isotropic bandwidth. For the anisotropic bandwidth, the estimation is less stable and the estimation of H becomes much more time-consuming.

The relative error in norm is smaller for VME (with isotropic and anisotropic bandwidth) than for the toy example 2, whereas the dimension is higher. That shows that the quality of estimation depends not only on the dimension but also on the regularity of the model, if the bandwith is sufficiently well estimated.

Bad results are found for the variance, 99% and 1% quantiles. The relative error for the mean is also very high. The real mean is quite close to zero, so that in the relative error computation, the numerator is divided by a quantity close to zero.

Figure 21 represents the estimation of the $\mathbf{P}(VAN < 0)$ for isotropic and anisotropic bandwidths. In this case, choosing an anisotropic bandwidth does not bring information.

3.2.3. *Functional decomposition methods*

In this section, the four functional decompositions MMP with L^2 and Hellinger distances, AQM and CPCA are compared. Figure 22 represents, in logarithmic scale, the relative errors on the different norms, modes and quantiles versus the basis size (from 1 to 20). First, the relative errors are low for all quantities of interest,

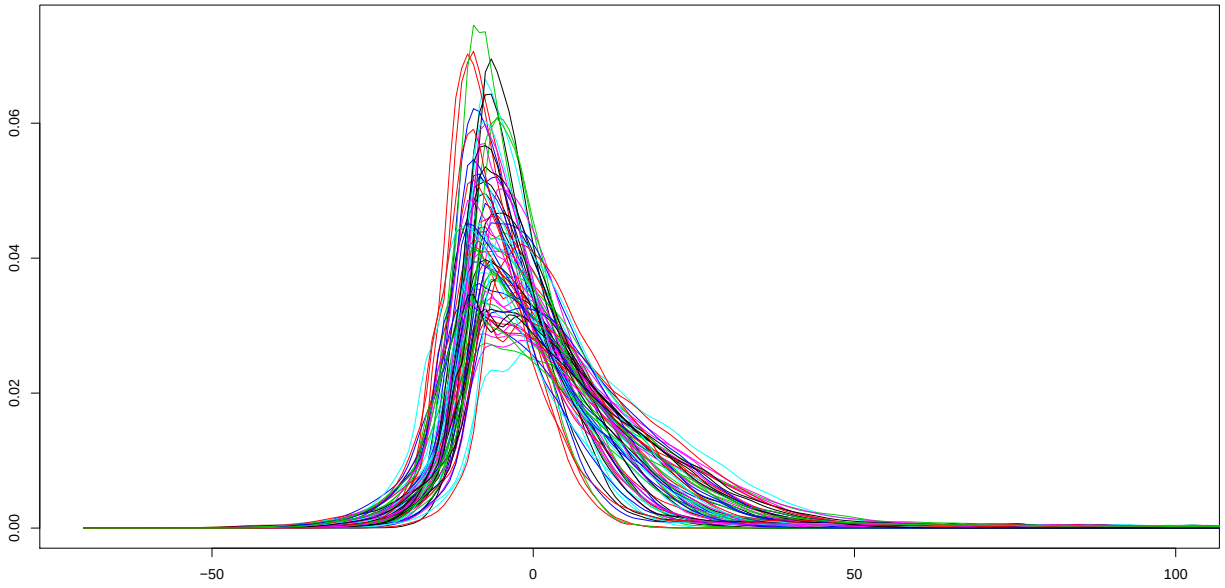

FIGURE 18. VME: representations of outputs for $N = 75$ different parameters.

except for the mean and variance. For the variance, the errors are over 20% with a decomposition basis with 20 functions. The very high errors on the mean can be explained by the fact that the mean to be estimated is close to zero for most of densities in the VME case. The errors on mean and variance have moreover a very chaotic behavior. The errors on distances decrease very quickly. For modes and quantiles, the errors of the four methods do not decrease steadily for all quantities of interest. CPCA outperforms other methods for the L^1, L^2, Hellinger distances, the variance and the 99% and 25% quantiles. The performances of AQM and MMP are very close for modes and quantiles. CPCA seems more stable than MMP and AQM on most of the quantities of interest.

In Figure 23, the errors on the probability for the VAN to be negative $P(VAN < 0)$ for the different decomposition methods are represented in function of the basis size. The errors are quite low and decrease quickly. The CPCA method outperforms the others for both learning sets sizes. The other three methods give similar results.

3.3. **IFPEN application: GIBBS test case**

3.3.1. *Molecular Modelling*

At IFP Energies Nouvelles, the researchers of the Department of Thermodynamics and Molecular Modelling elaborate models for the structure of molecules such as hydrocarbons and alcohols. In the following numerical study we consider a system of *pentane* molecules at a *reduced temperature* of 0.75. Pentane is an *alkane* with five carbon atoms (C_5H_{12}, *cf.* Figure 24). Here we use the Anisotropic United Atoms model (AUA) [29] to represent the molecular structure of pentane. The carbon (C) and hydrogen (H) atoms in each of the two terminal *methyls* (CH_3) and each of the three *methylene* bridges (CH_2) are treated as a single *interaction center*. To each of these two types of interaction centers correspond three parameters of the model: an energy parameter (ε_{CH_3} and ε_{CH_2}), a size parameter (σ_{CH_3} and σ_{CH_2}), and a displacement parameter (δ_{CH_3} and δ_{CH_2}). These six parameters

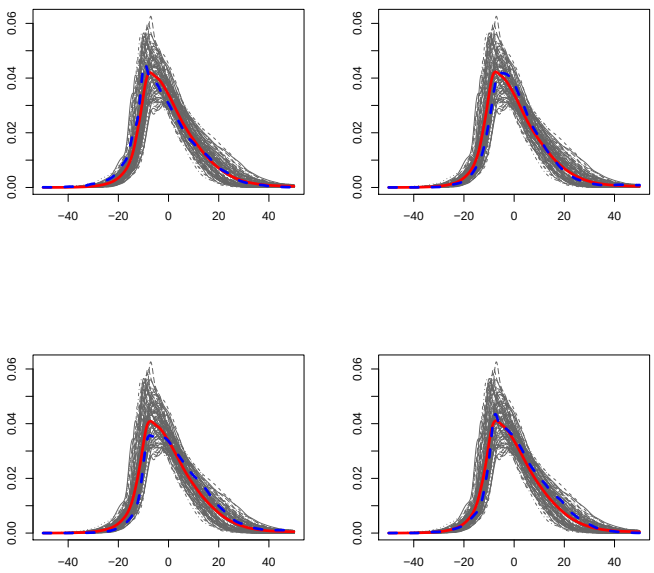

FIGURE 19. VME: curves in dashed gray lines are probability density functions, red plain line represents the estimation of a density for a fixed x_0 and dashed blue line is the true probability density function for $N = 100$.

will be the components of the set of input parameters of the numerical codes under consideration:

$$\mathbf{x} = (\varepsilon_{\mathrm{CH_3}}, \varepsilon_{\mathrm{CH_2}}, \sigma_{\mathrm{CH_3}}, \sigma_{\mathrm{CH_2}}, \delta_{\mathrm{CH_3}}, \delta_{\mathrm{CH_2}}) \in \mathbb{R}^6.$$

Here the matter of interest is the prediction of two macroscopic properties of a pentane system in *chemical equilibrium*: the *volumetric mass density* $\rho_{\mathrm{liq}}^{\mathrm{eq}}$ of the liquid phase and the *vapor pressure* $P_{\mathrm{gas}}^{\mathrm{eq}}$ (pressure of the gas phase). The algorithm carried out to numerically predict $\rho_{\mathrm{liq}}^{\mathrm{eq}}$ and $P_{\mathrm{gas}}^{\mathrm{eq}}$ – for a given set of parameters $\mathbf{x}$ – relies on the *fundamental postulate of statistical mechanics*: an isolated system in equilibrium is found with equal probability in each of its accessible *microstates*. Thus, $\rho_{\mathrm{liq}}^{\mathrm{eq}}$ (respectively $P_{\mathrm{gas}}^{\mathrm{eq}}$) can be approximated by the average of the values of the volumetric mass density of the liquid phase ρ_{liq} (respectively the vapor pressure P_{gas}) of a large number of accessible microstates of the system. Therefore the algorithm is essentially a loop, each step consisting of the following two instructions:

(1) randomly generate an accessible microstate of the system (characterized by the positions of the molecules and the volume of the system),
(2) compute the value of ρ_{liq} (respectively P_{gas}) given by the model for this particular microstate.

Let us denote $G_{\rho_{\mathrm{liq}}}(\mathbf{x})$ (respectively $G_{P_{\mathrm{gas}}}(\mathbf{x})$) the probability density function of the random value ρ_{liq} (respectively P_{gas}) computed at each loop step. The two instructions above are iterated a hundred million times. After the execution of the loop, the arithmetical mean of these millions of values is computed – it is a Monte Carlo approximation of the integral of $G_{\rho_{\mathrm{liq}}}(\mathbf{x})$ (respectively $G_{P_{\mathrm{gas}}}(\mathbf{x})$) – and is returned by the algorithm as the prediction of ρ_{liq} (respectively P_{gas}.) The whole algorithm is carried out in about twenty-four hours on a supercomputer (for each set $\mathbf{x}$ of input parameters.)

In order to circumvent this time-consuming process, we want to build a metamodel of the numerical codes

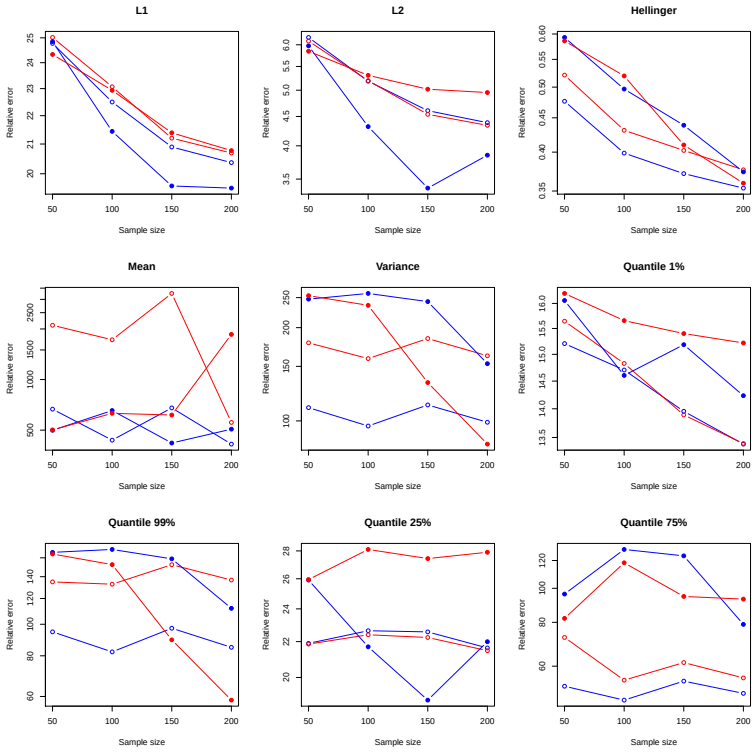

FIGURE 20. VME: comparison of the relative error for the two estimators with an isotropic and anisotropic (plain cicrle) bandwidth. Blue: estimators given by the L^2 norms. Red : estimator given by the Hellinger distance

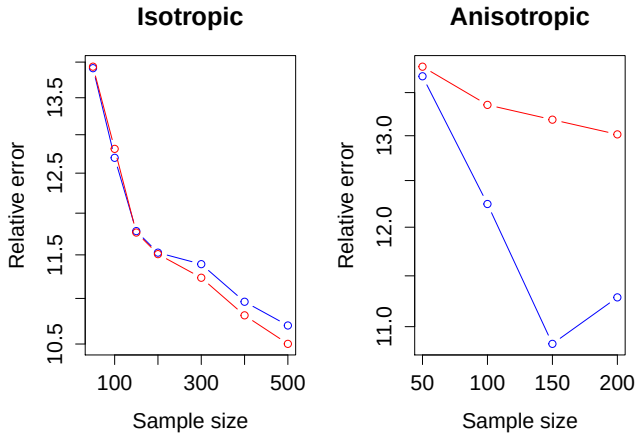

FIGURE 21. VME: comparison of the relative error for the $\mathbf{P}(VAN < 0)$. Left: isotropic bandwidth. Right: anisotropic bandwidth. Blue: estimators given by the L^2 norms. Red : estimator given by the Hellinger distance

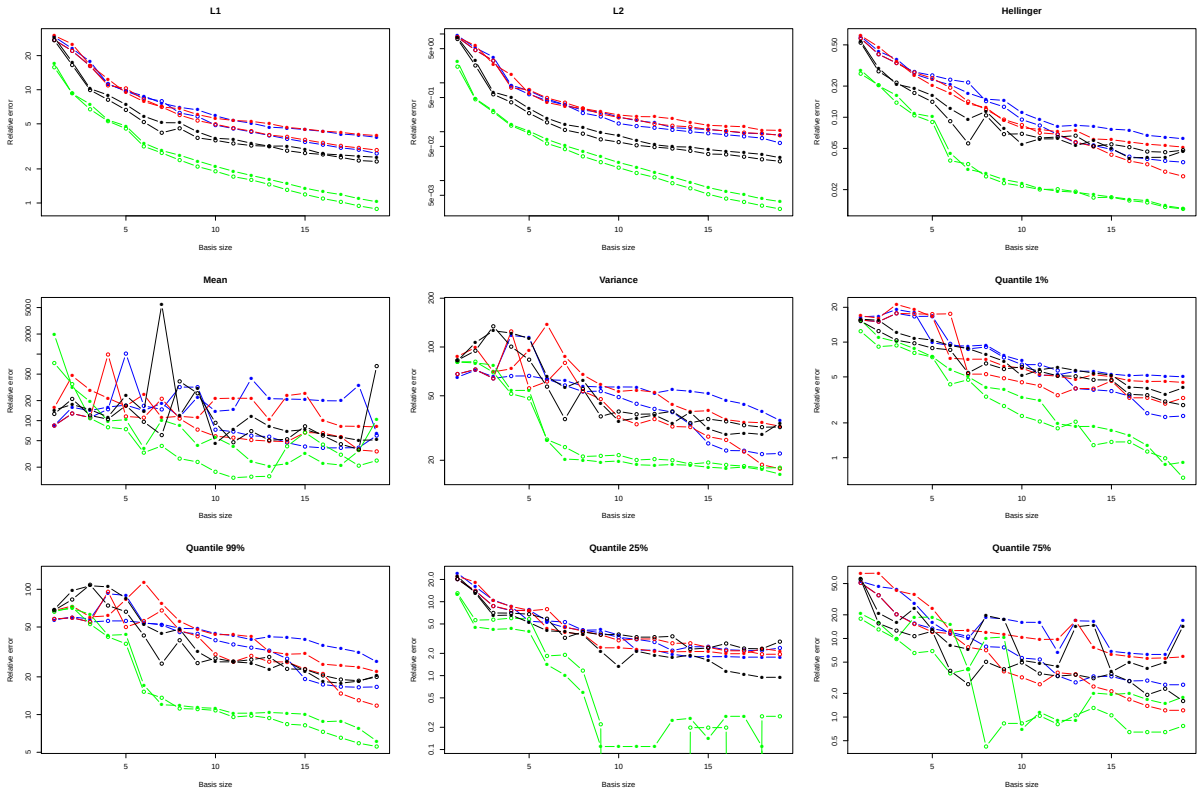

FIGURE 22. VME: comparison of the relative error for the 9 quantities in function of the size of the basis q. Blue: MMP decomposition given by L^2 norm with a learning set size of $N = 50$ (circles) and 100 (filled circles). Red: MMP decomposition given by the Hellinger distance. Black: AQM decomposition. Green: CPCA decomposition.

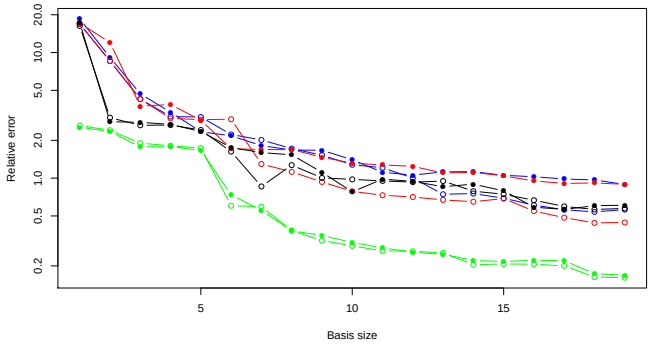

FIGURE 23. VME: comparison of the relative error for $P(VAN < 0)$ in function of the size of the basis q. Blue: MMP decomposition given by L^2 norm with a learning set size of $N = 50$ (circles) and 100 (filled circles). Red: MMP decomposition given by the Hellinger distance. Black: AQM decomposition. Green: CPCA decomposition.

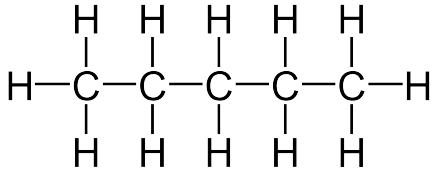

FIGURE 24. The structural formula of pentane. (Wikipedia)

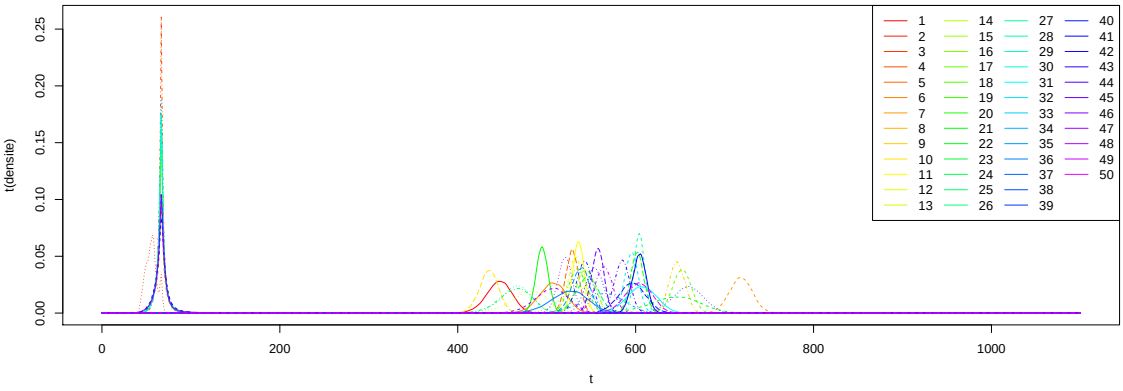

FIGURE 25. GIBBS test case: distributions of the volumetric mass density for different sets of input parameters.

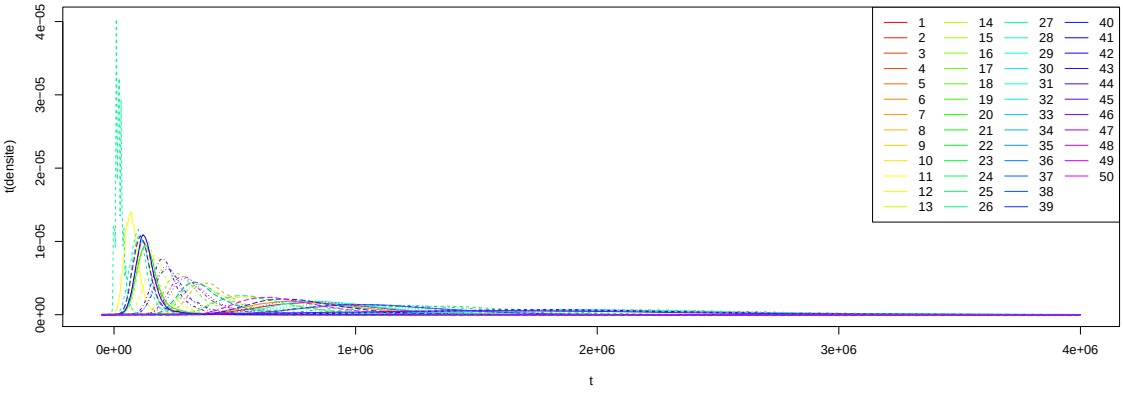

FIGURE 26. GIBBS test case: distributions of the vapor pressure for different sets of input parameters.

mapping the input parameters $\mathbf{x}$ to the probability density functions $G_{P_{\text{gas}}}(\mathbf{x})$ and $G_{\rho_{\text{liq}}}(\mathbf{x})$ of the random outputs. We had at our disposal a sample of size $N = 50$. Figure 25 and Figure 26 show the corresponding 50 probability functions for volumetric mass density and vapor pressure respectively.

L1	L2	Hellinger
151.1%	375.9%	7.089%
Mean	Variance	Quantile 1%
50.77%	98.07%	513.8%
Quantile 99%	Quantile 25%	Quantile 75%
30.44%	198.4%	41.52%

TABLE 5. GIBBS test case: kernel regression relative errors of the L^2 estimator for the volumetric mass density.

L1	L2	Hellinger
123.7%	254.1%	0.04878%
Mean	Variance	Quantile 1%
69.86%	80.72%	667.63%
Quantile 99%	Quantile 25%	Quantile 75%
74.82%	156.1%	73.5%

TABLE 6. GIBBS test case: KR relative errors of the L^2 estimator for the vapor pressure.

3.3.2. *Kernel regression*

We applied the kernel regression method on the numerical codes providing the volumetric mass density of the liquid phase and the vapor pressure. The results provided by the L^2 estimator with isotropic bandwidth are provided in Table 5 for the volumetric mass density and in Table 6 for the vapor pressure. The errors have been computed by the Leave-One-Out methodology as in Section 3.1.2. We can see that most of the results are disappointing: all relative errors are higher than 30%, excepted – surprisingly – the relative Hellinger error of the approximation. The latter amounts to nearly 7% for the volumetric mass density, and only nearly 0.05% for the vapor pressure. The errors for quantile 1% and quantile 25% are particularly high. The Hellinger estimator give almost the same results: the difference between the error for the L^2 estimator and the error for the Hellinger estimator is lower than 10^{-5} in every case. That is why they are not charted here.

In Figure 27 and Figure 28 we plot the probability density functions of the volumetric mass density and the vapor pressure with their kernel regression approximations, for arbitrarily chosen sets of input parameters. We can see how poorly the densities are approximated.

3.3.3. *Functional decomposition methods*

In this section, the four functional decompositions MMP with L^2 and Hellinger distances, AQM and CPCA are compared. Figure 29 represents, in logarithmic scale, the relative errors on the different norms, modes and quantiles versus the basis size (from 1 to 20) for the volumetric mass density of the liquid phase. The error curves are globally decreasing excepted the one corresponding to the variance relative error for AQM. Moreover, the relative error on variance is high even for basis with 20 functions and especially for AQM. The behavior of the AQM approximation is more erratic for the 99% quantile. Both MMP methods perform worse on the 25%

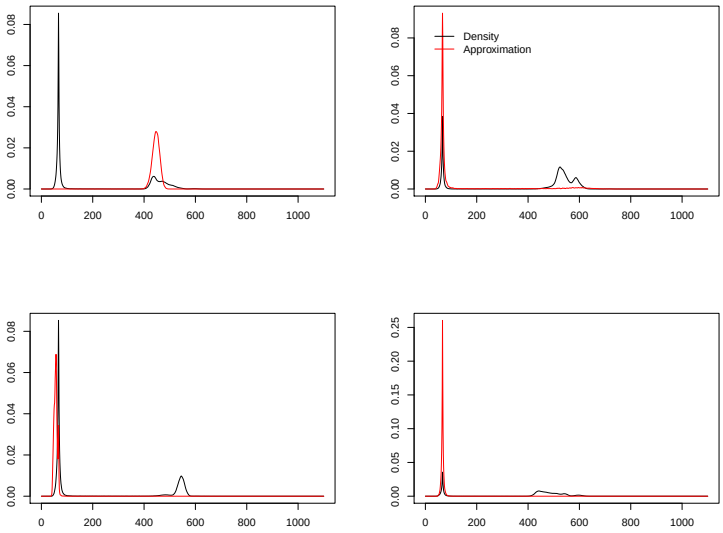

FIGURE 27. GIBBS test case: distribution of the volumetric mass density and its KR approximation for the sets of input parameters 1, 2, 3 and 4.

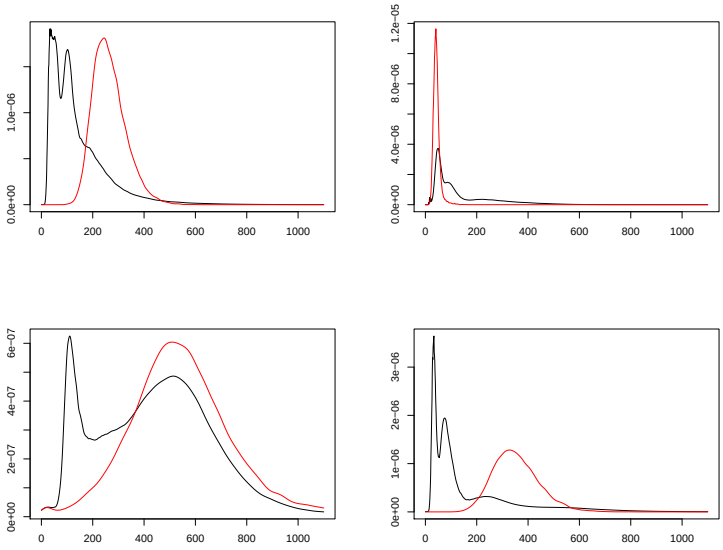

FIGURE 28. GIBBS test case: distribution of the vapor pressure and its KR approximation for the sets of input parameters 31, 32, 33 and 34.

and 75% quantiles than other methods. Other relative errors are low. CPCA gives better results for L^1, L^2, Hellinger distances and the 25% quantile. For 25% and 75% quantiles, errors on both MMP bases are higher than for others methods.

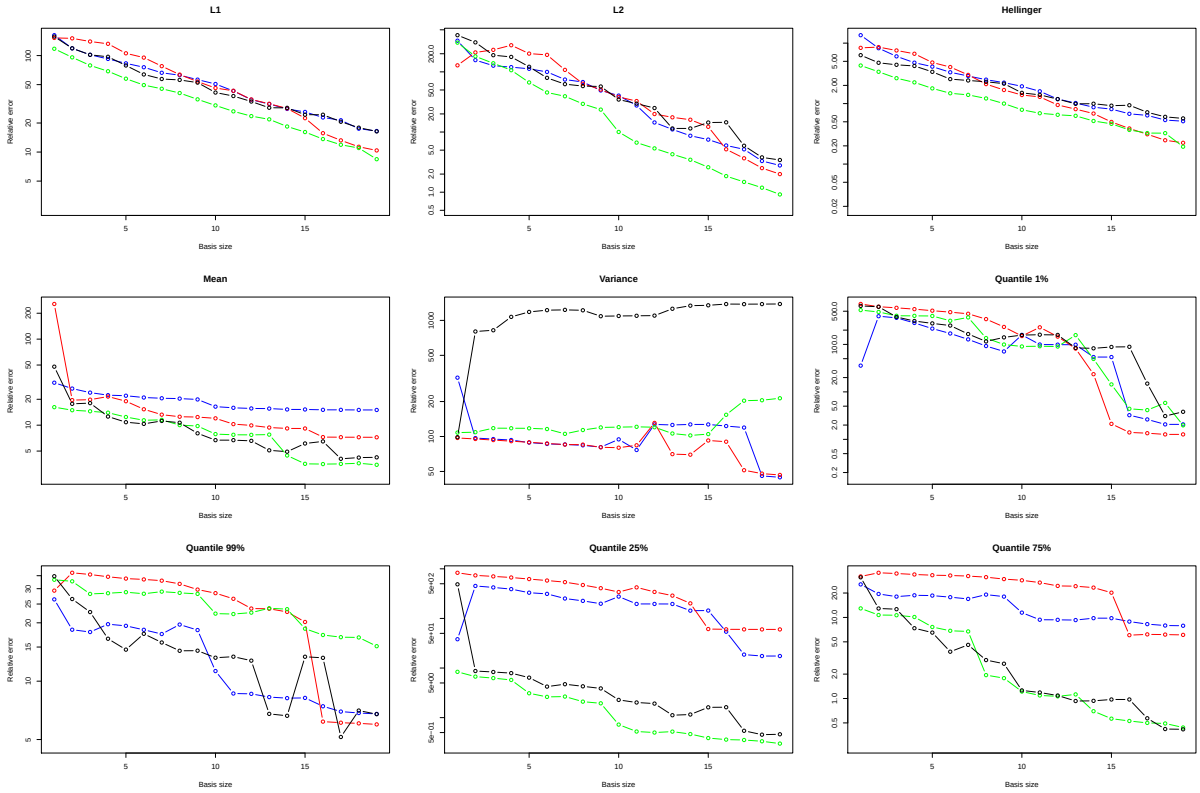

FIGURE 29. GIBBS test case: comparison of the relative errors on different quantities for the volumetric mass density in function of the size of the basis q. Blue: MMP decomposition given by L^2 norm. Red: MMP decomposition given by the Hellinger distance. Black: AQM decomposition. Green: CPCA decomposition.

The results for the vapor pressure are presented in Figure 30. The errors on all quantities of interest are more steady than for the volumetric mass. In this case, CPCA and AQM have very different behavior compared to both MMP methods. For all quantities of interest except L^1, CPCA and AQM errors have a plateau for higher basis sizes. For higher basis sizes, MMP method performs better than AQM and CPCA for all studied quantities, except the 25% quantile. Thus, contrary to what happens in the other presented numerical examples, CPCA method does not perform better than AQM and MMP for distances. For MMP decompositions, the errors on the mean and quantiles display a jump for basis sizes of 12 or 13.

4. CONCLUSION

The aim of this work was to build a metamodel for code outputs which are probability density functions. A first idea to design such a model was to build a convex combination of the sample probability density functions. To this mean, we first adapted the well-known kernel regression developed by [20] and [30]. Second, we proposed to approximate the sample probability density functions on a functional basis in order to reduce the problem dimension. In this way, the probability density functions are characterized by their coefficients on the basis, so that the problem becomes finite-dimensional. Two methods have been proposed to build a functional basis. The first method is adapted from Magic Points Method [16] and builds iteratively the basis by adding the sample function that maximizes the approximation error on the previous basis. The second one, called Alternate

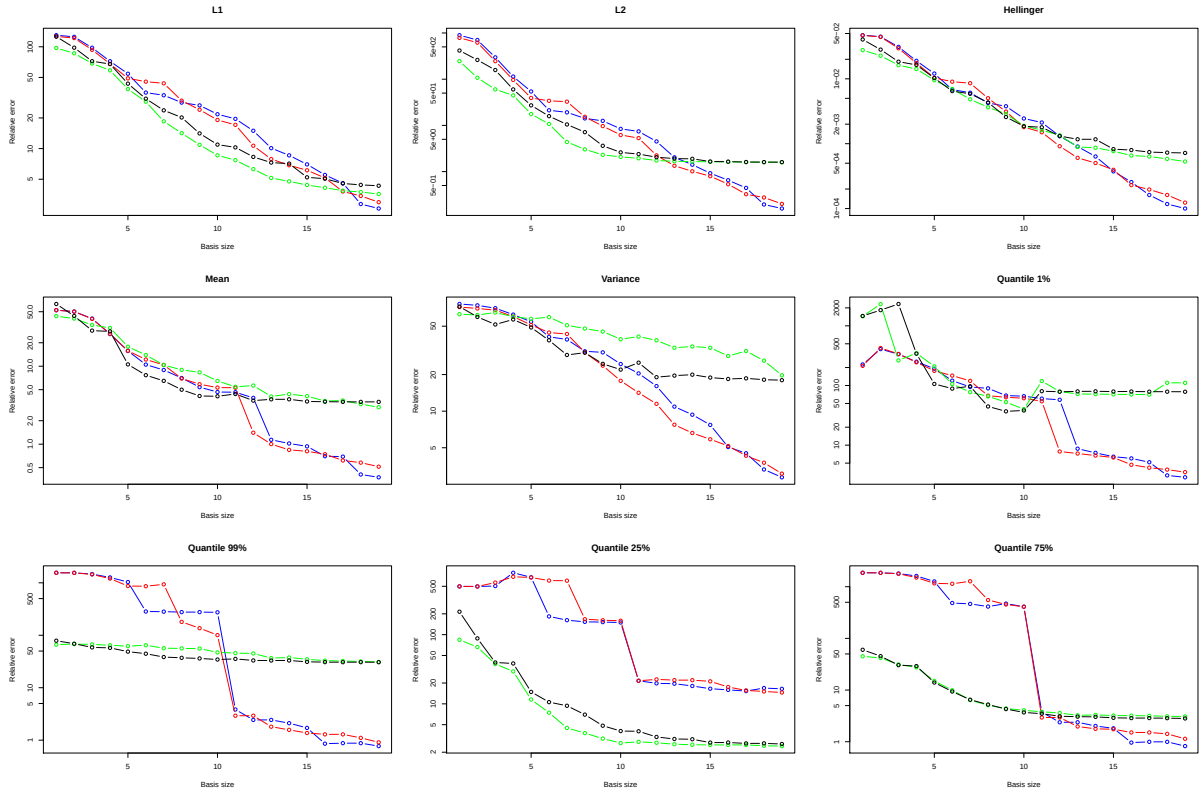

FIGURE 30. GIBBS test case: comparison of the relative errors on different quantities for the vapor pressure in function of the size of the basis q. Blue: MMP decomposition given by L^2 norm. Red: MMP decomposition given by the Hellinger distance. Black: AQM decomposition. Green: CPCA decomposition.

Quadratic Minimization (AQM), aims at minimizing the L^2 approximation error under the defined constraints. Both methods require the resolution of constrained optimization problems. Then, in future work, a metamodel could be adjusted on the coefficients, to link them with the uncertain inputs, and provide a global metamodel for probability density function outputs.

Our methods have been tested on two analytical test cases and three industrial applications proposed by CEA, EDF and IFPEN. The kernel regression-based metamodel performs well on the analytical test cases, but shows its limits on the three numerical codes. Indeed, kernel regression is efficient in low dimension, but the difficulty to estimate the parameters increases with the dimension and the size of the sample. Moreover, this method is not adapted for numerical codes for which the influence of the controllable parameters is much more important than one of the uncontrollable parameters. Both Modified Magic Points (MMP) and AQM methods give good results on all the test cases. However, in most cases, the method proposed by Kneip and Utikal [14] performs better than the proposed ones.

Metamodels linking the uncertain inputs of the code and the functional basis coefficients remains to be defined. The main difficulty to build this metamodel lies in ensuring that the sum of the functional basis coefficients is equal to one. One of the limits of our methods is that all quantities of interest are not as well approximated. In particular, quantiles are often poorly approximated. Indeed, all the proposed methods are based on norm minimization, and the norms used are not well suited for the study of extreme values. It could

be interesting to adapt the proposed methods to norms giving more weights on the tails of the densities. The use of the Wasserstein metric between probability measures will be also studied.

