POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Analyse de sensibilité pour la réduction de dimension en optimisation sans dérivée

ROMAIN VANDEN BULCKE
Département de mathématiques et génie industriel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Mathématiques appliquées

Juin 2020

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Analyse de sensibilité pour la réduction de dimension en optimisation sans dérivée

présenté par **Romain VANDEN BULCKE**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Dominique ORBAN, président
Charles AUDET, membre et directeur de recherche
Sébastien LE DIGABEL, membre et codirecteur de recherche
Stéphane ALARIE, membre

DÉDICACE

A mes amis et à ma famille

REMERCIEMENTS

Je profite de ce moment pour remercier tous ceux qui m'ont aidé lors de la réalisation de ce travail, à commencer par mes directeurs de recherche Charles Audet et Sébastien Le Digabel ainsi que Miguel Diago Martinez pour leur disponibilité, leurs conseils ainsi que pour leur contributions qui ont été très appréciés. Tout cela a permis de construire le travail présenté dans ce document. Je remercie également les membres du GERAD que j'ai pu côtoyés au cours de ma maîtrise.

J'exprime également ma gratitude aux membres du jury qui ont accepté d'évaluer ce travail.

Pour finir, je tiens à remercier mes proches qui m'ont soutenu durant toute ma scolarité ainsi que Shana qui était présente à mes côtés cette dernière année.

RÉSUMÉ

A l'heure actuelle, le monde industriel regorge de processus et de calculs complexes et l'optimisation de ceux-ci se retrouve au cœur de la recherche et du développement d'entreprises. Ces problèmes ont souvent des caractéristiques qui nécessitent de faire appel à des méthodes d'optimisation sans dérivée.

Il s'agit d'algorithmes d'optimisation qui permettent de gérer des fonctions non linéaires, non différentiables, bruitées ou encore non définies en certains points du domaine. La classe d'algorithme Mads rassemble des méthodes qui permettent de résoudre des problèmes contraints sous forme de boîtes noires correspondant aux résultats d'un code informatique. Par ailleurs, l'exploration d'un espace de recherche dont aucune information n'est disponible nécessite un grand nombre d'évaluations. Néanmoins, l'évaluation d'une boîte noire est souvent coûteuse ; ceci constitue la principale difficulté du domaine, la recherche d'un minimum d'une boîte noire en un nombre limité d'évaluations.

Cette limite du budget d'évaluations et d'autant plus importante lorsque le problème d'intérêt est de grande dimension. Il s'agit de la principale motivation pour appliquer une méthode de réduction de dimension au cours de l'optimisation du problème. L'algorithme Stats-Mads applique tout d'abord une méthode d'analyse de sensibilité basée sur une analyse de variance pour identifier les variables ayant le plus d'influence sur l'objectif. Ensuite, l'algorithme alterne entre une optimisation en petite dimension, où les variables les moins influentes sont fixées, et une optimisation en grande dimension. Les phases d'optimisation en petite dimension ont un rôle prépondérant dans la diminution de la valeur de l'objectif, et donc dans l'optimisation du problème.

Nous proposons un nouvel algorithme de la classe Mads qui permet de s'attaquer à des problèmes de grande dimension. Celui-ci applique une analyse de sensibilité basée sur une analyse en composante principale qui permet d'extraire des combinaisons de variables ayant le plus d'impact sur la fonction objectif. Cet algorithme a donc été nommé Pca-Mads. D'une manière similaire à Stats-Mads, l'algorithme Pca-Mads alterne entre une optimisation en petite et en grande dimension. Toutefois, la structure de l'algorithme permet de poursuivre l'optimisation en petite dimension tant que celle-ci fournit des solutions améliorant la valeur de la fonction objectif.

L'algorithme Pca-Mads, principalement basé sur l'instance LTMads, a été implémenté en MATLAB™. A la lumière des résultats obtenus sur des problèmes allant jusqu'à 1500 variables, l'algorithme Pca-Mads est comparé à d'autres algorithmes d'optimisation sans

dérivée dont CMA-ES, MADS et principalement STATS-MADS afin de pouvoir conclure de ses performances. Ces tests indiquent clairement l'intérêt de l'approche de PCA-MADS.

ABSTRACT

In today's industry, the look for highest productivity at smallest costs naturally creates optimization problems. Precise models often create complex problems along with the need for derivative free optimization methods.

Those are methods which can handle non-linear, non-differentiable or noisy objective functions. Mads algorithms are well-known black box optimization methods which solve this type of problem through calls to a black box, *i.e.* some kind of computer code. When little is known about the problem, the exploration of the search space requires a large number of black box evaluations. However, in the context of black box optimization, problems take the form of expensive-to-evaluate functions. The total number of evaluations is therefore very limited and this constitutes the main challenge of the field.

When considering black box problems in large dimensions, the limited budget of evaluations is even more constraining. Standard black box algorithms need to be adapted, for example through dimension reduction scheme. Stats-Mads is a Mads-based algorithm which applies an analysis of variance to rank most influential input variables. Then the method alternates between optimizing the problem in a smaller dimension, where least influential variables were fixed, and the problem in its original large dimension. Most of the improvement of the objective value was done during the optimization in the small dimension.

We propose a new Mads algorithm conceived to handle large-scale black box problems. This method applies a principal component analysis to identify most influential directions in the search space and is called Pca-Mads. Similarly to Stats-Mads, Pca-Mads alternates between an optimization in a smaller dimension, where the input can only evolve in the few most influential directions, and a poll in the large dimension. However, its structure allows to skip the poll in the large dimension as long as the optimization in the smaller dimension generates new improving solutions.

A MATLAB™ implementation of the Pca-Mads method, based on the LTMads instance was run on problems of up to 1500 variables. Its performances are compared to other derivative free methods such as CMA-ES, Mads and mainly Stats-Mads. The results of these tests clearly indicate the value of the approach developed for Pca-Mads.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

FAST	*Fourier amplitude sensitivity test*
GPS	Recherche par motif généralisée (*Generalized pattern search*)
LTMADS	Lower Triangular MADS
MADS	Recherche directe par treillis adaptatif (*Mesh adaptive direct search*)
NOMAD	Non Linear Optimization with the MADS algorithm
ORTHOMADS	MADS avec directions orthogonales
PCA-MADS	MADS avec analyse en composante principale
PSD-MADS	*Parallel space decomposition of the mesh direct search algorithm*
STATS-MADS	MADS avec analyse de sensibilité statistique
SRC	*Standardized Regression Coefficient*
SRRC	*Standardized Rank Regression Coefficient*

Ω	Ensemble réalisable
f	Fonction objectif
$\mathcal{X}$	Domaine de la fonction objectif
$c_i, i = 1, \ldots, m$	Fonctions des contraintes
$\mathbb{R}$	Ensemble des nombres réels
$\bar{\mathbb{R}}$	$\mathbb{R} \cup \{-\infty, +\infty\}$
k	Compteur d'itérations
x^k	Solution courante à l'itération k
$\mathbb{D}^k$	Ensemble générateur positif à l'itération k
M^k	Treillis conceptuel à l'itération k
P^k	Cadre de sonde à l'itération k
δ^k	Taille du treillis à l'itération k
Δ^k	Taille du cadre de sonde à l'itération k

CHAPITRE 1 Introduction

L'optimisation est le domaine des mathématiques appliquées où l'on cherche à minimiser ou à maximiser une certaine quantité, éventuellement sous la contrainte que les variables d'entrée du système appartiennent à un ensemble prédéfini. De nos jours, ce genre de problèmes est extrêmement présent dans notre société ; on peut aisément imaginer une entreprise qui cherche à minimiser ses coûts de production ou maximiser son rendement. Mais l'optimisation est également utilisée pour la gestion de réseaux ou de ressources, le calcul d'horaires ou encore l'apprentissage automatisé.

Un problème d'optimisation général s'écrit

$$\begin{aligned} \min_{x \in \mathcal{X}} \quad & f(x) \\ \text{sujet à} \quad & x \in \Omega. \end{aligned}$$

La fonction f est appelée fonction objectif, ou plus simplement objectif, tandis que x représente les variables du problème. L'ensemble $\mathcal{X} \subseteq \mathbb{R}^n$ est l'ensemble de définition de la fonction objectif $f : \mathcal{X} \mapsto \mathbb{R}$, et n est la dimension du problème. L'ensemble $\Omega \subseteq \mathcal{X}$ est l'ensemble réalisable du problème et est généralement défini par une série de fonctions appelées contraintes, $\Omega = \{c_i(x) \leq 0, i = 1, 2, \ldots, m\}$. Il est également possible de définir un problème d'optimisation avec d'autres types de variables, par exemple entières ou binaires. Bien que la recherche d'un maximum fasse également partie du domaine de l'optimisation, la convention préfère écrire les problèmes sous forme de minimisation. Toutefois, il existe une certaine équivalence entre la recherche d'un minimum et d'un maximum. Pour cela, l'équivalence suivante est utilisée.

$$\operatorname*{argmin}_{x \in \Omega} f(x) \quad \Longleftrightarrow \quad \operatorname*{argmax}_{x \in \Omega} -f(x)$$

La difficulté d'un problème d'optimisation peut venir de la nature de l'objectif (fonction linéaire, quadratique, convexe, lisse ou non lisse...) ou des contraintes (linéaires, quadratiques, entières...) ou encore de la dimension du problème. Les problèmes les plus simples peuvent être résolus à la main ou graphiquement mais les problèmes issus du monde réel sont souvent bien trop compliqués pour cela. Il faut donc utiliser des algorithmes d'optimisation prévus pour le type de problèmes auquel on a affaire.

1.1 Concepts de base et contexte du projet

Dans le cadre de ce projet, nous nous concentrons plus sur l'optimisation de boîtes noires. Il s'agit de problèmes dont l'objectif et les contraintes ne sont pas connus analytiquement. On ne peut évaluer l'objectif et les contraintes qu'en fixant une valeur pour chaque variable. Ce fonctionnement est représenté graphiquement à la figure 1.1. En pratique, une boîte noire prend souvent la forme d'une simulation informatique qui ne peut pas être écrite analytiquement.