We would like to thank our supervisors Nicolas Bousquet (EDF), Frédéric Delbos (IFPEN), Bertrand Iooss (EDF), Jérôme Lonchampt (EDF), Rafael Lugo (IFPEN), Amandine Marrel (CEA) and Nadia Pérot (CEA). We would also like to thank Anthony Nouy (École Centrale de Nantes) and Sébastien Da Veiga (IFPEN) for their guidance. Furthermore, we would like to thank Nicolas Champagnat (Institut Élie Cartan de Lorraine) and Tony Lelièvre (École des Ponts ParisTech) who organized the CEMRACS 2013 in collaboration with Anthony Nouy. Finally, we would like to thank CEA, EDF and IFPEN for funding this research project.

References

[1] John Affleck-Graves, A.H. Money, and C.G. Troskie. A principal component index subject to constraints. *Investment Analysts Journal*, November 1979.

[2] John Aitchison. The statistical analysis of compositional data. *Journal of the Royal Statistical Society. Series B (Methodological)*, 44(2):139–177, 1982.

[3] Bruce Ankenman, Barry L Nelson, and Jeremy Staum. Stochastic kriging for simulation metamodeling. *Operations research*, 58(2):371–382, 2010.

[4] Mario Bebendorf, Yvon Maday, and Benjamin Stamm. Comparison of some reduced representation approximations. *arXiv preprint arXiv:1305.5066*, 2013.

[5] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[6] Nguyen Ngoc Cuong. Reduced-basis approximations and a posteriori error bound for nonaffine and nonlinear partial differential equations: Application to inverse analysis, 2005.

[7] Sébastien Da Veiga and Amandine Marrel. Gaussian process modeling with inequality constraints. In *Annales de la faculté des sciences de Toulouse Mathématiques*, volume 21, pages 529–555. Université Paul Sabatier, Toulouse, 2012.

[8] Pedro Delicado. Dimensionality reduction when data are density functions. *Computational Statistics & Data Analysis*, 55(1):401–420, 2011.

[9] Karine Fessart and Jérome Lonchampt. Optimisation d'un programme d'investissements : méthode IPOP. *Conférence lambda-mu, La Rochelle, France*, 2010.

[10] Donald Goldfarb and Ashok Idnani. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical programming*, 27(1):1–33, 1983.

[11] Wolfgang Hardle, James Stephen Marron, et al. Optimal bandwidth selection in nonparametric regression function estimation. *The Annals of Statistics*, 13(4):1465–1481, 1985.

[12] Trevor J Hastie and Robert J Tibshirani. *Generalized Additive Models*. Chapman & Hall, London, 1990.

[13] Bertrand Iooss and Mathieu Ribatet. Global sensitivity analysis of computer models with functional inputs. *Reliability Engineering & System Safety*, 94(7):1194 – 1204, 2009.

[14] Alois Kneip and Klaus J Utikal. Inference for density families using functional principal component analysis. *Journal of the American Statistical Association*, 96(454):519–542, 2001.

[15] J. Lonchampt and K. Fessart. An optimization approach for life cycle management applied to large power transformers. *1023033 EPRI Report*.

[16] Yvon Maday, Ngoc Cuong Nguyen, Anthony T Patera, and George SH Pau. A general, multipurpose interpolation procedure: the magic points. In *Proceedings of the 2nd International Conference on Scientific Computing and Partial Differential Equations*, 2007.

[17] Amandine Marrel, Bertrand Iooss, Sébastien Da Veiga, and Mathieu Ribatet. Global sensitivity analysis of stochastic computer models with joint metamodels. *Statistics and Computing*, 22(3):833–847, 2012.

[18] Michael D McKay, Richard J Beckman, and William J Conover. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[19] Hans-Georg Muller and Ulrich Stadtmuller. Variable bandwidth kernel estimators of regression curves. *The Annals of Statistics*, 15(1):182–201, 1987.

[20] Elizbar A Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964.

[21] Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

[22] J. O. Ramsay and B. W. Silverman. *Functional Data Analysis*. Springer Series in Statistics. Springer, 2nd edition edition, 2005.

[23] Brian J Reich, Eric Kalendra, Curtis B Storlie, Howard D Bondell, and Montserrat Fuentes. Variable selection for high dimensional bayesian density estimation: application to human exposure simulation. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 61(1):47–66, 2012.
[24] RA Rigby and DM Stasinopoulos. Construction of reference centiles using mean and dispersion additive models. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 49(1):41–50, 2000.
[25] Murray Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, pages 832–837, 1956.
[26] Gianluigi Rozza. Reduced basis methods for stokes equations in domains with non-affine parameter dependence. *Computing and Visualization in Science*, 12(1):23–35, 2009.
[27] George R Terrell and David W Scott. Variable kernel density estimation. *The Annals of Statistics*, pages 1236–1265, 1992.
[28] Timo Tonn and Karsten Urban. A reduced-basis method for solving parameter-dependent convection-diffusion problems around rigid bodies. In *ECCOMAS CFD*, 2006.
[29] Philippe Ungerer, Christèle Beauvais, Jérôme Delhommelle, Anne Boutin, Bernard Rousseau, and Alain H. Fuchs. Optimization of the anisotropic united atoms intermolecular potential for n-alkanes. *Journal of Chemical Physics*, 112(12):5499–5510, March 2000.
[30] Geoffrey S Watson. Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 359–372, 1964.
[31] Isabelle Zabalza-Mezghani, Emmanuel Manceau, Mathieu Feraille, and Astrid Jourdan. Uncertainty management: From geological scenarios to production scheme optimization. *Journal of Petroleum Science and Engineering*, 44(1):11–25, 2004.

A. Appendix

A.1. L^2 distance-based kernel estimator

We prove here the equivalence between the classical kernel estimator and the L^2 distance-based kernel estimator introduced in Sections 1.1.1 and 1.1.2 respectively. Indeed the classical kernel estimator (1) can be retrieved by writing the optimality condition of the minimization problem (3) defining the L^2 distance-based kernel estimator:

$$\begin{aligned}
&\sum_{i=1}^{N} -2K_H(\mathbf{x}_i,\mathbf{x}_0)\int_I \left(f_i-\hat{f}_0\right)=0\\
\iff\quad &\int_I \left(\sum_{i=1}^{N} K_H(\mathbf{x}_i,\mathbf{x}_0)\left(f_i-\hat{f}_0\right)\right)=0\\
\iff\quad &\sum_{i=1}^{N} K_H(\mathbf{x}_i,\mathbf{x}_0)\left(f_i-\hat{f}_0\right)=0\\
\iff\quad &\hat{f}_0=\sum_{i=1}^{N}\frac{K_H(\mathbf{x}_i,\mathbf{x}_0)}{\sum_{j=1}^{N} K_H(\mathbf{x}_j,\mathbf{x}_0)}f_i.
\end{aligned}$$

A.2. Hellinger distance-based kernel estimator

We prove here the equivalence between expressions (4) and (5) of the Hellinger distance-based kernel estimator given in Section 1.1.2. The constraint that the integral of $\hat{f}_0$ is 1 in the optimization problem (4) is handled using the associated Lagrangian function. The problem becomes

$$\hat{f}_0=\arg\min_{f,\lambda} L(f,\lambda),$$

with

$$L(f,\lambda)=\sum_{i=1}^{N} K_H(\mathbf{x}_i,\mathbf{x}_0)\int_I\left(\sqrt{f_i}-\sqrt{f}\right)^2-\lambda\left(\int_I f-1\right).$$

The first order optimality conditions are

$$\begin{cases} 0 = \dfrac{\partial L}{\partial f} = \displaystyle\sum_{i=1}^{N} K_H(\mathbf{x}_i, \mathbf{x}_0) \int_I \left(1 - \frac{\sqrt{f_i}}{\sqrt{\hat{f}_0}}\right) - \lambda \int_I 1, \\ 0 = \dfrac{\partial L}{\partial \lambda} = 1 - \displaystyle\int_I \hat{f}_0. \end{cases} \tag{16}$$

The first condition in (16) gives

$$\begin{aligned} & \int_I \left(\sum_{i=1}^{N} K_H(\mathbf{x}_i, \mathbf{x}_0) \left(1 - \frac{\sqrt{f_i}}{\sqrt{\hat{f}_0}}\right) - \lambda \right) = 0 \\ \iff \quad & \sum_{i=1}^{N} K_H(\mathbf{x}_i, \mathbf{x}_0) \left(1 - \frac{\sqrt{f_i}}{\sqrt{\hat{f}_0}}\right) - \lambda = 0 \\ \iff \quad & \hat{f}_0 = \left(\sum_{i=1}^{N} \frac{K_H(\mathbf{x}_i, \mathbf{x}_0)\sqrt{f_i}}{\sum_{j=1}^{N} K_H(\mathbf{x}_j, \mathbf{x}_0) - \lambda} \right)^2 = \frac{\left(\sum_{i=1}^{N} K_H(\mathbf{x}_i, \mathbf{x}_0)\sqrt{f_i} \right)^2}{\left(\sum_{j=1}^{N} K_H(\mathbf{x}_j, \mathbf{x}_0) - \lambda \right)^2}. \end{aligned} \tag{17}$$

Then we replace $\hat{f}_0$ in the second condition of (16) by the expression given in (17):

$$\left(\sum_{j=1}^{N} K_H(\mathbf{x}_j, \mathbf{x}_0) - \lambda \right)^2 = \int_I \left(\sum_{i=1}^{N} K_H(\mathbf{x}_i, \mathbf{x}_0)\sqrt{f_i} \right)^2. \tag{18}$$

The combined equations (17) and (18) entail the formula given by (5) for the kernel estimator associated to the Hellinger distance:

$$\hat{f}_0 = \frac{\left(\sum_{i=1}^{N} K_H(\mathbf{x}_i, \mathbf{x}_0)\sqrt{f_i} \right)^2}{\displaystyle\int_I \left(\sum_{i=1}^{N} K_H(\mathbf{x}_i, \mathbf{x}_0)\sqrt{f_i} \right)^2}.$$

Conclusion et perspectives

Nous avons développé quatre méthodes de construction de métamodèle pour des fonctions dont les valeurs sont des densités de probabilité : *Kernel regression* (KR), *Constrained principal component analysis* (CPCA), *Modified magic points* (MMP) et *Alternate quadratic minimization* (AQM). Seule la méthode KR est complètement aboutie. Elle obtient de bons résultats sur les deux exemples analytiques, mais souffre de la dimension plus élevée des trois cas industriels. Les méthodes CPCA, MMP et AQM se sont révélées efficaces lors de tous nos différents tests. Elles nécessitent cependant le développement d'estimateurs liant les paramètres d'entrée aux coefficients de la décomposition (2.1) de la sortie. Une approche par krigeage a été proposée par les auteurs de [15, 16] pour la méthode MMP ; le métamodèle complet est ensuite intégré dans une adaptation de l'algorithme EGO pour la résolution d'un problème de maximisation de quantile. Un tel métamodèle complet pourra être mis en œuvre pour obtenir de premiers résultats sur le problème inverse de l'optimisation des champs de forces en modélisation moléculaire. Il s'agit de trouver les paramètres du modèle — les entrées du code GIBBS — qui permettent de reproduire au mieux les distributions expérimentales de données macroscopiques du système, telles que la masse volumique de la phase liquide et la pression de la phase gazeuse.

Chapitre 3

Recherche directe pour les fonctions soumises à un bruit borné

Dans ce chapitre nous revenons sur l'Algorithme 1 rappelé dans la Section 1.1.1. Nous analysons l'effet d'un bruit borné portant sur les valeurs de la fonction objectif sur l'efficacité de cet algorithme dans la Section 3.1. La mise en œuvre des résultats de cette analyse sur des fonctions test et deux applications industrielles est présentée dans la Section 3.2.

3.1 Influence d'un bruit borné sur la recherche directionnelle directe

En pratique il arrive que les valeurs de la fonction objectif soient seulement connues avec une précision limitée. Les algorithmes d'optimisation en général et l'algorithme de recherche directe en particulier sont cependant habituellement mis en œuvre sans prendre cette inexactitude en compte. Nous pouvons alors nous poser les trois questions suivantes.

1. Quelle précision peut-on espérer obtenir, étant donnée l'inexactitude ?
2. À quel prix cette précision peut-elle être atteinte ?
3. Quels critères d'arrêt permettent de garantir cette précision ?

Dans le projet de publication reproduit à la section suivante nous répondons ces trois questions pour l'algorithme de recherche directe directionnelle appliqué à des fonctions objectifs dont les valeurs sont connues avec une erreur uniformément bornée par un nombre positif e. Plus précisément nous montrons les trois résultats suivants.

1. L'infimum des normes des gradients aux itérés est majoré à une constante multiplicative près par $\sqrt{e}$.
2. Le nombre d'itérations nécessaires pour obtenir un itéré dont la norme du gradient est inférieure à un seuil ϵ est majoré à une constante multiplicative près par ϵ^{-2}.
3. Nous établissons également un critère d'arrêt basé sur la taille du pas qui assure la norme du gradient au dernier itéré est sous la borne donnée par notre premier résultat.

Le troisième résultat nous donne un critère d'arrêt pour un niveau de précision e donné. Nous en déduisons un algorithme adaptatif pour utiliser efficacement des oracles de niveaux de précision distincts.

Direct Search Based on Inaccurate Function Values

F. Delbos S. Gratton B. Pauwels Z. Zhang

1 Introduction

How to minimize a function if its derivatives or subdifferentials are not available and its function values can only be evaluated inaccurately?

This question arises, for instance, in some real world optimization problems whose objective functions involve the output of computer simulations [4, 35]. To be specific, let us confine ourselves to a smooth unconstrained problem

$$\min_{x\in\mathbb{R}^n} f(x).$$

Due to the inaccessibility of derivatives, we are encountering a so-called derivative-free optimization (DFO) problem [9], and direct search [21] may be the simplest method to try. Although direct search may signify a large family of algorithms (see Subsection 2.1), we focus on the following one.

Algorithm 1.1 (Direct search)

Initialization: Select $x_0 \in \mathbb{R}^n$, $\alpha_0 > 0$, $c > 0$, and $0 < \theta < 1 \leq \gamma$. Set $k = 0$.

Do Until Converge

1. **Polling:** Choose a set D_k of directions[1]. If there exists $d_k \in D_k$ such that $f(x_k + \alpha_k d_k) < f(x_k) - c\,\alpha_k^2/2$, then set $x_{k+1} = x_k + \alpha_k d_k$, and declare the iteration successful. Otherwise declare the iteration unsuccessful and set $x_{k+1} = x_k$.
2. **Step size update:** If the iteration is successful, set $\alpha_{k+1} = \gamma\alpha_k$. Otherwise, set $\alpha_{k+1} = \theta\alpha_k$. Increment k by one.

End

Even if we feed this algorithm with inaccurate function values, it may still minimize the objective function to some degree. In fact, [21, Section 6.1] argues that direct search is well adapted to problems with "numerical noise" in function values, because it "will easily realize decrease in the function so long as the step size is large relative to the noise". Nevertheless, to the best of our knowledge, no detailed theory has been established on the behavior of direct search working with inaccurate function values. This paper attempts to fill the gap.

The behavior of Algorithm 1.1 is fairly well understood when accurate function values are available. Its global convergence has been studied thoroughly [23, 21, 3, 34]. For smooth

[1] One conventionally chooses D_k to be a positive spanning set [9, Chapters 2 and 7]. See Subsection 2.1 for more information on how to define D_k.

problems, its worst case complexity is shown to be the same as classical gradient methods; in other words, Algorithm 1.1 drives $\|\nabla f(x_k)\|$ below a given positive constant ϵ within $\mathcal{O}(\epsilon^{-2})$, $\mathcal{O}(\epsilon^{-1})$, and $\mathcal{O}(\log \epsilon^{-1})$ iterations for nonconvex, convex, and strongly convex objective functions respectively [33, 14]. It is interesting to investigate what will happen when the function values are inaccurate, because it is not an uncommon situation in optimization, especially in derivative-free optimization (see Subsection 2.2).

The objective of this paper is twofold. The first is to understand the behavior of direct search given a certain magnitude of inaccuracy in the function values. More concretely, we will consider the situation that the objective function f is approximately evaluated by an oracle F_e with

$$|F_e(x) - f(x)| \leq e,$$

where $e > 0$ indicates the magnitude of inaccuracy in F_e, and we will address three questions as follows.

1. What kind of result is Algorithm 1.1 able to achieve?
2. What is the worst case complexity for Algorithm 1.1 to attain an achievable result?
3. When should Algorithm 1.1 terminate in order to guarantee the quality of the result and also avoid "over-optimization"?

Most interestingly, for a smooth but possibly nonconvex f, we will prove that Algorithm 1.1 renders $\|\nabla f(x_k)\| \leq \mathcal{O}(\sqrt{e})$ in finite iterations, where the bound $\mathcal{O}(\sqrt{e})$ cannot be improved, and more quantitatively, $\|\nabla f(x_k)\|$ is driven below ϵ within $\mathcal{O}(\epsilon^{-2})$ iterations as long as ϵ is not lower than $\mathcal{O}(\sqrt{e})$, meaning that the inaccuracy does not affect the worst case complexity of Algorithm 1.1 as long as the requirement on the optimality is realistic. We will also see that the initial step size α_0 is critical for the performance of Algorithm 1.1, and that setting $\gamma > 1$ is more advantageous than choosing $\gamma = 1$ when the function values are inaccurate.

The second objective is to understand how to manipulate the accuracy of the function values so that Algorithm 1.1 can generate reasonably good result with low expense. This part assumes that we have control over the accuracy of function evaluation but the control is limited, or more precisely, we have a family of oracles $\{F_e : e \in E\}$ in hand, where E is a set of positive numbers with a minimum $e_{\min} > 0$ corresponding to the most accurate oracle. We will design an oracle-selecting strategy for Algorithm 1.1 so that the following requirements are satisfied.

1. Instead of sticking to $F_{e_{\min}}$, Algorithm 1.1 chooses its oracle adaptively according to the progress of the computation.
2. Algorithm 1.1 will achieve a result equally good as $F_{e_{\min}}$ can yield, which means that $\|f(x_k)\| \leq \mathcal{O}(\sqrt{e_{\min}})$ is guaranteed within finite iterations.
3. The worst case complexity of Algorithm 1.1 is not impaired; that is to say, $\|f(x_k)\| \leq \epsilon$ is attained within $\mathcal{O}(\epsilon^{-2})$ iterations for any ϵ not lower than $\mathcal{O}(\sqrt{e_{\min}})$.

If the expense of a function evaluation increases rapidly with the improvement of its accuracy, our strategy will improve the overall performance of Algorithm 1.1 significantly.

The remainder of the paper will be organized in the following way. Section 2 will introduce some background knowledge on derivative-free optimization and direct search, and review some

literature concerning optimization based on inaccurate information. Section 3 will summarize briefly the convergence theory of direct search based on accurate function values, and we will present a new way of proving its worst case complexity. Section 4 will study the behavior of direct search when the function values are provided by an oracle with a certain magnitude of inaccuracy, and the analysis will be based on the techniques presented in Section 3. The oracle-selecting strategy will be discussed in Section 5. Several concluding remarks will be made in Section 6.

2 Background knowledge and related work

2.1 Derivative-free optimization and direct search

As mentioned in the introduction, some real-world optimization problems occur without usable derivative or subdifferentials, and they are known as derivative-free optimization problems. A typical situation is that the objective function is not defined by an explicit formula but by a black box that does not provide access to the first-order information, for example, a computer simulation. These problems have stimulated the development of derivative-free optimization algorithms. For applications and elementary theories of derivative-free optimization, we refer to [19, 29, 9] and the references therein.

Although this paper will focus on the direct search defined in Algorithm 1.1, the word "direct search" can also mean a broad class of derivative-free optimization algorithms including, in addition to Algorithm 1.1, the Nelder-Mead simplex method [27, 36], the Mesh Adaptive Direct Search (MADS) [32, 2, 1, 22], to name but a few. These algorithms are characterized by taking actions based on finite sampling of the objective function without explicit derivative approximation or model building. More information on these algorithms can be found in the survey paper [21] and [9, Chapter 7].

Now let us turn our attention back to Algorithm 1.1.

Clearly, the choice of the direction sets D_k ($k = 0, 1, 2 \dots$) is critical for performance of Algorithm 1.1. Traditionally, one choose D_k to be positive spanning sets [12, 21, 9], span $\mathbb{R}^n$ by linear combinations with nonnegative coefficients.

In practice, each iteration of Algorithm 1.1 can include an optional search step before the polling. Such a step samples the objective function at a finite number of points to look for an x satisfying $f(x) < f(x_k) - c\,\alpha_k^2/2$. If x could be found, then one would define $x_{k+1} = x$, declare the iteration successful, and skip the polling. Such a procedure enables the user to integrate heuristics or *a priori* information into the algorithm. In numerical analysis of direct search (for instance, [21, 33]), it is common to omit the search step and focus on the polling, which is the essential part of the algorithm. We will follow this convention.

2.2 Optimization based on inaccurate information

Traditional numerical analysis of optimization algorithms usually assumes that the objective function can be accurately evaluated when its value is explicitly involved in the numerical computation, unless the algorithm under consideration is particularly designed to deal with noisy problems. In reality, however, the function values used in computations are usually corrupted with some kind of inaccuracy, which can be deterministic or stochastic in nature, even if the underlying optimization problem is considered to be noise-free. Keeping in mind that computers

can only perform finite precision computation, it is not difficult to realize that the optimization algorithms designed to work with accurate function values are generally running based on inaccurate ones. Even if we assume that the inaccuracy caused by rounding errors is insignificant enough to be neglected, which may be wishful thinking unless it is guaranteed by a solid theory, there are still plenty of cases where inaccuracy is inevitable, especially if the definition of the objective function involves another mathematical problem or physical process. For example, it is often impossible to evaluate an objective function accurately if it relies on the solution of a PDE or the output of a computer simulation, which is the case for the optimal design problems studied in [4, 35] and commented in [21, Section 1.2.1]. For similar reasons, algorithms that are designed to utilize accurate gradients (or subgradients) are in fact running mostly with inaccurate ones when we think about computations in reality.

Therefore, it is necessary to investigate how optimization algorithms behave with inaccurate information if they were developed without much consideration on the inaccuracy. Such investigation is often a subject of algorithm benchmarking, where the algorithms are numerically compared on solving noisy problems even if they were primarily designed for noise-free ones (see [25], for example). On the theoretical side, there exist two categories of investigations. The first category mainly concentrates on diminishing inaccuracy that progressively vanishes along the iterations, and studies the necessary conditions on the inaccuracy to guarantee convergence to stationary points and certain convergence rates, examples including [17, 8]. The second category, on the contrast, concerns situations when the inaccuracy cannot be diminished along the iterations.

3 Direct search based on accurate function values

Throughout this paper we make the following classical assumptions.

Assumption 3.1

- *f is bounded below:* $\inf_{x\in\mathbb{R}^n} f(x) > -\infty$.
- *∇f is Lipschitz continuous:*

$$L = \sup\{\|\nabla f(x) - \nabla f(y)\|/\|x - y\| : x, y \in \mathbb{R}^n, x \neq y\} < \infty.$$

In particular

$$|f(y) - f(x) - \nabla f(x)^\top (y - x)| \leq \frac{L}{2}\|x - y\|^2 \tag{1}$$

for all x and y in $\mathbb{R}^n$.

Assumption 3.2 *Every direction of the positive spanning set D is normalized: $\|d\| = 1$ for all d in D. We denote κ the cosine measure of D, which lies in $(0, 1]$:*

$$\kappa = \min_{\|v\|=1} \max_{d\in D} v^\top d.$$

In the rest of this paper we denote g_k the gradient of f at the iterate x_k, for each integer k:

$$g_k = \nabla f(x_k), \quad k \geq 0.$$

We recall the following two classical results on the global convergence and worst case complexity of Algorithm 1.1 under the previous assumptions.

Theorem 3.1 (Torczon 1997, Kolda, Lewis, and Torczon 2003) *Let the previous assumptions hold. Then the sequence $\{x_k\}_{k\geq 0}$ generated by direct search satisfies*

$$\liminf_{k\to\infty} \|g_k\| = 0.$$

Theorem 3.2 (Vicente 2013) *Let the previous assumptions hold and ϵ be in $(0, 1)$, and k_ϵ be the first integer such that $\|g_{k_\epsilon}\| \leq \epsilon$. Then*

$$k_\epsilon = \mathcal{O}(\epsilon^{-2}).$$

4 Direct search based on a single inaccurate oracle

In this section, after presenting a motivating numerical example, we first introduce some useful lemmas and then present our main results: an upper bound on the infimum of the norms of the iterates gradients, an upper bound on the first iteration driving the gradient norm below a given tolerance, and a stopping criterion based on the step size.

4.1 A motivating example

Conventional analysis of optimization algorithms usually assumes that the information available (function values, derivatives) are accurate, unless the algorithm is particularly designed for noisy problems. However, in reality, it happens very often that people apply algorithms designed to solve

What will happen if we ignore the inaccuracy and apply Algorithm 1.1 as if the function values were accurate? We hope that the algorithm will still minimize the objective function in some sense, despite the inaccuracy in the information. Obviously, it depends on the magnitude of the inaccuracy, and also the initial step size. If the initial step size is large enough, the algorithm may well "overlook" the inaccuracy in the function values and behave well until the step size becomes too small. On the contrary, if the initial step size is too small, the algorithm will suffer since the very beginning, because the inaccuracy in the function values will dominate the variation of the objective function when the trial points are too close to each other, and consequently the algorithm may hardly make any correct move.

To see this, let us do a simple numerical experiment. In this experiment, we use Algorithm 1.1 to minimize

$$f(x) = \frac{1}{20}x^\top x, \quad x \in \mathbb{R}^2,$$

but we do not feed it with the accurate function values. Instead, every time Algorithm 1.1 request a function evaluation, we give it an inaccurate value that is the true value perturbed by a random number uniformly distributed in $[-0.1, 0.1]$, and the perturbations are mutually independent. More precisely,

$$F(x) = f(x) + eR_x,$$

with $e = 0.1$ and, for each x in $\mathbb{R}^2$, R_x is an independent sample from the uniform distribution on $[-1, 1]$. Algorithm 1.1 is configured with $c = 1/20$, $\theta = 1/2$, $\gamma = 1$, and $D_k = \{v_1, -v_1, v_2, -v_2\}$, where v_1 and v_2 are the coordinate vectors in $\mathbb{R}^2$. The starting point is taken at random from the uniform distribution on $10\,\mathbb{S}^1$ (the sphere whose center is 0 and radius is 10).

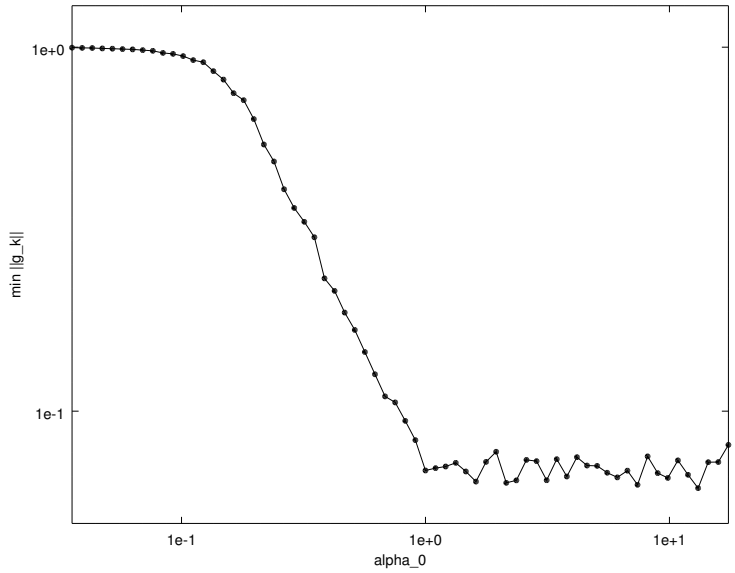

Figure 1: The initial step size significantly influences the minimal gradient norm attainable

The minimum gradient norm achieved by Algorithm 1.1 relative to the initial step size α_0 is represented on Figure 1. We observe that the performance of the algorithm is at its best only if the initial value of the step size is big enough. This dependence will be explained by our results on the global convergence of Algorithm 1.1 for functions with inaccurate function values.

4.2 Lemmas

The following lemma is an adaptation of a classical gradient norm inequality for inaccurate function values.

Lemma 4.1 *Let k be an integer. If iteration k is unsuccessful, then*

$$\kappa\|g_k\| \leq \frac{(c+L)\alpha_k}{2} + \frac{2e}{\alpha_k}. \tag{2}$$

Proof. Let d be a direction in D such that $\kappa\|g_k\| \leq -g_k^\top d$. If iteration k is unsuccessful, then

$$\begin{aligned} 0 &\leq F_e(x_k + \alpha_k d) - F_e(x_k) + \frac{c}{2}\alpha_k^2 \\ &\leq f(x_k + \alpha_k d) - f(x_k) + 2e + \frac{c}{2}\alpha_k^2 \\ &\leq \alpha_k g_k^\top d + \frac{L}{2}\alpha_k^2 + 2e + \frac{c}{2}\alpha_k^2 \\ &\leq -\kappa\|g_k\|\alpha_k + \frac{c+L}{2}\alpha_k^2 + 2e, \end{aligned}$$

where we used the Lipschitz continuity (1) of f. Hence the result. □

We refine the previous result with the following lemma, which will be a key to our main results in the next subsections.

Lemma 4.2 *Let k be an integer and α be a positive number. If $\alpha_{k+1} < \alpha \leq \alpha_k$, then*

$$\kappa\|g_k\| \leq \max\left\{\frac{(c+L)\alpha}{2} + \frac{2e}{\alpha},\ \frac{(c+L)\alpha}{2\theta} + \frac{2\theta e}{\alpha}\right\}. \tag{3}$$

Proof. According to Algorithm 1.1, iteration k is unsuccessful, and $\alpha_k = \theta^{-1}\alpha_{k+1}$. Thus Equation (2) holds. The right-hand side of (2) is the sum of two convex functions of the variable α_k in $(0, \infty)$, therefore it is convex with respect to the positive number α_k. Since $\alpha \leq \alpha_k < \theta^{-1}\alpha$ holds, the right-hand side of (2) is less than or equal to that of (3) by convexity. □

The following lemma is an adaptation of a result of [16] providing a bound on the sum of the squared stepsizes. It ensures the sequence $\{\alpha_k\}_{k\geq 0}$ goes to 0, and will also be useful to prove our results on the worst case complexity and stopping of the algorithm for inaccurate function values.

Lemma 4.3 *Let k be a positive integer. Then*

$$\sum_{i=0}^{k-1} \alpha_i^2 \ < \ \frac{\gamma^2(4e+\beta)}{c(1-\theta^2)},$$

where β is the positive number defined below:

$$\beta \ = \ 2\left[\rho(\gamma^{-1}\alpha_0) + f(x_0) - \inf_{x\in\mathbb{R}^n} f(x)\right]. \tag{4}$$

Proof. The same analysis as in the proof of [16, Lemma 4.1] leads to

$$\frac{c}{2}\sum_{i=0}^{k-1} \alpha_i^2 \leq \frac{\gamma^2}{1-\theta^2}\left[\rho(\gamma^{-1}\alpha_0) + F(x_0) - \inf_{x\in\mathbb{R}^n} F(x)\right].$$

Hence

$$\sum_{i=0}^{k-1} \alpha_i^2 \leq \frac{2\gamma^2}{c(1-\theta^2)}\left[\rho(\gamma^{-1}\alpha_0) + f(x_0) - \inf_{x\in\mathbb{R}^n} f(x) + 2e\right]$$

and the result follows. □

4.3 Global convergence

We now present our first global convergence result, extending Theorem 3.1 to functions with inaccurate values.

Theorem 4.1 *The iterates generated by Algorithm 1.1 satisfy*

$$\inf_{k\geq 0} \|g_k\| \;\leq\; \max\left\{\frac{\tilde{\alpha}}{\alpha_0},\, 1\right\} \frac{(1+\theta)\sqrt{c+L}}{\kappa\sqrt{\theta}}\sqrt{e}, \tag{5}$$

where

$$\tilde{\alpha} \;=\; \frac{2\sqrt{\theta e}}{\sqrt{c+L}}.$$

Proof. Let α be in $(0, \alpha_0]$. Lemma 4.3 implies the sequence $\{\alpha_k\}_{k\geq 0}$ goes to 0. Let us denote k the first integer such that $\alpha_{k+1} < \alpha$. Then k is an unsuccessful iteration and $\alpha_{k+1} < \alpha \leq \alpha_k$. Therefore Lemma 4.2 ensures inequality (3) holds; let us denote $m(\alpha)$ the right-hand side of (3). We deduce the following upper bound for the infimum of the gradient norms at the iterates along the algorithm:

$$\kappa \inf_{k\geq 0} \|g_k\| \leq \inf_{\alpha\in(0,\alpha_0]} m(\alpha). \tag{6}$$

Let α be a positive number. The following inequality

$$\frac{(c+L)\alpha}{2} + \frac{2e}{\alpha} \leq \frac{(c+L)\alpha}{2\theta} + \frac{2\theta e}{\alpha}$$

is equivalent to

$$\frac{c+L}{2\theta}\alpha \geq \frac{2e}{\alpha}, \;\text{ and to }\; \alpha^2 \geq \frac{4\theta e}{c+L} = \tilde{\alpha}^2.$$

Therefore $m(\alpha)$ is given by the equality below:

$$m(\alpha) = \begin{cases} \dfrac{(c+L)\alpha}{2} + \dfrac{2e}{\alpha} & \text{if } \alpha \leq \tilde{\alpha}, \\ \dfrac{(c+L)\alpha}{2\theta} + \dfrac{2\theta e}{\alpha} & \text{if } \alpha \geq \tilde{\alpha}. \end{cases}$$

If α is lower than $\tilde{\alpha}$ then $m'(\alpha) = (c+L)/2 - 2e/\alpha^2$ is negative. If α is greater than $\tilde{\alpha}$ then $m'(\alpha) = (c+L)/(2\theta) - 2\theta e/\alpha^2$ is positive. Since m is continuous at $\tilde{\alpha}$, equalities $\inf_{\alpha\in(0,\tilde{\alpha})} m(\alpha) = m(\tilde{\alpha})$ and $\inf_{\alpha\in(\tilde{\alpha},\infty)} m(\alpha) = m(\tilde{\alpha})$ hold. Consequently the function m decreases over $(0, \tilde{\alpha}]$ and increases over $[\tilde{\alpha}, \infty)$. If α_0 is lower than $\tilde{\alpha}$ then the infimum of m over $(0, \alpha_0]$ is attained at α_0 and

$$\begin{aligned} m(\alpha_0) &= \frac{(c+L)\alpha_0}{2} + \frac{2e}{\alpha_0} \\ &= \left[\frac{\sqrt{\theta}\alpha_0}{\tilde{\alpha}} + \frac{\tilde{\alpha}}{\sqrt{\theta}\alpha_0}\right]\sqrt{(c+L)e} \\ &= \frac{(\theta\alpha_0^2 + \tilde{\alpha}^2)\sqrt{(c+L)e}}{\sqrt{\theta}\alpha_0\tilde{\alpha}} \\ &\leq \frac{\tilde{\alpha}(1+\theta)\sqrt{(c+L)e}}{\sqrt{\theta}\alpha_0}. \end{aligned}$$

Otherwise — α_0 is greater than or equal to $\tilde{\alpha}$ — the infimum is attained at $\tilde{\alpha}$ and

$$m(\tilde{\alpha}) = \frac{(1+\theta)\sqrt{(c+L)e}}{\sqrt{\theta}}.$$

In both cases Equation (6) ensures that Equation (5) holds. □

Thus the minimization of f can only be guaranteed to a certain extent, which is $\mathcal{O}(\sqrt{e})$, depending on the initial step size. If α_0 is chosen greater than or equal to $\tilde{\alpha}$ then the bound

$$\inf_{k\geq 0} \|g_k\| \;\leq\; \frac{(1+\theta)\sqrt{c+L}}{\kappa\sqrt{\theta}}\sqrt{e}$$

is guaranteed. Our next result shows that, when $\gamma > 1$, a tighter bound on the infimum of the gradient norms is obtained provided the gradient norm of the initial point is big enough.