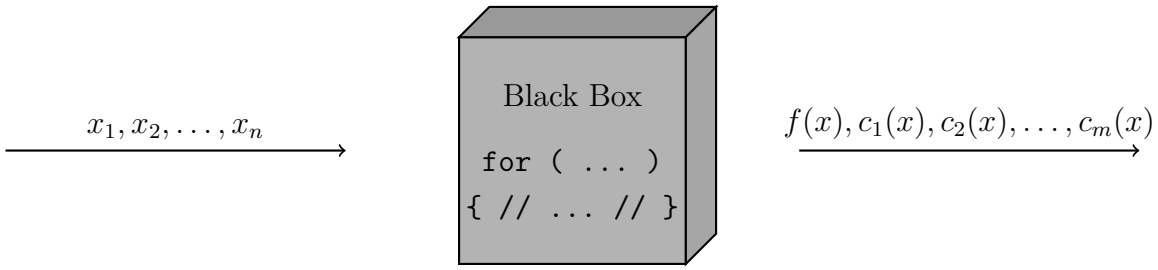


Figure 1.1 Schéma simplifié du fonctionnement d'une boîte noire

Parmi les principales caractéristiques des boîtes noires, on retrouve un long temps d'exécution, des évaluations qui peuvent mener à des erreurs, des boîtes noires très peu lisses ou encore bruitées. Le temps d'évaluation de la boîte noire est souvent le facteur critique et définit un budget d'évaluation maximum. Les dérivées de la fonction objectif ne sont pas accessibles et/ou n'existent pas. Les méthodes basées sur le gradient ou sur la matrice Hessienne ne sont donc pas utilisables dans ce contexte. Ces deux outils sont très appréciés en optimisation, car ils donnent des informations quant à la pente ou la courbure de l'objectif. De plus, vu que la boîte noire est souvent coûteuse à évaluer, il devient difficile d'estimer les dérivées en un temps raisonnable. Il faut donc se tourner vers des méthodes propres à l'optimisation sans dérivée. Il en existe plusieurs qui seront décrites au chapitre 3. La plupart ont de bonnes performances lorsque les boîtes noires sont d'une taille raisonnable, de l'ordre de quelques dizaines de variables.

1.2 Problématique

Dans le cadre de ce projet, on considère des problèmes de minimisation d'une boîte noire sous contraintes de bornes sur les variables :

$$\min_{x \in \mathcal{X} \subseteq \mathbb{R}^n} \quad f(x)$$

où $\mathcal{X} = \{x \in \mathbb{R}^n : l \leq x \leq u\}$ où $l, u \in \bar{\mathbb{R}}^n$ définit des bornes explicitement connues. Les problèmes considérés dans ce document possèdent uniquement des variables réelles. Plus particulièrement, les problèmes étudiés sont des problèmes d'optimisation sans dérivée de grande taille, de l'ordre de quelques centaines à quelques milliers de variables. Ces problèmes sont trop grands pour utiliser des méthodes d'optimisation de boîtes noires classiques en un temps raisonnable.

La principale difficulté des problèmes d'optimisation de boîtes noires est que l'on ne possède que peu d'information sur le problème. L'idée de départ de ce projet est donc d'en apprendre un peu plus sur le fonctionnement de la boîtes noires en appliquant des méthodes d'analyse de sensibilité. Ces méthodes devraient permettre d'identifier des variables ou des combinaisons de variables qui ont plus d'influence que les autres. On suppose en effet que certaines combinaisons de variables sont plus critiques. Cela permettra de concentrer la recherche d'un minimum dans un sous-espace où seules les valeurs de ces combinaisons de variables pourront être modifiées, au lieu de l'espace complet. Ce projet propose donc un nouvel algorithme basé sur un algorithme d'optimisation connu qui applique une méthode d'analyse de sensibilité pour réduire l'espace de recherche.

1.3 Structure du document

Ce document est structuré de la façon suivante. Le chapitre 2 présente plusieurs méthodes d'analyse de sensibilité. Celles-ci ont été classées en différentes catégories en fonction de leur fonctionnement et de leur portée. Le chapitre 3 présente plusieurs méthodes et algorithmes propres à l'optimisation sans dérivée. La dernière section de ce chapitre porte sur les méthodes d'optimisation de boîtes noires qui ciblent plus particulièrement les problèmes de grandes tailles. Dans le chapitre 4, on propose un algorithme d'optimisation sans dérivée utilisant une méthode d'analyse de sensibilité, ainsi qu'une courte analyse de sa convergence, c'est-à-dire une analyse de son comportement lorsque son nombre d'itérations tend vers l'infini. Le chapitre 5 donne quelques résultats numériques concrets ainsi que des comparaisons avec d'autres algorithmes connus. Le chapitre 6 présente une discussion des résultats et une conclusion de ce projet.

CHAPITRE 2 Méthodes d'analyse de sensibilité

Une analyse de sensibilité cherche à identifier l'influence des variables d'entrée sur la sortie d'un modèle. Les variables sont aussi appelées facteurs du modèle. Ce projet considère un modèle à n variables et une sortie :

$$y = f(x_1, x_2, \ldots, x_n).$$

Etant donné que ce projet porte sur de l'optimisation de boîte noire, le modèle étudié est la fonction objectif du problème d'optimisation.

En premier lieu, la section 2.1 présente des méthodes locales. Comme leur nom l'indique, il s'agit des méthodes capables d'identifier la sensibilité du modèle autour d'un point. La section 2.2 porte sur des méthodes de criblage ; il s'agit d'une généralisation plus globale de méthodes locales. Contrairement aux méthodes locales, les méthodes globales ont pour but de donner des mesures de sensibilité qui seraient valides dans tout le domaine des variables d'entrée et indépendantes d'un choix arbitraire de points nominaux. Les sections suivantes portent sur des méthodes globales. Parmi celles-ci, les méthodes basées sur la variance, à la section 2.3, supposent que le modèle est aléatoire et appliquent une analyse de variance. Les sections 2.4 et 2.5 cherchent à construire des modèles de la fonction à analyser pour en déduire sa sensibilité aux variables. La dernière section présente une méthode d'analyse en composante principale.

2.1 Méthodes locales

Dans le cas d'une fonction différentiable, la première mesure de sensibilité qui vient à l'esprit est sans doute une mesure basée sur les dérivées partielles. Un coefficient S_i est défini pour chaque facteur d'entrée x_i, $i = 1, \ldots, n$,

$$S_i = \frac{\partial f}{\partial x_i}(x^0) \tag{2.1}$$

où $x^0 = (x_1^0, \ldots, x_n^0)$ est un point de dimension n, appelé point nominal. Toutefois, cette mesure semble ne pas être toujours appropriée. Considérons l'exemple d'une facture pour n objets. On note $x_i > 0$ le prix du i^e objet, $i = 1, \ldots, n$, et $y = f(x) = \sum_{i=1}^n x_i$. La mesure de sensibilité décrite dans (2.1) donnerait $S_i = 1$ pour chaque objet, indépendamment du prix de chacun de ces objets. Il serait alors intéressant de normaliser la mesure (2.1) en fonction

de la valeur du facteur et la valeur de la sortie :

$$S_i = \frac{x_i^0}{f(x^0)} \frac{\partial y}{\partial x_i}(x^0).$$

De cette manière, le coefficient de sensibilité de l'objet le plus cher sera plus grand que ceux des autres objets.

Une méthode simple pour le calcul numérique d'une approximation des dérivées partielles est un calcul par différences finies. En faisant varier légèrement un paramètre à la fois, il est possible de comparer les variations de la sortie du modèle. Ces variations sont appelées effets élémentaires et sont notées E_i^0 s'ils sont calculés au point x^0. Par exemple, pour une fonction différentiable,

$$\frac{\partial f}{\partial x_i}(x^0) \approx E_i^0 = \frac{f(x^0 + \Delta e_i) - f(x^0)}{\Delta}, \tag{2.2}$$

où e_i est un vecteur ne contenant que des 0 et un 1 à la i^e composante et $\Delta \in \mathbb{R}$. Une petite valeur pour le paramètre Δ donne, en général, une bonne approximation. Toutefois, trouver la valeur idéale reste compliqué. Cependant, d'autres approximations par différences finies existent dont certaines utilisent plus de points.

Les mesures basées sur les dérivées premières calculées par différences finies dépendent donc grandement du paramètre Δ et du point nominal x^0. Rien ne peut garantir la qualité de la mesure en dehors d'un voisinage du point nominal. De plus, cette mesure néglige une possible interaction entre les différents facteurs en entrée. Bien entendu, des mesures basées sur des approximations des dérivées d'ordre supérieur tiendraient compte de telles interactions, mais demanderaient plus de calcul. D'ailleurs, l'idée centrale de ces mesures est d'approximer les dérivées partielles de la fonction. Dans le cas où celle-ci n'est pas différentiable, le calcul des différences finies de l'équation (2.2) est toujours possible, mais cela ne donne aucune garantie de la qualité d'une telle mesure.

Cette méthode requiert de pouvoir choisir les points d'évaluation afin de calculer des mesures de sensibilité. Ceci n'est pas applicable dans notre situation, car les points mis à disposition sont issus des évaluations faites à partir d'un algorithme d'optimisation MADS ; celui-ci sera décrit à la section 3.3.3.

2.2 Méthodes de criblage

La question centrale de l'analyse de sensibilité est, parmi toutes les variables d'entrée, lesquelles sont *réellement importantes*, c'est-à-dire plus importantes que les autres. Une hypothèse simple est de supposer que le nombre de variables plus importantes que les autres est

relativement petit par rapport à leur nombre total, sinon elles ne seraient pas *réellement importantes*.