Theorem 4.2 *If $\gamma > 1$ and*

$$\alpha_0\|g_0\| \geq \frac{2(1+\theta)e}{\min\{\kappa,\ \gamma-1\}},$$

then

$$\inf_{k\geq 0} \|g_k\| \;\leq\; \frac{(1+\theta)\sqrt{c+L}}{\kappa\sqrt{\theta}}\sqrt{e}. \tag{7}$$

Proof. We prove by contradiction, assuming that (7) is false:

$$\frac{(1+\theta)\sqrt{c+L}}{\kappa\sqrt{\theta}}\sqrt{e} < \inf_{k\geq 0} \|g_k\|. \tag{8}$$

Observe that, in the proof of Theorem 4.1, the parameter α_0 can be any positive number lower than or equal to $\max_{k\geq 0}\alpha_k$. In particular we can replace α_0 with $\max_{k\geq 0}\alpha_k$ and obtain

$$\inf_{k\geq 0} \|g_k\| \;\leq\; \max\left\{\frac{\tilde{\alpha}}{\max_{k\geq 0}\alpha_k},\ 1\right\}\frac{(1+\theta)\sqrt{c+L}}{\kappa\sqrt{\theta}}\sqrt{e}. \tag{9}$$

Combining Equations (8) and (9) we get

$$\max_{k\geq 0}\alpha_k < \tilde{\alpha}. \tag{10}$$

We will show by induction that

$$\alpha_k\|g_k\| \geq \frac{2(1+\theta)e}{\min\{\kappa,\ \gamma-1\}},\ \ k \geq 0. \tag{11}$$

Let k be an integer. When $k = 0$, this is true. If it is true for k, then

$$\alpha_k\|g_k\| \geq \frac{2(1+\theta)e}{\kappa} = \frac{2e}{\kappa} + \frac{c+L}{2\kappa}\tilde{\alpha}^2 > \frac{2e}{\kappa} + \frac{c+L}{2\kappa}\alpha_k^2.$$

It follows from Lemma 4.1 that the k-th iteration is successful. Thus $\gamma\alpha_k = \alpha_{k+1} \leq \tilde{\alpha}$, and

$$\alpha_{k+1}\|g_{k+1}\| \geq \gamma\alpha_k(\|g_k\| - L\alpha_k) = \alpha_k\|g_k\| + (\gamma-1)\alpha_k\|g_k\| - \gamma L\alpha_k^2,$$

where we used the Lipschitz continuity of the gradient. Equations (10) and (11) give

$$\begin{aligned}(\gamma-1)\|g_k\| - \gamma L\alpha_k &\geq \frac{2(1+\theta)e}{\alpha_k} - L\alpha_{k+1} \\ &\geq \frac{2(1+\theta)e}{\tilde{\alpha}} - L\tilde{\alpha} \\ &= \frac{(1+\theta)\sqrt{(c+L)e}}{\sqrt{\theta}} - \frac{2\sqrt{\theta e}L}{\sqrt{c+L}} \\ &\geq \frac{(1-\theta)\sqrt{(c+L)e}}{\sqrt{\theta}} \geq 0.\end{aligned}$$

Therefore $\alpha_{k+1}\|g_{k+1}\| \geq \alpha_k\|g_k\|$, which completes the induction. Equation (11) contradicts the fact that $\{\alpha_k\}_{k\geq 0}$ goes to 0. □

Suppose $\gamma > 1$ holds and the starting point x_0 satisfies

$$\|g_0\| > \frac{(1+\theta)\sqrt{c+L}}{\min\{\kappa, \gamma-1\}\sqrt{\theta}} \geq \frac{(1+\theta)\sqrt{c+L}}{\kappa\sqrt{\theta}}.$$

Notice that the bound (7) is not met at the beginning of the algorithm. Let us denote

$$\bar{\alpha} = \frac{2(1+\theta)e}{\min\{\kappa,\ \gamma-1\}\|g_0\|}.$$

Then the following inequality holds: $\bar{\alpha} < \tilde{\alpha}$. It follows from Theorem 4.2 that in this case requiring $\alpha_0 \geq \bar{\alpha}$, rather than $\alpha_0 \geq \tilde{\alpha}$, is enough to achieve (7).

4.4 Worst case complexity

We now extend Theorem 3.2 to functions with inaccurate values: the worst case complexity of directional direct search is still $\mathcal{O}(\epsilon^{-2})$. However the tolerance ϵ is required to be realistic—given Theorem 4.1—and the initial step to be big enough.

Theorem 4.3 *For each positive number ϵ such that*

$$\epsilon \ \geq \ \frac{(1+\theta)\sqrt{c+L}}{\kappa\sqrt{\theta}}\sqrt{e}, \tag{12}$$

and

$$\alpha_0 \ \geq \ \frac{2\kappa\theta\epsilon}{(1+\theta)(c+L)},$$

the first integer k_ϵ driving the gradient norm below ϵ satisfies

$$k_\epsilon \ \leq \ \frac{\gamma^2(c+L)}{c\,\theta(1-\theta^2)} + \frac{\beta\gamma^2(c+L)^2}{c\,\kappa^2\theta^2(1-\theta^2)}\epsilon^{-2},$$

where β is the positive number given by (4).

Proof. Let us consider the following positive integer

$$k = \left\lceil \frac{\gamma^2(c+L)}{c\,\theta(1-\theta^2)} + \frac{\beta\gamma^2(c+L)^2}{c\,\kappa^2\theta^2(1-\theta^2)}\epsilon^{-2} \right\rceil$$

and the positive number below

$$\alpha = \frac{2\kappa\theta\epsilon}{(1+\theta)(c+L)}. \tag{13}$$

It follows from Equation (12) and Lemma 4.3 that

$$\begin{aligned} k\alpha^2 &\geq \frac{\gamma^2(c+L)}{c\theta(1-\theta^2)}\left[1 + \frac{\beta(c+L)}{\kappa^2\theta\epsilon^2}\right]\frac{4\kappa^2\theta^2\epsilon^2}{(1+\theta)^2(c+L)^2} \\ &= \frac{4\gamma^2}{c(1-\theta^2)}\left[\frac{\kappa^2\theta\epsilon^2}{(1+\theta)^2(c+L)} + \frac{\beta}{(1+\theta)^2}\right] \\ &\geq \frac{4\gamma^2}{c(1-\theta^2)}\left[e + \frac{\beta}{(1+\theta)^2}\right] \\ &> \sum_{i=0}^{k-1}\alpha_i^2. \end{aligned}$$

Therefore there exists i in $\{0, 1, \ldots, k-1\}$ such that

$$\alpha_i < \alpha.$$

Since $\alpha_0 \geq \alpha$, we know that $i > 1$. Consequently there exists an unsucessful iteration i_0 in $\{0, 1, \ldots, i-1\}$ satisfying $\alpha_{i_0} \geq \alpha > \alpha_{i_0+1}$. Applying Lemma 4.2, we have

$$\|g_{i_0}\| \leq \max\left\{\frac{\alpha(c+L)}{2\kappa} + \frac{2e}{\kappa\alpha},\ \frac{\alpha(c+L)}{2\theta\kappa} + \frac{2\theta e}{\kappa\alpha}\right\}. \tag{14}$$

Now we check that the right-hand side of (14) is at most ϵ, which could be done by straightforward calculations with the help of (13) and (12):

$$\frac{\alpha(c+L)}{2\kappa} + \frac{2e}{\kappa\alpha} \leq \frac{\theta\epsilon}{1+\theta} + \frac{\kappa\theta\epsilon^2}{(1+\theta)^2(c+L)}\cdot\frac{(1+\theta)(c+L)}{\kappa\theta\epsilon} = \frac{\theta\epsilon}{1+\theta} + \frac{\epsilon}{1+\theta} = \epsilon,$$

and

$$\frac{\alpha(c+L)}{2\theta\kappa} + \frac{2\theta e}{\kappa\alpha} \leq \frac{\epsilon}{1+\theta} + \frac{\kappa\theta^2\epsilon^2}{(1+\theta)^2(c+L)}\cdot\frac{(1+\theta)(c+L)}{\kappa\theta\epsilon} = \frac{\epsilon}{1+\theta} + \frac{\theta\epsilon}{1+\theta} = \epsilon.$$

By the definition of k_ϵ, i_0, and i, we have

$$k_\epsilon \leq i_0 \leq i-1 \leq k-2,$$

which completes the proof. □

The previous result shows the worst case complexity of the algorithm is not affected by the accuracy, so long as we expect a tolerance on the gradient norm within the bounds of Theorems 4.1 and 4.2. As for both these results, the initial step size is required to be big enough.

4.5 Stopping criterion

The following result provides a stopping criterion ensuring a gradient norm accuracy close to the best that can be guaranteed—considering Theorems 4.1 and 4.2— and a sufficient number of iterations that is $\mathcal{O}(e^{-1})$.

Theorem 4.4 *Let λ be any non-negative number. If*

$$\alpha_0 \geq \frac{2\sqrt{\theta e}}{\sqrt{c+\lambda}} \tag{15}$$

and k is the first integer such that

$$\alpha_k < \frac{2\sqrt{\theta e}}{\sqrt{c+\lambda}}, \tag{16}$$

then

$$\|g_k\| \;\leq\; \frac{2c+\lambda+L}{\kappa\sqrt{\theta(c+\lambda)}}\sqrt{e} \tag{17}$$

and

$$k \;\leq\; \frac{\gamma^2(c+\lambda)}{c\,\theta(1-\theta^2)} + \frac{\beta\gamma^2(c+\lambda)}{4c\,\theta(1-\theta^2)}e^{-1}. \tag{18}$$

Proof. According to (15) and the definition of k, we know that $k \geq 1$ and

$$\alpha_{k-1} \;\geq\; \frac{2\sqrt{\theta e}}{\sqrt{c+\lambda}} \;>\; \alpha_k.$$

By Lemma 4.2,

$$\begin{aligned}\|g_{k-1}\| \;&\leq \max\left\{\frac{(c+L)\sqrt{\theta}}{\kappa\sqrt{c+\lambda}}\sqrt{e} + \frac{\sqrt{c+\lambda}}{\kappa\sqrt{\theta}}\sqrt{e},\; \frac{(c+L)}{\kappa\sqrt{\theta(c+\lambda)}}\sqrt{e} + \frac{\sqrt{\theta(c+\lambda)}}{\kappa}\sqrt{e}\right\}\\ &\leq \frac{2c+\lambda+L}{\kappa\sqrt{\theta(c+\lambda)}}\sqrt{e}.\end{aligned}$$

Meanwhile, it is obvious that the $(k-1)$-th iteration is unsuccessful, which implies $g_k = g_{k-1}$. Hence inequality (17) is true.

Let

$$\ell \;=\; \left\lceil \frac{\gamma^2(c+\lambda)}{c\,\theta(1-\theta^2)} + \frac{\beta\gamma^2(c+\lambda)}{4c\,\theta(1-\theta^2)}e^{-1} \right\rceil.$$

It follows from Lemma 4.3 that

$$\ell \geq \frac{(4e+\beta)\gamma^2(c+\lambda)}{4ec\theta(1-\theta^2)} > \frac{c+\lambda}{4\theta e}\sum_{i=0}^{\ell-1}\alpha_i^2.$$

Thus there exists $i \leq \ell - 1$ satisfying

$$\alpha_i < \frac{2\sqrt{\theta e}}{\sqrt{c+\lambda}}.$$

Consequently $k \leq i \leq \ell - 1$ and (18) holds. □

The constant λ should be thought as an approximation of the Lispchitz constant L. Suppose $\lambda = L$. Then the gradient norm of the first iteration k satisfying Equation (16) is bounded as below:

$$\|g_k\| \leq \frac{2\sqrt{c+L}}{\kappa\sqrt{\theta}}\sqrt{e}.$$

This upper bound on $\|g_k\|$ is very similar to the one given by Theorem 4.1. Therefore Theorem 4.4 provides us with a stopping criterion that guarantees the gradient norm accuracy of the current iterate is nearly the best that can be ensured.

Let us incorporate this stopping criterion in the algorithm as follows.

Algorithm 4.1

Initialization

Choose $x_0 \in \mathbb{R}^n$, $0 < \theta < 1 \leq \gamma$, $c > 0$, $\lambda \geq 0$, and

$$\alpha_0 \geq \frac{2\sqrt{\theta e}}{\sqrt{c+\lambda}}.$$

Set $k = 0$.

While $\alpha_k \geq 2\sqrt{\theta e}/\sqrt{c+\lambda}$ **do**

1. **Search step**
2. **Poll step**
3. **Step size update**, increment k by one.

End

5 Direct search with adaptive oracle accuracy

Suppose that we have a sequence of oracles $\{F^{(0)}, F^{(1)}, \ldots, F^{(i_{\max})}\}$ corresponding to error levels $\{e^{(0)}, e^{(1)}, \ldots, e^{(i_{\max})}\}$, and

$$e^{(i+1)} \leq \theta^2 e^{(i)}, \qquad i = 0, \ldots, i_{\max} - 1.$$

Then we propose the following heuristic algorithm adapting the oracle accuracy along the iterations. We begin as in Algorithm 4.1 applied to the available oracle with the greatest error level $e^{(0)}$—that is $F^{(0)}$—with the following condition on the initial step size:

$$\alpha_0 \;\geq\; \frac{2\sqrt{\theta e^{(0)}}}{\sqrt{c+\lambda}}.$$

We know that the stopping criterion corresponding to $e^{(0)}$ given by Theorem 4.4 is reached by the sequence of step sizes within a finite number of iterations. When that occurs we are no longer ensured by Theorem 4.4 that the gradient norm of the iterate may decrease while using

$F^{(0)}$. Therefore we switch to the more accurate oracle $F^{(1)}$. We re-evaluate the current iterate with $F^{(1)}$ and restart with the current step size, now evaluating the objective function with an error level of $e^{(1)}$. Again, when the stopping criterion associated with $e^{(1)}$ is attained we switch to the next more accurate oracle $F^{(2)}$, re-evaluate the current iterate, restart, and so on. If, while using an oracle $F^{(i)}$ such that $i \geq 1$, the stopping criterion associated with $e^{(i-1)}$ is exceeded by the step size, then Theorem 4.4 guarantees that the gradient norm of the iterate may be decreased using oracle $F^{(i-1)}$. Thus we switch back to the less accurate oracle $F^{(i-1)}$, this time without re-evaluating. With this mechanism we ensure that we are using an error level appropriate to the step size value. If the stopping criterion corresponding to $e^{(i_{\max})}$ is met while using oracle $F^{(i_{\max})}$ then we stop: we can no longer guarantee a decrease in the gradient norm of the iterate, given the oracles available. This heuristic algorithm is summarized below.

Algorithm 5.1

Initialization

Choose $x_0 \in \mathbb{R}^n$, $0 < \theta < 1 \leq \gamma$, $c > 0$, $\lambda \geq 0$, and

$$\alpha_0 \ \geq \ \frac{2\sqrt{\theta e^{(0)}}}{\sqrt{c+\lambda}}.$$

Let $i_0 = 0$ and $F_0 = F^{(i_0)}$. Set $k = 0$.

While $i_k \leq i_{\max}$ **do**

1. **Search step**
2. **Poll step**
3. **Step size update**
4. **Oracle update:** Set

$$i_{k+1} = \begin{cases} i_k + 1 & \text{if } \alpha_{k+1} < 2\sqrt{\theta e^{(i_k)}}/\sqrt{c+\lambda}, \\ \max\{i_k - 1,\, 0\} & \text{if } \alpha_{k+1} > 2\gamma\sqrt{\theta e^{(i_k-1)}}/\sqrt{c+\lambda}, \\ i_k & \text{else,} \end{cases}$$

and

$$F_{k+1} = F^{(i_{k+1})}.$$

If $i_k < i_{k+1} \leq i_{\max}$, then re-evaluate $f(x_k)$ by the new oracle; else, do not re-evaluate.

End

6 Conclusions

In this paper we analyzed the behavior of the classical directional direct search algorithm for the unconstrained minimization a smooth function with inaccurate values. The error between the true value of the objective value and the accessible inaccurate value was assumed to be uniformly upper bounded by a positive accuracy magnitude e. We proved that the gradient

norm achieved is of the order $\mathcal{O}(\sqrt{e})$, and is attained within $\mathcal{O}(e^{-1})$ iterations. We observed that the choice of the initial step size α_0 is essential to the performance of the algorithm. Moreover the gradient norm is driven below a positive ϵ within $\mathcal{O}(\epsilon^{-2})$ iterations as long as ϵ is not lower than the achievable bound. Thus the inaccuracy on the function values does not affect the worst case complexity of the algorithm as long as the requirement on the optimality is realistic. Based on these results we proposed a heuristic algorithm for situations when a family of oracles with various accuracy magnitudes is at hand. This algorithm aims at achieving the accuracy magnitude of the finest oracle, while taking advantage of the other coarser oracles to alleviate the computational cost, which is assumed directly related to the accuracy.

References

[1] C. Audet and J. E. Dennis Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM J. Optim.*, 17:188–217, 2006.

[2] C. Audet and J. E. Dennis Jr. Analysis of generalized pattern searches. *SIAM J. Optim.*, 13:889–903, 2002.

[3] C Bogani, M. G. Gasparo, and A. Papini. Generating set search methods for piecewise smooth problems. *SIAM J. Optim.*, 20:321–335, 2009.

[4] J. Borggaard, D. Pelletier, and K. Vugrin. On sensitivity analysis for problems with numerical noise. AIAA Paper 2002–5553, presented at the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, GA, 2002.

[5] D. M. Bortz and C. T. Kelley. The simplex gradient and noisy optimization problems. In J. Borggaard, J. Burns, E. Cliff, and S. Schreck, editors, *Computational methods for optimal design and control*, volume 24 of *Progress in Systems and Control Theory*, pages 77–90. Springer Science+Business Media, New York, 1998.

[6] R. G. Carter. On the global convergence of trust region algorithms using inexact gradient information. *SIAM J. Numer. Anal.*, 28:251–265, 1991.

[7] R. G. Carter. Numerical experience with a class of algorithms for nonlinear optimization using inexact function and gradient information. *SIAM J. Sci. Comput.*, 14, 1993.

[8] T. D. Choi and C. T. Kelley. Superlinear convergence and implicit filtering. *SIAM J. Optim.*, 10:1149–1162, 2000.

[9] A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to Derivative-Free Optimization.* MPS-SIAM Series on Optimization. SIAM, Philadelphia, 2009.

[10] A. R. Conn and L. N. Vicente. Bilevel derivative-free optimization and its application to robust optimization. *Optim. Methods Softw.*, 27:561–577, 2012.

[11] A. d'Aspremont. Smooth optimization with approximate gradient. *SIAM J. Optim.*, 19:1171–1183, 2008.

[12] C. Davis. Theory of positive linear dependence. *Amer. J. Math.*, 76:733–746, 1954.

[13] O. Devolder, F. Glineur, and Y. Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Math. Program.*, 146:37–75, 2014.

[14] M. Dodangeh and L. N. Vicente. Worst case complexity of direct search under convexity. *Math. Program.*, to appear.

[15] T. Glad and A. Goldstein. Optimization of functions whose values are subject to small errors. *BIT*, 17:160–169, 1977.

[16] S. Gratton, C. W. Royer, L. N. Vicente, and Z. Zhang. Direct search based on probabilistic descent. *SIAM J. Optim.*, 25:1515–1541, 2015.

[17] M. Heinkenschloss and L. N. Vicente. Analysis of inexact trust-region SQP algorithms. *SIAM J. Optim.*, 12:283–302, 2002.

[18] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, second edition, 2002.

[19] C. T. Kelley. *Iterative Methods for Optimization*, volume 18 of *Frontiers in Applied Mathematics*. SIAM, Philadelphia, 1999.

[20] K. C. Kiwiel. A proximal bundle method with approximate subgradient linearizations. *SIAM J. Optim.*, 16:1007–1023, 2006.

[21] T. G. Kolda, R. M. Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Rev.*, 45:385–482, 2003.

[22] S. Le Digabel. Algorithm 909: NOMAD: Nonlinear optimization with the MADS algorithm. *ACM Trans. Math. Software*, 37:44:1–44:15, 2011.

[23] S. Lucidi and M. Sciandrone. On the global convergence of derivative-free methods for unconstrained optimization. *SIAM J. Optim.*, 13:97–116, 2002.

[24] J. J. Moré. Recent developments in algorithms and software for trust region methods. In A. Bachem, M. Grötschel, and B. Korte, editors, *Mathematical Programming: The State of Art*. Springer-Verlag, Berlin, 1983.

[25] J. J. Moré and S. M. Wild. Benchmarking derivative-free optimization algorithms. *SIAM J. Optim.*, 20:172–191, 2009.

[26] A. Nedić and D. P. Bertsekas. The effect of deterministic noise in subgradient methods. *Math. Program.*, 125:75–99, 2010.

[27] J. A. Nelder and R. Mead. A simplex method for function minimization. *Comput. J.*, 7:308–313, 1965.

[28] B. T. Polyak. Nonlinear programming methods in the presence of noise. *Math. Program.*, 14:87–97, 1978.

[29] M. J. D. Powell. A view of algorithms for optimization without derivatives. Technical Report DAMTP 2007/NA03, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, 2007.

[30] M. V. Solodov and S. K. Zavriev. Error stability properties of generalized gradient-type algorithms. *J. Optim. Theory Appl.*, 98:663–680, 1998.

[31] Ph. L. Toint. Global convergence of aa of trust-region methods for nonconvex minimization in hilbert space. *IMA J. Numer. Anal.*, 8:231–252, 1988.

[32] V. Torczon. On the convergence of pattern search algorithms. *SIAM J. Optim.*, 7:1–25, 1997.

[33] L. N. Vicente. Worst case complexity of direct search. *EURO Journal on Computational Optimization*, 1:143–153, 2013.

[34] L. N. Vicente and A. L. Custódio. Analysis of direct searches for discontinuous functions. *Math. Program.*, 133:299–325, 2012.

[35] K. Vugrin. On the effect of numerical noise in simulation-based optimization. Master's thesis, Virginia Polytechnic Institute and State University, 2003.

[36] M. H. Wright. Direct search methods: Once scorned, now respectable. In D. F. Griffiths and Waston G. A., editors, *Numerical Analysis 1995*, volume 344 of *Pitman Research Notes in Mathematics Series*, pages 191–208. Longman, 1996.

3.2 Expériences numériques

Dans cette section nous testons numériquement les résultats obtenus sur l'algorithme de recherche directe directionnelle ainsi que l'algorithme adaptatif présentés dans la section précédente. Nous considérons d'abord des problèmes de l'environnement CUTEr (*Constrained and Unconstrained Testing Environment, revisited* [10]), puis un problème de placement de puits pétrolier avec un oracle inexact, et enfin un problème de minimisation de surface avec trois oracles de précisions distinctes.

3.2.1 Applications à des fonctions de test

Nous mettons ici en œuvre l'algorithme de recherche directe directionnelle et sa variante adaptative sur des problèmes d'optimisation sans contrainte issus de l'environnement CUTEr — *Constrained and Unconstrained Testing Environment, revisited* [10]. Les fonctions objectifs sont normalisées de sorte qu'elles prennent leurs valeurs dans l'intervalle $[0, 1]$, la valeur 0 étant optimale.

Paramètres Dans chaque expérience nous choisissons les valeurs de paramètres $c = 1$, $\gamma = 1$, $\theta = 0,5$, et $\lambda = 0$. Le pas initial de l'algorithme est fixé à $\alpha_0 = 1$, qui est supérieur à $2\sqrt{\theta e}/\sqrt{c+\lambda} = \sqrt{2e}$ pour des niveaux d'erreur e inférieurs à $0,5$ — ici nous considérerons 2^{-12}, 2^{-15} et 2^{-18}—, de sorte que la condition de notre résultat d'arrêt est respectée. Lors des tests sur des fonctions bruitées l'algorithme est arrêté lorsque le critère d'arrêt sur le pas est atteint. Si la fonction n'est pas bruitée nous considérerons le critère d'arrêt correspondant à un niveau d'erreur de 2^{-21}. Dans tous les cas le budget d'évaluations de f est limité à $2000n$. Par ailleurs les éléments de l'ensemble $\mathcal{D}$ des directions de recherche sont les vecteurs $d_1, \ldots, d_n$ de la base canonique de $\mathbb{R}^n$ et leurs opposés $-d_1, \ldots, -d_n$. À la première itération les directions sont testées selon l'ordre $d_1, \ldots, d_n, -d_1, \ldots, -d_n$. Si l'itération est un succès dans la direction d_i, l'itération suivante commence par tester de nouveau la direction d_i (avec un pas multiplié par γ), autrement l'itération débute avec la direction suivante dans l'ordre $d_1, \ldots, d_n, -d_1, \ldots, -d_n$.

Convergence Intéressons-nous dans un premier temps à l'influence d'un bruit borné sur l'algorithme de recherche directe directionnelle. Nous considérons les problèmes `bdqrtic`, `vardim` et `arglina`, dont nous normalisons les valeurs des fonctions objectif pour les ramener dans l'intervalle $[0, 1]$. Nous affectons ces valeurs par un bruit borné par $e = 10^{-1}$ de la façon suivante : à chaque évaluation de la fonction objectif f en un point x de $\mathbb{R}^n$ est ajoutée une erreur de ξe, où ξ est tiré uniformément dans $[-1, 1]$, de sorte que $F(x) = f(x) + \xi e$. Une réalisation de l'évolution de la valeur de F au point courant au cours de l'algorithme est représentée sur les Figures 3.1, 3.2 et 3.3 pour les problèmes `bdqrtic`, `vardim` et `arglina` respectivement. Observons que la convergence de l'algorithme n'est pas retardée par le bruit.
Pour chacun de ces trois problèmes nous estimons la norme du gradient de f à l'itéré final par la technique du pas complexe [74] : $\|g_k\|$ est approché par $|\Im(f(x+iv))|/\|v\|$. Chaque

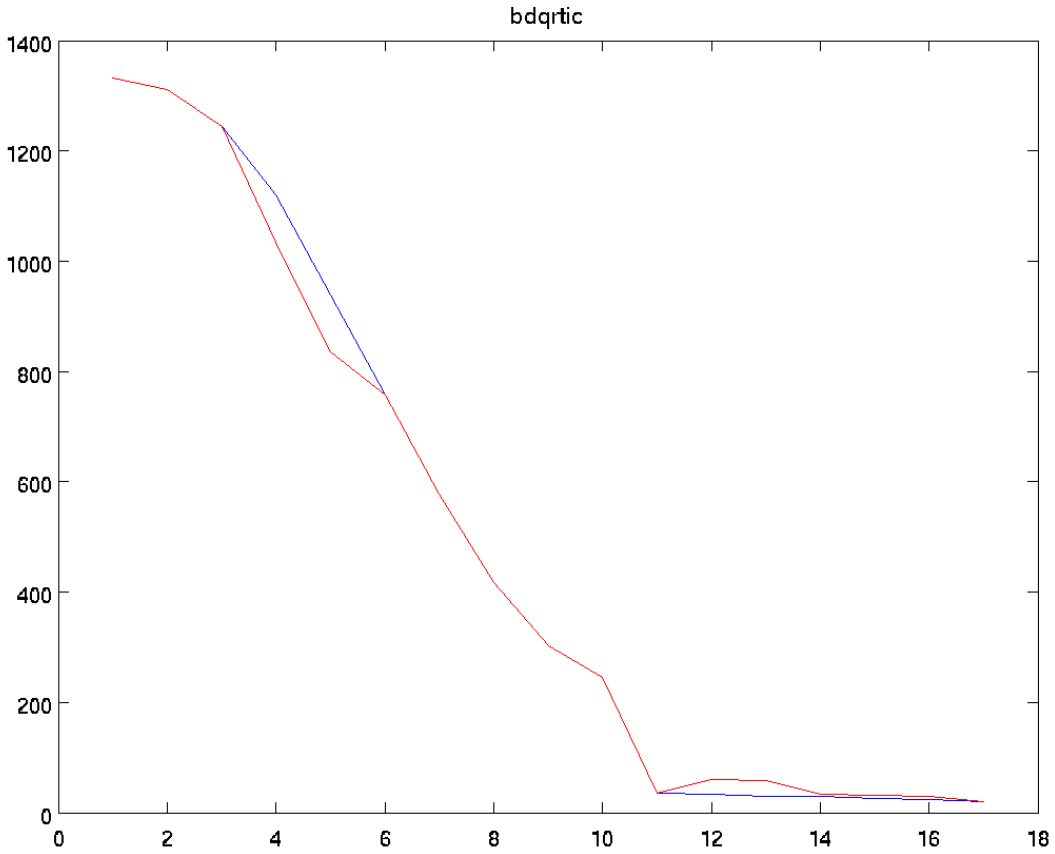

Figure 3.1 – Historique de convergence pour le problème `bdqrtic` non bruité, en bleu, et soumis à un bruit borné par $e = 0.1$, en rouge.

problème est résolu dix fois et la valeur moyenne de cette norme est représentée en fonction de l'ordre de grandeur de l'erreur, qui varie entre 10^{-1} et 10^{-16}, pour les problèmes `bdqrtic`, `vardim` et `arglina` sur les Figures 3.4, 3.5 et 3.6 respectivement. Ces résultats sont en accord avec la théorie, qui donne un majorant sur la norme du gradient proportionnel à la racine carrée de l'erreur, donc représentés par une droite parallèle à la droite tracée en rouge sur les Figures 3.4, 3.5 et 3.6.

Comparaison entre le cas adaptatif et le cas fin Comparons à présent les performances entre un algorithme adaptif à trois oracles d'ordres de précision différents — que nous qualifierons de "grossier", "modéré" et "fin" par ordre croissant de précision — sur 27 problèmes de l'environnement CUTEr. Chacun de ces problèmes est résolu de trois façons différentes :

1. par une approche adaptative à trois oracles donnant des valeurs approchées de f avec des précisions grossière $e_1 = 2^{-12}$, modérée $e_2 = 2^{-15}$ et fine $e_3 = 2^{-18}$;
2. par recherche directe directionnelle classique avec l'oracle de précision fine ;
3. par recherche directe directionnelle avec la véritable valeur de f (précision parfaite).

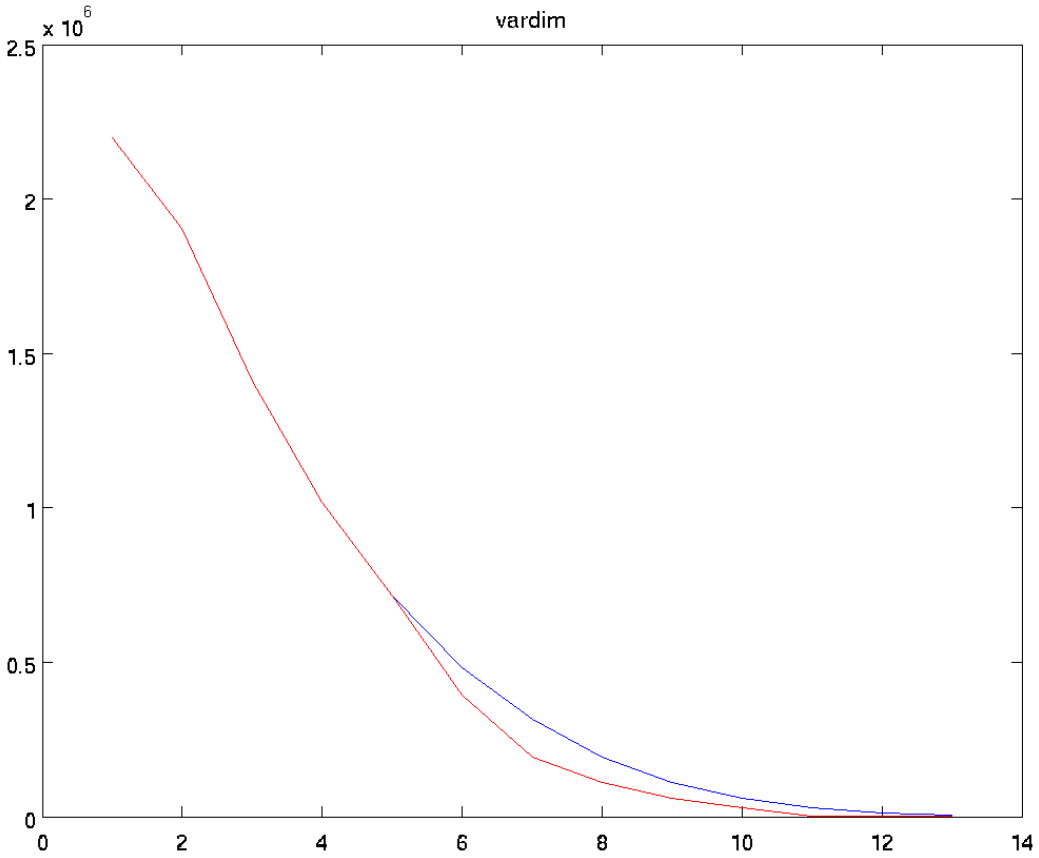

Figure 3.2 – Historique de convergence pour le problème `vardim` non bruité, en bleu, et soumis à un bruit borné par $e = 0.1$, en rouge.