Les méthodes de criblage sont en général utilisées pour analyser de grands modèles, avec plusieurs centaines de facteurs d'entrée par exemple. Elles sont considérées comme des méthodes à faible coût, mais donnent uniquement une analyse qualitative, c'est-à-dire elles peuvent classer les entrées par ordre d'importance ou du moins identifier les variables qui ont un effet négligeable par rapport aux autres, mais ne peuvent quantifier la différence d'influence entre deux facteurs. Une utilisation possible des méthodes de criblage est d'exploiter un petit nombre d'évaluations pour identifier des variables peu influentes sur le modèle, pour ensuite simplifier le modèle et appliquer d'autres méthodes d'analyse de sensibilité, souvent plus précises mais plus coûteuses. L'idée derrière les méthodes de criblage est de discrétiser les valeurs possibles de chaque variable en plusieurs niveaux, et de faire varier les entrées en fonction de ces niveaux.

Parmi les types de méthodes de criblage, les *designs One At a Time* (OAT) sont les plus utilisés. Ils étudient l'importance d'une entrée en faisant varier un facteur à la fois. Bien que cela ressemble à une analyse locale, les auteurs de [53] présentent une adaptation globale des méthodes de criblage. Chaque variable peut prendre deux ou trois valeurs possibles, en général une valeur nominale et deux extrêmes. Ensuite, on applique une étude statistique d'un échantillon d'effets élémentaires de chaque variable. Il existe d'autres méthodes de type OAT moins coûteuses que la méthode de Morris décrite dans [53] mais cette dernière reste considérée comme plus complète [39]. Toutefois, si le nombre d'évaluations est petit, plus faible que le nombre de variables d'entrée par exemple, d'autres méthodes sont à privilégier dont certaines sont présentées dans [16, 28, 46].

La méthode de Morris [53] cherche à classer les variables en trois groupes : celles avec un effet négligeable, celles avec un effet linéaire et celles avec un effet non linéaire et/ou des interactions. On suppose que l'on veut analyser un modèle à n variables $x_1, \ldots, x_n$ et une sortie $y = f(x_1, \ldots, x_n)$, et que chaque facteur peut prendre $p > 1$ valeurs entre 0 et 1, à savoir $x_i \in \left\{0; \frac{1}{p-1}; \frac{2}{p-1}; \ldots; \frac{p-2}{p-1};\ 1\right\}$. A partir d'un point nominal x^0, un effet élémentaire du facteur i est défini comme

$$E_i^0 = \frac{f(x^0 + \Delta e_i) - f(x^0)}{\Delta},$$

où $\Delta \in \left\{\frac{1}{p-1}, \ldots, 1 - \frac{1}{p-1}\right\}$. Le calcul d'un effet élémentaire peut être répété à partir de plusieurs points nominaux. La moyenne de ces effets élémentaires à partir de r points $\{x^j\}_{j=1}^r$ est notée

$$\mu_i = \frac{1}{r} \sum_{j=1}^{r} EE_i^j.$$

De même, on peut calculer une moyenne des valeurs absolues μ_i^* et l'écart-type σ_i des effets élémentaires de chaque variable,

$$\mu_i^* = \frac{1}{r}\sum_{j=1}^{r} |EE_i^j|,$$
$$\sigma_i^2 = \frac{1}{r-1}\sum_{j=1}^{r}(EE_i^j - \mu_i)^2.$$

La moyenne μ_i^* donne une mesure de sensibilité du modèle. Si μ_i^* est petit, la variable i a un effet peu important. L'écart-type σ_i donne une idée de l'importance des interactions entre les variables. En effet, si σ_i est grand, cela indique que les effets élémentaires du facteur i dépendent du point nominal auquel ils sont calculés, et donc des valeurs des autres variables. A l'opposé, si σ_i est faible, cela indique que l'effet d'une variable ne dépend pas de la valeur des autres entrées, et donc que la variable i a un effet plutôt linéaire.

Ces effets élémentaires peuvent avoir des impacts opposés en fonction du point nominal et cela peut résulter en un faible μ_i. Dans ce cas, μ_i^* sera grand. A partir de μ_i, μ_i^*, σ_i^*, on peut déduire l'importance relative de chaque variable $i = 1, \ldots, n$, et donc les classer en trois groupes. Cela peut se faire en représentant les variables dans un graphe représentant σ et fonction de μ^*.

Exemple 2.1 *Par exemple, considérons la fonction suivante :*

$$f : [-1;1]^7 \mapsto \mathbb{R}, x \to 110x_1 + 100x_2 + 10x_3 + 60x_3x_4 + 50x_5^2 + 5x_6 + 2x_7 \qquad (2.3)$$

En appliquant la méthode de Morris en considérant $r = 5$ répétitions, ce qui donne $r(n+1) = 40$ évaluations du modèle, nous pouvons construire la figure 2.1. Nous remarquons que l'on peut distinguer facilement trois groupes de variables. Les premières sont les deux variables proches de l'origine. Il s'agit des variables qui ont un impact négligeable sur la fonction, les variables x_6 et x_7. En effet, aussi bien la moyenne que l'écart-type de leurs effets élémentaires sont très faibles. Le deuxième groupe reprend les variables en bas à droite, c'est-à-dire x_1 et x_2 qui ont un effet uniquement linéaire, tout comme x_6 et x_7, mais non négligeable ; on peut le remarquer car l'écart-type de leurs effets élémentaires est nul. Le troisième groupe reprend les variables qui ont un μ^* et un σ relativement importants. Il s'agit donc de variables ayant un effet non négligeable et non linéaire sur le modèle. On peut trouver les variables x_3, x_4 et x_5 dans ce groupe.

Grâce à ce genre de méthodes, il est possible d'explorer l'entièreté du domaine et avoir une

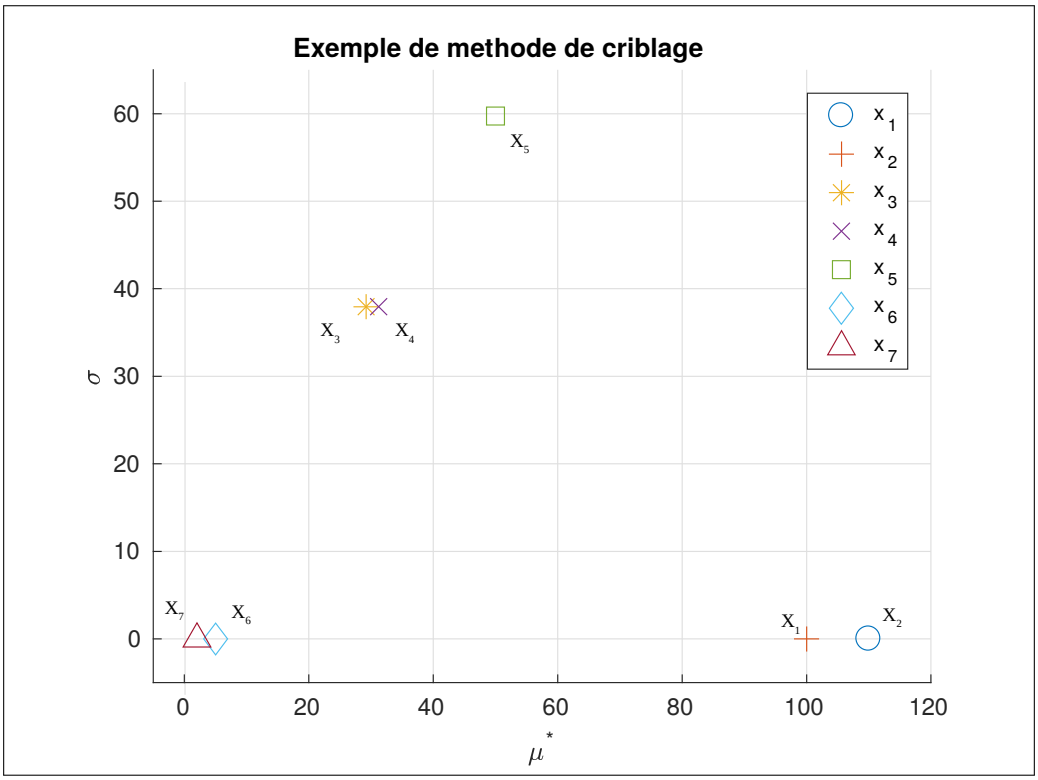


Figure 2.1 Exemple d'application de la méthode de Morris sur la fonction (2.3)

idée de l'impact global des variables sur le modèle. Toutefois, les résultats d'une telle analyse dépendent du choix des points nominaux auxquels les effets élémentaires sont calculés.

Cette méthode n'est pas directement applicable dans notre situation, et ce pour la même raison que précédemment. On veut exploiter l'ensemble des évaluations effectuées par un algorithme d'optimisation ; le choix de points spécifiques pour calculer une mesure de sensibilité n'est donc pas possible. Par contre, une telle méthode permet, en un faible nombre d'évaluations, d'avoir une idée des variables non influentes.

2.3 Méthodes basées sur la variance

Plusieurs méthodes d'analyse de sensibilité sont basées sur la variance. L'intérêt de cette mesure est son indépendance par rapport au modèle. Elle permet également de mettre en évidence les effets dus à la variation d'une variable particulière, ainsi que les effets dus aux interactions entre les variables.

L'idée derrière l'analyse de sensibilité basée sur la variance est de considérer les variables d'entrée x_i et de la sortie $y = f(x)$ comme des variables aléatoires, notées X_i et Y. La sensibilité de la sortie à une variable correspond à la réduction de variance observée si cette

variable d'entrée est fixée. La variance de la sortie peut être décomposée de la manière suivante :

$$Var[Y] = Var_{X_i}[E(Y|X_i)] + E_{X_i}(Var[Y|X_i]), \tag{2.4}$$

$$1 = \frac{Var_{X_i}[E(Y|X_i)]}{Var[Y]} + \frac{E_{X_i}(Var[Y|X_i])}{Var[Y]}, \tag{2.5}$$

où $Var_{X_i}[E(Y|X_i)]$ est la variance de l'espérance de Y conditionnelle à X_i et $E_{X_i}(Var[Y|X_i])$ est l'espérance de la variance de Y conditionnelle à X_i. L'équation (2.5) est obtenue en divisant l'équation (2.4) par $Var[Y]$.