La valeur de la fonction objectif au dernier itéré pour chacune de ces trois approches est présentée dans la Table 3.1. Remarquons à la lecture des troisième et quatrième colonnes que, sur les 27 problèmes considérés, les valeurs finales obtenues par l'algorithme adaptatif et l'algorithme classique avec l'oracle fin sont du même ordre de grandeur dans 10 cas, et éloignées d'un ordre de grandeur seulement dans 9 cas. Ici la précision de l'oracle de précision fine est telle que les résultats obtenus avec cet oracle seul sont sensiblement les mêmes que ceux obtenus avec un oracle parfait (voir les deux dernières colonnes de la Table 3.1).
La norme du gradient au dernier itéré obtenu par chacune des trois expériences est présentée dans la Table 3.2. En comparant les troisième et quatrième colonnes de celle-ci nous observons que l'algorithme adaptatif permet d'atteindre un gradient de norme du même ordre grandeur que celui obtenu par l'algorithme classique appliqué à l'oracle de meilleure précision dans 20 cas sur 27. Les Tables 3.1 et 3.2 montrent que l'algorithme adaptatif à trois oracles permet en général d'obtenir des résultats comparables à la recherche directe directionnelle appliquée à l'oracle le plus fin. Il reste à voir si adopter une telle stratégie est avantageux en termes de coût de calcul. Nous mesurons ce coût à l'aide du nombre d'évaluations des différents oracles, qui sont supposés d'autant plus exigeants en ressources processeur qu'ils sont précis..

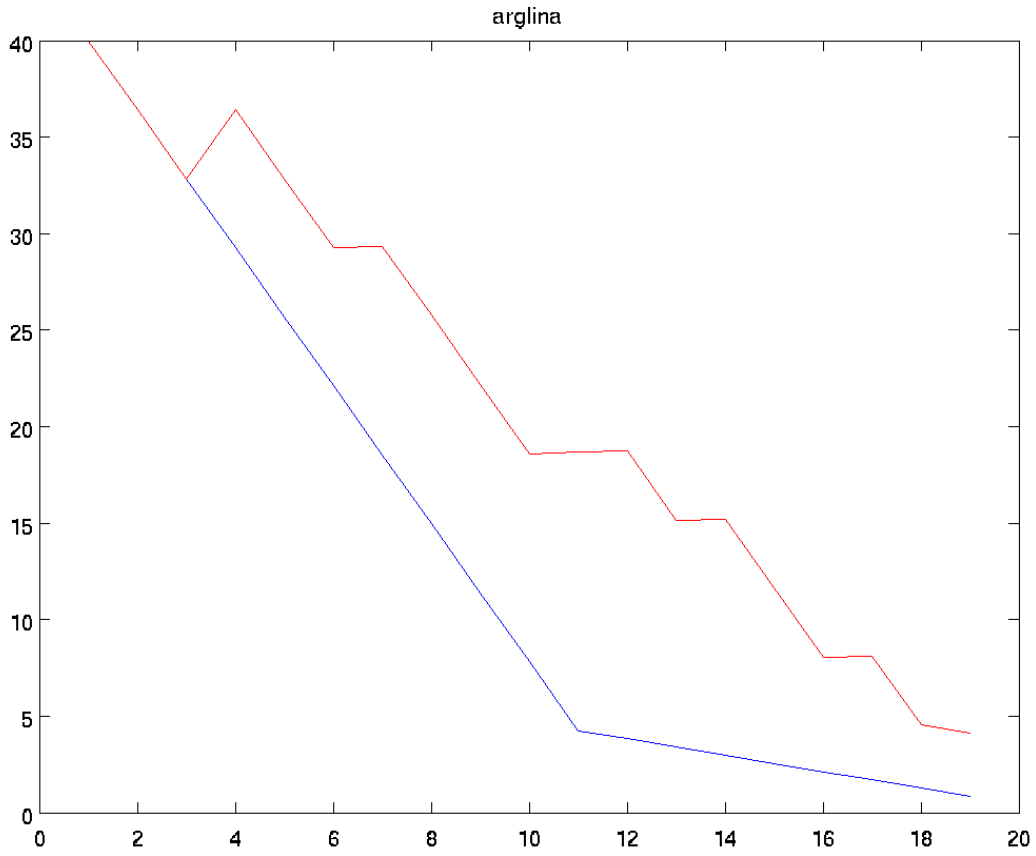

Figure 3.3 – Historique de convergence pour le problème `arglina` non bruité, en bleu, et soumis à un bruit borné par $e = 0.1$, en rouge.

Nous représentons le nombre d'appels aux différents oracles au cours de chacune des trois expériences dans la Table 3.3. La Table 3.3 montre que l'algorithme adaptatif et l'algorithme classique appliqué à l'oracle fin recquièrent des nombres d'évaluations comparables. Cependant, dans la grande majorité des problèmes testés, la quantité d'appels à l'oracle fin effectués par l'algorithme adaptatif est moindre comparée au nombre d'évaluations réclamées par l'algorithme classique appliqué à l'oracle de meilleure précision. À supposer que les oracles de précisions grossière et modérée nécessitent significativement moins de ressources processeur que l'oracle fin, ceci se traduit par un temps de calcul plus intéressant pour l'algorithme adaptatif tout en obtenant des résultats satisfaisants (voir Tables 3.1 et 3.2).

3.2.2 Placement de puits pétrolier

Placement de puits Le placement de puits sur un champ pétrolifère est un problème classique en ingénierie de réservoir [13, 14]. Il consiste à choisir les emplacements d'un ou plusieurs puits injecteurs d'eau (favorisant l'écoulement de l'huile dans le réservoir) ou pro-

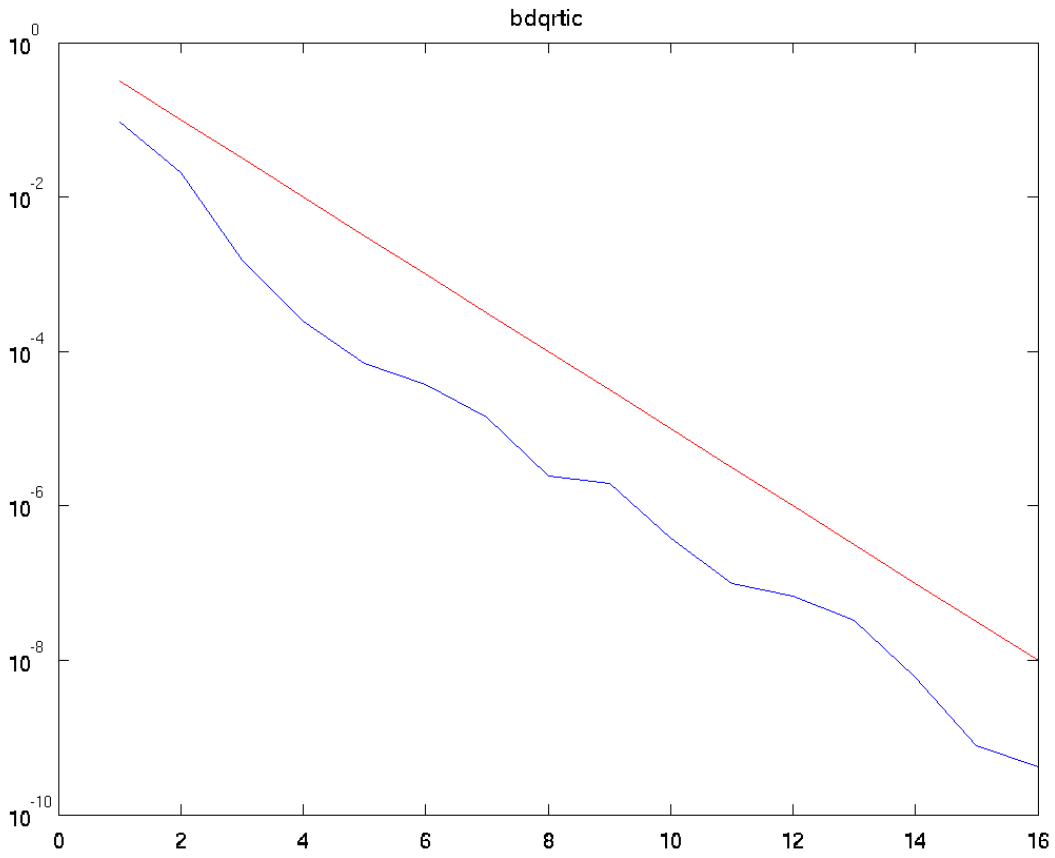

Figure 3.4 – En bleu : valeur estimée de la norme du gradient du dernier itéré $\|g_k\|$ en fonction de p, où $e = 10^{-p}$, en échelle semi-logarithmique ; en rouge : la droite de pente -0.5 passant par l'origine.

ducteurs (d'huile et d'eau) dans le but de maximiser la production d'huile du champ sur un intervalle de temps donné. Nous caractériserons un puits par un couple de coordonnées cartésiennes horizontales (x_1, x_2) définissant son emplacement à la surface du champ pétrolifère, modélisée par une partie $\mathcal{X}$ de $\mathbb{R}^2$. Dans le cas du champ que nous étudierons, présenté plus bas, l'ensemble réalisable $\mathcal{X}$ sera un rectangle de $\mathbb{R}^2$. Nous supposerons qu'un puits injecteur d'eau est déjà présent au centre du champ et nous chercherons à placer un puits producteur décrit par une variable $x = (x_1, x_2)$ dans $\mathcal{X}$ maximisant la production du champ sur dix ans. Le critère à maximiser est la *valeur actuelle nette*, un indicateur économique permettant d'attribuer une valeur dans le présent à des flux d'argent futurs. Le forage du puits a un coût fixe C_{W} et un coût $C_{\mathrm{WL}}\ell$ proportionnel à sa longueur ℓ, qui ne sera pas considérée ici comme un paramètre de décision. Chaque année le puits injecteur d'eau occasionne un coût de fonctionnement $C_{\mathrm{WI}}V_{\mathrm{WI},k}$ proportionnel au volume d'eau injecté $V_{\mathrm{WI},k}$ durant l'année k (avec k dans $\{1, \ldots, 10\}$). Le puits producteur induit un coût annuel de production d'eau $C_{\mathrm{WP}}V_{\mathrm{WP},k}$ et un coût (ou plutôt un revenu) annuel de production d'huile $C_{\mathrm{OP}}V_{\mathrm{OP},k}$ proportionnels aux volumes d'eau $V_{\mathrm{WP},k}$ et d'huile $V_{\mathrm{OP},k}$ produits respectivement. Ces volumes produits dépendent

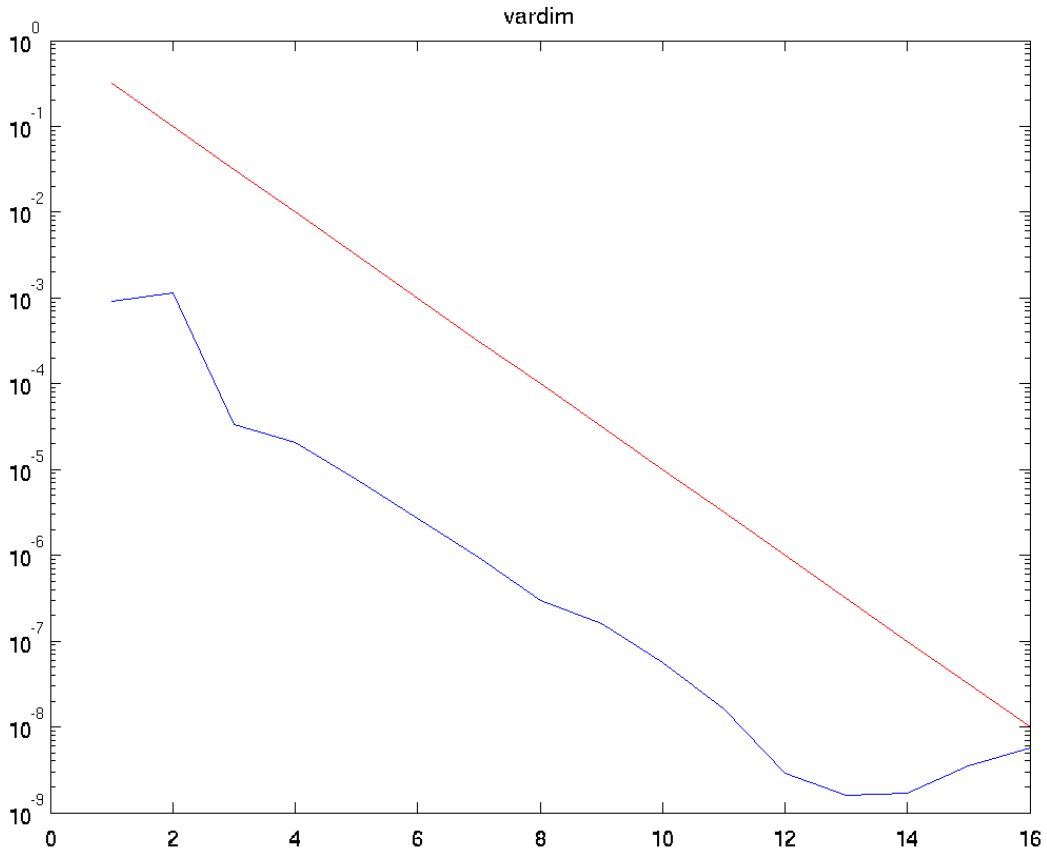

Figure 3.5 – En bleu : valeur estimée de la norme du gradient du dernier itéré $\|g_k\|$ en fonction de p, où $e = 10^{-p}$, en échelle semi-logarithmique ; en rouge : la droite de pente -0.5 passant par l'origine.

naturellement de l'emplacement x du puits producteur. De plus, nous ferons intervenir un vecteur aléatoire u de $\mathbb{R}^2$ — défini plus bas — qui affectera la géologie du réservoir et dont dépendront donc les volumes V_{WP} et V_{OP}. Chacun des trois coûts cités précédemment est multiplié par un facteur d'actualisation $(1+R)^{-k}$, où R est un *taux d'actualisation* compris entre 0 et 1, d'autant plus faible que l'annuité k est éloignée dans le temps. En effet plus un flux d'argent est éloigné dans la futur moins nous lui accordons de valeur dans le présent. La valeur actuelle nette (aléatoire du fait de u) est ainsi donnée par

$$f(x;u) = -C_{\mathrm{W}} - C_{\mathrm{WL}}\ell \log \ell - C_{\mathrm{WI}} \sum_{k=1}^{10} \frac{V_{\mathrm{WI},k}(x;u)}{(1+R)^k} - C_{\mathrm{WP}} \sum_{k=1}^{10} \frac{V_{\mathrm{WP},k}(x;u)}{(1+R)^k} + C_{\mathrm{OP}} \sum_{k=1}^{10} \frac{V_{\mathrm{OP},k}(x;u)}{(1+R)^k}, \tag{3.1}$$

pour tout x de $\mathcal{X}$. Si x est un vecteur du complémentaire de $\mathcal{X}$ dans $\mathbb{R}^2$, correspondant à un emplacement hors du champ pétrolier, nous attribuerons une valeur positive infinie presque sûrement à $f(x;u)$. Les valeurs des constantes économiques apparaissant dans la définition (3.1) sont précisées dans la Table 3.4. Nous sommes donc face à un problème de

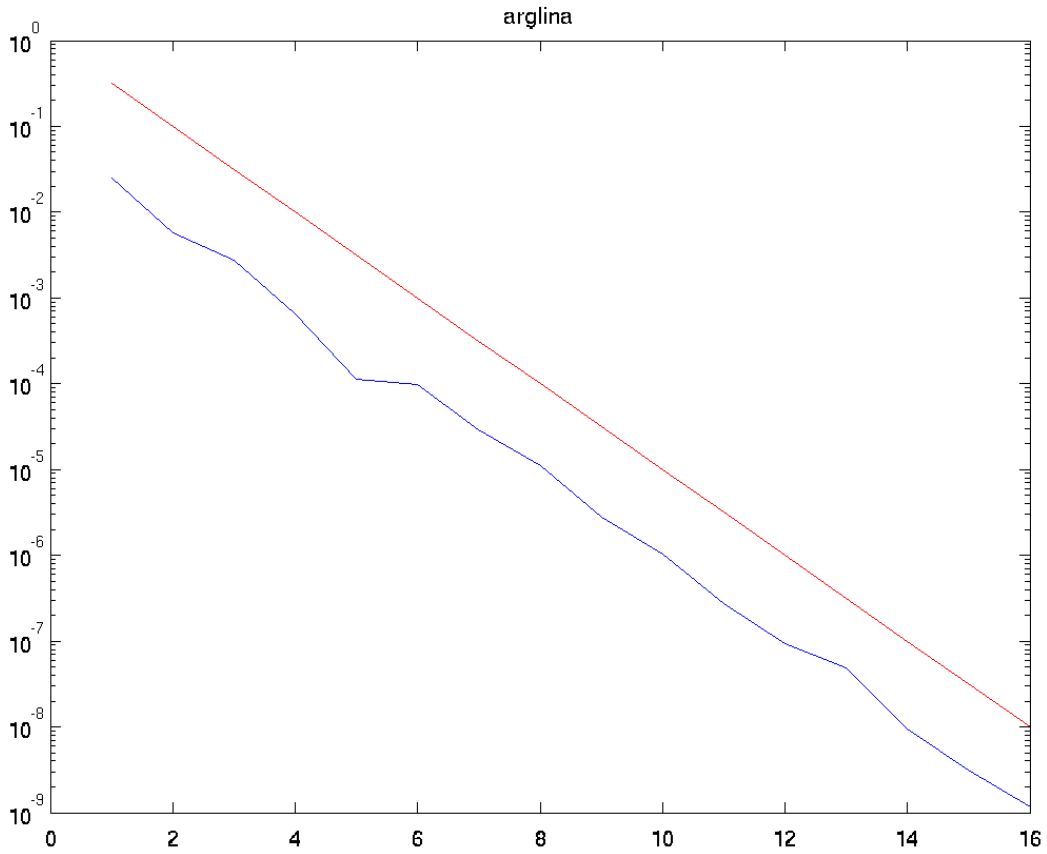

Figure 3.6 – En bleu : valeur estimée de la norme du gradient du dernier itéré $\|g_k\|$ en fonction de p, où $e = 10^{-p}$, en échelle semi-logarithmique ; en rouge : la droite de pente -0.5 passant par l'origine.

maximisation d'une fonction à valeurs aléatoires $f(x; u)$ sous la contrainte $x \in \mathcal{X}$.

Cas SPE10 Le modèle géologique que nous allons étudier est le cas SPE10. Il s'agit d'un exemple de référence proposé lors du dixième projet comparatif de méthodes d'*upscaling* organisé en 2001 [24] par la SPE (*Society of Petroleum Engineers*). Nous nous intéressons à la partie inférieure du modèle géologique, de dimensions $1200 \times 2200 \times 100$ — largeur × longueur ×profondeur en pieds. Le modèle initial est constitué d'une grille cartésienne de $60 \times 220 \times 50$ cellules. La porosité ϕ du modèle — fraction du volume d'une cellule non occupée par la roche, où les fluides peuvent s'écouler — est représentée sur la Figure 3.7. Supposons qu'un système composé d'eau et d'huile liquides, supposées incompressibles, s'écoule dans les pores du réservoir, dont la roche est également supposée incompres-

Problème	n	Adaptif	Fin	Non bruité
arglina	10	4.06e−4	0.00e+0	0.00e+0
arglinb	10	2.26e+0	2.14e+0	2.14e+0
arglinc	10	3.76e+0	3.65e+0	3.65e+0
arwhead	10	4.58e−5	0.00e+0	0.00e+0
bard	3	9.05e−3	9.01e−3	9.01e−3
bdqrtic	10	1.20e+1	1.19e+1	1.19e+1
box3d	3	1.83e−5	0.00e+0	0.00e+0
brownal	10	3.26e−3	7.91e−4	7.91e−4
broydn3d	10	5.59e−4	3.87e−5	3.87e−5
cube	10	7.03e−2	3.11e−4	3.11e−4
dqrtic	10	2.78e+0	0.00e+0	0.00e+0
engval1	10	9.20e+0	9.18e+0	9.18e+0
freuroth	10	1.19e+3	1.18e+3	1.18e+3
heart8ls	8	3.57e+0	1.07e+0	1.07e+0
helical	4	1.34e−1	1.08e−2	1.08e−2
integreq	10	1.04e−5	1.04e−6	1.04e−6
kowosb	4	2.23e−3	6.88e−4	6.88e−4
mancino	10	6.67e+7	1.49e+0	1.49e+0
meyer	3	3.89e+9	7.14e+6	7.14e+6
nondquar	10	4.99e−3	2.29e−4	2.29e−4
osborne1	5	1.11e+0	1.11e+0	1.11e+0
osborne2	11	4.90e−1	4.03e−2	4.03e−2
powellsing	4	2.16e−4	0.00e+0	0.00e+0
rosenbrock	2	0.00e+0	0.00e+0	0.00e+0
sinquad	10	1.24e−1	3.86e−3	3.86e−3
vardim	10	3.22e+2	1.70e−3	1.70e−3
watson	10	5.28e+0	1.37e−1	1.37e−1

Table 3.1 – Valeur de f au dernier itéré.

sible. Cet écoulement peut être décrit par le système d'équations couplées ci-dessous [22,63] :

$$\nabla \cdot v = \frac{q_w}{\rho_w} + \frac{q_o}{\rho_o}, \qquad v = -K[\lambda \nabla p - (\lambda_w \rho_w + \lambda_o \rho_o) G] \tag{3.2}$$

$$\phi \frac{\partial s_w}{\partial t} + \nabla \cdot (f_w [v + \lambda_o (\rho_w - \rho_o) K G]) = \frac{q_w}{\rho_w}. \tag{3.3}$$

Les différents paramètres intervenant dans ce système sont décrits dans la Table 3.5. Les hypothèses et l'établissement de ces équations sont décrit dans [1]. Les variables (fonctionnelles) du système (3.2)-(3.3) sont la saturation en eau s_w — la fraction du volume des pores de la roche occupée par l'eau —, et la pression globale p, qui est une fonction de s_w. Par ailleurs, les perméabilités relatives k_{rw} et k_{ro} dépendent également de la saturation en eau, et donc λ_w, λ_0, λ et f_w aussi. Les perméabilités absolues K et la porosité ϕ sont des caractéristiques de la roche supposées constantes relativement à p et s_w (incompressibilité de la roche). De même les masses volumiques, ρ_w et ρ_o, et les viscosités, μ_w et μ_o, sont supposées constantes

Problème	n	Adaptif	Fin	Non bruité
arglina	10	1.92e−4	7.78e−5	0.00e+0
arglinb	10	3.96e−5	5.89e−5	8.72e−7
arglinc	10	1.01e−4	5.19e−5	3.11e−6
arwhead	10	2.13e−4	0.00e+0	0.00e+0
bard	3	6.73e−4	3.53e−4	6.57e−5
bdqrtic	10	3.05e−4	1.13e−4	3.85e−6
box3d	3	2.67e−4	1.50e−4	0.00e+0
brownal	10	1.02e−4	1.14e−4	3.67e−5
broydn3d	10	2.97e−3	2.73e−3	8.74e−4
cube	10	1.79e−7	1.74e−7	2.52e−9
dqrtic	10	1.89e−4	2.39e−7	0.00e+0
engval1	10	3.59e−4	1.16e−4	3.66e−6
freuroth	10	2.32e−4	2.02e−4	4.18e−5
heart8ls	8	6.55e−3	7.47e−3	2.54e−3
helical	4	1.81e−2	1.58e−2	1.78e−3
integreq	10	2.55e−2	3.17e−2	8.18e−3
kowosb	4	2.69e−3	2.61e−3	1.30e−3
mancino	10	5.97e−5	4.18e−6	9.42e−9
meyer	3	5.31e−5	6.63e−5	2.86e−6
nondquar	10	5.46e−4	4.17e−4	6.16e−5
osborne1	5	2.13e−3	2.10e−3	4.25e−4
osborne2	11	3.13e−3	2.98e−4	4.71e−5
powellsing	4	5.83e−4	4.26e−4	0.00e+0
rosenbrock	2	0.00e+0	0.00e+0	0.00e+0
sinquad	10	1.43e−2	1.18e−2	8.59e−3
vardim	10	2.09e−5	3.54e−6	9.43e−9
watson	10	4.33e−4	1.14e−4	3.77e−5

Table 3.2 – Estimation de la norme du gradient f au dernier itéré.

vis-à-vis de p et s_w (incompressibilité des fluides). Enfin les flux q_w et q_o, entrant et sortant du réservoir, sont commandés au niveau des puits. Un puits injecteur d'eau est déjà placé au centre du champ et l'emplacement d'un puits producteur est donné par x. Le système d'équations (3.2)-(3.3) est couplé. Nous le résolvons à l'aide de l'implémentation MRST (*MATLAB Reservoir Simulation Toolbox*) [62]. La stratégie adoptée consiste non pas à résoudre directement le système couplé, mais à résoudre dans premier temps l'Équation (3.2) en fonction de la pression p, puis d'injecter le résultat dans l'Équation (3.3) et de la résoudre par rapport à la saturation s_w. Cette résolution est effectuée sur chacun des 10 intervalles d'un an qui composent la durée d'exploitation étudiée du champ. Un fois le système résolu pour l'année k, nous déduisons de la solution les volumes d'eau injectée $V_{\mathrm{WI},k}$, d'eau produite V_{WP} et d'huile produite V_{OP}, et donc la valeur actuelle nette (3.1). En pratique nous observons que le temps nécessaire à la résolution du système est proportionnel au nombre de cellules de la grille. Elle est initialement composée de $60 \cdot 220 \cdot 50 = 660000$ cellules. Nous effectuons un *upscaling* afin de réduire ce nombre et de diminuer le temps de calcul. Nous nous ramenons ainsi à une grille

Problème	n	Oracles multiples				Oracle simple	
		grossier	modéré	fin	total	fin	parfait
arglina	10	287.9	73.5	57.9	419.3	343.1	376
arglinb	10	442.4	74.4	40.5	557.3	537.9	591
arglinc	10	414.2	55.5	44.2	513.9	441.9	389
arwhead	10	142.5	54.3	37.6	234.4	206.8	240
bard	3	63.8	27.7	14.9	106.4	93.7	108
bdqrtic	10	303.9	122.9	135.1	561.9	448.9	506
box3d	3	52.5	18.9	13.4	84.8	71.8	82
brownal	10	1295	174	194	1663	1661	3054
broydn3d	10	266.3	123.9	64.8	455	439.8	558
cube	10	3994	95	46	4135	4133	4243
dqrtic	10	357.9	69.8	93.7	521.4	351	367
engval1	10	474.3	140.3	238.4	853	601.4	772
freuroth	10	457.8	78.7	95.1	631.6	729.5	1169
heart8ls	8	367.5	254.6	626.2	1248.3	1449.3	15988
helical	4	98.2	160.1	693.9	952.2	906.9	1851
integreq	10	189	92.9	35.1	317	315	383
kowosb	4	64.6	84.1	310.3	459	339.9	3544
mancino	10	4116.1	54.1	36.7	4206.9	2686.1	2983
meyer	3	45.9	18.2	12	76.1	3173	5994
nondquar	10	362.6	323.2	822.7	1508.5	945	4359
osborne1	5	104.6	29.6	19.1	153.3	136.8	154
osborne2	11	660.3	583.2	1938.5	3182	1453.5	19798
powellsing	4	70.9	26.5	23	120.4	92.3	102
rosenbrock	2	28	9	5	42	40	48
sinquad	10	165	67	98.9	330.9	317.7	12156
vardim	10	179	64.8	50.2	294	308.4	13386
watson	10	400.1	230.4	1729	2359.5	1620.2	19984

Table 3.3 – Nombre d'évaluations de la fonction f suivant la méthode employée.

grossière de dimensions $6 \times 22 \times 25$, soit 3300 cellules chacune de dimensions $200 \times 100 \times 4$ en pieds. La porosité du modèle obtenu après changement d'échelle est représentée sur la Figure 3.8.

Paramètres incertains Comme nous l'avons précisé plus haut, nous faisons intervenir un vecteur aléatoire $u = (u_1, u_2)$ de $\mathbb{R}^2$ pour faire appraître artificiellement de l'incertitude dans la géologie du modèle. Nous définissons u_1 et u_2 comme des variables aléatoires réelles indépendantes et uniformément distribuées sur l'intervalle $[1, 10]$. Nous partageons la grille dans le sens de la largeur en deux parties égales et nous multiplions les perméabilités relatives de la première moitié par u_1 et les perméabilités relatives de la seconde moitié par u_2. Les incertitudes ainsi générées sur la géologie du modèle affectent les solutions du système (3.2)-(3.3), et donc les volumes injectés et produits apparaissant dans la définition de la valeur

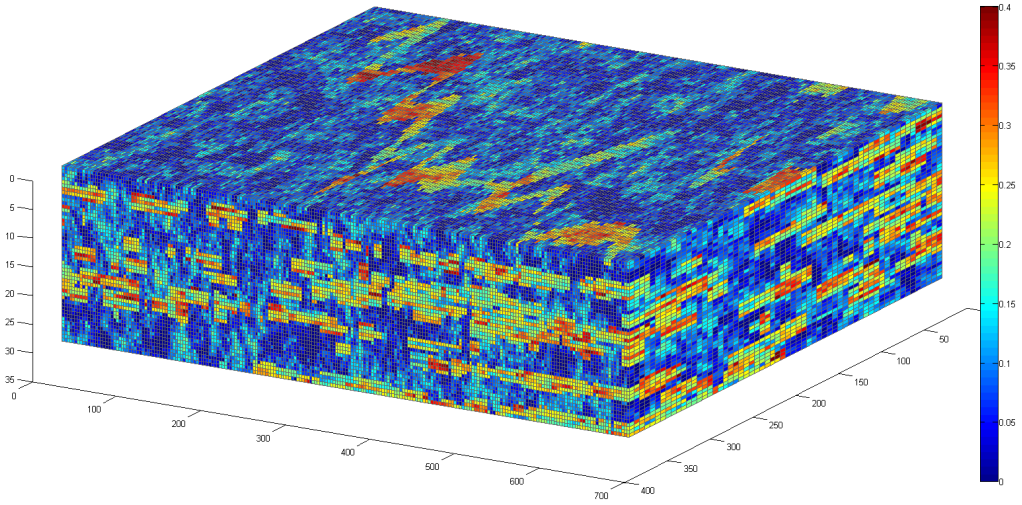

Figure 3.7 – Porosité du modèle géologique SPE10. (L'unité de longueur sur les axes est le mètre.)

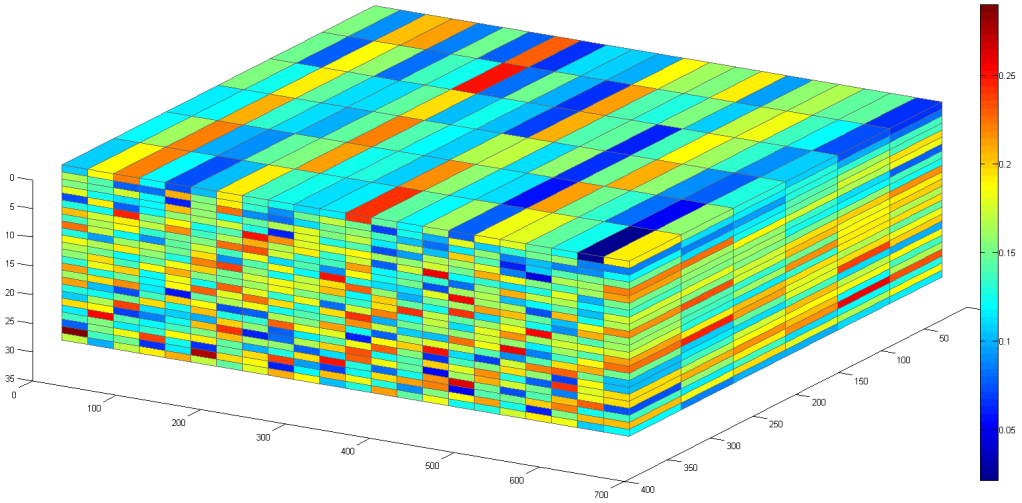

Figure 3.8 – Porosité du modèle simplifié.

C_{W}	Coût fixe de forage	1 M$
C_{WL}	Coût par unité de longueur	45 $ par pied
l	Longueur du puits	100 pieds
C_{WI}	Coût d'injection par baril d'eau	5 $/bbl
C_{OP}	Revenu de production d'huile par baril	50 $/bbl
C_{WP}	Coût de production par baril d'eau	5 $/bbl
R	Taux d'actualisation	$0,1$

Table 3.4 – Valeurs des constantes économiques.

ϕ	porosité de la roche : fraction volumique de vide dans le réservoir
K	matrice 3×3 des perméabilités absolues de la roche
v	vitesse de l'écoulement du système eau-huile
ρ_w, ρ_o	masses volumiques de l'eau et de l'huile
μ_w, μ_o	viscosités, supposées constantes
k_{rw}, k_{ro}	perméabilités relatives, fonctions de la saturation en eau s_w
λ_w, λ_o	mobilités, définies par $\lambda_w = k_{rw}/\mu_w$, $\lambda_o = k_{ro}/\mu_o$
λ	mobilité totale définie par $\lambda = \lambda_w + \lambda_o$
f_w	fonction de s_w définie par $f_w = \lambda_w/\lambda$
q_w	flux d'eau injectée et produite
q_o	flux d'huile produite
G	force de pesanteur

Table 3.5 – Grandeurs physiques apparaissant dans les équations d'écoulement.

actuelle nette (3.1). Afin de prendre en compte cette incertitude nous choisissons l'espérance de la valeur actuelle nette comme fonction objectif : pour tout x dans $\mathbb{R}^2$ nous définissons $g(x) = \mathrm{E}[f(x;u)]$. Nous avons ainsi modélisé le problème de placement de puits par le programme d'optimisation suivant :

$$\begin{aligned} &\text{maximiser} \quad g(x) \\ &\text{s. c.} \quad x \in \mathcal{X}. \end{aligned} \tag{3.4}$$

Oracle et erreur Nous n'avons pas directement accès à l'espérance de la valeur actuelle nette pour un vecteur x fixé. Nous l'approchons par une simple méthode de Monte Carlo : donnons-nous un échantillon $u^1, \ldots, u^N$ de tirages indépendants de la loi du vecteur aléatoire u et à chaque évaluation en un point x de $\mathcal{X}$ approchons $f(x;u)$ par la moyenne des $f(x;u^i)$ pour $i = 1, \ldots, N$. Notre oracle bruité G est ainsi défini pour tout x dans $\mathbb{R}^2$ par

$$G(x) = \begin{cases} \frac{1}{N}\sum_{i=1}^{N} f(x;u^i) & \text{si } x \in \mathcal{X}, \\ -\infty & \text{sinon.} \end{cases}$$

Nous estimons le niveau d'erreur grâce à l'inégalité de Hoeffding [47] : pour tout x dans $\mathcal{X}$ et tout réel positif e,

$$\mathrm{P}(|G(x) - g(x)| \leq e) \geq 1 - \exp(-2Ne^2). \tag{3.5}$$

Fixons les valeurs $e = 0,1$ et $N = 150$. La propriété (3.5) assure que l'inégalité $|G(x) - g(x)| \leq e$ est vérifiée avec une probabilité supérieure à $0,95$.