Le premier terme de (2.5) est considéré comme une bonne mesure de sensibilité puisqu'il représente la proportion de variance de Y qui peut être attribuée à X_i. Il s'agit d'une variance de l'espérance de conditionnelle (VCE), notée

$$VCE[X_i] = Var_{X_i}[E(Y|X_i)].$$

Les différentes méthodes présentées ci-dessous décrivent plusieurs moyens d'estimer la VCE de chaque facteur. Dans ce document, la mesure du rapport de corrélation, la méthode des indices de Sobol' et la méthode *Fourier amplitude sensitivity test* (FAST) sont présentées.

2.3.1 Rapport de corrélation

La méthode des rapports de corrélation cherche à construire un estimateur du rapport de corrélation d'une variable. Le rapport de corrélation η_i de la variable x_i, $i = 1, 2, \ldots, n$ est défini comme

$$\eta_i^2 = \frac{Var_{X_i}[E(Y|X_i)]}{Var[Y]}, \quad i = 1, 2, \ldots, n.$$

Pour estimer un rapport de corrélation, il faut construire un estimateur de la variance de Y, $\widehat{Var[Y]}$, et de la variance de l'espérance conditionnelle $\widehat{VCE[X_i]}$. Ensuite l'estimation du rapport de corrélation se calcule par le rapport entre ces deux estimateurs ;

$$\widehat{\eta_i}^2 = \frac{\widehat{VCE[X_i]}}{\widehat{Var[Y]}}.$$

Les deux estimateurs nécessaires peuvent être calculés à partir de r réplications d'un échan-

tillonnage par hypercube latin de m éléments. On pose

$$\widehat{Var[Y]} = \frac{1}{mr}\sum_{j=1}^{m}\sum_{l=1}^{r}(y_{jl} - \bar{y})^2$$

où y_{jl} correspond à la sortie d'un échantillon et $\bar{y} = (1/mr)\sum_{j=1}^{m}\sum_{l=1}^{r} y_{jl}$ est la moyenne de toutes les sorties. La moyenne d'un échantillon est notée $\bar{y_j}$. Le deuxième estimateur est

$$\widehat{VCE[X_i]} = \frac{1}{m}\sum_{j=1}^{m}(\bar{y_j} - \bar{y})^2 - \frac{1}{mr^2}\sum_{j=1}^{m}\sum_{l=1}^{r}(y_{jl}^{(i)} - \bar{y_j})^2,$$

où $y_{jl}^{(i)}$ est obtenu en fixant les valeurs prises par la variable X_i à ses valeurs dans la première réplication, dans chacune des r réplications.

Dans le cadre de ce projet, cette méthode est intéressante, car elle peut être utilisée à partir d'un ensemble d'évaluations déjà effectuées. D'ailleurs, elle a déjà été utilisée dans des projets similaires [2–4, 15]. Ceux-ci seront décrits à la section 3.4.

2.3.2 Indices de Sobol'

La méthode des indices de Sobol' [65] se base sur une décomposition de la fonction du modèle à étudier. Pour décrire cette méthode, on suppose que l'espace des variables est un hypercube unitaire en dimension n, c'est-à-dire

$$f : \Omega^n = [0; 1]^n \to \mathbb{R}.$$

Chaque sous-ensemble de variables peut avoir un effet sur le modèle. Il est alors possible de décomposer une fonction f sous une somme de fonctions $f_{i_1,\ldots,i_m}$ représentant l'effet de chaque sous-ensemble de variables,

$$f(x) = f_0 + \sum_i f_i(x_i) + \sum_i \sum_{j>i} f_{ij}(x_i, x_j) + \cdots + f_{12\ldots k}(x_1, x_2, \ldots, x_k). \tag{2.6}$$

En choisissant chaque fonction carré intégrable, comme décrit dans [65], cette décomposition existe et est unique.

Dans le cadre d'une analyse de sensibilité, nous avons un vecteur aléatoire $X = (X_1, \ldots, X_n)$ composé de n variables indépendantes et $Y = f(X)$ la sortie d'un modèle déterministe $f(\cdot)$. Nous pouvons donc appliquer une analyse de variance fonctionnelle. La variance totale D de

la fonction f est définie comme

$$D = \int_{\Omega^k} f(\mathbf{x}) d\mathbf{x} - f_0^2$$

tandis que les variances partielles sont

$$D_{i_1,i_2,\ldots,i_s} = \int_0^1 \ldots \int_0^1 f^2_{i_1 i_2 \ldots i_s}(x_{i_1}, x_{i_2}, \ldots, x_{i_s}) dx_{i_1} dx_{i_2} \ldots dx_{i_s},$$

où $1 \leq i_1 < i_2 < \cdots < i_s$ et $s = 1, \ldots, k$.

En prenant le carré de (2.6) et en intégrant sur le domaine, on obtient

$$D = \sum_{i=1}^{k} D_i + \sum_{1 \leq i < j \leq k} D_{ij} + \cdots + D_{i,2,\ldots,k}.$$

On note alors que $D = Var[Y]$, $D_i = Var[E(Y|X_i)]$, $D_{ij} = Var[E(Y|X_i, X_j)] - D_i - D_j$ et ainsi de suite. Les indices de sensibilité sont définis comme

$$S_{i_1,\ldots,i_s} = \frac{D_{i_1,\ldots,i_s}}{D}.$$

S_i est l'indice de sensibilité de premier ordre du facteur i, $i = 1, \ldots, n$, et mesure l'effet du facteur i sur la sortie. S_{ij} est l'indice de sensibilité du deuxième ordre des facteurs i et j, $i < j$, et il mesure l'effet de l'interaction des facteurs i et j. Une propriété intéressante de ces indices est

$$\sum_{i=1}^{n} S_i + \sum_{1 \leq i < j \leq n} S_{ij} + \cdots + S_{1,2,\ldots,n} = 1.$$

Le nombre d'indices, correspondant au nombre de sous-ensembles de variables, croît exponentiellement avec la dimension du problème. En pratique, les indices d'ordres supérieurs à deux ne sont pas calculés.

Les auteurs de [37] présentent également des indices de sensibilité totaux ST_i d'une variable i comme la somme de tous les indices qui font intervenir la variable en question. Par exemple, l'indice de sensibilité totale de la variable x_1 d'un modèle à trois variables est

$$ST_1 = S_1 + S_{12} + S_{13} + S_{123}.$$

Lorsque le nombre de variables est important, il peut être judicieux de ne calculer que les indices du premier ordre et les indices totaux.

Les indices de sensibilité du premier ordre peuvent être estimés à partir de deux échantillons

de Monte Carlo de taille N. Les estimateurs sont construits de la façon suivante :

$$\begin{aligned}
\widehat{f_0} =& \frac{1}{N}\sum_{m=1}^{N} f(\mathbf{x}_m), \\
\widehat{D} =& \frac{1}{N}\sum_{m=1}^{N} (f(\mathbf{x}_m) - \widehat{f_0})^2, \\
\widehat{D_i} =& \frac{1}{N}\sum_{m=1}^{N} f(\mathbf{x}^{(1)}_{(\sim i)m}, x^{(1)}_{im}) f(\mathbf{x}^{(2)}_{(\sim i)m}, x^{(1)}_{im}) - \widehat{f_0}^2,
\end{aligned}$$

où les exposants (1) et (2) indiquent l'échantillon, $\mathbf{x}_m$ est un point dans Ω^n et

$$\mathbf{x}_{(\sim i)m} = (x_{1m}, x_{2m}, \ldots, x_{(i-1)m}, x_{(i+1)m}, \ldots x_{nm}).$$

Des manières pratiques de calculer les indices de Sobol' du premier ordre et les indices totaux grâce à deux échantillons de Monte Carlo sont décrites dans [59, 65] et [61, section 4.6].

Le calcul de ces indices de sensibilité requiert toutefois un grand nombre d'évaluations du modèle. Les auteurs de [39] décrivent un taux de convergence en $\sqrt{N}$ où N est la taille de l'échantillon, ce qui revient en général à un ordre de 10^4 évaluations du modèle.

Bien que cette méthode soit intéressante, le nombre d'évaluations nécessaire la rend peu utilisable dans le cadre de l'optimisation de boîte noire. En effet, le budget d'évaluations de la boîte noire est normalement très limité.

2.3.3 Méthode du *Fourier amplitude sensitivity test*

Le *Fourier amplitude sensitivity test* (FAST) [26] est une méthode qui permet d'estimer l'espérance et la variance de la sortie d'un modèle, ainsi que la contribution d'une variable à la variance. Celle-ci se base sur la transformée de Fourier, calculée en certains points et pour certaines fréquences. L'idée est de transformer des intégrales en n dimensions en des intégrales en une dimension.

On pose

$$X_i = G_i(\sin \omega_i s), \quad i = 1, 2, \ldots, n. \tag{2.7}$$

En choisissant des valeurs adaptées pour les fréquences ω_i et de bonnes transformations G_i, il est possible d'approximer l'espérance de la sortie Y comme

$$\mathbb{E}[Y] \approx \frac{1}{2\pi}\int_{-\pi}^{\pi} f(s)ds,$$

où $f(s) = f(G_1(\sin\omega_1 s), G_2(\sin\omega_2 s), \ldots, G_n(\sin\omega_n s))$. De même, l'approximation de la variance peut être calculée :

$$\begin{aligned}\mathbb{V}[Y] &= \frac{1}{2\pi}\int_{-\pi}^{\pi} f^2(s)ds - [\mathbb{E}(Y)]^2 \\ &\approx \sum_{j=-\infty}^{\infty}(A_j^2 + B_j^2) - (A_0^2 + B_0^2) \\ &\approx 2\sum_{j=1}^{\infty}(A_j^2 + B_j^2),\end{aligned}$$

où A_j et B_j sont les coefficients de Fourier

$$A_j = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(s)\cos(js)ds, \quad B_j = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(s)\sin(js)ds.$$

Le choix des transformations G_i est crucial. Différentes sources proposent les différentes transformations du tableau 2.1. Une bonne transformation devrait proposer une distribution uniforme pour chaque facteur. Les auteurs de [60] notent que les transformations (C) et (D) proposent une meilleure distribution que les transformations (A) et (B), et devraient donc être préférées à ces dernières.

Le calcul des indices de sensibilité se fait en calculant les coefficients A_j et B_j à leur fréquence fondamentale ω_i pour $i = 1, \ldots, n$, ainsi qu'à leurs harmoniques $p\omega_i$ pour $p = 1, 2, \ldots$. La contribution du facteur i à la variance de la sortie peut être approximée comme

$$D_{\omega_i} \approx 2\sum_{p=1}^{\infty}(A_{p\omega_i}^2 + B_{p\omega_i}^2).$$

En remarquant que les amplitudes de Fourier décroissent lorsque p croit, l'estimation de D_{ω_i}

Tableau 2.1 Proposition de transformations pour l'éqation (2.7) [60]

Tranformation	$\mathbf{G_i}$	**Référence**
(A)	$x_i = \bar{x_i}e^{\bar{v_i}\sin(\omega_i s)}$	[26]
(B)	$x_i = \bar{x_i}(1 + \bar{v_i}\sin(\omega_i s))$	[42]
(C)	$x_i = \frac{1}{2}\frac{1}{\pi}\arcsin(\sin(\omega_i s))$	[62]
(D)	$x_i = \frac{1}{2}\frac{1}{\pi}\arcsin(\sin(\omega_i s + \phi_i))$	[62]

peut être calculée par

$$\widehat{D}_{\omega_i} = 2\sum_{p=1}^{M}(A_{p\omega_i}^2 + B_{p\omega_i}^2),$$

où M correspond à la plus grande harmonique considérée, en général $M = 4$ ou $M = 6$ [27,60, section 8.4]. A partir de $\widehat{D_{\omega_i}}$, il est possible de calculer des coefficients équivalents aux indices de Sobol' du premier ordre. De plus, [60, section 8.4.3] présente une méthode basée sur le FAST permettant de calculer des indices totaux.

Même si cette méthode est moins coûteuse que la méthode présentée à la section précédente, elle reste coûteuse et est parfois biaisée ou instable lorsque la dimension du modèle devient trop grande, supérieur à 10 environ, comme précisé dans [66]. Cette méthode sera donc difficilement exploitable dans le cadre de ce projet, car les problèmes considérés ont un nombre de variables de l'ordre de quelques centaines voire quelques milliers de variables.

2.4 Méthode basée sur un modèle linéaire

En supposant que le modèle à étudier est linéaire, il est possible de calculer des coefficients de régression linéaire. En considérant un échantillon de N points, on suppose la relation

$$Y_j = \beta_0 + \sum_{i=1}^{N}\beta_i X_{ij} + \epsilon_j, \qquad \text{pour } j = 1, 2, \ldots, N, \tag{2.8}$$

où X_{ij} représente l'observation j de la variable x_i, pour $i = 1, \ldots, n$ et Y_j correspond à la sortie de l'échantillon $(X_{1j}, X_{2j}, \ldots, X_{kj})$. Les β_i sont des coefficients de régression et ϵ_j des termes d'erreur. Les coefficients sont à calculer par une méthode de moindres carrés.

En supposant les variables indépendantes entre elles, et qu'elles le soient également des erreurs, il est possible de décomposer la variance du modèle (2.8) comme

$$Var[Y] = \sum_{i=1}^{k} V_i + V_e,$$

où V_i donne la contribution de la variable X_i à la variance totale, et V_e la contribution des termes d'erreur. Les auteurs de [71] montrent qu'une estimation de V et V_i est possible,

$$\widehat{V} = \frac{1}{N-1}\sum_{i=1}^{N}(Y_i - \bar{Y})^2,$$
$$\widehat{V_i} = \frac{1}{N-1}\beta_i^2\sum_{i=1}^{N}(X_{ij} - \bar{X_i})^2.$$

Un coefficient de sensibilité S_i peut alors être construit,

$$S_i = \sqrt{\frac{\widehat{V_i}}{\widehat{V}}}.$$

Celui-ci est connu sous le nom de *standardized regression coefficient* ou SRC [19].

Si on suppose le modèle monotone, un autre indicateur connu sous le nom de *standardized rank regression coefficient* ou SRRC peut également être utilisé. Cela consiste à attribuer 1 à la plus petite valeur de sortie, 2 à la suivante jusqu'à N à la plus grande valeur. La même procédure est appliquée aux valeurs de chaque variable. Ensuite le calcul se fait de manière similaire que pour les SRC.

Il existe également d'autres coefficients [39] :

— coefficient de corrélation de Pearson :

$$\rho(X_i, Y) = \frac{\sum_{j=1}^{N} \Big(X_{ij} - \mathbb{E}(X_i)\Big)\Big(Y_j - \mathbb{E}(Y)\Big)}{\sqrt{\sum_{j=1}^{N} \Big(X_{ij} - \mathbb{E}(X_i)\Big)^2} \sqrt{\sum_{j=1}^{N} \Big(Y_j - \mathbb{E}(Y)\Big)^2}}.$$

Ce coefficient cherche à indiquer un lien linéaire entre une variable et la sortie du modèle. Il vaut 1 ou -1 si elles ont une relation linéaire et 0 si elles sont indépendantes.

— coefficient de corrélation partielle [38] :

$$PCC_i = \rho\Big(X_i - \widehat{X_{-i}}, Y - \widehat{Y_{-i}}\Big),$$

où $\widehat{X_{-i}}$ est la prédiction du modèle linéaire et $\widehat{Y_{-i}}$ est la prédiction du modèle linéaire lorsque X_i est absent.

Ces méthodes sont applicables dans notre contexte car les mesures sont calculées à partir d'un nuage de point. Néanmoins, elles se basent sur une hypothèse forte, que la fonction étudiée ait un certain caractère linéaire. Il existe des coefficients qui permettent de confirmer cette hypothèse :

— coefficient de détermination R^2 :

$$R^2 = \frac{\sum_{j=1}^{N} \Big(\hat{Y}_j - \mathbb{E}(Y)\Big)^2}{\sum_{j=1}^{N} \Big(Y_j - \mathbb{E}(Y)\Big)^2}$$

où $\hat{Y}_j$ correspond à la prédiction du modèle ;

— coefficient prédictif Q^2 :

$$Q^2 = 1 - \frac{\sum_{j=1}^{M} \left(Y_j - \widehat{Y_j}\right)^2}{\sum_{j=1}^{M} \left(Y_j - \mathbb{E}[Y]\right)^2},$$

où l'échantillon de taille M n'a pas été utilisé pour construire le modèle et $\widehat{Y_j}$ correspond à la prédiction du modèle pour le point $\mathbf{X_j} = (X_{1j}, \ldots, X_{nj})$.

Ces deux coefficients permettent d'exprimer la part de la variabilité de la sortie de la fonction étudiée qui peut être expliquée par le modèle linéaire. En règle générale, les problèmes étudiés dans le cadre de l'optimisation de boîte noire ne sont pas linéaires, ce qui rend cette méthode peu pertinente à ce projet.

2.5 Méthode basée sur les métamodèles

Une boîte noire ayant un long temps d'évaluation ne donne que peu de points disponibles pour une analyse de sensibilité. L'idée des méthodes basées sur des métamodèles est de trouver un substitut ou *surrogate* de la boîte noire ayant un comportement similaire à la fonction étudiée, mais dont le temps d'évaluation est grandement réduit. Il est alors possible d'évaluer le substitut un grand nombre de fois et d'appliquer l'une ou l'autre méthode d'analyse de sensibilité sur les points obtenus. Il existe un grand nombre de techniques possibles pour créer ce substitut, par exemple un modèle quadratique ou polynomial, l'utilisation de splines ou des modèles de krigeage. D'ailleurs, il existe des méthodes d'optimisation sans dérivée qui exploitent des substituts pour diminuer l'objectif. Celles-ci sont décrites un peu plus en détails à la section 3.2.

Etant donné que le substitut et la fonction à analyser sont différents, les résultats d'une analyse de sensibilité effectuée sur un substitut peuvent ne pas correspondre parfaitement à la fonction originale. Toutefois, si le substitut est de bonne qualité, celui-ci peut être utilisé avec l'une des méthodes d'analyse de sensibilité décrites dans ce chapitre. Il serait alors possible d'exploiter le substitut pour avoir un grand nombre de points à analyser tout en préservant le budget d'évaluations limité de la boîte noire décrivant l'objectif.

L'optimisation sans dérivée à l'aide de substitut est un domaine qui mérite un certain intérêt mais n'est pas l'objet du présent document. En effet, dans le cadre de ce projet, nous ne supposons pas avoir accès à un modèle simplifié de l'objectif, ce qui rend cette méthode inexploitable.

2.6 Analyse en composante principale

Dans l'introduction de son livre [40], Jolliffe décrit l'idée derrière l'analyse en composante principale comme celle *"de réduire la dimension d'un ensemble de données composées d'un grand nombre de variables liées entre elles, tout en retenant autant que possible la variabilité de l'ensemble de données. Cela est réalisé en le transformant en un nouvel ensemble de variables, les composantes principales, qui sont non corrélées, et qui sont ordonnées telles que les* quelques *premières variables retiennent la majorité de la variabilité présente dans* toutes *les variables originales"* ([40, chapitre 1], notre traduction).

Concrètement, on suppose X un vecteur aléatoire de taille n et les liens entre ces variables, leur structure de corrélation ou de covariance, nous intéressent. Pour éviter de regarder les n variances et les $\frac{n(n-1)}{2}$ covariances ou corrélations, il serait intéressant de regarder $p \ll n$ nouvelles variables qui retiennent la majeure partie de l'information contenue dans ces covariances. Ces nouvelles variables seront construites comme des combinaisons linéaires des variables originales de la manière suivante.