Résolution À présent que nous disposons d'un oracle bruité G et d'une borne e pour le problème (3.4), nous lui appliquons l'algorithme de recherche directe directionnelle avec les même paramètres que précédemment (donnés au second paragraphe de la Section 3.2.1). Pour simplifier le compte-rendu de l'algorithme nous parlerons de cellule x sur la grille de taille 6×22 représentant la surface du champ pétrolifère plutôt que de point de $\mathbb{R}^2$. Sur la Figure 3.9 la première coordonnée est représentée en ordonnée et la seconde en abscisse (de même que sur les Figures 3.7 et 3.8). Le puits injecteur est situé sur la cellule $(3, 11)$ représentée en bleu.

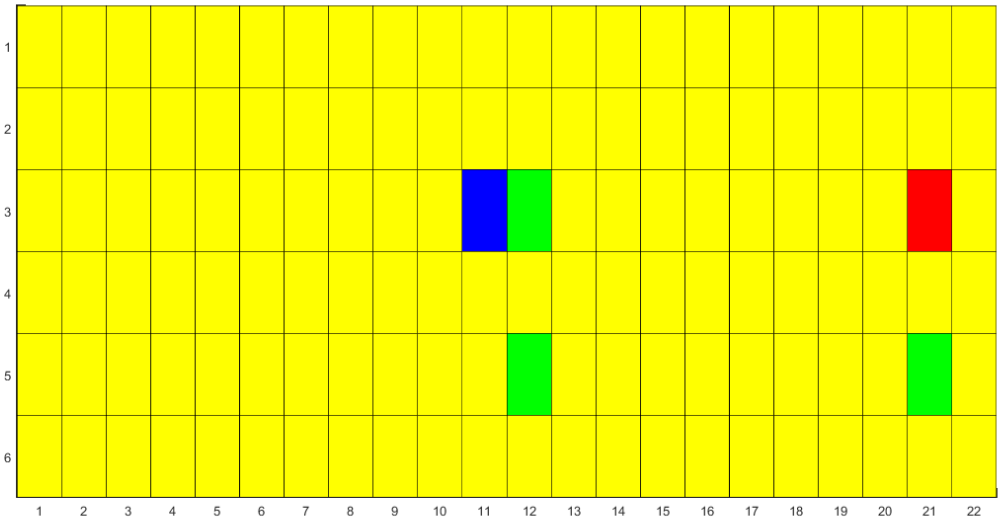

Figure 3.9 – Positions du puits injecteur (en bleu) et des itérés (en vert et rouge) sur le champ pétrolifère.

Nous choisissons comme point de départ x_0 la cellule adjacente $(3, 12)$. Chaque évaluation de G en un point x de $\mathcal{X}$ nécessite un peu moins de 7 minutes en moyenne ; l'algorithme est exécuté en 34 minutes et 30 secondes. La trajectoire de l'algorithme est présentée dans la Table 3.6 et l'évaluation de la valeur en l'itérée sur la Figure 3.10. L'itéré de l'algorithme au moment de l'arrêt est le point $(5, 21)$. La valeur actuelle nette du champ pétrolifère sur 10 ans si le puits producteur est placé en $(5, 21)$ est de 277 802 178 $.

Observons qu'ici la trajectoire de l'algorithme s'est orientée vers la partie droite du champ — tel que représenté sur la Figure 3.9. Un choix différent des lois des paramètres aléatoires pourrait orienter la recherche différemment. Par ailleurs, la discrétisation du réservoir est telle que les dimensions de la surface supérieure d'une cellule sont de 200 pieds sur 100 — soit environ 61 mètres sur 30 —, et nous avons choisi un niveau d'erreur de $e = 0,1$. Une meilleure précision sur l'emplacement du puits à ajouter pourrait être obtenue en utilisant une grille plus fine, et un niveau d'erreur plus bas sur les valeurs de g pourraient être atteint en augmentant la taille N de l'échantillon, au prix d'un temps de calcul plus élevé. Ce coût

Itération	Cellule	Valeur	Test
0	$(3, 12)$	131 M$	(départ)
1	$(5, 12)$	182 M$	succès
2	$(7, 12)$	∞	(non réalisable)
2	$(5, 21)$	278 M$	succès
3	$(5, 30)$	∞	(non réalisable)
3	$(3, 21)$	287 M$	échec
3	$(5, 12)$	182 M$	échec
3	$(7, 21)$	∞	(non réalisable)

Table 3.6 – Trajectoire de la recherche directe directionnelle pour le cas SPE10.

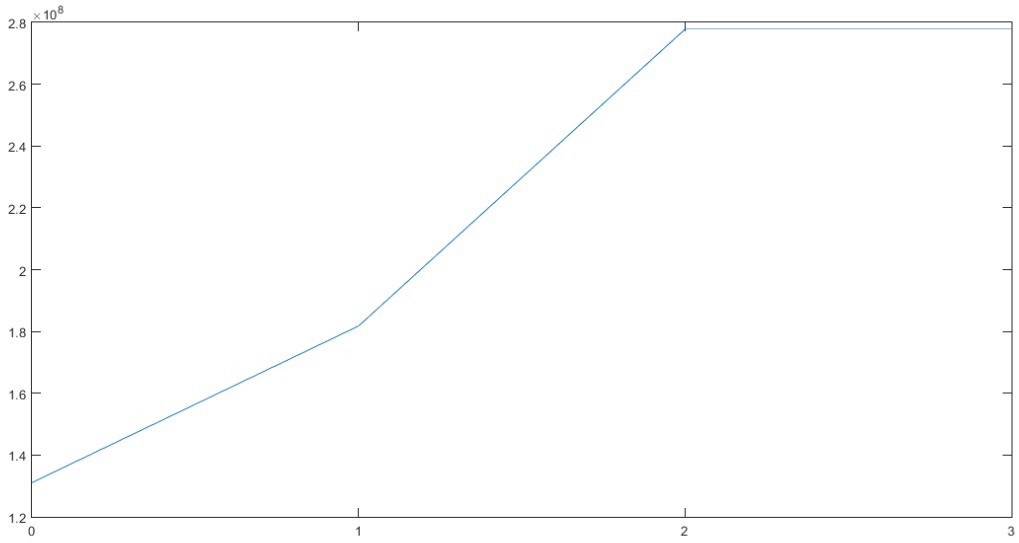

Figure 3.10 – Évolution de la valeur de G en l'itéré.

pourrait être modéré en utilisant notre algorithme adaptatif avec des oracles basés sur des tailles de grilles et d'échantillons distinctes, dont on aura évalué les niveaux de précision respectifs.

3.2.3 Minimisation de surface

Le problème de minimisation de surface MINS-OB [42] est un problème issu du calcul des variations. Notons $C = [0, 1] \times [0, 1]$ le carré unité $\mathbb{R}^2$, ∂C sa frontière, et $\mathcal{H}^1(C)$ l'espace Hilbert des fonctions à support compact dans C dont le gradient ∇v est de carré intégrable. Le problème consiste à minimiser la forme intégrale $\int_C \sqrt{1 + \|\nabla v\|^2}$ sous la contrainte de

borne v_0 définie de la manière suivante : pour tout point y à la frontière de C,

$$v_0(y) = \begin{cases} \sin(4\pi y_1) + \sin(120\pi y_1)/10 & \text{si } y_2 = 0,\ y_1 \in [0,1], \\ 0 & \text{si } y_1 = 0,\ y_2 \in [0,1], \\ \sin(4\pi y_1) + \sin(120\pi y_1)/10 & \text{si } y_2 = 1,\ y_1 \in [0,1], \\ 0 & \text{si } y_1 = 1,\ y_2 \in [0,1]. \end{cases}$$

Le problème de minimisation de surface s'écrit donc comme suit :

$$\begin{aligned} \text{Minimiser} \quad & \int_C \sqrt{1 + \|\nabla v\|^2} \\ \text{s. c.} \quad & v \in \mathcal{H}^2(C), \\ & v = v_0 \text{ sur } \partial C. \end{aligned}$$

Ce problème convexe est discrétisé avec une base d'élements finis définie par une triangulation uniforme du carré unité C, avec un pas de grille identique dans chacune des deux coordonnées. Les fonctions de base utilisées sont les classiques fonctions $P1$, linéaires sur chaque triangle et valant 0 ou 1 sur les sommets.
Nous considérons trois niveaux de discrétisation pour l'approximation de la forme intégrale : fin, modéré et grossier. Nous obtenons ainsi trois oracles ayant des niveaux de précision différents, liés à la taille de la discrétisation. Appliquons l'algorithme adaptatif avec ces trois oracles, ainsi que l'algorithme classique avec l'oracle le plus fin, au problème de minimisation de surface, avec les même paramètres que précédemment (donnés au second paragraphe de la Section 3.2.1). Les résultats sont présentés dans la Table 3.7.

Problème	Oracles multiples			Oracle fin
	grossier	modéré	fin	
Erreur	$1,533 \cdot 10^{-1}$	$6,043 \cdot 10^{-2}$	$2,111 \cdot 10^{-2}$	$2,111 \cdot 10^{-2}$
Évaluations	13	10	15	28
Opérations	$1,797 \cdot 10^{7}$			$3,169 \cdot 10^{7}$
Valeur	$8,061 \cdot 10^{-3}$			$8,061 \cdot 10^{-3}$

Table 3.7 – Résultats des algorithmes adaptatif et classique appliqués au problème de la minimisation de surface.

L'algorithme adaptatif permet d'obtenir une valeur finale sensiblement égale à celle obtenue avec simplement l'oracle le plus fin tout en faisant près de deux fois moins appels à l'oracle le plus fin et en réclamant un nombre total d'opérations à virgule flottante presque deux fois moindre.

Conclusions et perspectives

L'analyse de l'algorithme de recherche directe directionnelle appliquée à des fonctions inexactes, dont l'erreur est uniformément bornée, a permis de mettre en évidence une borne sur la précision que l'on peut espérer atteindre sur la norme du gradient, une borne sur le nombre d'itérations nécessaires pour atteindre une précision donnée, ainsi qu'un critère d'arrêt basé sur la taille du pas. Nous avons vérifié ces résultats théoriques par des expériences numériques et observé le comportement de l'algorithme sur le problème du placement de puits pétroliers en ingénierie de réservoir, dont la fonction objectif est soumise à des paramètres incertains. Nous avons ici considéré que l'écart entre les évaluations de f par oracles ou les véritables valeurs était uniformément borné. Une voie de recherche future serait de considérer le cas d'un bruit non borné, gaussien par exemple, appartenant à un certain intervalle avec une probabilité supérieure à un seuil donné. Ces résultats nous ont également conduit à définir un algorithme adaptatif pour les situations où différent oracles d'évaluation sont disponibles avec des niveaux de précision distincts. Nous avons testé cet algorithme sur un problème de minimisation de surface pour lequel trois oracles d'ordres de précision différents étaient disponibles. Nous avons appliqué l'algorithme de recherche directe directionnelle au problème de placement de puits en utilisant une discrétisation de taille modeste pour obtenir rapidement de premiers résultats. Une prochaine étape consisterait à mettre à profit l'algorithme adaptatif que nous proposons sur des modèles numériques de résolutions supérieures pour obtenir des résultats plus précis, avec un temps de calcul modéré.

Chapitre 4

Optimisation en présence de paramètres flous

Dans ce chapitre nous nous intéressons à des problèmes d'optimisation dont la fonction objectif comporte des paramètres imprécis modélisés par des nombres flous. Les programmes linéaires ont été particulièrement étudiés dans la littérature sur l'optimisation floue ; les problèmes non linéaires ont reçu moins d'attention, sauf peut-être en programmation quadratique floue. C'est pourquoi nous proposons une méthode par tri itératif pour une plus large classe de problèmes, ainsi qu'un critère pour évaluer la précision des solutions obtenues, dans la Section 4.2.2.

4.1 Optimisation floue

Optimisation avec paramètres flous Dans ce chapitre nous nous intéressons spécifiquement à l'optimisation en présence de paramètres flous, c'est-à-dire aux problèmes dans lesquels des paramètres flous interviennent dans la définition de la fonction objectif et des contraintes. Considérons une fonction f définie sur $\mathbb{R}^n \times \mathbb{R}^m$ et à valeurs dans $\mathbb{R}$, et un vecteur $\tilde{u}$ de taille m dont les coefficients sont des nombres flous $\tilde{u}_1, \ldots, \tilde{u}_m$. Ces paramètres flous sont définis par des fonctions d'appartenance $\mu_{\tilde{u}_1}, \ldots, \mu_{\tilde{u}_m}$. Le degré d'appartenance s'interprète ici comme un niveau de précision. La fonction d'appartenance du vecteur $\tilde{u}$ est donnée par $\mu_{\tilde{u}} = \min_{i=1,\ldots,m} \mu_{\tilde{u}_i}$. Les valeurs (floues) de la fonction objectif sont données par $f(x;\tilde{u})$ pour tout x dans $\mathbb{R}^n$. Rappelons que $f(x;\tilde{u})$ est une partie floue de $\mathbb{R}$ définie par le principe d'extension [119–121], pour chaque x dans $\mathbb{R}^n$:

$$\mu_{f(x;\tilde{u})}(t) = \sup\{\mu_{\tilde{u}}(u) : u \in \mathbb{R}^m,\ f(x;u) = t\} \quad t \in \mathbb{R}. \tag{4.1}$$

Considérons également une partie $\mathcal{X}$ de $\mathbb{R}^n$ modélisant les contraintes portant sur les variables de décisions $x_1, \ldots, x_n$. Dans les méthodes que nous passons en revue dans la suite de cette section, selon les cas, l'ensemble $\mathcal{X}$ sera un ensemble flou ou un ensemble au sens classique ; nous le préciserons. En revanche, dans la méthode que nous proposons à la Section 4.2.2, la

partie $\mathcal{X}$ sera un ensemble non flou. Résumons les données du problème flou sous la forme suivante :

$$\begin{aligned} \text{minimiser} \quad & f(x;\tilde{u}) \\ \text{s. c.} \quad & x \in \mathcal{X}. \end{aligned} \tag{PF}$$

Le problème (PF) est — pour l'instant — mal défini puisque nous n'avons pas défini de relation d'ordre sur l'espace des ensembles flous de $\mathbb{R}$, espace dans lequel la fonction objectif prend ses valeurs. Il existe en fait différentes approches dans la littérature pour donner un sens à la résolution du problème (PF).

1. La plus directe — et la plus grossière — consiste à remplacer les paramètres flous par des valeurs scalaires bien choisies (par exemple des valeurs dont le degré d'appartenance est le plus élevé possible).
2. L'approche consistant à définir un ordre sur les parties floues de $\mathbb{R}$ a été très étudiée dans la littérature. Il peut s'agir d'une relation binaire au sens classique, où tout couple de nombres flous $\tilde{a}$ et $\tilde{b}$ comparables sont ordonnés [11, 17, 79, 92] : $\tilde{a}$ est soit plus petit que $\tilde{b}$, soit plus grand que $\tilde{b}$, il n'y a pas d'autre possibilité. Dans [81] une relation d'ordre floue est proposée : une décision x est considérée meilleure qu'une décision x' avec un un degré d'appartenance donné par

   $$\eta(x,x') = \sup\{\min\{\mu_{f(x;\tilde{u})}(t), \mu_{f(x';\tilde{u})}(t')\} : t \leq t', t \in \mathbb{R}, t' \in \mathbb{R}\}.$$

   Une décision x n'est alors pas meilleure qu'une décision x' dans l'absolu, mais seulement relativement à un degré de possibilité compris en 0 et 1. Remarquons qu'il s'agit de l'application du principe d'extension à la relation binaire définissant l'ordre sur $\mathbb{R}$, pour les ensembles flous $f(x;\tilde{u})$ et $f(x',\tilde{u})$. La suite de cette approche consiste à déterminer une décision qui soit optimale pour cette relation d'ordre floue avec un degré aussi élevé que possible. D'autres approches [102, 104] pour donner un sens aux inégalités floues sont passées en revue dans [90].
3. L'approche que nous allons adopter consiste à raisonner à partir des coupes des valeurs de la fonction objectif, dans l'esprit du Théorème de décomposition donné à la Section 1.2.4. Nous serons ainsi amenés à nous intéresser aux problèmes suivants :

   $$\begin{aligned} \text{minimiser} \quad & [f(x;\tilde{u})]_\alpha \\ \text{s. c.} \quad & x \in \mathcal{X}, \end{aligned} \tag{PI_α}$$

   où α est un nombre dans $[0,1]$. Dans la mesure où, pour x dans $\mathbb{R}^n$ et α dans $[0,1]$, la coupe $[f(x;\tilde{u})]_\alpha$ est un ensemble *a priori* quelconque, le problème (PI_α) est mal défini. Sous des hypothèses que nous préciserons plus loin ces coupes seront des intervalles et les problèmes (PI_α) seront donc des problèmes de minimisation d'intervalles, auxquels nous avons déjà donné un sens dans la Section 1.2.2. Nous étudions au paragraphe suivant l'état de l'art de la résolution des problèmes de la forme (PF) avec des considérations sur les coupes.

Approche par les coupes Les paramètres apparaissant dans les valeurs de la fonction objectif du problème (PF) sont généralement supposés être des nombres flous compacts.

Autrement dit les coupes $[\tilde{u}_i]_\alpha$ de chacun des paramètres flous, $i = 1, \ldots, m$, est un intervalle non vide, fermé et borné de $\mathbb{R}$, $\alpha \in [0, 1]$. Nous vérifierons que cette hypothèse garantit que, dans les cas linéaire et quadratique, les valeurs de la fonction objectif sont également des nombres flous compacts dont les coupes sont données par

$$[f(x; \tilde{u})]_\alpha = f(x; [\tilde{u}]_\alpha), \quad x \in \mathbb{R}^n, \alpha \in [0, 1].$$

Passons en revue quelques méthodes proposées dans le cas de la programmation linéaire floue, qui a été abondamment étudiée dans la littérature [29, 92, 102]. Les valeurs de la fonction objectif de (PF) sont alors de la forme

$$f(x; \tilde{c}) = \tilde{c}^\top x = \sum_{i=1}^{n} x_i \tilde{c}_i, \tag{4.2}$$

pour tout x dans $\mathbb{R}^n$, où $\tilde{c}$ est un vecteur de taille n dont les coefficients $\tilde{c}_1, \ldots, \tilde{c}_n$ sont des nombres flous compacts. Rappelons que l'addition des nombres flous et la multiplication d'un nombre flou par un réel ont été définies à la Section 1.2.4. L'ensemble réalisable est un polyèdre donné par $\mathcal{X} = \{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\}$, où A et b sont respectivement une matrice de taille $m \times n$ et un vecteur de taille m à coefficients réels. Le problème (PF) est alors un programme linéaire flou (PLF) de la forme ci-dessous.

$$\begin{aligned} \text{Minimiser} \quad & \tilde{c}^\top x \\ \text{s. c.} \quad & Ax \leq b,\ x \geq 0. \end{aligned} \tag{PLF}$$

Les auteurs de [91] supposent que l'information sur les fonctions d'appartenance des paramètres imprécis est limitée : seules N coupes, de niveaux $\alpha_1, \ldots, \alpha_N$ décroissants, sont connues pour chaque paramètre.Dans cette approche le programme linéaire flou (PLF) est remplacé par le problème de la minimisation à $2N$ critères des bornes $(\tilde{c}^\top x)^L_\alpha$, $(\tilde{c}^\top x)^U_\alpha$ des coupes de niveau $\alpha = \alpha_1, \ldots, \alpha_N$, sous contrainte $Ax \leq b$, $x \geq 0$. Une méthode est mise en œuvre afin d'obtenir une solution particulière de ce problème multicritère ; cette solutions est alors définie comme la solution du problème initial (PLF). Observons que, du fait de la linéarité de f relativement aux paramètres (4.2), les bornes des coupes des valeurs de la fonction objectifs se calculent aisément. Prenons par exemple un point x de $\mathbb{R}^n$ dont les coordonnées sont positives, alors

$$\begin{aligned} (\tilde{c}^\top x)^L_\alpha &= \min\{x_1 c_1 + \cdots + x_n c_n : (c_1, \ldots, c_n) \in [\tilde{c}_1]_\alpha \times \cdots \times [\tilde{c}_n]_\alpha\} \\ &= (\tilde{c}_1)^L_\alpha x_1 + \cdots + (\tilde{c}_n)^L_\alpha x_n, \end{aligned}$$

et de même $(\tilde{c}^\top x)^U_\alpha = (\tilde{c}_1)^U_\alpha x_1 + \cdots + (\tilde{c}_n)^U_\alpha x_n$. L'approche adoptée dans [20, 30, 94] suppose, elle, la connaissance complète des fonctions d'appartenance des paramètres flous. Elle consiste à définir un ensemble flou de solutions optimales en attribuant à chaque point réalisable x de $\mathcal{X}$ un degré d'appartenance dans $[0, 1]$. Notons, pour chaque α dans $[0, 1]$, Eff_α l'ensemble des points efficaces (Pareto-optimaux) du problème de minimisation bicritère suivant — cette notion a été définie à la Section 1.2.2 :

$$\begin{aligned} \text{minimiser} \quad & \left((\tilde{c}^\top x)^L_\alpha, (\tilde{c}^\top x)^U_\alpha\right) \\ \text{s. c.} \quad & Ax \leq b,\ x \geq 0. \end{aligned} \tag{4.3}$$

Le degré d'appartenance d'un point réalisable x à l'ensemble des solutions optimales est alors défini comme la mesure de Lebesgue de l'ensemble des niveaux α pour lesquels x est une solution efficace du problème (4.3) : $\{\alpha \in [0,1] : x \in \mathrm{Eff}_\alpha\}$. Les auteurs de [30, 94] présentent une méthode pour calculer cette mesure en considérant des conditions d'optimalité linéaire.
L'optimisation non linéaire floue a été relativement peu étudiée [67, 100] en comparaison avec la programmation linéaire floue. Les problèmes traités sont principalement les programmes quadratiques flous [18, 64, 65], en particulier en vue d'applications en gestion de portefeuille financier [2, 66]. Les valeurs de la fonction objectif du problème (PF) prennent alors la forme

$$f(x; \tilde{Q}, \tilde{c}) = \frac{1}{2} x^\top \tilde{Q} x - \tilde{c}^\top x = \frac{1}{2} \sum_{i,j=1}^{n} x_i x_j \tilde{Q}_{ij} - \sum_{i=1}^{n} x_i \tilde{c}_i, \tag{4.4}$$

où $\tilde{Q}$ et $\tilde{c}$ sont respectivement une matrice de taille $n \times n$ et un vecteur de taille n sont les coefficients sont des nombres flous compacts. Observons qu'une nouvelle fois la dépendance de f en les paramètres flous (4.4) est linéaire. Ceci facilite le calcul des coupes des valeurs de la fonction objectif, comme nous l'avons montré précédemment. L'ensemble réalisable est ici un polyèdre flou donné par $\tilde{A}x \leq \tilde{b}$, $x \geq 0$, où $\tilde{A}$ et $\tilde{b}$ sont respectivement une matrice de taille $m \times n$ et un vecteur de taille m dont les coefficients sont des nombres flous compacts. Le sens donné à la satisfaction de contraintes flous sera inclu dans la définition des méthodes à suivre. Le problème (PF) prend alors la forme du programme quadratique flou (PQF) ci-dessous.

$$\begin{aligned} \text{Minimiser} \quad & \frac{1}{2} x^\top \tilde{Q} x - \tilde{c}^\top x \\ \text{s. c.} \quad & \tilde{A}x \leq \tilde{b},\ x \geq 0. \end{aligned} \tag{PQF}$$

Les auteurs de [64, 65] proposent de calculer la fonction d'appartenance de la valeur optimale de (PQF), qui dépend des paramètres flous. Pour tous paramètres non flous (Q, c, A, b), notons comme suit la valeur optimale du problème quadratique non flou associé :

$$v(Q, c, A, b) = \inf \left\{ \frac{1}{2} x^\top Q x - c^\top x : Ax \leq b, x \geq 0 \right\}.$$

La valeur optimale (floue) de (PQF) est alors la partie floue $v(\tilde{Q}, \tilde{c}, \tilde{A}, \tilde{b})$ de $\mathbb{R}$ dont la fonction d'appartenance est définie par le principe d'extension. Ici ce sont plutôt ses coupes qui sont calculées ; la fonction d'appartenance s'en déduit alors grâce au Théorème de décomposition. Les bornes de chacune des coupes de $v(\tilde{Q}, \tilde{c}, \tilde{A}, \tilde{b})$ sont obtenues par deux programmes quadratiques biniveaux non flous. Les auteurs de [2] suggèrent quant à eux de résoudre, pour chaque α dans $[0, 1]$, les problèmes quadratiques non flous de paramètres non flous $(\tilde{Q}^L_\alpha, \tilde{c}^L_\alpha, \tilde{A}^L_\alpha, \tilde{b}^L_\alpha)$ et $(\tilde{Q}^U_\alpha, \tilde{c}^U_\alpha, \tilde{A}^U_\alpha, \tilde{b}^U_\alpha)$ respectivement. Les valeurs optimales de ces deux programmes sont alors prises comme approximations des bornes de le coupe de niveau α de $v(\tilde{Q}, \tilde{c}, \tilde{A}, \tilde{b})$, $\alpha \in [0, 1]$. Cependant cette méthode d'approximation ne s'appuie pas sur des résultats théoriques.
Citons également une méthode [71] destinée aux problèmes multicritères flous et présentons-là pour les problèmes monocritères de la forme (PF). Il s'agit de remplacer les valeurs floues de la fonction objectif par une approximation sous forme d'intervalle (*nearest interval approximation* [44]). Le problème flou initial est ainsi ramené un problème de minimisation d'intervalles. Nous faisons remarquer dans la section suivante que, lorsque f dépend linéairement des paramètres — comme c'est le cas pour les programmes linéaires flous ou quadratiques flous — et

que les nombres flous sont triangulaires, cette approche revient à ne considérer que les coupes de niveaux $0,5$ des valeurs de la fonction objectif. Le problème est donc simplifié au prix d'un niveau de précision moyen.

4.2 Méthode par tri non-dominé itératif

Nous avons pu remarquer dans la section précédente que la littérature actuelle en optimisation avec paramètres flous est focalisée sur les problèmes linéaires et quadratiques. Dans la Section 4.2.1 nous définissons une classe plus large de problèmes quvi pourraient être résolus avec une méthode s'inspirant de cet état de l'art. Nous nous restreignons à un niveau de précision raisonnable sur les paramètres et nous examinons le niveau de précision que nous pouvons espérer sur l'optimalité des solutions. Nous définissons dans la Section 4.2.2 une méthode pour les problèmes de cette classe et nous la comparons avec diverses méthodes de la littérature sur la base d'un critère théorique ainsi que sur des exemples numériques.

4.2.1 Classe de problèmes abordée

Dans cette section nous faisons à nouveau l'hypothèse que les paramètres flous $\tilde{u}_1, \ldots, \tilde{u}_m$ sont des nombres flous compacts. En particulier leurs coupes sont des intervalles non vides, fermés et bornés de $\mathbb{R}$.

Encadrement des vraies valeurs de l'objectif Les auteurs de [91] font remarquer que, dans le cas du programme linéaire flou (PLF), il est possible d'encadrer la valeur de la fonction objectif pour tout x dans $\mathbb{R}^n$. En effet, si le vecteur des véritables valeurs des paramètres est noté $\bar{c}$ et que la fonction d'appartenance de $\tilde{c}$ est telle que $\mu_{\tilde{c}}(\bar{c}) = 1$, alors $\bar{c}$ est élément de chacune des coupes $[\tilde{c}]_\alpha$, $\alpha \in [0,1]$. Ainsi, en notant $\tilde{c}_\alpha^L$ et $\tilde{c}_\alpha^U$ les vecteurs définis respectivement par

$$\tilde{c}_\alpha^L = \left[(\tilde{c}_1)_\alpha^L \quad \cdots \quad (\tilde{c}_n)_\alpha^L\right] \quad \text{et} \quad \tilde{c}_\alpha^U = \left[(\tilde{c}_1)_\alpha^U \quad \cdots \quad (\tilde{c}_n)_\alpha^U\right],$$

nous obtenons les deux inégalités (composante par composante) suivantes : $\tilde{c}_\alpha^L \leq \bar{c} \leq \tilde{c}_\alpha^U$, pour tout α dans $[0,1]$. Il s'ensuit l'encadrement suivant pour tout x de $\mathbb{R}^n$ à coordonnées positives :

$$\min[\tilde{c}^\top x]_\alpha = (\tilde{c}_\alpha^L)^\top x \leq \bar{c}^\top x \leq (\tilde{c}_\alpha^U)^\top x = \max[\tilde{c}^\top x]_\alpha, \quad \alpha \in [0,1]. \tag{4.5}$$

Un encadrement similaire peut être obtenu pour le problème quadratique flou (PQF). Voyons sous quelles hypothèses plus générales sur f nous pouvons obtenir un résultat similaire à (4.5). Notons $\bar{u}$ le vecteur des vraies valeurs (inconnues) des paramètres. Supposons que les nombres flous $\tilde{u}_1, \ldots, \tilde{u}_m$ modélisant les paramètres imprécis soient tel que $\mu_{\tilde{u}}(\bar{u}) = 1$. Donnons-nous un vecteur x de $\mathbb{R}^n$. D'après le principe d'extension (4.1) le degré d'appartenance du nombre $f(x;\bar{u})$ à la partie floue $f(x;\tilde{u})$ de $\mathbb{R}$ est 1. Par conséquent le nombre $f(x;\bar{u})$ appartient à chaque coupe $[f(x;\tilde{u})]_\alpha$, $\alpha \in (0,1]$, de la partie floue $f(x;\tilde{u})$. Nous obtenons ainsi l'encadrement suivant pour la valeur $f(x;\bar{u})$:

$$\min[f(x;\tilde{u})]_\alpha \leq f(x;\bar{u}) \leq \min[f(x;\tilde{u})]_\alpha, \quad \alpha \in [0,1]. \tag{4.6}$$

Observons que le calcul du minorant et du majorant dans (4.6) nécessite de faire appel au principe d'extension (4.1) et une connaissance parfaite des fonctions d'appartenances $\mu_{\tilde{u}_1}, \ldots, \mu_{\tilde{u}_m}$ des paramètres flous.
Dans le cas linéaire (4.2), il est possible de vérifier grâce à l'arithmétique d'intervalles que $[f(x;\tilde{u})]_\alpha = f(x;[\tilde{u}]_\alpha)$ pour tout x dans $\mathbb{R}^n$ et tout α dans $[0,1]$. Nous démontrerons dans la Section 4.2.2 que ces relations sont encore vérifiées si la fonction f est continue relativement aux paramètres — c'est en fait une conséquence du Lemme 2.

Proposition 1. *Soit x un vecteur de $\mathbb{R}^n$. Supposons que la fonction $f(x;\cdot)$ soit continue sur $\mathbb{R}^m$ et que $\tilde{u}_1, \ldots, \tilde{u}_m$ soient des nombres flous compacts. Alors $f(x;\tilde{u})$ est un nombre flou compact et ses coupes sont données par*

$$[f(x;\tilde{u})]_\alpha = f(x;[\tilde{u}]_\alpha), \quad \alpha \in [0,1].$$

Sous l'hypothèse que la fonction f est continue relativement aux paramètres, la Proposition 1 permet de modifier l'encadrement (4.6) de la façon suivante :

$$\min\{f(x;u) : u \in [\tilde{u}]_\alpha\} \leq f(x;\bar{u}) \leq \max\{f(x;u) : u \in [\tilde{u}]_\alpha\}, \quad \alpha \in [0,1]. \tag{4.7}$$

Le minorant et le majorant dans l'encadrement (4.7) ne dépendent maintenant que des coupes du vecteur $\tilde{u}$ des paramètres flous et des évaluations de la fonction f. L'hypothèse de continuité sera suffisante pour définir l'algorithme itératif présenté dans la Section 4.2.2.
Observons que si la fonction f est de plus supposée monotone relativement aux paramètres imprécis pour x dans $\mathbb{R}^n$ fixé — c'est en particulier le cas si f est linéaire relativement aux paramètres imprécis, comme dans les cas linéaire (4.2) et quadratique (4.4) —, cet encadrement peut être amélioré. Nous détaillerons cette remarque à la section suivante. Les bornes sur $f(x;\bar{u})$ de l'encadrement (4.7) peuvent dans ce cas être calculées directement, de sorte que nous connaissons un intervalle — dépendant de α — auquel appartient $f(x;\bar{u})$, pour un x donné de $\mathbb{R}^n$. Cela est particulièrement intéressant pour les x retournés par les algorithmes de résolution du problème (PF).

Restriction des données Dans la pratique il est difficile de définir parfaitement les fonctions d'appartenance des paramètres flous $\tilde{u}_1, \ldots, \tilde{u}_m$. Le choix fait dans [91] est de ne supposer que la connaissance d'un nombre restreint K de coupes de niveaux décroissants $\alpha_1, \ldots, \alpha_K$ dans $[0,1]$. Ainsi, au lieu de connaître exactement le degré d'appartenance de chaque vecteur u de $\mathbb{R}^m$, nous savons seulement si ce degré est supérieur ou égal à α_1, à α_2, *etc.* Les paramètres flous étant supposés être des nombres flous compacts, les données se résument donc à K intervalles non vides, fermés et bornés de $\mathbb{R}$.

Degré d'optimalité L'ambition des auteurs de [64, 65] est de connaître la fonction d'appartenance de la valeur optimale du problème PF, qui dépend des paramètres flous $\tilde{u}$. Dans le même esprit — mais du côté de l'espace des variables — les auteurs de [20, 30, 94] suggèrent de définir un ensemble flou de solutions au problème PF en attribuant à chaque élement x

de $\mathcal{X}$ un degré d'appartenance dans $[0,1]$.
Voyons comment nous pourrions évaluer le "degré d'optimalité" d'un point x de $\mathcal{X}$ pour le problème (PF). Supposons qu'il existe un vecteur de paramètres u tel que x soit une solution optimale du problème non flou suivant :

$$\begin{aligned} \text{minimiser} \quad & f(x;u) \\ \text{s. c.} \quad & x \in \mathcal{X}. \end{aligned} \tag{P_u}$$

Nous nous attendons alors à ce que, dans le même esprit que le Principe d'extension, le degré d'optimalité de x soit au moins égal à $\mu_{\tilde{u}}(u)$. C'est pourquoi nous proposons la définition suivante pour le *degré d'optimalité* du point x relativement au problème (PF) :

$$\sup\{\mu_{\tilde{u}}(u) : f(x;u) = \inf f(\mathcal{X};u)\}. \tag{4.8}$$

Ainsi, si l'on connaît un vecteur u de paramètres tel que x est solution du problème (P_u), la définition (4.8) attribue à x un degré d'optimalité supérieur ou égal à $\mu_{\tilde{u}}(u)$. Afin de garantir un degré d'optimalité le plus élevé possible les points intéressants sont donc les solutions des problèmes de la forme (P_u) où u est un élément de $[\tilde{u}]_{\alpha_1}$, la coupe disponible de niveau le plus grand. Cette remarque sera fondamentale pour notre algorithme itératif.

L'algorithme que nous présentons à la section suivante est destiné aux problèmes d'optimisation avec paramètres flous compacts de la forme (PF) tels que la fonction f est continue relativement aux paramètres. Seul un nombre restreint de coupes des paramètres flous sont supposées connues ; l'algorithme garantit un degré d'optimalité au moins égal au niveau de coupe le plus élevé.

4.2.2 Algorithme par tri non-dominé itératif

Nous reproduisons ci-contre le projet de publication *Iterative non-dominated sorting for fuzzy optimization problems* détaillant l'algorithme itératif et le degré d'optimalité introduits plus haut.

Iterative non-dominated sorting for fuzzy optimization problems

B. Pauwels[a,*], S. Gratton[b], F. Delbos[a]

[a] *Department of Applied Mathematics, IFPEN, 1&4 avenue de Bois-Préau, 92852 Rueil-Malmaison Cedex, France*
[b] *CERFACS, 42 avenue Gustave Coriolis, 31057 Toulouse Cedex, France*

Abstract

We consider an optimization problem with imprecise parameters in the objective function. This imprecision is modeled by fuzzy numbers of which only a limited number of α-level cuts are assumed accessible. We present a method to solve such a fuzzy optimization problem by iterative non-dominated sorting. Only continuity of the objective with respect to the imprecise parameters is required, although monotonic dependency dramatically alleviate the computational cost. These assumptions cover fuzzy linear programming, fuzzy quadratic programming as well as more general fuzzy polynomial objective functions. We also define an 'optimality degree' to compare the solutions obtained with several methods including ours and prove that we achieve an optimality degree greater than or equal to the highest cut level assumed accessible.

Keywords: Fuzzy mathematical programming, Multiobjective programming

1. Introduction

Imprecision often arises in real life mathematical programming problems as a result of a lack of information or knowledge. In most situations probability theory is applied. However imprecision may have nothing to do with randomness and be a significantly different type of uncertainty [22]. This is the case when imprecision stems from a lack of physical measures, scarcity of data, or subjective expert judgements. Fuzzy sets theory [21] has been developped specifically for dealing with that kind of uncertainty. This theoretical tools were largely applied to mathematical programming under imprecision giving rise to the field of fuzzy optimization [11].

We consider a single-objective mathematical program with fuzzy parameters in the objective function, the fuzziness of these parameters modeling imprecision. We denote f a function defined on $\mathbb{R}^n \times \mathbb{R}^m$ mapping an n-dimensional variables vector $x = (x_1, \ldots, x_n)$ and an m-dimensional parameters vector $u = (u_1, \ldots, u_m)$ to a real value of interest $f(x; u)$. The parameters are assumed to be fuzzy numbers $\tilde{u}_1, \ldots, \tilde{u}_m$ with membership functions $\mu_{\tilde{u}_1}, \ldots, \mu_{\tilde{u}_m}$. Let us denote $\tilde{u} = (\tilde{u}_1, \ldots, \tilde{u}_m)$ the fuzzy parameters vector with membership function given by $\mu_{\tilde{u}}(u) = \min\{\mu_{\tilde{u}_i}(u_i) : i = 1, \ldots, m\}$ for every u in $\mathbb{R}^m$. Therefore the program consists in minimizing—in a meaning to be clarified—the fuzzy quantity $f(x; \tilde{u})$ with respect to $x_1, \ldots, x_n$. Remember that the membership function of this fuzzy quantity is given by the classical extension principle—or fuzzification principle [1]. We require the solution x to belong to a feasible subset $\mathcal{X}$ of $\mathbb{R}^n$. We summarize the fuzzy mathematical program at stake in the following compact form:

$$\begin{aligned} &\text{minimize} && f(x; \tilde{u}) \\ &\text{subject to} && x \in \mathcal{X}. \end{aligned} \tag{FP}$$

We will assume that the fuzzy numbers are only partially known: we only have access to a finite number of the α-level cuts of $\tilde{u}_1, \ldots, \tilde{u}_m$, α in $[0, 1]$. This seems a reasonable assumption since information on fuzzy

*Corresponding author
Email addresses: be.pauwels@gmail.com (B. Pauwels), gratton@cerfacs.fr (S. Gratton), frederic.delbos@ifpen.fr (F. Delbos)

quantities typically rely on subjectivity and suffers from a lack of information. Actually such an assumption is made in [15]. In practice it means only a few points of the membership functions are known, namely both bounds of each accessible α-level cut.

As stated in [12] there is no defined optimum optimorum for a problem such as (FP). Indeed there is no *a priori* order on the space were lies the objective values—that is the space of fuzzy subsets of $\mathbb{R}$. Therefore we rather call *satisficing* any solution of (FP) advocated by a given method. Several approaches have been investigated to single out a solution for such a problem. A classical strategy of defuzzification consists in considering the α-level cuts of the fuzzy quantities rather than their whole membership functions. We take on such an approach in Section 2. Another path consists in replacing each fuzzy quantity by an appropriately chosen approximating interval, called *nearest interval approximation* [12]. In Section 3.2 we draw comparative observations with this approach, which actually coincides with ours in the case of triangular fuzzy parameters when only the 0.5-level cuts are considered. To our knowledge these methods are specifically dedicated to fuzzy linear programming [5, 12, 15] and may not be easily generalized.

In this paper we suggest an iterative process based on the well-known non-dominated sorting [4, 7]. First efficient points for the biobjective minimization of the α-level cut bounds with the highest α accessible are sought. Then this set of efficient solutions is iteratively refined repeating the same process with the bounds of lower α-level cuts. We only assume that f is continuous with respect to the parameter values. We also stress the fact that computations are much less time-consuming when f is also monotone—either non-decreasing or decreasing—with respect to each parameter value u_i, $i = 1, \dots, m$. These assumptions cover the case of polynomial functions with fuzzy coefficients. In particular the fuzzy objective may be linear

$$f(x; \tilde{c}) = \tilde{c}^{\mathsf{T}} x = \sum_{j=0}^{m} \tilde{c}_j x_j,$$

or quadratic

$$f(x; \tilde{Q}, \tilde{c}) = x^{\mathsf{T}} \tilde{Q} x - \tilde{c}^{\mathsf{T}} x = \sum_{i,j=1}^{n} \tilde{Q}_{ij} x_i x_j - \sum_{j=1}^{n} \tilde{c}_j x_j.$$

In Section 2 we make our assumptions explicit, provide the theoretical requirements for our method and then present our algorithm. Then in Section 3 we apply our method to a fuzzy linear program and a fuzzy quadratic program, drawing comparisons with inspiring approaches found in the literature. In Section 4 we suggest a criterion to discriminate between satisficing solutions attained by the different methods considered. Finally Section 5 will be devoted to conluding remarks.

2. Method

2.1. Compact fuzzy numbers and continuous monotonic dependence

We make the classical assumption that the fuzzy subsets of $\mathbb{R}$ at stake $\tilde{u}_1, \dots, \tilde{u}_m$ are fuzzy numbers. Let i be in $\{1, \dots, m\}$. We mean their α-level cuts $[\tilde{u}_i]_\alpha = \{t \in \mathbb{R} : \mu_{\tilde{u}_i}(t) \geq \alpha\}$ are non-empty and convex, for each α in $(0, 1]$. Throughout this paper we (improperly) denote $[\tilde{u}_i]_0$ the support of $\tilde{u}_i$—the closure of $\{t \in \mathbb{R} : \mu_{\tilde{u}_i}(t) > 0\}$—and call it the 0-level cut of $\tilde{u}_i$ for the sake of simplicity. Since $\tilde{u}_i$ is a fuzzy number $[\tilde{u}_i]_0$ is also non-empty and convex. We require further these fuzzy numbers to be *bounded* and *closed* in the sense their α-level cuts are also bounded and closed, for each α in $[0, 1]$. We call such fuzzy numbers *compact*. In particular $[\tilde{u}]_\alpha = [\tilde{u}_1]_\alpha \times \cdots \times [\tilde{u}_m]_\alpha$ is also non-empty, convex, bounded and closed set—even a hypercube—for every α in $[0, 1]$. We call $\tilde{u}$ a compact fuzzy vector.

Assumption 1. The fuzzy numbers $\tilde{u}_1, \dots, \tilde{u}_n$ are compact: for every α in $[0, 1]$ the α-level cut $[\tilde{u}_i]_\alpha$ is a non-empty bounded closed interval of $\mathbb{R}$, $i = 1, \dots, m$.

Moreover we assume we only have access to a finite number K of α-level cuts for each of the fuzzy parameters, corresponding to levels denoted in decreasing order $\alpha_1, \dots, \alpha_K$ so that α_1 is the highest accessible cut level. Observe that if the whole membership functions were known we could meet this assumption by

choosing K of the α-level cuts. However the question of how to properly select the K cut levels is beyond the scope of this paper. The worst-case complexity of our algorithm is proportional to that number K. On the one hand the computational cost may be alleviated if K is low. On the other hand a higher K potentially means more reduction of the final set of satisficing solutions and an easier choice for the decision-maker. In terms of possibility the values of $\alpha_1, \ldots, \alpha_K$ should clearly be as high as possible.

Fuzzy numbers $\tilde{u}_1, \ldots, \tilde{u}_m$ model the imprecision relative to the true (unknown) values $\bar{u}_1, \ldots, \bar{u}_m$ of the parameters. Even though we cannot directly compute $t = f(x; \bar{u})$ for a given x in $\mathcal{X}$—where we denote $\bar{u} = (\bar{u}_1, \ldots, \bar{u}_m)$—we know theoretical lower and upper bounds for this value t. Suppose $\mu_{\tilde{u}_i}(\bar{u}_i) = 1$ for every i in $\{1, \ldots, m\}$. Then the extension principle entails $\mu_{f(x;\tilde{u})}(t) = 1$, so that t belongs to each α-level cut of $f(x; \tilde{u})$, α in $[0, 1]$. Therefore the pair of bounds of each α-level cut provide us with a range for $f(x; \bar{u})$, and the higher the cut level α the tighter the bounds. Remember that we denoted α_1 the highest cut level available, hence the following bounds for the true objective value $f(x; \bar{u})$:

$$f(x; \tilde{u})^L_{\alpha_1} \leq f(x; \bar{u}) \leq f(x; \tilde{u})^U_{\alpha_1}, \tag{1}$$

where we denote $f(x; \tilde{u})^L_{\alpha_1} = \inf[f(x; \tilde{u})]_{\alpha_1}$ and $f(x; \tilde{u})^U_{\alpha_1} = \sup[f(x; \tilde{u})]_{\alpha_1}$—we will use similar notations for all cut bounds throughout this paper. Not only the bounds in Inequality (1) are hard to estimate in general but here any computation is made impossible by the ignorance of $\mu_{\tilde{u}}$. In fact the set $[f(x; \tilde{u})]_{\alpha_1}$ cannot be determined. A continuity assumption on f will bend this infeasibility.

Assumption 2. For each x in $\mathcal{X}$, the mapping $f(x; \cdot) : u \mapsto f(x; u)$ is continuous from $\mathbb{R}^m$ to $\mathbb{R}$.

This assumption makes sufficient the knowledge of a given α-level cut of $\tilde{u}$ to infer the corresponding α-level cut of $f(x; \tilde{u})$ for each x in $\mathcal{X}$ and each α in $(0, 1]$.

Lemma 1. Let Assumptions 1 and 2 hold. Then, for every x in $\mathcal{X}$, $f(x; \tilde{u})$ is a compact fuzzy number and its α-level cuts are bounded closed intervals of $\mathbb{R}$ given by

$$[f(x; \tilde{u})]_\alpha = f(x; [\tilde{u}]_\alpha), \qquad \alpha \in [0, 1]. \tag{2}$$

Proof. Let x be in $\mathcal{X}$ and denote $g(u) = f(x; u)$ for each u in $\mathbb{R}^m$. Let α be in $[0, 1]$. First let us consider the inclusion $[g(\tilde{u})]_\alpha \subset g([\tilde{u}]_\alpha)$. Let t be in $[g(\tilde{u})]_\alpha$.
If $\alpha = 0$ then $[g(\tilde{u})]_\alpha$ is the support of $g(\tilde{u})$ and there exists a sequence $\{t^k\}_{k\in\mathbb{N}}$ in $\mathbb{R}$ converging towards t such that $\mu_{g(\tilde{u})}(t^k) > 0$ for each k in $\mathbb{N}$. The extension principle states

$$\mu_{g(\tilde{u})}(t) = \sup\{\mu_{\tilde{u}}(v) : v \in \mathbb{R}^m / g(v) = t\}. \tag{3}$$

Consequently, for each k in $\mathbb{N}$, there exists u^k in $\mathbb{R}^m$ such that $g(u^k) = t^k$ and $\mu_{\tilde{u}}(u^k) > 0$. Thus $\{u^k\}_{k\in\mathbb{N}}$ is a sequence of the compact set $[\tilde{u}]_0$ and admits a subsequence converging towards a point u in $[\tilde{u}]_0$. The continuity of g entails $g(u) = t$, therefore t belongs to $g([\tilde{u}]_0)$.
Let us now suppose $\alpha > 0$. It follows from $\mu_{g(\tilde{u})}(t) \geq \alpha$ and (3) that, for each k in $\mathbb{N} \setminus \{0\}$, there exists u^k in $\mathbb{R}^m$ such that $g(u^k) = t$ and $\mu_{\tilde{u}}(u^k) \geq (1 - 2^{-k})\alpha$. Since $\alpha > 0$ the sequence $\{u^k\}_{k\in\mathbb{N}\setminus\{0\}}$ is included in the compact set $[\tilde{u}]_0$ and there exists a subsequence of $\{u^k\}_{k\in\mathbb{N}\setminus\{0\}}$ converging towards a point u in $[\tilde{u}]_0$. Since $\mu_{\tilde{u}_i}$ is upper semi-continuous for every i in $\{1, \ldots, m\}$ (Assumption 1) the function $\mu_{\tilde{u}} = \min\{\mu_{\tilde{u}_i} : i = 1, \ldots, m\}$ is also upper semi-continuous and

$$\mu_{\tilde{u}}(u) \geq \limsup_{k\to\infty} \mu_{\tilde{u}}(u^k) \geq \limsup_{k\to\infty} (1 - 2^{-k})\alpha = \alpha.$$

Consequently u is in $[\tilde{u}]_\alpha$. Moreover the continuity of g implies $g(u) = t$, therefore t is in $g([\tilde{u}]_\alpha)$.
Now let us consider the reciprocal inclusion for $\alpha > 0$. Consider u in $[\tilde{u}]_\alpha$ and denote $t = g(u)$. It follows from (3) that $\mu_{g(\tilde{u})}(t) \geq \mu_{\tilde{u}}(u) \geq \alpha$, therefore t belongs to $[g(\tilde{u})]_\alpha$.
At this point Equation (2) is proven for $\alpha > 0$. We conclude with the following

$$g([\tilde{u}]_0) = g\left(\overline{\cup_{\alpha\in(0,1]}[\tilde{u}]_\alpha}\right) \subset \overline{g\left(\cup_{\alpha\in(0,1]}[\tilde{u}]_\alpha\right)} = \overline{\cup_{\alpha\in(0,1]} g([\tilde{u}]_\alpha)} = \overline{\cup_{\alpha\in(0,1]}[g(\tilde{u})]_\alpha} = [g(\tilde{u})]_0,$$

where the inclusion results from the continuity of g.
Now Equation (2) holds for each α in $[0,1]$. The α-level cut of $f(x;\tilde{u})$ is the image of the non-empty connected compact $[\tilde{u}]_\alpha$ by the continuous function g, therefore $[f(x;\tilde{u})]_\alpha$ is a non-empty bounded closed interval of $\mathbb{R}$, for each α in $[0,1]$. □

Inequalities (1) may now be written as follows:

$$\min\{f(x;u) : u \in [\tilde{u}]_{\alpha_1}\} \leq f(x;\bar{u}) \leq \max\{f(x;u) : u \in [\tilde{u}]_{\alpha_1}\}. \tag{4}$$

Nevertheless both bounds in (4) are still troublesome to evaluate. Indeed they both are optimal values of m-dimensional optimization problems. We add a third assumption to alleviate the practical computation of the objective level cuts bounds. We assume f is monotone with respect to each parameter. However this assumption is unnecessary to define properly the algorithm presented in Section 2.3.

Assumption 3. Let x be in $\mathcal{X}$ and u be in $\mathbb{R}^m$. The function $f(x;u_1,\ldots,u_{i-1},\cdot,u_{i+1},\ldots,u_m)$ is nondecreasing or nonincreasing, for each i in $\{1,\ldots,n\}$.

The next lemma follows directly from Assumption 3.

Lemma 2. Let Assumptions 1, 2 and 3 hold. Let α be in $[0,1]$, x be in $\mathcal{X}$ and denote $u_\alpha^{\min} \in \operatorname{Argmin}\{f(x;u) : u \in [\tilde{u}]_\alpha\}$ and $u_\alpha^{\max} \in \operatorname{Argmax}\{f(x;u) : u \in [\tilde{u}]_\alpha\}$. Then

$$f(x;\tilde{u})_\alpha^L = f\left(x;u_\alpha^{\min}\right) \text{ and } f(x;\tilde{u})_\alpha^U = f\left(x;u_\alpha^{\max}\right).$$

Furthermore, denoting $u_\alpha^{\min} = (u_{\alpha,1}^{\min},\ldots,u_{\alpha,m}^{\min})$ and $u_\alpha^{\max} = (u_{\alpha,1}^{\max},\ldots,u_{\alpha,m}^{\max})$,

$$u_{\alpha,i}^{\min} = \begin{cases} (\tilde{u}_i)_\alpha^L & \text{if } f(x;\cdot) \text{ is non-decreasing with respect to } u_i, \\ (\tilde{u}_i)_\alpha^U & \text{otherwise,} \end{cases}$$

and

$$u_{\alpha,i}^{\max} = \begin{cases} (\tilde{u}_i)_\alpha^U & \text{if } f(x;\cdot) \text{ is non-decreasing with respect to } u_i, \\ (\tilde{u}_i)_\alpha^L & \text{otherwise,} \end{cases}$$

for every i in $\{1,\ldots,m\}$.

Observe that vectors $u_\alpha^{\min}$ and $u_\alpha^{\max}$ depend on the solution vector x. However there are various classical situations when this not the case. Suppose f depends linearly on the parameters and the signs of the linear coefficients—depending on the components of x—do not change. Then vectors $u_\alpha^{\min}$ and $u_\alpha^{\max}$ do not depend on x and Inequalities (4) may now be written as follows:

$$f(x;u_\alpha^{\min}) \leq f(x;\bar{u}) \leq f(x;u_\alpha^{\max}). \tag{5}$$

For example this situation occurs when the objective function is a polynomial of the non-negative variables $x_1,\ldots,x_n$ with parameters $\tilde{u}_1,\ldots,\tilde{u}_m$ as coefficients. In Section 3 we will examine a linear case

$$f(x;\tilde{c}) = \tilde{c}^\mathsf{T} x = \sum_{j=1}^n \tilde{c}_j x_j$$

and a quadratic case

$$f(x;\tilde{Q},\tilde{c}) = x^\mathsf{T}\tilde{Q}x - \tilde{c}^\mathsf{T}x = \sum_{i,j=1}^n \tilde{Q}_{ij}x_i x_j - \sum_{j=1}^n \tilde{c}_j x_j.$$

2.2. Defuzzification

Optimization problem (FP) is ill-defined. Indeed the objective space $\tilde{\mathcal{Y}} = \{f(x;\tilde{u}) : x \in \mathcal{X}\}$ is not equipped with an *a priori* order relation. Some authors suggest to define such an order to address this shortcoming [2, 3, 13, 14, 17, 19]. We rather proceed by defuzzification: one or more crisp optimization programs are substituted to the initial fuzzy program (FP) and they are solved with classical methods [11]. A typical defuzzification strategy consists in dealing with the α-level cuts of the fuzzy quantities involved rather their membership functions [5, 10, 11, 15]. It has also been suggested to replace the fuzzy objectives by their *nearest interval approximations* [12]. We carry on with the former strategy. Let α be in $\{\alpha_1, \ldots, \alpha_K\}$. We replace the fuzzy objective values $f(x;\tilde{u})$ with their α-level cuts $[f(x;\tilde{u})]_\alpha$, for each x in $\mathcal{X}$, which are bounded closed intervals according to Lemma 1. Therefore the new optimization problem, indexed by α, is the interval minimization program below:

$$\begin{aligned} \text{minimize} \quad & [f(x;\tilde{u})]_\alpha \\ \text{subject to} \quad & x \in \mathcal{X}. \end{aligned} \tag{IP_α}$$

To give meaning to (IP_α) we need to discuss which order on compact intervals to choose so as to compare them. Let a_1, b_1, a_2 and b_2 be real numbers such that $a_1 \leq b_1$ and $a_2 \leq b_2$. We denote $[a_1, b_1] = \{t \in \mathbb{R} : a_1 \leq t \leq b_1\}$ and we use similar notations for bounded closed intervals in the rest of this article. If the lowest achievable value of the first interval, a_1, is smaller than or equal to a_2, and its worst possible value, b_1, is lower than or equal to b_2, then it seems reasonable to say the first interval is less than or equal to the second one. Thus we define a binary relation $\preceq$ on bounded closed intervals of $\mathbb{R}$ as follows: $[a_1, b_1] \preceq [a_2, b_2]$ if and only if $a_1 \leq a_2$ and $b_1 \leq b_2$ both hold. This order was also chosen in [12]. It is straigthforward to verify that $\preceq$ is a partial order on bounded closed intervals of $\mathbb{R}$. Two bounded closed intervals are not necessarily comparable with respect to $\preceq$. In other words it may happen that neither $[a_1, b_1] \preceq [a_2, b_2]$ nor $[a_2, b_2] \preceq [a_1, b_1]$ holds, for example consider $[0, 3]$ and $[1, 2]$. More precisely, $[a_1, b_1]$ and $[a_2, b_2]$ cannot be compared by $\preceq$ if and only if either $a_1 < a_2 \leq b_2 < b_1$ or $a_2 < a_1 \leq b_1 < b_2$ holds. Both these cases correspond to situations where one of the two intervals is a subset of the other and they do not share a common bound. Thus $\preceq$ is not a total order on bounded closed intervals. Consequently the collection $\{[f(x;\tilde{u})]_\alpha : x \in \mathcal{X}\}$ of bounded closed intervals may not have a infimum—which would be a bounded closed interval itself—with respect to $\preceq$, let alone a least element. Therefore we rather seek minimal elements of this collection with respect to $\preceq$. Thus we will say that x^* is a satisficing solution of (IP_α) if $[f(x^*;\tilde{u})]_\alpha$ is a minimal element of $\{[f(x;\tilde{u})]_\alpha : x \in \mathcal{X}\}$ with respect to $\preceq$.

The previous definition is closely related to the notion of Pareto efficiency [7]. Indeed a solution x^* in $\mathcal{X}$ is a satisficing solution of (IP_α) if and only if there is no x in $\mathcal{X}$ such that both $[f(x;\tilde{u})]_\alpha \preceq [f(x^*;\tilde{u})]_\alpha$ and $[f(x;\tilde{u})]_\alpha \neq [f(x^*;\tilde{u})]_\alpha$ hold. In other words there is no x in $\mathcal{X}$ such that $f(x;\tilde{u})^L_\alpha \leq f(x^*;\tilde{u})^L_\alpha$ and $f(x;\tilde{u})^U_\alpha \leq f(x^*;\tilde{u})^U_\alpha$ both hold with at least one strict inequality. This is exactly the definition of the (Pareto) efficiency of x^* for the following biobjective program:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \left(f(x;\tilde{u})^L_\alpha, f(x;\tilde{u})^U_\alpha\right) \\ \text{subject to} \quad & x \in \mathcal{X}. \end{aligned} \tag{BP_α}$$

Thus the set of satisficing solutions of (IP_α) is merely the set of efficient points—the efficient set—associated with (BP_α). In order to lighten the notations from now on we will denote $x \preceq_\alpha x'$ instead of $[f(x;\tilde{u})]_\alpha \preceq [f(x';\tilde{u})]_\alpha$ for every x and x' in $\mathcal{X}$. Furthermore we will denote $\mathrm{Eff}_\alpha(\mathcal{X})$ the efficient set of (BP_α). There may very well be several satisficing solutions. Since they are minimal elements with respect to $\preceq_\alpha$, they cannot be compared with each other using $\preceq_\alpha$ and, as emphasized in the previous paragraph, one of the corresponding α-level cuts of the two fuzzy objective values is included in the other without sharing a common bound.

2.3. Algorithm

The elements of $\mathbb{R}$ with the greatest membership degrees relatively to a fuzzy objective value $f(x;\tilde{u})$, x in $\mathcal{X}$, are in the highest α-level cut accessible. This cut is $[f(x;\tilde{u})]_{\alpha_1}$ for each feasible solution x. Therefore

we advocate to use the α_1-level cut to build a first solution set $\mathcal{X}_1$ to be refined iteratively using the lower accessible cut levels with the following process. We start with computing the set of satisficing solutions of (IP_{α_1}), that is the efficient set $\mathcal{X}_1 = \mathrm{Eff}_{\alpha_1}(\mathcal{X})$ of the biobjective problem (BP_{α_1}). As we explained in the previous section, the efficient solutions in $\mathcal{X}_1$ cannot be compared with $\preceq_{\alpha_1}$. If x and x' are two distinct solutions in $\mathcal{X}_1$ then one of the two intervals $[f(x;\tilde{u})]_{\alpha_1}$ and $[f(x';\tilde{u})]_{\alpha_1}$ is a subset of the other with no common bound. The originality of our method resides in using iteratively the order relations $\preceq_{\alpha_2}, \ldots, \preceq_{\alpha_K}$ corresponding to lower cut levels to exclude the respective inefficient solutions. Thus in the second iteration of the algorithm we compute the set $\mathcal{X}_2 = \mathrm{Eff}_{\alpha_2}(\mathcal{X}_1)$, then the set $\mathcal{X}_3 = \mathrm{Eff}_{\alpha_3}(\mathcal{X}_2)$ in the third iteration, and so on until all of the accessible cut levels have been considered, in decreasing order. We summarize this algorithm in the following paragraph.

Algorithm 1.

1. Set $\mathcal{X}_0 = \mathcal{X}$.
2. For $k = 1, \ldots, K$ compute $\mathcal{X}_k = \mathrm{Eff}_{\alpha_k}(\mathcal{X}_{k-1})$.
3. The final set of satisficing solutions is $\mathcal{X}_K$.

In order to unify the notations we denote $\mathcal{X}_0 = \mathcal{X}$. Let k be in $\{1, \ldots, K\}$. Observe that the elements of $\mathcal{X}_k$ are the solutions in $\mathcal{X}_{k-1}$ which are efficient with respect to the order $\preceq_{\alpha_k}$ within the set $\mathcal{X}_{k-1}$. In other words, the elements of $\mathcal{X}_k$ are the minimal elements of $\mathcal{X}_{k-1}$ with respect to the order $\preceq_{\alpha_k}$. In particular $\mathcal{X}_k$ is a subset of $\mathcal{X}_{k-1}$. As a consequence $\{\mathcal{X}_k\}_{0 \leq k \leq K}$ is a non-increasing family of sets. In particular the final set $\mathcal{X}_K$ is a subset of $\mathcal{X}_1$. Thus the final satisficing solutions were derived from the greatest cut level available, α_1, and then have been selected from the others based on the information given by lower α-level cuts.

3. Numerical examples and comparison with other methods

In this section we apply the algorithm presented in Section 2 to two numerical examples—a linear one and a quadratic one—and we make comparative observations relatively to other methods found in the literature [12, 15].

3.1. A linear example

We consider a fuzzy linear program studied in [15] with a most inspiring approach. After defining the problem we will solve it with Algorithm 1 then compare our method and results with the contents of [15]. The fuzzy objective values depend linearly on two real decision variables x_1 and x_2: $f(x;\tilde{c}) = -x_1\tilde{c}_1 - x_2\tilde{c}_2$ where $\tilde{c}_1$ and $\tilde{c}_2$ are fuzzy numbers assumed to satisfy Assumption 1. For details about the arithmetic of fuzzy numbers we refer the reader to [6, 9]. The feasible set is a polytope of $\mathbb{R}^2$ given by $\mathcal{X} = \{x \in \mathbb{R}^2 : Ax \leq b, x \geq 0\}$, where A and b are respectively a 9×2 real matrix and a 9-dimensional real vector defined as follows:

$$A^{\mathsf{T}} = \begin{bmatrix} 1 & 1 & 1 & 3 & 3 & 7 & 1 & 3 & 2 \\ 4 & 3 & 2 & 5 & 4 & 8 & 1 & 2 & 1 \end{bmatrix},$$
$$b^{\mathsf{T}} = \begin{bmatrix} 100 & 76 & 53 & 138 & 120 & 260 & 36 & 103 & 68 \end{bmatrix}.$$

The feasible polytope $\mathcal{X}$ is represented in Figure 1. We denote (FLP) this fuzzy linear program.

$$\begin{aligned} &\text{Minimize} && -\tilde{c}^{\mathsf{T}}x \\ &\text{subject to} && x \in \mathcal{X}. \end{aligned} \tag{FLP}$$

Only four α-level cuts of each fuzzy parameter are known, corresponding to $\alpha = 0, 0.25, 0.5, 0.75$. They are shown in Table 1. We can observe they are compatible with Assumption 1: the cuts are closed bounded intervals. Let us apply Algorithm 1 to (FLP). Let x be in $\mathcal{X}$. Since $f(x;\cdot)$ is linear on $\mathbb{R}^m$ Assumption 2 is satisfied. Therefore $f(x;\tilde{c})$ is a compact fuzzy number (Lemma 1). Thus Algorithm 1 is properly defined

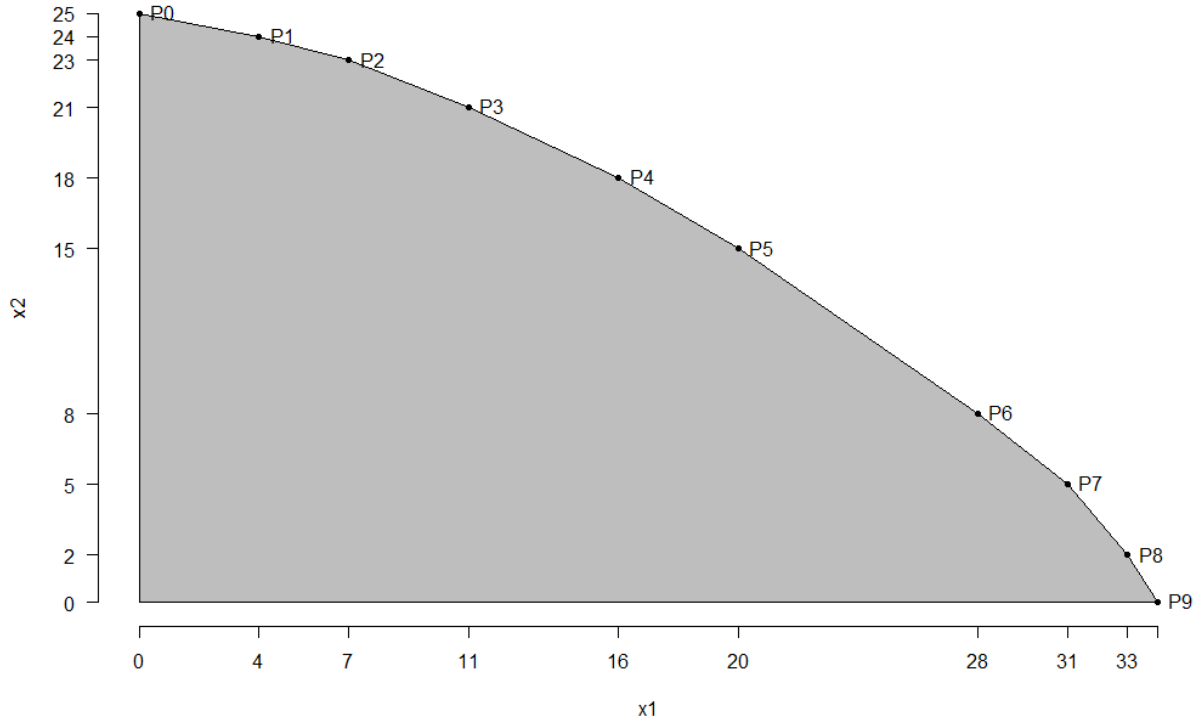

Figure 1: Feasible polytope $\mathcal{X}$ with vertices $P_0, \ldots, P_9$ and $(0,0)$.

v

for this problem. Furthermore Assumption 3 is satisfied and the non-negativity of the feasible decisions's components entails that the vectors $c_\alpha^{\min}$ and $c_\alpha^{\max}$ defined in Lemma 2—with the letter u—do not depend on x:

$$c_\alpha^{\min} = \begin{bmatrix}(\tilde{c}_1)\alpha & (\tilde{c}_2)\alpha\end{bmatrix}^\mathsf{T} \text{ and } c_\alpha^{\max} = \begin{bmatrix}(\tilde{c}_1)_\alpha^L & (\tilde{c}_2)_\alpha^L\end{bmatrix}^\mathsf{T}.$$

As a consequence the cut bounds of $f(x;\tilde{c})$ are straigthforward to compute:

$$\begin{aligned} f(x;\tilde{c})_\alpha^L &= -x_1(\tilde{c}_1)\alpha - x_2(\tilde{c}_2)\alpha, \\ f(x;\tilde{c})\alpha &= -x_1(\tilde{c}_1)_\alpha^L - x_2(\tilde{c}_2)_\alpha^L, \end{aligned} \quad \alpha = 0, 0.25, 0.5, 0.75. \tag{6}$$

Since the objectives $f(\cdot\,;\tilde{c})_\alpha^L$ and $f(\cdot\,;\tilde{c})\alpha$ are linear (6) and $\mathcal{X}$ is a polytope the objective space $\mathcal{Y}_\alpha = \{(f(x;\tilde{c})_\alpha^L, f(x;\tilde{c})\alpha) : x \in \mathcal{X}\}$ is also a polytope whose vertices are the images of the vertices of $\mathcal{X}$—that is $(0,0)$ and the points denoted $P_0, \ldots, P_9$ in Figure 1. The set $\mathcal{Y}_{0.75}$ is partially represented in the top left graph of Figure 2—we already know the image of the vertex $(0,0)$ of $\mathcal{X}$ is $(0,0)$, so we focus on the other vertices whose images have negative coordinates. We can see on this graph that the set of efficient solutions $\mathrm{Eff}_{0.75}(\mathcal{X})$ is the edge P_2P_3 of $\mathcal{X}$. Let us precise that the image of P_3 is $(-207, -171.7)$ and the image of P_4 is $(-207, -168.2)$ so that P_4 is indeed dominated by P_3 and is not an efficient point of $\mathrm{Eff}_{0.75}(\mathcal{X})$. Thus the first iteration in Algorithm 1 gives $\mathcal{X}_1 = \mathrm{Eff}_{0.75}(\mathcal{X}) = P_2P_3$. We verify on the other three graphs of Figure 2 that the points of P_2P_3 remain efficient for $\alpha = 0, 0.25, 0.5$, therefore $\mathcal{X}_2 = \mathrm{Eff}_{0.5}(\mathcal{X}_1) = P_2P_3$, $\mathcal{X}_3 = \mathrm{Eff}_{0.25}(\mathcal{X}_2) = P_2P_3$ and $\mathcal{X}_4 = \mathrm{Eff}_0(\mathcal{X}_3) = P_2P_3$. Thus the algorithm does not return a single solution but a whole edge of the polytope. The parameters of the problem (FLP) are such that, for each α in $[0,1]$,

α	$[\tilde{c}_1]_\alpha$	$[\tilde{c}_2]_\alpha$
0	$[0.5, 10]$	$[1.4, 11]$
0.25	$[1, 7]$	$[5, 10]$
0.50	$[2, 5]$	$[6, 9]$
0.75	$[3.2, 4.5]$	$[6.5, 7.5]$

Table 1: α-level cuts of $\tilde{c}_1$ and $\tilde{c}_2$ for $\alpha = 0, 0.25, 0.5, 0.75$.

the bounds $f(x;\tilde{c})^L_\alpha$ and $f(x;\tilde{c})\alpha$ for $x = (1-\lambda)P_2 + \lambda P_3$ decreases and increases respectively as λ increases in $[0,1]$. In particular the α-level cuts of $f(x;\tilde{c})$ for x in P_2P_3 are subsets and supsets of the α-level cuts corresponding to P_3 and P_2 respectively. For the sake of lisibility in Figure 3 we have only represented the bounds for P_2 ($\lambda = 0$) with squares and P_3 ($\lambda = 1$) with triangles.