Dans un premier temps, une transformation linéaire $\alpha_1^\top X = \sum_{i=1}^n \alpha_{1,i} X_i$ est construite où $\alpha_1 \in \mathbb{R}^n$. Puisque la majorité de la variance doit être préservée, celle-ci est construite avec une variance maximale. Ensuite, on cherche une transformation linéaire $\alpha_2^\top x$, non corrélée avec $\alpha_1^\top x$, qui maximise également sa variance. La troisième transformation linéaire ne doit être corrélée ni avec $\alpha_1^\top x$ ni avec $\alpha_2^\top x$ tout en maximisant sa variance et ainsi de suite. Les vecteurs α_i sont appelés les composantes principales. Il existe jusqu'à n transformations, mais on espère que les $p \ll n$ premières transformations contiendront la majeure partie de la variance de l'ensemble de données.

Le calcul des composantes principales se fait à partir de la matrice de covariance $\Sigma \in \mathbb{R}^{n \times n}$ du vecteur aléatoire X. On note $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ ses valeurs propres. La première composante principale, le vecteur $\alpha_1 \in \mathbb{R}^n$, maximise $Var[\alpha_1^\top x] = \alpha_1^\top \Sigma \alpha_1$. Pour s'assurer que α_1 soit unique, il est utile d'imposer que ce dernier soit normé. Cette transformation α_1 est donc la solution du problème d'optimisation

$$\max_{\alpha_1 \in \mathbb{R}^n} \quad \alpha_1^\top \Sigma \alpha_1 \tag{2.9}$$

$$\text{sujet à} \quad \alpha_1^\top \alpha_1 = 1. \tag{2.10}$$

Pour résoudre ce problème, il est utile d'introduire le multiplicateur de Lagrange $\lambda \in \mathbb{R}$ et le Lagrangien

$$\mathcal{L}(\alpha_1, \lambda) = \alpha_1^\top \Sigma \alpha_1 - \lambda(\alpha_1^\top \alpha_1 - 1). \tag{2.11}$$

En imposant que le gradient de (2.11) soit nul, on obtient

$$\Sigma\alpha_1 - \lambda\alpha_1 = 0 \iff (\Sigma - \lambda I_n)\alpha_1 = 0,$$

où $I_n \in \mathbb{R}^{n\times n}$ est la matrice identité. Le multiplicateur de Lagrange λ est donc une valeur propre de Σ et α_1 le vecteur propre correspondant. La quantité à maximiser étant

$$\alpha_1^\top \Sigma\alpha_1 = \alpha_1^\top \lambda\alpha_1 = \lambda\alpha_1^\top\alpha_1 = \lambda,$$

le multiplicateur de Lagrange λ est égal à la plus grande valeur propre de Σ, $\lambda = \lambda_1$, et α_1 le vecteur propre correspondant.

Pour trouver la deuxième composante principale $\alpha_2 \in \mathbb{R}^n$, un problème d'optimisation similaire à (2.10) est construit, en ajoutant une contrainte. En effet, les composantes principales doivent être non-corrélées entre elles. Cette nouvelle contrainte est

$$Cov[\alpha_1^\top X, \alpha_2^\top X] = \alpha_1^\top\Sigma\alpha_2 = \alpha_2^\top\Sigma\alpha_1 = \alpha_2^\top\lambda_1\alpha_1 = \lambda_1\alpha_2^\top\alpha_1 = 0. \tag{2.12}$$

En ajoutant une contrainte de normalisation sur α_2, on obtient le problème

$$\max_{\alpha_2\in\mathbb{R}^n} \quad \alpha_2^\top\Sigma\alpha_2 \tag{2.13}$$

$$\text{sujet à} \quad \alpha_2^\top\alpha_2 = 1 \tag{2.14}$$

$$\alpha_2^\top\alpha_1 = 0. \tag{2.15}$$

Pour le résoudre, deux multiplicateurs de Lagrange $\lambda, \mu \in \mathbb{R}$ et le Lagrangien

$$\mathcal{L}(\alpha_2, \lambda, \mu) = \alpha_2^\top\Sigma\alpha_2 - \lambda(\alpha_2^\top\alpha_2 - 1) - \mu\alpha_2^\top\alpha_1$$

sont introduits. En annulant son gradient par rapport à α_2, on obtient l'équation

$$\Sigma\alpha_2 - \lambda\alpha_2 - \mu\alpha_1 = 0. \tag{2.16}$$

Il est possible de multiplier cette équation par $\alpha_1^\top \neq 0$, ce qui donne

$$\alpha_1^\top\Sigma\alpha_2 - \lambda\alpha_1^\top\alpha_2 - \mu\alpha_1^\top\alpha_1 = 0.$$

Puisque $\alpha_1^\top\alpha_1 = 1$ (2.10), $\alpha_1^\top\alpha_2 = 0$ (2.15) et $\alpha_1^\top\Sigma\alpha_2 = 0$ (2.12), l'équation se réduit à $\mu = 0$. Alors (2.16) devient

$$\Sigma\alpha_2 - \lambda\alpha_2 = 0 \iff (\Sigma - \lambda I_n)\alpha_2 = 0,$$

et λ est également une valeur propre de Σ et α_2 son vecteur propre correspondant. Puisqu'il faut maximiser λ et que les vecteurs propres de Σ, α_1 et α_2, doivent satisfaire $\alpha_2^\top \alpha_1 = 0$, λ correspond à la deuxième plus grande valeur propre de Σ, $\lambda = \lambda_2$, et α_2 à son vecteur propre correspondant.

Via un raisonnement similaire pour les $(n-2)$ composantes principales suivantes, on en déduit qu'elles correspondent aux $(n-2)$ vecteurs propres associés aux $(n-2)$ valeurs propres de Σ suivantes, ordonnées par ordre décroissant.

Alors, pour $k = 1, 2, \ldots, n$, la k^e composante principale est $z_k = \alpha_k^\top x$ où α_k est le vecteur propre de Σ associé à sa k^e plus grande valeur propre λ_k. De plus, si α_k est unitaire, $Var[z_k] = \lambda_k$.

Si Σ n'est pas connue, elle peut être estimée grâce à un échantillon de réalisations du vecteur aléatoire X. A partir d'un nuage de N points, il faut estimer la matrice de covariance. On suppose que D est une matrice N lignes n colonnes. Chaque ligne représente une observation des variables aléatoires. Une estimation de la matrice de covariance serait une matrice $S \in \mathbb{R}^{n \times n}$ où

$$S_{ij} = \frac{1}{N-1} \sum_{l=1}^{N} (D_{li} - \bar{D}_i)(D_{lj} - \bar{D}_j),$$

où $\bar{D}_j = \frac{1}{N} \sum_{i=1}^{N} D_{ij}$ est la moyenne des réalisations de la j^e variable aléatoire.

Ce projet porte sur l'optimisation de boîte noire. Dans ce contexte, une analyse en composante principale pourrait être utilisée pour en apprendre plus sur la fonction objectif à minimiser. En effet, une telle analyse pourrait être appliquée pour déterminer des liens entre les variables, et plus particulièrement mettre en avant des liens entre certaines variables et la fonction objectif. Un des avantages de l'analyse en composante principale est qu'elle peut s'appliquer à un nuage de points. Il ne sera donc pas nécessaire d'utiliser des évaluations pour faire cette analyse. De plus, lors de l'analyse, des combinaisons de variables apparaissent naturellement, ce qui est un peu plus général que des coefficients de sensibilité pour chaque variable.

CHAPITRE 3 Méthodes d'optimisation sans dérivée

Ce chapitre se concentre sur les méthodes d'optimisation sans dérivée. Ces méthodes sont développées pour résoudre des problèmes mathématiques de la forme :

$$\begin{aligned} \min_{x \in \mathcal{X}} \quad & f(x) \\ \text{sujet à} \quad & x \in \Omega, \end{aligned}$$

où les dérivées de la fonction objectif f ne sont pas accessibles ou n'existent pas.

Les algorithmes présentés sont classés dans trois catégories. La première catégorie, décrite à la section 3.1, reprend des méthodes heuristiques. Il s'agit de méthodes qui ne sont pas supportées par une analyse de convergence prouvée mathématiquement. La section 3.2 reprend des méthodes basées sur des modèles, tandis que la section 3.3 présente des méthodes de recherche directe. Les premières tentent de construire un modèle de la fonction objectif plus simple à optimiser mais gardant un comportement similaire et les secondes ont une structure spécifique qui permet d'analyser leur convergence. La dernière section revient sur des algorithmes d'optimisation sans dérivée qui utilisent des méthodes d'analyse de sensibilité.

3.1 Méthodes heuristiques

Bien que les méthodes heuristiques ne soient pas supportées par des résultats de convergence prouvés mathématiquement, elles sont fortement utilisées en pratique. En effet, elles sont souvent plus simples à comprendre et à implémenter que d'autres méthodes d'optimisation. De plus, elles peuvent être modifiées afin de mieux correspondre au problème et peuvent offrir de bonnes performances. Ces méthodes ont un intérêt supplémentaire, en effet elles peuvent être utilisées comme sous-méthodes dans d'autres algorithmes qui eux garantissent des résultats de convergence. Ce document présente rapidement quelques heuristiques connues en optimisation, notamment la famille des *algorithmes génétiques* et l'algorithme de *Nelder-Mead*.

3.1.1 Algorithme *Hit-and-run*

Décrit dans [18,58,64], l'algorithme *Hit-and-run* est d'une conception assez simple. A chaque itération, la méthode compare la solution courante x^k à l'itération k à un autre candidat

généré aléatoirement. Si ce candidat améliore la solution, alors l'itéré est mis à jour. La génération aléatoire des candidats se fait en deux étapes. Dans un premier temps, une direction d est choisie aléatoirement à partir d'une distribution dense dans la sphère unité. Ensuite, la longueur du pas s est générée à partir d'une distribution uniforme, telle que $x^k + sd$ soit réalisable. Bien qu'une analyse de convergence de cet algorithme soit proposée dans [14], l'algorithme *Hit-and-run* est placé dans la catégorie des heuristiques, car il a un comportement fortement aléatoire et est d'une conception assez simple. En effet, sa convergence repose sur le fait qu'en évaluant suffisamment de points aléatoirement, la solution finira par être trouvée.