Since $\text{Eff}_{0.75} = P_2P_3$ the bounding of the true objective value (5) for each x in P_2P_3 becomes

$$(c_\alpha^{\min})^\mathsf{T} x \leq \bar{c}^\mathsf{T} x \leq (c_\alpha^{\max})^\mathsf{T} x$$

where $\bar{c}$ is the true (unaccessible) vector of parameters.

In [15] the solutions of (FLP) are also defined as efficient points of the mulitiobjective minimization of the available α-cut bounds. However, contrary to Algorithm 1, all the cut bounds are considered simultaneously, whatever their cut level, in a single $2K$-objectives optimization program:

$$\begin{aligned} \text{minimize} \quad & \Big(f(x;\tilde{u})^L_{\alpha_1}, \ldots, f(x;\tilde{u})^L_{\alpha_K}, f(x;\tilde{u})\alpha_1, \ldots, f(x;\tilde{u})\alpha_K \Big) \\ \text{subject to} \quad & x \in \mathcal{X}. \end{aligned} \tag{MOP}$$

Thus the cuts corresponding to lower levels are taken into account from the start, not as part of an iterative process to refine a first solution set $\mathcal{X}_1$ as in Algorithm 1. When considering the resolution of (FLP) the actual set of efficient solutions of (MOP) is found to be P_2P_3, which is consistent with the results of our method. However the authors of [15] are not looking for every efficient points of (MOP). They take advantage of a method presented in [23] to single out an efficient solution as follows. A membership function is defined for each objective of (MOP) as a linear transform of it. For each k in $\{1, \ldots, K\}$ the membership function $\mu^L_{\alpha_k}$ is given by

$$\mu^L_\alpha(x) = \begin{cases} \dfrac{f(x\alpha;\tilde{u})^L_\alpha - f(x;\tilde{u})^L_\alpha}{f(x\alpha;\tilde{u})^L_\alpha - f(x^L_\alpha;\tilde{u})^L_\alpha} & \text{if } f(x^L_\alpha;\tilde{u})^L_\alpha \leq f(x;\tilde{u})^L_\alpha < f(x\alpha;\tilde{u})^L_\alpha, \\ 0 & \text{otherwise,} \end{cases}$$

for all x in $\mathcal{X}$, where x^L_α and $x\alpha$ are respectively minimizers of $f(\cdot\,;\tilde{u})^L_\alpha$ and $f(\cdot\,;\tilde{u})\alpha$ over $\mathcal{X}$. A linear transform $\mu\alpha$ of the upper bound $f(\cdot\,;\tilde{u})\alpha$ is defined similarly. Then the following single-objective optimization program is solved:

$$\begin{aligned} \text{maximize} \quad & \min\{\mu^L_{\alpha_k}(x), \mu\alpha_k(x) : k = 1, \ldots, K\} \\ \text{subject to} \quad & x \in \mathcal{X}. \end{aligned} \tag{7}$$

The optimal solutions of (7) are efficient solutions of (MOP) since we are dealing with a linear program (see [23] for details). The efficient solution attained in [15] when considering the linear example (FLP) is $(9, 22)$. This point is the middle of the edge P_2P_3—whose cuts are represented on Figure 3—, thus a solution consistent with the results of Algorithm 1. An advantage of this method is that a single efficient solution is returned so that the decision-maker does not have to choose among undifferentiated solutions. All things considered this method requires $2K$ single-objective linear minimizations to compute $x^L_{\alpha_k}$ and $x\alpha_k$ for $k = 1, \ldots, K$ and an additional single-objective linear minimization to solve (7)—which can indeed be written as a linear program [15, 23]. In a nonlinear framework a $2K$-objectives optimization program,

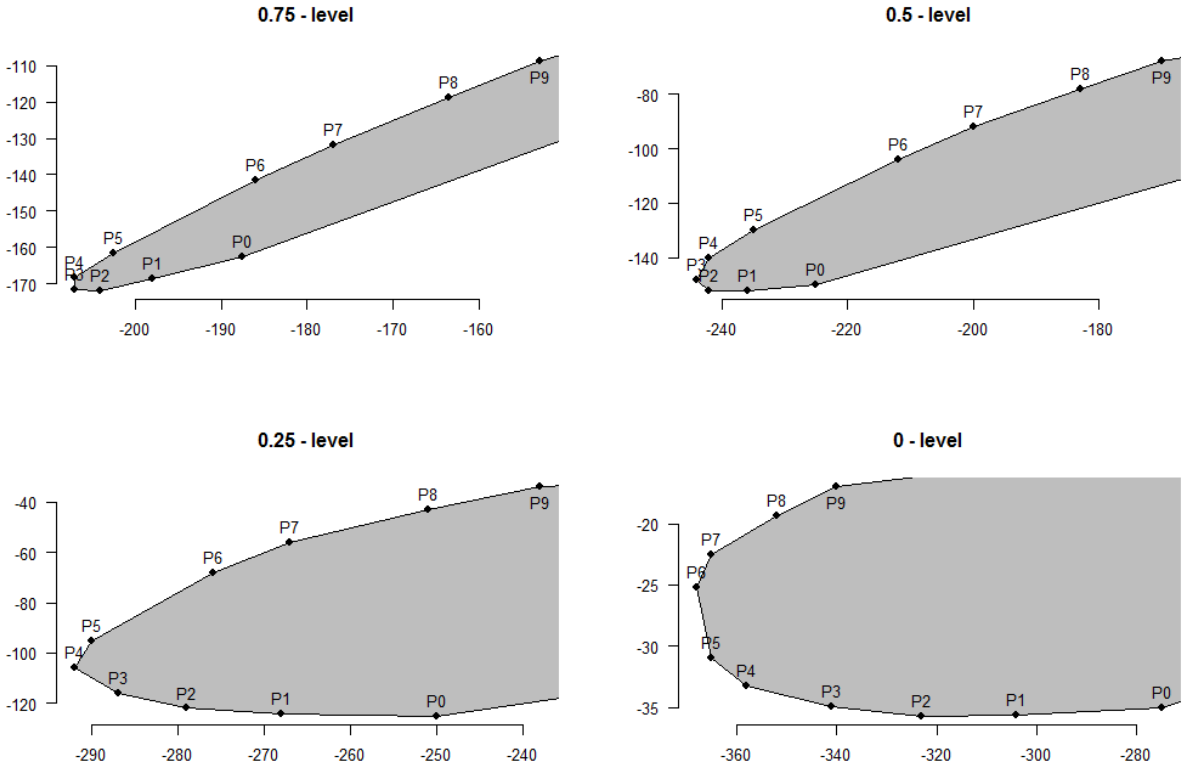

Figure 2: South-west part of $\mathcal{Y}_\alpha = \{(f(x;\tilde{c})^L_\alpha, f(x;\tilde{c})\alpha) : x \in \mathcal{X}\}$—with lower bound on the horizontal axis and upper bound on the vertical axis.

namely (MOP), would have to be solved. Most of all giving same importance to bounds of different cuts disregarding their α-level seems surprising. It would be reasonable that the significance of an α-level cut increases with α. Even though this approach singles out a point of $\mathcal{X}_1 = P_2P_3$ we will discuss in Section 4 why we think Algorithm 1 provides better guarantees.

3.2. A quadratic example

We propose to study the following fuzzy quadratic program. Here the function f depends on three real variables x_1, x_2, x_3 and on twelve fuzzy parameters: the nine coefficients values of a 3×3 symmetric matrix $Q = [Q_{ij}]_{1\leq i,j\leq 3}$ and the three coefficients values of a three-dimensional vector $c = [c_1\ c_2\ c_3]^\mathsf{T}$. The objective value of a point $x = [x_1\ x_2\ x_3]^\mathsf{T}$ in $\mathbb{R}^3$ is given by

$$\begin{aligned} f(x;\tilde{Q},\tilde{c}) &= \frac{1}{2}x^\mathsf{T}\tilde{Q}x - \tilde{c}^\mathsf{T}x \\ &= \frac{1}{2}(\tilde{Q}_{11}x_1^2 + \tilde{Q}_{22}x_2^2 + \tilde{Q}_{33}x_3^2) + \tilde{Q}_{12}x_1x_2 + \tilde{Q}_{13}x_1x_3 + \tilde{Q}_{23}x_2x_3 - \tilde{c}_1x_1 - \tilde{c}_2x_2 - \tilde{c}_3x_3, \end{aligned} \tag{8}$$

where $\tilde{Q}$ and $\tilde{c}$ are respectively a matrix and a vector with fuzzy numbers as coefficients. We consider box constraints: $\mathcal{X} = \{x \in \mathbb{R}^3 : 0 \leq x_1 \leq 3, 0 \leq x_2 \leq 3, 0 \leq x_3 \leq 3\}$. We denote (FQP) this fuzzy quadratic

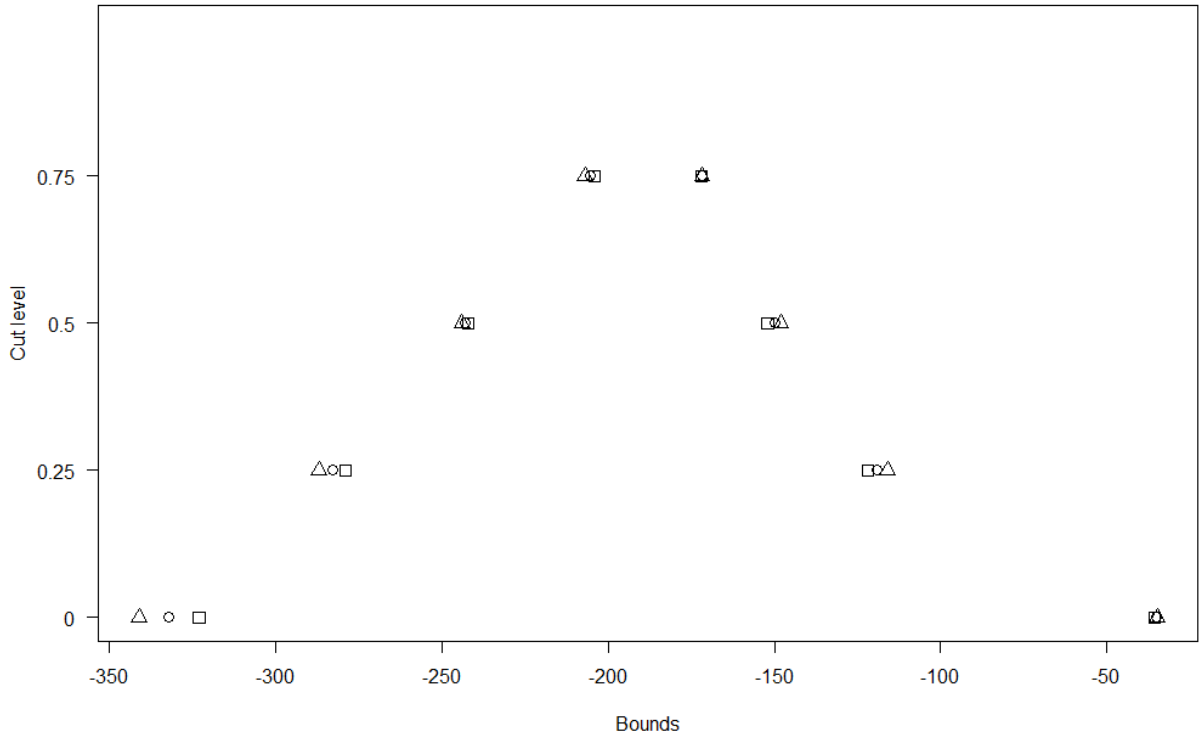

Figure 3: Bounds of $[f(x;\tilde{c})]_\alpha$ for $x = P_2$ (squares), $x = P_3$ (triangles) and $x = (P_2 + P_3)/2$ (circles), and cut levels $\alpha = 0, 0.25, 0.5, 0.75$.

program.

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} x^\mathsf{T} \tilde{Q} x - \tilde{c}^\mathsf{T} x \\ & 0 \leq x_1 \leq 3 \\ \text{subject to} \quad & 0 \leq x_2 \leq 3 \\ & 0 \leq x_3 \leq 3. \end{aligned} \tag{FQP}$$

We consider a fuzzy matrix $\tilde{Q}$ and a fuzzy vector $\tilde{c}$ with *triangular* fuzzy numbers as coefficients. We denote $\langle m; a, b\rangle$—where m, a, b are real numbers with a and b positive—the triangular fuzzy number defined by the following membership function: for all t in $\mathbb{R}$,

$$\mu_{\langle m;a,b\rangle}(t) = \begin{cases} 1 - \dfrac{m-t}{a} & \text{if } m - a \leq t \leq m, \\ 1 - \dfrac{t-m}{b} & \text{if } m \leq t \leq m + b, \\ 0 & \text{otherwise.} \end{cases}$$

We suppose we know the α-level cuts corresponding to $\alpha = 0, 0.25, 0.5, 0.75$ of the following fuzzy matrix $\tilde{Q}$ and fuzzy vector $\tilde{c}$ with triangular coefficients:

$$\tilde{Q} = \begin{bmatrix} \tilde{Q}_{11} & \tilde{Q}_{12} & \tilde{Q}_{13} \\ \tilde{Q}_{21} & \tilde{Q}_{22} & \tilde{Q}_{23} \\ \tilde{Q}_{31} & \tilde{Q}_{32} & \tilde{Q}_{33} \end{bmatrix} = \begin{bmatrix} \langle 9; 1, 3\rangle & \langle -1; 1, 1\rangle & \langle 1; 1, 1\rangle \\ \langle -1; 1, 1\rangle & \langle 5; 2, 2\rangle & \langle -1; 1, 1\rangle \\ \langle 1; 1, 1\rangle & \langle -1; 1, 1\rangle & \langle 9; 3, 1\rangle \end{bmatrix} \text{ and } \tilde{c} = \begin{bmatrix} \tilde{c}_1 \\ \tilde{c}_2 \\ \tilde{c}_3 \end{bmatrix} = \begin{bmatrix} \langle 50; 99, 4\rangle \\ \langle 50; 40, 40\rangle \\ \langle 20; 4, 99\rangle \end{bmatrix}. \tag{9}$$

We do not suppose the whole membership functions of the fuzzy parameters (9) are known. We only assume knowledge of four of their cuts. We recall that the bounds of the α-level cut of a triangular fuzzy number $\langle m; a, b\rangle$ are given by

$$\langle m; a, b\rangle_\alpha^L = m - (1-\alpha)a \quad \text{and} \quad \langle m; a, b\rangle\alpha = m + (1-\alpha)b, \quad \alpha \in [0, 1]. \tag{10}$$

Since triangular fuzzy numbers are particular cases of compact fuzzy numbers Assumption 1 holds. Let x be in $\mathcal{X}$. Function $f(x; \cdot\,)$ is obviously linear (8), therefore Assumption 2 is verified and we can apply Algorithm 1 to solve (FQP). Since the coordinates of the feasible solutions are non-negative Assumption 3 holds as well. Furthermore the following holds

$$\tilde{Q}_\alpha^{\min} = \tilde{Q}_\alpha^L, \;\; \tilde{Q}_\alpha^{\max} = \tilde{Q}\alpha, \;\; \tilde{c}_\alpha^{\min} = \tilde{c}\alpha \text{ and } \tilde{c}_\alpha^{\max} = \tilde{c}_\alpha^L.$$

Consequently the bounds of the α-level cuts of $f(x; \tilde{Q}, \tilde{c})$ are given as below

$$\begin{matrix} f(x; \tilde{Q}, \tilde{c})_\alpha^L = x^\mathsf{T}\tilde{Q}_\alpha^L x - (\tilde{c}\alpha)^\mathsf{T} x, \\ f(x; \tilde{Q}, \tilde{c})\alpha = x^\mathsf{T}\tilde{Q}\alpha x - (\tilde{c}_\alpha^L)^\mathsf{T} x, \end{matrix} \quad \alpha = 0, 0.25, 0.5, 0.75.$$

Here we compute the set $\mathcal{X}_1 = \mathrm{Eff}_{\alpha_1}(\mathcal{X})$ numerically thanks to the NSGA-II multiobjective optimization algorithm [4]. A remarkable feature of this algorithm is to produce a population of efficient points whose images—*i.e.* the empirical Pareto front—are well spread out in the objective space. Here we set the population size to $|\mathcal{X}_1| = 40$ and run 100 iterations of the NSGA-II algorithm. The subsequent sets $\mathcal{X}_k$ are obtained comparing pairwise the current solutions in $\mathcal{X}_{k-1}$ with respect to $\preceq_{\alpha_k}$, $k = 2, \ldots, K$. The 40 points of the Pareto front obtained are represented on the top left graph of Figure 4.

The size of the solution set $\mathcal{X}_k$, $k = 1, 2, 3, 4$, decreases during the process as shown in Table 2.

k	α_k	$\lvert\mathcal{X}_k\rvert$
1	0.75	40
2	0.5	30
3	0.25	24
4	0	21

Table 2: Size of the solution set along the process.

The final solution set $\mathcal{X}_4$ contains 21 decision points instead of 40 at the start of the process.

Here we could once again compare our results with the approach found in [15]. That would mean solving a $2K$-objectives optimization problem similar to (MOP), by means of the NSGA-II algorithm for example. The worst-case complexity of both approaches is the same. Denoting N the desired number of efficient points—set to 40 above—and M the number of objectives the worst-case complexity of NSGA-II is $\mathcal{O}(MN^2)$. Algorithm 1 requires K biobjectives minimizations while the other approach requires one $2K$-objectives minimization. Therefore they both have a worst-case complexity of $\mathcal{O}(KN^2)$. As stated before, we believe our method guarantees more satisfying solutions and we will explain why in Section 4. Instead this time we will draw comparisons with a recently published method originally intended for fuzzy multiobjective programs [12]. This paper suggests to define a surrogate problem by substituting the fuzzy objectives values by their *nearest interval approximations* (NIA) [8] and then solving a multiobjective interval minimization

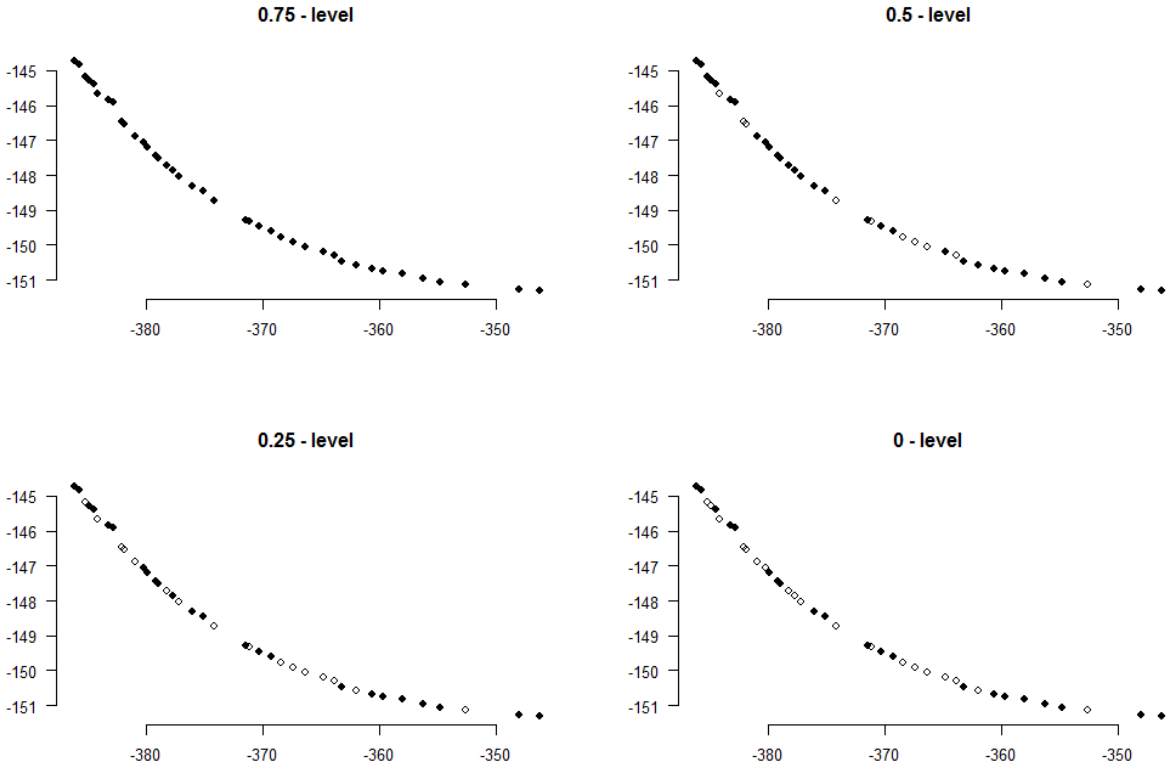

Figure 4: Pareto front of (IP_α) for $\alpha = 0.75$ in the quadratic example—with the lower bound on the horizontal axis and the upper bound on the vertical one. The points discarded during the second, third and fourth iterations are represented by white circles.

program. In the fuzzy single-objective case (FP) this approach consist in replacing the fuzzy objectives values $f(x;\tilde{u})$, x in $\mathcal{X}$, by their nearest interval approximations and then solve the interval minimization program obtained in the same way as presented in Section 2.2. This approach actually leads to a special case of our method when the components of $\tilde{u}$ are triangular fuzzy numbers and f is linear with respect to these. We refer the reader to [8] for the definition and properties of the nearest interval approximation of a fuzzy number. Here we only need to know that the nearest interval approximation NIA($\tilde{a}$) of a compact fuzzy number $\tilde{a}$ is given by

$$\mathrm{NIA}(\tilde{a}) = \left[\int_0^1 \tilde{a}_\alpha^L \mathrm{d}\alpha, \int_0^1 \tilde{a}\alpha \mathrm{d}\alpha\right]. \tag{11}$$

Following this approach the fuzzy quadratic objective of (FQP) is replaced by its nearest interval approximation. Since the coefficients of $\tilde{Q}$ and $\tilde{c}$ are triangular fuzzy numbers and the underlying function f is linear with respect to the fuzzy parameters, the objective values $f(x;\tilde{Q},\tilde{c})$, x in $\mathcal{X}$, are also triangular fuzzy numbers. We refer the reader to [9] for details on the arithmetic of triangular fuzzy numbers. For each α in $[0,1]$ the bounds of the α-level cut of a triangular fuzzy number $\tilde{a} = \langle m; a, b\rangle$ are given by (10). It follows from (11) that $\mathrm{NIA}(\tilde{a}) = [\tilde{a}]_{0.5}$. Consequently the surrogate program of (FP) suggested in [12] is (IP_α) for $\alpha = 0.5$. Therefore this approach coincides with the method presented in Section 2 with $K = 1$ and $\alpha_1 = 0.5$—only a single α-level cut is known—, when f is linear with respect to the parameters and the latter are triangular fuzzy numbers.

Observe that the NIA approach cannot be applied when only a few α-level cuts of the fuzzy parameters are known—as in the linear example of Section 3.1. Indeed the whole membership functions of the parameters

are required to compute their nearest interval approximations. When the bounds of the objective interval $[f(\cdot\,;\tilde{u})]_\alpha$ are convex and differentiable, as is the case for problems (FLP) and (FQP), it is advised in [12] to consider the minimizers of $f(\cdot\,;\tilde{u})^L_{0.5} + f(\cdot\,;\tilde{u})0.5$ as satisficing solutions of the initial problem (FP). This approach leads to a more computationally tractable method, the solution points being sought as solutions of a KKT system. However doing so only a subset of $\mathrm{Eff}_{0.5}(\mathcal{X})$ is taken into account rather than the totality of efficient points—refer to the *weighted sum method* [7].

4. Optimality degree

In this section we suggest the definition of an 'optimality degree' criterion to assess the quality of a feasible solution of (FP) in terms of precision level.

In Section 3 we compared our method with two other approaches. The first one considers all the cut levels available $\alpha_1, \ldots, \alpha_K$ at once while ours selects solutions among the efficient points corresponding to the highest level α_1. We feel that this first approach should not be able to guarantee an 'optimality degree' higher than the lowest cut level α_K. Indeed the objective values at stake corresponding to α_K might have a more significant influence on a solution x than the ones associated with $\alpha_1, \ldots, \alpha_{K-1}$, *e.g.* x is efficient only because it minimizes $f(\cdot\,;\tilde{u})^L_{\alpha_K}$ or $f(\cdot\,;\tilde{u})^U_{\alpha_K}$. In contrast our method exclusively selects points x which are efficient solutions of (IP_α) for $\alpha = \alpha_1$, the highest cut level at hand, therefore the 'optimality degree' of such points x should be greater than or equal to α_1. The second approach considers nearest interval approximation (NIA) and we verified in Section 3.2 that it is equivalent to our method when only the 0.5-level cut is taken into account. Therefore the 'optimality degree' of solutions attained with this second approach should not be guaranteed higher than 0.5.

To define an 'optimality degree' for each feasible solution of (FP) we start from the following remark. For each vector u in $\mathbb{R}^m$ let us denote (P_u) the crisp optimization program below:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x;u) \\ \text{subject to} & x \in \mathcal{X}. \end{array} \qquad (P_u)$$

Let x be a feasible solution. In the spirit of the extension principle, the 'optimality degree' of x should be at least as big as the membership degree $\mu_{\tilde{u}}(u)$ of any parameters vector u such that x is an optimal solution of (P_u). Let us denote $\operatorname{Argmin} f(\mathcal{X};u) = \{x \in \mathcal{X} : f(x;u) = \inf f(\mathcal{X};u)\}$ the set of optimal solutions of (P_u) for each u in $\mathbb{R}^m$.

Definition 1. We define the *optimality degree* of x as the supremum of the membership degrees $\mu_{\tilde{u}}(u)$ of the parameters vectors u of $\mathbb{R}^m$ such that x belongs to $\operatorname{Argmin} f(\mathcal{X};u)$:

$$\sup\{\mu_{\tilde{u}}(u) : u \in \mathbb{R}^m, x \in \operatorname{Argmin} f(\mathcal{X};u)\}. \qquad (12)$$

The optimality degree of x can be interpreted as the possibility for x to be in the fuzzy set $\operatorname{Argmin} f(\mathcal{X};\tilde{u})$. Obviously the quantity (12) may be difficult to compute in general, however the straightforward following observation will allow us to bound the optimality degree below in several situations—especially under some convexity assumptions that will be precised further. If x is an optimal solution of (P_u) for some u in $[\tilde{u}]_\alpha$ and α in $[0,1]$ then the optimality degree of x is greater than or equal to α. This results directly from (12) and inequality $\mu_{\tilde{u}}(u) \geq \alpha$.

Now we will show that under some convexity assumptions of $\mathcal{X}$ and f it is possible to bound below the optimality degree of a solution of (FP) obtained with Algorithm 1.

Assumption 4.

1. The set $\mathcal{X}$ is a convex part of $\mathbb{R}^n$.
2. For every u in $[\tilde{u}]_{\alpha_1}$ the function $f(\cdot\,;u)$ is convex on $\mathcal{X}$.
3. For every x in $\mathcal{X}$ the function $f(x;\cdot\,)$ is linear on $\mathbb{R}^m$.

We recall the following classical result on the weighted sum method in the convex case [7].

Proposition 1. Let $g_1, \ldots, g_p$ be real-valued convex functions defined on $\mathbb{R}^n$ and x^* be in the convex feasible set $\mathcal{X}$. The point x^* is an efficient solution of the multiobjective problem

$$\begin{aligned} \text{minimize} \quad & (g_1(x), \ldots, g_p(x)) \\ \text{subject to} \quad & x \in \mathcal{X} \end{aligned}$$

if and only if there exist non-negative weights $\lambda_1, \ldots, \lambda_p$ satisfying $\sum_{k=1}^{p} \lambda_k = 1$ such that x^* is an optimal solution of the minimization problem $\min_{x \in \mathcal{X}} \sum_{k=1}^{p} \lambda_k g_k(x)$.

The solutions ultimately selected by Algorithm 1 are the elements of $\mathcal{X}_K$. Let Assumptions 1, 2, 3 and 4 hold and consider such a solution point x^*. Since $\mathcal{X}_K$ is a subset of the efficient set of (BP_α) for $\alpha = \alpha_1$, namely $\mathcal{X}_1$, it follows from Proposition 1 that x^* is an optimal solution of $\min\{(1-\lambda) f(x; u_{\alpha_1}^{\min}) + \lambda f(x; u_{\alpha_1}^{\max}) : x \in \mathcal{X}\}$ for some λ in $[0, 1]$. Furthermore, for every x in $\mathcal{X}$,

$$(1-\lambda) f(x; u_{\alpha_1}^{\min}) + \lambda f(x; u_{\alpha_1}^{\max}) = f(x; (1-\lambda) u_{\alpha_1}^{\min} + \lambda u_{\alpha_1}^{\max}),$$

where we used the linearity of $f(x; \cdot)$. Since $\tilde{u}$ is a vector of fuzzy numbers, $[\tilde{u}]_{\alpha_1}$ is a convex set and contains $u = (1-\lambda) u_{\alpha_1}^{\min} + \lambda u_{\alpha_1}^{\max}$. As a consequence x^* is in $\text{Argmin}\{f(x; u) : x \in \mathcal{X}\}$. It follows from inequality $\mu_{\tilde{u}}(u) \geq \alpha_1$ and Equation (12) that the optimality degree of x^* is greater than or equal to α_1. Thus every solution point provided at the end of Algorithm 1 has an optimality degree guaranteed to be greater than or equal to α_1.

As was verified in Section 3.2 the approach by nearest interval approximation is equivalent to Algorithm 1 under Assumption 4 in the presence of triangular fuzzy numbers—which are a very common type of fuzzy numbers—with $K = 1$ and $\alpha_1 = 0.5$. Therefore the solutions provided by the NIA approach have an optimality degree greater than or equal to 0.5. The other approach considered in Section 3.1 consists in solving a $2K$-objectives optimization program (MOP) gathering all the cut bounds without distinction of cut level α. The satisficing solutions of (FP) thus are defined as the efficient solutions of (MOP). Since the problem is linear Assumption 4 is obviously satisfied. Let x^* be an efficient solution of (MOP). Proposition 1 entails the existence of $2K$ non-negative weights $\lambda_{\alpha_1}^{\max}, \lambda_{\alpha_1}^{\min}, \ldots, \lambda_{\alpha_K}^{\max}, \lambda_{\alpha_K}^{\min}$ adding up to 1 such that x^* is an optimal solution of the crisp minimization problem below:

$$\begin{aligned} \text{minimize} \quad & -(\lambda_{\alpha_1}^{\max} c_{\alpha_1}^{\max} + \lambda_{\alpha_1}^{\min} c_{\alpha_1}^{\min} + \cdots + \lambda_{\alpha_K}^{\max} c_{\alpha_K}^{\max} + \lambda_{\alpha_K}^{\min} c_{\alpha_K}^{\min})^{\mathsf{T}} x \\ \text{subject to} \quad & Ax \leq b, x \geq 0. \end{aligned}$$

The objective vector $c = \lambda_{\alpha_1}^{\max} c_{\alpha_1}^{\max} + \lambda_{\alpha_1}^{\min} c_{\alpha_1}^{\min} + \cdots + \lambda_{\alpha_K}^{\max} c_{\alpha_K}^{\max} + \lambda_{\alpha_K}^{\min} c_{\alpha_K}^{\min}$ is in $[\tilde{c}]_{\alpha_K}$ by convexity, but we have no guarantee that c belongs to $[\tilde{c}]_{\alpha_k}$ for any k greater than K, that is an available cut of higher level. Therefore we can only say that the optimality degree of the satisficing solutions obtained by $2K$-objectives minimization is greater than or equal to α_K — which is barely useful when $\alpha_K = 0$.

5. Concluding remarks

We presented a method to solve optimization problems with fuzzy parameters in the objective function by iterative non-dominated sorting when only a few cuts of the fuzzy parameters are known. In theory this approach only requires the continuity of f with respect to the imprecise parameters. However monotonic dependence significantly alleviate the computational cost of the algorithm. We applied our method to two examples featuring continuous monotonic dependence—a linear one and a quadratic one—and drew comparisons with two other approaches found in the literature. In order to assess the guarantees provided by the satisficing solutions of the various methods considered we suggested the definition of an optimality degree. We showed that this criterion may be easily bounded below under some convexity assumptions. We proved that our method yields solutions with an optimality degree greater than or equal to the highest cut level available in such situations.

[1] Bellman, R. E. and Zadeh, L. A. (1970). Decision-making in a fuzzy environment. *Management science*, 17(4):B–141.
[2] Bortolan, G. and Degani, R. (1985). A review of some methods for ranking fuzzy subsets. *Fuzzy sets and Systems*, 15(1):1–19.
[3] Campos, L. and Verdegay, J. (1989). Linear programming problems and ranking of fuzzy numbers. *Fuzzy sets and systems*, 32(1):1–11.
[4] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197.
[5] Dempe, S. and Ruziyeva, A. (2012). On the calculation of a membership function for the solution of a fuzzy linear optimization problem. *Fuzzy Sets and Systems*, 188(1):58–67.
[6] Dubois, D. J. (1980). *Fuzzy sets and systems: theory and applications*, volume 144. Academic press.
[7] Ehrgott, M. (2006). *Multicriteria optimization*. Springer Science & Business Media.
[8] Grzegorzewski, P. (2002). Nearest interval approximation of a fuzzy number. *Fuzzy Sets and systems*, 130(3):321–330.
[9] Hanss, M. (2005). *Applied fuzzy arithmetic*. Springer.
[10] Liu, S.-T. (2009). Quadratic programming with fuzzy parameters: A membership function approach. *Chaos, Solitons & Fractals*, 40(1):237–245.
[11] Luhandjula, M. (2014). Fuzzy optimization: Milestones and perspectives. *Fuzzy Sets and Systems*.
[12] Luhandjula, M. K. and Rangoaga, M. J. (2014). An approach for solving a fuzzy multiobjective programming problem. *European Journal of Operational Research*, 232(2):249–255.
[13] Nasseri, S. (2008). Fuzzy nonlinear optimization. *The Journal of Nonlinear Sciences and its Applications*, 1(4):230–235.
[14] Orlovski, S. (1985). Mathematical programming problems with fuzzy parameters. *Management decision support systems using fuzzy sets and possibility theory*, pages 136–145.
[15] Rommelfanger, H., Hanuscheck, R., and Wolf, J. (1989). Linear programming with fuzzy objectives. *Fuzzy sets and systems*, 29(1):31–48.
[16] Rommelfanger, H. and Słowiński, R. (1998). Fuzzy linear programming with single or multiple objective functions. In *Fuzzy sets in decision analysis, operations research and statistics*, pages 179–213. Springer.
[17] Rommelfanger, H. J. (1996). Fuzzy linear programming and applications. *European Journal of Operational Research*, 92(3):512–527.
[18] Sakawa, M. and Yano, H. (1991). Feasibility and pareto optimality for multiobjective nonlinear programming problems with fuzzy parameters. *Fuzzy Sets and Systems*, 43(1):1–15.
[19] Tanaka, H. and Asai, K. (1984). Fuzzy linear programming problems with fuzzy numbers. *Fuzzy sets and systems*, 13(1):1–10.
[20] Tanaka, H., Okuda, T., and Asai, K. (1973). On fuzzy-mathematical programming.
[21] Zadeh, L. (1965). Fuzzy sets. *Information and Control*, 8:338–353.
[22] Zadeh, L. (1978). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1(1):3–28.
[23] Zimmermann, H.-J. (1978). Fuzzy programming and linear programming with several objective functions. *Fuzzy sets and systems*, 1(1):45–55.
[24] Zimmermann, H.-J. (2001). *Fuzzy set theory and its applications*. Springer Science & Business Media.

Conclusions et perspectives

Nous avons introduit un algorithme de résolution par tri non-dominé itératif pour résoudre une large classe de problèmes d'optimisation floue. Nous l'avons testé et comparé avec d'autres méthodes sur un problème linéaire et un problèmes quadratique. L'algorithme mériterait d'être appliqué à des problèmes convexes plus généraux — notons qu'il n'existe pas, ou peu, d'éléments de comparaison dans la littérature. Nous avons également introduit un critère pour estimer le niveau de précision de nos solutions en termes de degré d'appartenance, et l'avons également appliqué à trois méthodes de la littérature. Il serait intéressant d'évaluer la précision garantie par d'autres approches, en fonction du niveau de précision sur les données, afin de la comparer à celle obtenue avec notre algorithme. Une extension à des problèmes dont, non seulement la fonction objectif, mais aussi les contraintes, sont paramétrées par des nombres flous est envisageable.

Conclusion et perspectives

Dans ce mémoire nous avons présenté des contributions à l'optimisation sans dérivées sous incertitudes dans les thématiques de l'émulation de codes de simulation stochastiques, de la recherche directe directionnelle appliquée à des fonctions aux valeurs inexactes, et à l'optimisation de fonctions dépendant de paramètres imprécis modélisés par des nombres flous.
Nous avons présenté quatre constructions de métamodèles au Chapitre 2 pour des fonctions dont les valeurs sont des densités de probabilité. Nous les avons testé sur deux exemples analytiques et mis en application sur trois cas industriels. Cependant une seule de ces quatre approches est complètement aboutie (la méthode *Kernel Regression*). En effet les trois autres nécessitent l'élaboration de modèles liant les variables d'entrée aux coefficients de l'estimateur. Les auteurs de [15, 16] proposent une approche par krigeage pour la méthode *Modified Magic Points* et intègrent le métamodèle complet dans une adaptation de l'algorithme EGO pour la résolution d'un problème de maximisation de quantile. Un tel métamodèle complet pourrait être mis en œuvre pour obtenir de premiers résultats sur le problème inverse de l'optimisation des champs de forces en modélisation moléculaire : trouver les paramètres du modèle — les entrées du code GIBBS — qui permettent de reproduire au mieux les distributions expérimentales de données macroscopiques du système, telles que la masse volumique de la phase liquide et la pression de la phase gazeuse.
L'analyse menée au Chapitre 3 sur l'algorithme de recherche directe directionnelle appliquée à des fonctions inexactes, dont l'erreur est uniformément bornée, a permis de mettre en évidence une borne sur la précision que l'on peut espérer atteindre sur la norme du gradient, une borne sur le nombre d'itérations nécessaires pour atteindre une précision donnée, ainsi qu'un critère d'arrêt basé sur la taille du pas. Nous avons vérifié ces résultats théoriques par des expériences numériques et observé le comportement de l'algorithme sur le problème du placement de puits pétroliers en ingénierie de réservoir, dont la fonction objectif est soumise à des paramètres incertains. Ces tests numériques sont à approfondir en considérant davantage de problèmes, en étudiant les résultats plus en détails et en les comparant aux résultats obtenus avec d'autres algorithmes de la littérature. Nous avons ici considéré que l'écart entre les évaluations de f par oracles ou les véritables valeurs était uniformément borné. Une voie de recherche future serait de considérer le cas d'un bruit non borné, gaussien par exemple, appartenant à un certain intervalle avec une probabilité supérieure à un seuil donné. Ces résultats nous ont également conduit à définir un algorithme adaptatif pour les situations où différent oracles d'évaluation sont disponibles avec des niveaux de précision distincts. Nous avons testé cet algorithme sur un problème de minimisation de surface pour lequel trois oracles d'ordres de précision différents étaient disponibles. Nous avons appliqué l'algorithme de recherche di-

recte directionnelle au problème de placement de puits en utilisant une discrétisation de taille modeste pour obtenir rapidement de premiers résultats. Une prochaine étape consisterait à mettre à profit l'algorithme adaptatif que nous proposons sur des modèles numériques de résolutions supérieures pour obtenir des résultats plus précis, avec un temps de calcul modéré. L'utilisation de modèles numériques intermédiaires basés sur des maillages grossiers ou des descriptions simplifiées de la physique est à la base des approches multi-échelles en simulation des réservoirs pétroliers [41]. Ces méthodes ont un fort potentiel pour la modélisation numérique en géosciences et sont un sujet de recherche très actif. Dans le même esprit, notre algorithme permettrait d'atteindre des résultats aussi précis que le simulateur le plus fin peut le permettre, tout en faisant appel autant que possible aux simulateurs moins précis mais moins exigeants en ressources processeur.
Au Chapitre 4 nous avons proposé un algorithme de résolution par tri non-dominé itératif pour résoudre une large classe de problèmes d'optimisation floue incluant les programmes linéaires flous et les programmes quadratiques flous. Seule la connaissance d'un nombre finis de coupes des paramètres flous est supposée. Nous avons également introduit un critère pour estimer le niveau de précision de nos solutions en termes de degré d'appartenance. D'après ce critère notre algorithme permet d'obtenir des solutions avec un degré d'optimalité au moins égal au plus haut niveau de coupe disponible pour les problèmes convexes avec une dépendance linéaire en les paramètres imprécis. Nous avons testé notre algorithme sur un problème linéaire et un problème quadratique et comparé ses performances en termes de degré d'optimalité relativement à d'autres méthodes de la littérature. Contrairement à notre algorithme ces méthodes ne garantissent pas nécessairement un degré d'optimalité supérieur ou égal au niveau le plus haut de coupes disponibles. Des tests numériques sur des problèmes convexes plus généraux et des problèmes non convexes, ainsi que l'évaluation du degré d'optimalité garanti par d'autres méthodes existantes, seraient les bienvenus pour poursuivre ces travaux. L'extension à des problèmes dont les contraintes sont également paramétrées par des quantités floues pourrait être envisagée.

Bibliographie

[1] Aarnes, J. E., Gimse, T., and Lie, K.-A. An introduction to the numerics of flow in porous media using matlab. In *Geometric Modelling, Numerical Simulation, and Optimization*. Springer, 2007, pp. 265–306.

[2] Ammar, E., and Khalifa, H. Fuzzy portfolio optimization a quadratic programming approach. *Chaos, Solitons & Fractals 18*, 5 (2003), 1045–1054.

[3] Ankenman, B., Nelson, B. L., and Staum, J. Stochastic kriging for simulation metamodeling. *Operations research 58*, 2 (2010), 371–382.

[4] Asai, K., Tanaka, H., and Okuda, T. Decision making and its goal in a fuzzy environment. *Fuzzy sets and their applications to cognitive and decision processes* (1975), 257–277.

[5] Auder, B. *Classification et modélisation de sorties fonctionnelles de codes de calcul : application aux calculs thermo-hydrauliques accidentels dans les réacteurs à eau pressurisés (REP)*. PhD thesis, Paris 6, 2011.

[6] Bélisle, C. J. P., Romeijn, H. E., and Smith, R. L. Hit-and-run algorithms for generating multivariate distributions. *Mathematics of Operations Research 18*, 2 (1993), 255–266.

[7] Bellman, R. E., and Zadeh, L. A. Decision-making in a fuzzy environment. *Management Science 17*, 4 (1970), B–141.

[8] Beyer, H.-G., and Sendhoff, B. Robust optimization – a comprehensive survey. *Computer methods in applied mechanics and engineering 196*, 33 (2007), 3190–3218.

[9] Bhurjee, A. K., and Panda, G. Efficient solution of interval optimization problem. *Mathematical Methods of Operations Research 76*, 3 (2012), 273–288.

[10] Bongartz, I., Conn, A., Gould, N., and Toint, P. L. Constrained and unconstrained testing environment. *PUBLICATIONS DU DÉPARTEMENT DE MATHÉMATIQUE* (1993).

[11] Bortolan, G., and Degani, R. A review of some methods for ranking fuzzy subsets. *Fuzzy sets and Systems 15*, 1 (1985), 1–19.

[12] Bousquet, N. Accelerated monte carlo estimation of exceedance probabilities under monotonicity constraints. *Annales de la Faculté des Sciences de Toulouse. Série 6 21* (2012), 557–591.

[13] Bouzarkouna, Z. *Well placement optimization*. PhD thesis, Université Paris Sud-Paris XI, 2012.

[14] Bouzarkouna, Z., Ding, D. Y., and Auger, A. Well placement optimization with the covariance matrix adaptation evolution strategy and meta-models. *Computational Geosciences 16*, 1 (2012), 75–92.

[15] Browne, T. Développement d'émulateur de codes stochastiques pour la résolution de problèmes d'optimisation robuste par algorithmes génétiques. Master's thesis, Université Paris Descartes, 2014.

[16] Browne, T., Iooss, B., Gratiet, L. L., and Lonchampt, J. Stochastic simulators based optimization by gaussian process metamodels-application to maintenance investments planning issues. *arXiv preprint arXiv :1509.03880* (2015).

[17] Campos, L., and Verdegay, J. Linear programming problems and ranking of fuzzy numbers. *Fuzzy sets and systems 32*, 1 (1989), 1–11.

[18] Canestrelli, E., Giove, S., and Fullér, R. Stability in possibilistic quadratic programming. *Fuzzy Sets and Systems 82*, 1 (1996), 51–56.

[19] Carlsson, C., and Fuller, R. *Fuzzy reasoning in decision making and optimization*, vol. 82. Physica, 2012.

[20] Chanas, S., and Kuchta, D. Linear programming problem with fuzzy coefficients in the objective function. *Fuzzy Optimization, Physica-Verlag, Heidelberg* (1994), 148–157.

[21] Chanas, S., and Kuchta, D. Multiobjective programming in optimization of interval objective functions – a generalized approach. *European Journal of Operational Research 94*, 3 (1996), 594–598.

[22] Chavent, G., and Jaffré, J. *Mathematical models and finite elements for reservoir simulation*. North Holland, Amsterdam, 1982.

[23] Choquet, G. *Topologie*, vol. 2. Masson, 1969.

[24] Christie, M., Blunt, M., et al. Tenth spe comparative solution project : A comparison of upscaling techniques. In *SPE Reservoir Simulation Symposium* (2001), Society of Petroleum Engineers.

[25] Conn, A. R., Scheinberg, K., and Vicente, L. N. *Introduction to derivative-free optimization*, vol. 8. Siam, 2009.

[26] Conn, A. R., and Vicente, L. N. Bilevel derivative-free optimization and its application to robust optimization. *Optimization Methods and Software 27*, 3 (2012), 561–577.

[27] Da Veiga, S., and Marrel, A. Gaussian process modeling with inequality constraints. In *Annales de la Faculté des Sciences de Toulouse* (2012), vol. 21, pp. 529–555.

[28] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. A fast and elitist multiobjective genetic algorithm : Nsga-ii. *Evolutionary Computation, IEEE Transactions on 6*, 2 (2002), 182–197.

[29] Delgado, M., Verdegay, J., and Vila, M. Relating different approaches to solve linear programming problems with imprecise costs. *Fuzzy Sets and Systems 37*, 1 (1990), 33–42.

[30] Dempe, S., and Ruziyeva, A. On the calculation of a membership function for the solution of a fuzzy linear optimization problem. *Fuzzy Sets and Systems 188*, 1 (2012), 58–67.

[31] Dubois, D., and Prade, H. *Fuzzy Sets and Systems : Theory and Applications*, vol. 144. Academic Pr, 1980.

[32] Ehrgott, M. *Multicriteria optimization*. Springer Science & Business Media, 2006.

[33] Esogbue, A. Computational and data acquisition issues in fuzzy dynamic programming. In *Proceedings of ORSA/TIMS Conference, Anaheim, CA* (1991).

[34] Esogbue, A. O., and Kacprzyk, J. Fuzzy dynamic programming. In *Fuzzy sets in decision analysis, operations research and statistics*. Springer, 1998, pp. 281–307.

[35] Fang, K., Li, R., and Sudjianto, A. Design and modeling for com-puter experiments. *London : Chapman& Hall* (2005).

[36] Fessart, K., and Lonchampt, J. Optimisation d'un programme d'investissements : méthode IPOP. *Conférence lambda-mu, La Rochelle, France* (2010).

[37] Floris, F., Bush, M., Cuypers, M., Roggero, F., and Syversveen, A. R. Methods for quantifying the uncertainty of production forecasts : a comparative study. *Petroleum Geoscience 7*, S (2001), S87–S96.

[38] Fongang Fongang, D. Optimization under uncertainties. Master's thesis, École Nationale Supérieure des Mines de Saint-Étienne, 2011.

[39] Gabrel, V., Murat, C., and Thiele, A. Recent advances in robust optimization and robustness : An overview. Tech. rep., LAMSADE, Université Paris-Dauphine, 2012.

[40] Garcia-Brotons, R. Optimisation globale : application au problème de placement de puits en ingénierie de réservoir. Master's thesis, École Supérieure d'Ingénieurs Léonard de Vinci, 2012.

[41] Gardet, C., Le Ravalec, M., and Gloaguen, E. Multiscale parameterization of petrophysical properties for efficient history-matching. *Mathematical Geosciences 46*, 3 (2014), 315–336.

[42] Gratton, S., Mouffe, M., Sartenaer, A., Toint, P. L., and Tomanos, D. Numerical experience with a recursive trust-region method for multilevel nonlinear bound-constrained optimization. *Optimization Methods & Software 25*, 3 (2010), 359–386.

[43] Gratton, S., Royer, C., Vicente, L., and Zhang, Z. Direct search based on probabilistic descent. *preprint November* (2014).

[44] Grzegorzewski, P. Nearest interval approximation of a fuzzy number. *Fuzzy Sets and systems 130*, 3 (2002), 321–330.

[45] Hansen, N., and Ostermeier, A. Convergence properties of evolution strategies with the derandomized covariance matrix adaptation : The $(\mu/\mu_i, \lambda) - cma - es$. *Eufit 97* (1997), 650–654.

[46] Hanss, M. *Applied fuzzy arithmetic*. Springer, 2005.

[47] Hoeffding, W. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association 58*, 301 (1963), 13–30.

[48] Huyer, W., and Neumaier, A. Global optimization by multilevel coordinate search. *Journal of Global Optimization 14*, 4 (1999), 331–355.

[49] Igel, C., Hansen, N., and Roth, S. Covariance matrix adaptation for multi-objective optimization. *Evolutionary computation 15*, 1 (2007), 1–28.

[50] Inuiguchi, M. Robust optimization by fuzzy linear programming. In *Managing Safety of Heterogeneous Systems*. Springer, 2012, pp. 219–239.

[51] Inuiguchi, M., Ichihashi, H., and Tanaka, H. Fuzzy programming : a survey of recent developments. In *Stochastic versus fuzzy approaches to multiobjective mathematical programming under uncertainty*. Springer, 1990, pp. 45–68.

[52] Inuiguchi, M., and Ramik, J. Possibilistic linear programming : a brief review of fuzzy mathematical programming and a comparison with stochastic programming in portfolio selection problem. *Fuzzy sets and systems 111*, 1 (2000), 3–28.

[53] Inuiguchi, M., and Sakawa, M. Robust optimization under softness in a fuzzy linear programming problem. *International Journal of Approximate Reasoning 18*, 1 (1998), 21–34.

[54] Ishibuchi, H., and Tanaka, H. Multiobjective programming in optimization of the interval objective function. *European Journal of Operational Research 48*, 2 (1990), 219–225.

[55] Janusevskis, J., and Le Riche, R. Simultaneous kriging-based estimation and optimization of mean response. *Journal of Global Optimization 55*, 2 (2013), 313–336.

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

[57] Kennedy, J., and Eberhart, R. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on* (Nov 1995), vol. 4, pp. 1942–1948 vol.4.

[58] Kirkpatrick, S. Optimization by simulated annealing : Quantitative studies. *Journal of statistical physics 34*, 5-6 (1984), 975–986.

[59] Kleijnen, J. P. *Design and analysis of simulation experiments*, vol. 111. Springer Science & Business Media, 2007.

[60] Kneip, A., and Utikal, K. J. Inference for density families using functional principal component analysis. *Journal of the American Statistical Association 96*, 454 (2001), 519–542.

[61] Kolda, T. G., Lewis, R. M., and Torczon, V. Optimization by direct search : New perspectives on some classical and modern methods. *SIAM review 45*, 3 (2003), 385–482.

[62] Lie, K.-A. An introduction to reservoir simulation using matlab : User guid for the matlab reservoir simulation toolbox (mrst). *SINTEF ICT* (May 2014).

[63] Lie, K.-A., Krogstad, S., Ligaarden, I. S., Natvig, J. R., Nilsen, H. M., and Skaflestad, B. Open-source matlab implementation of consistent discretisations on complex grids. *Computational Geosciences 16*, 2 (2012), 297–322.

[64] Liu, S.-T. Quadratic programming with fuzzy parameters : A membership function approach. *Chaos, Solitons & Fractals 40*, 1 (2009), 237–245.

[65] Liu, S.-T. A revisit to quadratic programming with fuzzy parameters. *Chaos, Solitons & Fractals 41*, 3 (2009), 1401–1407.

[66] Liu, S.-T. A fuzzy modeling for fuzzy portfolio optimization. *Expert Systems with Applications 38*, 11 (2011), 13803–13809.

[67] Lodwick, W. A. *Fuzzy Optimization : Recent Advances and Applications*, vol. 254. Springer, 2010.

[68] Luhandjula, M. Fuzzy stochastic linear programming : survey and future research directions. *European Journal of Operational Research 174*, 3 (2006), 1353–1367.

[69] Luhandjula, M. K. Fuzzy mathematical programming : Theory, applications and extension. *Journal of Uncertain Systems 1*, 2 (2007), 124–136.

[70] Luhandjula, M. K. Fuzzy optimization : Milestones and perspectives. *Fuzzy Sets and Systems 274* (2015), 4 – 11. Fuzzy Modeling for Optimisation and Decision Support.

[71] Luhandjula, M. K., and Rangoaga, M. An approach for solving a fuzzy multiobjective programming problem. *European Journal of Operational Research 232*, 2 (2014), 249–255.

[72] Maday, Y., Nguyen, N. C., Patera, A. T., and Pau, S. H. A general multipurpose interpolation procedure : the magic points. *Communications on Pure and Applied Analysis 8*, 1 (2009), 383–404.

[73] Marrel, A., Iooss, B., Da Veiga, S., and Ribatet, M. Global sensitivity analysis of stochastic computer models with joint metamodels. *Statistics and Computing 22*, 3 (2012), 833–847.

[74] Martins, J. R., Sturdza, P., and Alonso, J. J. The complex-step derivative approximation. *ACM Transactions on Mathematical Software (TOMS) 29*, 3 (2003), 245–262.

[75] McKay, M. D., Beckman, R. J., and Conover, W. J. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics 21*, 2 (1979), 239–245.

[76] Moore, R. E. *Interval analysis*, vol. 4. Prentice-Hall Englewood Cliffs, 1966.

[77] Moutoussamy, V., Nanty, S., and Pauwels, B. Emulators for stochastic simulation codes. *ESAIM : Proceedings and Surveys 48* (2015), 116–155.

[78] Nadaraya, E. A. On estimating regression. *Theory of Probability & Its Applications 9*, 1 (1964), 141–142.

[79] Nasseri, S. Fuzzy nonlinear optimization. *The Journal of Nonlinear Sciences and its Applications 1*, 4 (2008), 230–235.

[80] Nelder, J. A., and Mead, R. A simplex method for function minimization. *The computer journal 7*, 4 (1965), 308–313.

[81] Orlovski, S. Mathematical programming problems with fuzzy parameters. *Management decision support systems using fuzzy sets and possibility theory* (1985), 136–145.

[82] Parzen, E. On estimation of a probability density function and mode. *The annals of mathematical statistics* (1962), 1065–1076.

[83] Powell, M. J. Restart procedures for the conjugate gradient method. *Mathematical programming 12*, 1 (1977), 241–254.

[84] Ramík, J. Optimal solutions in optimization problem with objective function depending on fuzzy parameters. *Fuzzy sets and systems 158*, 17 (2007), 1873–1881.

[85] Ramík, J., et al. Inequality relation between fuzzy numbers and its use in fuzzy optimization. *Fuzzy sets and systems 16*, 2 (1985), 123–138.

[86] Ramsay, J. O. *Functional data analysis*. Wiley Online Library, 2006.

[87] Reich, B. J., Kalendra, E., Storlie, C. B., Bondell, H. D., and Fuentes, M. Variable selection for high dimensional bayesian density estimation : application to human exposure simulation. *Journal of the Royal Statistical Society : Series C (Applied Statistics) 61*, 1 (2012), 47–66.

[88] Rigby, R., and Stasinopoulos, D. Construction of reference centiles using mean and dispersion additive models. *Journal of the Royal Statistical Society : Series D (The Statistician) 49*, 1 (2000), 41–50.

[89] Rios, L. M., and Sahinidis, N. V. Derivative-free optimization : A review of algorithms and comparison of software implementations. *Journal of Global Optimization* (2012), 1–47.

[90] Rommelfanger, H. Inequality relations in fuzzy constraints and its use in linear fuzzy optimization. *The Interface between Artificial Intelligence and Operations Research in Fuzzy Environment* (1989), 195–211.

[91] Rommelfanger, H., Hanuscheck, R., and Wolf, J. Linear programming with fuzzy objectives. *Fuzzy sets and systems 29*, 1 (1989), 31–48.

[92] Rommelfanger, H. J. Fuzzy linear programming and applications. *European Journal of Operational Research 92*, 3 (1996), 512–527.

[93] Rosenblatt, M., et al. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics 27*, 3 (1956), 832–837.

[94] Ruziyeva, A. *Fuzzy Bilevel Optimization*. PhD thesis, Technische Universität Bergakademie Freiberg, 2013.

[95] Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. Design and analysis of computer experiments. *Statistical science 4*, 4 (1989), 409–423.

[96] Sakawa, M. Fuzzy multiobjective and multilevel optimization. In *Multiple criteria optimization : State of the art annotated bibliographic surveys*. Springer, 2002, pp. 171–226.

[97] Sakawa, M. *Fuzzy sets and interactive multiobjective optimization*. Springer Science & Business Media, 2013.

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

[99] Shubert, B. O. A sequential method seeking the global maximum of a function. *SIAM Journal on Numerical Analysis 9*, 3 (1972), 379–388.

[100] Silva, R. C., Cruz, C., Verdegay, J. L., and Yamakami, A. A survey of fuzzy convex programming models. In *Fuzzy Optimization*. Springer, 2010, pp. 127–143.

[101] Soyster, A. L. Technical note - convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations research 21*, 5 (1973), 1154–1157.

[102] Tanaka, H., and Asai, K. Fuzzy linear programming problems with fuzzy numbers. *Fuzzy sets and systems 13*, 1 (1984), 1–10.

[103] Tanaka, H., Guo, P., and Türksen, I. B. Portfolio selection based on fuzzy probabilities and possibility distributions. *Fuzzy sets and systems 111*, 3 (2000), 387–397.

[104] Tanaka, H., Ichihashi, H., and Asai, K. A value of information in flp problems via sensitivity analysis. *Fuzzy Sets and Systems 18*, 2 (1986), 119–129.

[105] Tanaka, H., Okuda, T., and Asai, K. On fuzzy-mathematical programming. *Journal of Cybernetics 3*, 4 (1973), 37–46.

[106] Tanaka, H., Okuda, T., and Asai, K. On decision-making in fuzzy environment fuzzy information and decision making. *International Journal of Production Research 15*, 6 (1977), 623–635.

[107] Tang, J., and Wang, D. An interactive approach based on a genetic algorithm for a type of quadratic programming problems with fuzzy objective and resources. *Computers & Operations Research 24*, 5 (1997), 413–422.

[108] Tang, J., and Wang, D. A nonsymmetric model for fuzzy nonlinear programming problems with penalty coefficients. *Computers & operations research 24*, 8 (1997), 717–725.

[109] Torczon, V. On the convergence of pattern search algorithms. *SIAM Journal on optimization 7*, 1 (1997), 1–25.

[110] Vicente, L. N. Worst case complexity of direct search. *EURO Journal on Computational Optimization 1*, 1-2 (2013), 143–153.

[111] Wang, D. An inexact approach for linear programming problems with fuzzy objective and resources. *Fuzzy Sets and Systems 89*, 1 (1997), 61–68.

[112] Wang, S., and Watada, J. *Fuzzy stochastic optimization : theory, models and applications*. Springer Science & Business Media, 2012.

[113] Watson, G. S. Smooth regression analysis. *Sankhyā : The Indian Journal of Statistics, Series A* (1964), 359–372.

[114] Wu, H.-C. The karush-kuhn-tucker optimality conditions for multi-objective programming problems with fuzzy-valued objective functions. *Fuzzy Optimization and Decision Making 8*, 1 (2009), 1–28.

[115] Zabalza Mezghani, I. *Analyse statistique et planification d'expérience en ingénierie de réservoir*. PhD thesis, Université de Pau et des Pays de l'Adour, 2000.

[116] Zadeh, L. Pruf and its application to inference from fuzzy propositions. In *1977 IEEE Conference on Decision and Control including the 16th Symposium on Adaptive Processes and A Special Symposium on Fuzzy Set Theory and Applications* (1977), pp. 1359–1360.

[117] Zadeh, L. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems 1*, 1 (1978), 3 – 28.

[118] Zadeh, L. A. Fuzzy sets. *Information and control 8*, 3 (1965), 338–353.

[119] Zadeh, L. A. The concept of a linguistic variable and its application to approximate reasoning i. *Information sciences 8*, 3 (1975), 199–249.

[120] Zadeh, L. A. The concept of a linguistic variable and its application to approximate reasoning ii. *Information sciences 8*, 4 (1975), 301–357.

[121] Zadeh, L. A. The concept of a linguistic variable and its application to approximate reasoning-iii. *Information sciences 9*, 1 (1975), 43–80.

[122] Zimmermann, H.-J. Description and optimization of fuzzy systems. *International Journal of General System 2*, 1 (1975), 209–215.

Index

Résumé La modélisation de phénomènes complexes rencontrés dans les problématiques industrielles peut conduire à l'étude de codes de simulation numérique. Ces simulateurs peuvent être très coûteux en temps d'exécution (de quelques heures à plusieurs jours), mettre en jeu des paramètres incertains et même être intrinsèquement stochastiques. Fait d'importance en optimisation basée sur de tels simulateurs, les dérivées des sorties en fonction des entrées peuvent être inexistantes, inaccessibles ou trop coûteuses à approximer correctement. Ce mémoire est organisé en quatre chapitres. Le premier chapitre traite de l'état de l'art en optimisation sans dérivées et en modélisation d'incertitudes. Les trois chapitres suivants présentent trois contributions indépendantes — bien que liées — au champ de l'optimisation sans dérivées en présence d'incertitudes. Le deuxième chapitre est consacré à l'émulation de codes de simulation stochastiques coûteux — stochastiques au sens où l'exécution de simulations avec les mêmes paramètres en entrée peut donner lieu à des sorties distinctes. Tel était le sujet du projet CODESTOCH mené au Centre d'été de mathématiques et de recherche avancée en calcul scientifique (CEMRACS) au cours de l'été 2013 avec deux doctorants de Électricité de France (EDF) et du Commissariat à l'énergie atomique et aux énergies alternatives (CEA). Nous avons conçu quatre méthodes de construction d'émulateurs pour des fonctions dont les valeurs sont des densités de probabilité. Ces méthodes ont été testées sur deux exemples-jouets et appliquées à des codes de simulation industriels concernés par trois phénomènes complexes : la distribution spatiale de molécules dans un système d'hydrocarbures (IFPEN), le cycle de vie de grands transformateurs électriques (EDF) et les répercussions d'un hypothétique accident dans une centrale nucléaire (CEA). Dans les deux premiers cas l'émulation est une étape préalable à la résolution d'un problème d'optimisation. Le troisième chapitre traite de l'influence de l'inexactitude des évaluations de la fonction objectif sur la recherche directe directionnelle — un algorithme classique d'optimisation sans dérivées. Dans les problèmes réels, l'imprécision est sans doute toujours présente. Pourtant les utilisateurs appliquent généralement les algorithmes de recherche directe sans prendre cette imprécision en compte. Nous posons trois questions. Quelle précision peut-on espérer obtenir, étant donnée l'inexactitude ? À quel prix cette précision peut-elle être atteinte ? Quels critères d'arrêt permettent de garantir cette précision ? Nous répondons à ces trois questions pour l'algorithme de recherche directe directionnelle appliqué à des fonctions dont l'imprécision sur les valeurs — stochastique ou non — est uniformément bornée. Nous déduisons de nos résultats un algorithme adaptatif pour utiliser efficacement des oracles de niveaux de précision distincts. Les résultats théoriques et l'algorithme sont validés avec des tests numériques et deux applications réelles : la minimisation de surface en conception mécanique et le placement de puits pétroliers en ingénierie de réservoir. Le quatrième chapitre est dédié aux problèmes d'optimisation affectés par des paramètres imprécis, dont l'imprécision est modélisée grâce à la théorie des ensembles flous. Plusieurs méthodes ont déjà été publiées pour résoudre les programmes linéaires où apparaissent des coefficients flous, mais très peu pour traiter les problèmes non linéaires. Nous proposons un algorithme pour répondre à une large classe de problèmes par tri non-dominé itératif. Les distributions des paramètres flous sont seulement supposées partiellement connues. Nous définissons également un critère pour évaluer la précision des solutions obtenues et tirons des comparaisons avec d'autres algorithmes de la littérature. Nous démontrons également que notre algorithme garantit des solutions dont le niveau de précision est au moins égal à la précision des données disponibles.

Abstract The modeling of complex phenomena encountered in industrial issues can lead to the study of numerical simulation codes. These simulators may require extensive execution time (from hours to days), involve uncertain parameters and even be intrinsically stochastic. Importantly within the context of simulation-based optimization, the derivatives of the outputs with respect to the inputs may be inexistent, inaccessible or too costly to approximate reasonably. This thesis is organized in four chapters. The first chapter discusses the state of the art in derivative-free optimization and uncertainty modeling. The next three chapters introduce three independent—although connected—contributions to the field of derivative-free optimization in the presence of uncertainty. The second chapter addresses the emulation of costly stochastic simulation codes—stochastic in the sense simulations run with the same input parameters may lead to distinct outputs. Such was the matter of the CODESTOCH project carried out at the Summer mathematical research center on scientific computing and its applications (CEMRACS) during the summer of 2013, together with two Ph.D. students from Electricity of France (EDF) and the Atomic Energy and Alternative Energies Commission (CEA). We designed four methods to build emulators for functions whose values are probability density functions. These methods were tested on two toy functions and applied to industrial simulation codes concerned with three complex phenomena : the spatial distribution of molecules in a hydrocarbon system (IFPEN), the life cycle of large electric transformers (EDF) and the repercussions of a hypothetical accidental in a nuclear plant (CEA). Emulation was a preliminary process towards optimization in the first two cases. In the third chapter we consider the influence of inaccurate objective function evaluations on direct search—a classical derivative-free optimization method. In real settings inaccuracy may never vanish, however users usually apply direct search algorithms disregarding inaccuracy. We raise three questions. What precision can we hope to achieve, given the inaccuracy ? How fast can this precision be attained ? What stopping criteria can guarantee this precision ? We answer these three questions for directional direct search applied to objective functions whose evaluation inaccuracy stochastic or not is uniformly bounded. We also derive from our results an adaptive algorithm for dealing efficiently with several oracles having different levels of accuracy. The theory and algorithm are validated with numerical tests and two industrial applications : surface minimization in mechanical design and oil well placement in reservoir engineering. The fourth chapter considers optimization problems with imprecise parameters, whose imprecision is modeled with fuzzy sets theory. A number of methods have been published to solve linear programs involving fuzzy parameters, but only a few as for nonlinear programs. We propose an algorithm to address a large class of fuzzy optimization problems by iterative non-dominated sorting. The distributions of the fuzzy parameters are assumed only partially known. We also provide a criterion to assess the precision of the solutions and make comparisons with other methods found in the literature. We show that our algorithm guarantees solutions whose level of precision at least equals the precision on the available data.