3.1.2 Algorithme génétique

Les algorithmes génétiques, aussi appelés algorithmes évolutifs, se basent sur la théorie de l'évolution et de survie des individus les mieux adaptés. Pour préserver l'analogie biologique, une solution est appelée individu, un ensemble de solutions forme une population et les individus se combinent pour en créer de nouveaux. Le cadre général des algorithmes génétiques est décrit à l'algorithme 1. Celui-ci laisse beaucoup de flexibilité et d'interprétation, notamment au point 3 où aucun critère d'arrêt n'est précisé, ce qui correspond à une heuristique. De même, la sélection de la population initiale, la métrique d'*aptitude* (*fitness*), la sélection des parents ou des survivants, la création d'une progéniture ou la mutation de celle-ci sont des étapes clefs mais aucune méthode claire pour choisir ceux-ci n'est définie. De bonnes pratiques pour certaines de ces décisions existent, sans toutefois donner de règles fixes garantissant la convergence de la méthode.

C'est dans la flexibilité des algorithmes génétiques que réside leur force, mais aussi leur principal défaut. En effet, ceux-ci peuvent être adaptés à chaque problème particulier afin d'améliorer leur performance, mais rien ne garantit que deux algorithmes génétiques aient un comportement similaire face à un même problème d'optimisation.

La qualité d'un individu est appelée son aptitude. Bien que celle-ci doit être liée à la valeur de la fonction objectif, plusieurs façons de calculer une aptitude existent. Elles doivent toutefois respecter deux règles.

1. Pour un problème de minimisation d'une fonction f, si $f(x) < f(y)$, alors la valeur de l'aptitude de x doit être plus grande que celle de y. Cette règle s'inverse pour un problème de maximisation.
2. L'aptitude d'un individu doit être strictement positive.

Deux méthodes pour calculer une métrique d'aptitude utilisant la fonction objectif sont proposées dans [10], il s'agit de l'aptitude de rang, qui ordonne les points en fonction de

Algorithme 1 Algorithme génétique

Données : objectif $f : \mathbb{R}^n \mapsto \mathbb{R}$ et une population initiale $P^0 = \{x^1, x^2, \ldots, x^{\bar{p}}\}$

0. Initialisation

$\gamma \in (0; 1)$	Probabilité de mutation
$k \leftarrow 0$	Compteur d'itération

1. Aptitude

▷ Utiliser la valeur de la fonction objectif $f(x)$ pour déterminer une aptitude de survie de chaque individu $x \in P^k$

2. Croisement

▷ 2.1 Sélection : sélectionner 2 parents de la population P^k et aller à 2.2 ou sélectionner un survivant de P^k et aller à 2.4 ;

▷ 2.2 Croisement : utiliser les deux parents pour créer une progéniture ;

▷ 2.3 Mutation : avec une probabilité γ appliquer une mutation à la progéniture et vérifier la faisabilité de la mutation :

- ▷ Si la mutation n'est pas réalisable, déclarer la progéniture morte et aller à 2.1
- ▷ Sinon, déclarer la progéniture survivante et aller à 2.4 ;

▷ 2.4 Mise à jour de la nouvelle génération :

- ▷ Placer le survivant dans la population P^{k+1},
- ▷ Si $|P^{k+1}| \geq \bar{p}$, aller à 3
- ▷ Sinon, aller à 2.1

3. Mise à jour

▷ Incrémenter $k \leftarrow k + 1$, stop ou aller à 1

leur valeur, et de l'aptitude de valeur de la fonction, qui vaut $-f(x^i) + \bar{f} + 1$ où $\bar{f} = \max_{i=1,2,\ldots,\bar{p}} f(x^i)$ pour une population $\{x^1, x^2, \ldots, , x^p\}$.

Une fois la métrique d'aptitude déterminée, il faut établir une méthode de sélection. Cette méthode doit permettre de choisir un individu survivant ou deux individus parents d'une population. Sélectionner deux parents revient à choisir deux fois un survivant, à l'exception que les deux survivants doivent être différents. La sélection élitiste, la roulette et le tournoi font partie des méthodes de sélection les plus populaires. La première consiste à garder les individus avec la meilleure aptitude. La sélection par roulette choisit des individus aléatoirement avec des probabilités pondérées en fonction de leur aptitude. La dernière sélectionne une sous-population et applique une des deux méthodes précédentes.

Une fois la sélection effectuée viennent les étapes de croisement et de mutation. Celles-ci se basent sur un système d'encodage. Chaque individu est représenté par un ensemble de *bits*, souvent appelé chromosomes pour respecter la métaphore biologique. Plusieurs stratégies de croisement existent et se basent sur le fait de garder une partie des chromosomes de chaque parent pour créer un nouvel individu. Les stratégies de mutation consistent en l'inversion de

certains *bits* choisis aléatoirement.

A la troisième étape de l'algorithme 1, l'algorithme génétique a la possibilité de s'arrêter. Toutefois, cet algorithme n'utilise pas de mesure intrinsèque pour quantifier la convergence de la méthode. Les critères d'arrêt les plus utilisés se basent donc sur un budget d'évaluations ou sur un nombre maximum d'itérations sans amélioration de la valeur de l'objectif. A l'aide de certaines hypothèses sur la sélection et l'encodage, la convergence de l'algorithme génétique peut être étudiée. Des méthodes pour satisfaire ces hypothèses sont données dans [10, section 4.2-4.5]. Concrètement, les résultats de convergence de l'algorithme génétique peuvent se résumer à *"en essayant suffisamment de points, on finira par trouver la meilleure solution"*. Néanmoins, les algorithmes génétiques sont très populaires et offrent parfois des performances satisfaisantes. D'une manière générale, si on trouve des stratégies d'encodage et de croisement qui correspondent très bien au problème, les algorithmes génétiques auront de bonnes performances ; si les stratégies d'encodage et de croisement ne correspondent pas adéquatement au problème, alors il sera plus intéressant d'utiliser d'autres méthodes.

3.1.3 Algorithme de Nelder-Mead

L'algorithme de Nelder-Mead est une des méthodes d'optimisation sans dérivée les plus populaires. Celle-ci a été introduite sous le nom de méthode du simplexe [55]. Sa popularité vient à la fois de sa simplicité d'implémentation et de ses performances. L'algorithme original ne possédait pas de résultat de convergence, même sur des fonctions différentiables et convexes [48], mais des versions modifiées de l'algorithme existent et assurent sa convergence vers un point stationnaire [25, 41, 68].

Comme son nom original l'indique, la méthode de Nelder-Mead se base sur un simplexe. Dans un espace de dimension n, un simplexe est le sous-espace défini comme l'enveloppe convexe de $(n+1)$ points formant ses sommets.

A chaque itération k, il existe un ensemble de $(n+1)$ sommets formant le simplexe $Y_k = \{y_k^0, y_k^1, \ldots, y_k^n\}$, ordonnés par ordre croissant de la valeur de f. Ensuite une opération de réflexion, d'expansion et/ou de contraction est appliquée pour remplacer le dernier sommet. Le nouveau sommet est

$$y = y^c + \delta(y^c - y^n),$$

où $\delta \in \mathbb{R}$ et $y^c = \sum_{i=0}^{n-1} y^i/n$ est le centroïde des n premiers sommets. La valeur du paramètre δ définit l'opération appliquée et celle-ci dépend du succès des transformations précédentes. Une réduction peut également être appliquée ; celle-ci ne garde que le meilleur sommet et

remplace tous les autres de la manière suivante :

$$y^{s_i} = y^0 + \delta^s(y^i - y^0), \quad i = 1, 2, \ldots, n.$$

Les paramètres δ^s, δ^{ic}, δ^{oc}, δ^r, δ^e correspondent au paramètre δ pour les opérations de rétrécissement, de contraction interne, de contraction externe, de réflexion et d'expansion respectivement. Une description de l'algorithme de Nelder-Mead est disponible à l'algorithme 2.

3.2 Méthodes basées sur des modèles

Comme leur nom l'indique, les méthodes d'optimisation sans dérivée basées sur des modèles utilisent des modèles de la fonction objectif pour optimiser celle-ci. L'idée derrière ces méthodes est d'utiliser ou de construire un modèle plus simple, qui peut être optimisé plus facilement que le problème orignal. Si la qualité du modèle est satisfaisante, les solutions obtenues suite à son optimisation devraient être des solutions intéressantes du problème modélisé.

Il existe deux grandes catégories de modèles : les modèles statiques ou dynamiques. Les premiers sont plus souvent appelés substituts ou *surrogates*. Il s'agit de modèles qui ont un comportement similaire à la fonction objectif mais qui sont plus simples et plus rapides à évaluer. Ceux-ci sont en général fournis par l'utilisateur et restent les mêmes au cours de l'optimisation, d'où leur nom de statiques. L'utilisation la plus simple de ces modèles consiste à les optimiser en supposant que la solution du modèle sera une bonne solution du problème original.

Les modèles dynamiques sont construits et mis à jour au cours de l'optimisation. Prenons un modèle quadratique pour exemple. Celui-ci est construit à partir de la base des polynômes de degré inférieur ou égal à 2, qui possède $q + 1 = \frac{(n-1)(n-2)}{2}$ éléments :

$$\begin{aligned}\phi(x) &= (\phi_0(x), \phi_1(x), \ldots, \phi_q(x)) \\ &= \left(1, x_1, x_2, \ldots, x_n, \frac{x_1^2}{2}, \frac{x_2^2}{2}, \ldots, \frac{x_n^2}{2}, x_1x_2, x_1x_3, \ldots, x_{n-1}x_n\right).\end{aligned}$$

Le modèle quadratique de la fonction objectif m_f sera donc défini par $\alpha \in \mathbb{R}^{q+1}$, $m_f(x) = \alpha^\top \phi(x)$. La construction du modèle se fait à partir d'un ensemble de $p + 1$ points de $\mathbb{R}^n$, $Y = \{y^0, y^1, \ldots, y^p\}$. Puisque le modèle doit être le plus proche possible de la fonction objectif, il faut trouver $\alpha \in \mathbb{R}^{q+1}$ tel que $\sum_{y \in Y}(f(y) - m_f(y))^2$ soit minimal. En définissant

Algorithme 2 Algorithme de Nelder-Mead

0. Initialisation

- ▷ Evaluer f aux points de $Y_0 = \{y_0^0, y_0^1, \ldots, y_0^n\}$
- ▷ Choisir les paramètres $0 < \delta^s < 1, -1 < \delta^{ic} < 0 < \delta^{oc} < \delta^r < \delta^e$
- ▷ $k \leftarrow 0$

1. Itération

- ▷ **Ordonnancement**
  - ▷ Ordonner les points de Y_k tels que $f^0 = f(y_k^0) \leq f^1 = f(y_k^1) \leq \cdots \leq f^n = f(y_k^n)$
- ▷ **Réflexion**
  - ▷ Construire y^c et $y^r = y^c + \delta^r(y^c - y^n)$
  - ▷ Evaluer $f^r = f(y^r)$
  - ▷ Si $(f^0 \leq f^r < f^{n-1})$
    - ▷ $Y_{k+1} \leftarrow \{y_k^0, y_k^1, \ldots, y_k^{n+1}, y^r\}$
    - ▷ $k \leftarrow k + 1$ et aller à 1
- ▷ **Expansion** si $(f^r < f^0)$
  - ▷ Construire $y^e = y^c + \delta^e(y^c - y^n)$
  - ▷ Evaluer $f^e = f(y^e)$
  - ▷ Si $(f^e \leq f^r)$
    - ▷ $Y_{k+1} \leftarrow \{y_k^0, y_k^1, \ldots, y_k^{n+1}, y^e\}$
  - ▷ Sinon
    - ▷ $Y_{k+1} \leftarrow \{y_k^0, y_k^1, \ldots, y_k^{n+1}, y^r\}$
  - ▷ $k \leftarrow k + 1$ et aller à 1
- ▷ **Contraction** si $(f^r \geq f^{n+1})$
  - ▷ Si $(f^r < f^n)$: contraction externe
    - ▷ Construire $y^{oc} = y^c + \delta^{oc}(y^c - y^n)$
    - ▷ Evaluer $f^{oc} = f(y^{oc})$
    - ▷ Si $(f^{oc} \leq f^r)$
      - ▷ $Y_{k+1} \leftarrow \{y_k^0, y_k^1, \ldots, y_k^{n+1}, y^{oc}\}$
      - ▷ $k \leftarrow k + 1$, aller à 1
    - ▷ Sinon
      - ▷ Aller à **Rétrécissement**
  - ▷ Si $(f^r \geq f^n)$: contraction interne
    - ▷ Construire $y^{ic} = y^c + \delta^{ic}(y^c - y^n)$
    - ▷ Evaluer $f^{ic} = f(y^{ic})$
    - ▷ Si $(f^{ic} < f^r)$
      - ▷ $Y_{k+1} \leftarrow \{y_k^0, y_k^1, \ldots, y_k^{n+1}, y^{ic}\}$
      - ▷ $k \leftarrow k + 1$, aller à 1
- ▷ **Rétrécissement** si $(f^r \geq f^{n-1})$
  - ▷ Construire les n points $y^{s_i}, i = 1, 2, \ldots, n$
  - ▷ Evaluer f aux n points
  - ▷ $Y_{k+1} \leftarrow \{y^{s_1}, y^{s_2}, \ldots, y^{s_n}\}$
  - ▷ $k \leftarrow k + 1$ et aller à 1

une matrice $M(\phi, Y) \in \mathbb{R}^{(p+1)\times(q+1)}$,

$$M(\phi, Y) = \begin{bmatrix} \phi_0(y^0) & \phi_1(y^0) & \cdots & \phi_q(y^0) \\ \phi_0(y^1) & \phi_1(y^1) & \cdots & \phi_q(y^1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(y^p) & \phi_1(y^p) & \cdots & \phi_q(y^p) \end{bmatrix},$$

cela revient à résoudre $M(\phi, Y)\alpha = f(Y)$, où $f(Y) = (f(y^0), f(y^1), \ldots, f(y^p))^\top$.

Si $p > q$, plus de points que nécessaire sont disponibles. La construction du modèle se fait en résolvant

$$\min_{\alpha\in\mathbb{R}^{q+1}} \|M(\phi, Y)\alpha - f(Y)\|^2.$$

Si $p = q$, alors la solution est unique. Si $p < q$, nous ne disposons pas de suffisamment de points pour construire le modèle, et il existe une infinité de solutions. C'est le cas le plus fréquent en optimisation de boîte noire, puisque le budget d'évaluations est souvent limité. Dans ce cas, la solution choisie sera celle qui minimise la norme de Frobenius de la matrice Hessienne, c'est-à-dire qui minimise la courbure du modèle.

Il peut être intéressant de juger de la qualité d'un modèle donné. En supposant une fonction $f \in \mathcal{C}^1$ et ∇f Lipschitz continue, un modèle m_f est appelé complètement linéaire (CL) sur la boule centrée en y de rayon Δ, $B(y; \Delta)$, si

$$\begin{cases} |f(x) - m_f(x)| \leq \kappa_f \Delta^2 \\ |\nabla f(x) - \nabla m_f(x)| \leq \kappa_g \Delta \end{cases},$$

pour tout $x \in B(y; \Delta)$ et pour des constantes κ_f, κ_g ; Δ est appelé paramètre de précision du modèle. Il existe également une définition d'un modèle complètement quadratique (CQ). Soit $f \in \mathcal{C}^2$ et $\nabla^2 f$ Lipschitz continue, un modèle m_f est complètement quadratique si

$$\begin{cases} |f(x) - m_f(x)| \leq \kappa_f \Delta^3 \\ |\nabla f(x) - \nabla m_f(x)| \leq \kappa_g \Delta^2 \\ |\nabla^2 f(x) - \nabla^2 m_f(x)| \leq \kappa_h \Delta^3 \end{cases},$$

pour tout $x \in B(y; \Delta)$ et pour des constantes κ_f, κ_g et κ_h.

Un ensemble de modèles $\mathcal{M} = \{m : \mathbb{R}^n \mapsto \mathbb{R}, m \in \mathcal{C}^2\}$ est une classe complètement linéaire (resp. quadratique) s'il existe un modèle complètement linéaire (resp. quadratique) dans

$\mathcal{M}$ et s'il existe une procédure capable de, soit déterminer si un modèle est complètement linéaire (resp. quadratique) sur $B(x;\Delta)$, soit trouver un modèle complètement linéaire (resp. quadratique) sur $B(x;\Delta)$.

Vu qu'un modèle est généralement plus simple que la fonction objectif du problème d'optimisation, les limitations dues au temps d'évaluation de la boîte noire ne s'appliquent plus aux modèles. Les auteurs de [10] concluent que si les modèles sont de qualité suffisante, alors il est possible d'adapter des méthodes d'optimisation lisse à l'optimisation sans dérivée. La suite de cette section présente une généralisation de la méthode du gradient basée sur des modèles ainsi qu'une méthode de région de confiance.

3.2.1 Descente basée sur des modèles

La méthode du gradient [56, Chapitre 1] est une méthode d'optimisation bien connue. Celle-ci consiste à réaliser une recherche linéaire dans la direction de plus forte descente, celle opposée au gradient. Toute la méthode se base donc sur le calcul du gradient à chaque itération pour trouver la direction de plus forte descente.

Grâce à l'utilisation de modèles, une généralisation de cette méthode sans dérivée est possible. L'idée de base est la suivante. A chaque itération, un modèle local dérivable est construit, vient ensuite une recherche linéaire dans la direction opposée au gradient du modèle. La qualité du modèle est assez importante pour l'optimisation, c'est pourquoi celle-ci est vérifiée avant d'effectuer la recherche linéaire.

On introduit les notations suivantes. $x_0 \in \mathbb{R}^n$ représente le point de départ et $x_k \in \mathbb{R}^n$ l'itéré courant, m_f est un modèle complètement linéaire de précision $\Delta_k > 0$. Un paramètre $\mu_0 > 0$ vérifie la précision de l'évaluation du modèle, $\eta \in]0;1[$ le paramètre d'Armijo, $\epsilon_d \in]0;1[$ l'angle minimum de descente et $\epsilon \geq 0$ une tolérance d'arrêt. L'algorithme de descente basée sur des modèles est décrit à l'algorithme 3.

3.2.2 Méthodes de région de confiance

Les méthodes de région de confiance exploitent un modèle simple, en général lisse et facile à évaluer, et supposent qu'il a un comportement similaire à l'objectif dans un voisinage, la région de confiance, de la solution courante.

Un modèle linéaire peut être utilisé car il n'a besoin que de $\mathcal{O}(n)$ points pour être construit mais il ne donne aucune information quant à la courbure du problème. Un modèle quadratique

Algorithme 3 Descente basée sur des modèles

0. Initialisation
- ▷ Choisir x_0, Δ_0, μ_0, ϵ_d et ϵ
- ▷ $k \leftarrow 0$.

1. Construction du modèle
- ▷ Construire m_f^k à partir de Δ_k et d'un nombre de points

2. Vérification du modèle
- ▷ Si ($\Delta_k < \epsilon$ et $\|\nabla m_f^k(x_k)\| < \epsilon$) : Stop
- ▷ Si ($\Delta_k > \mu \|\nabla m_f^k(x_k)\|$) :
  - ▷ Modèle pas précis
  - ▷ $\Delta_{k+1} \leftarrow \Delta_k/2, \mu_{k+1} \leftarrow \mu_k, x_{k+1} \leftarrow x_k$
  - ▷ $k \leftarrow k+1$, aller à l'étape 1
- ▷ Si ($\Delta_k \leq \mu \|\nabla m_f^k(x_k)\|$) :
  - ▷ Modèle précis
  - ▷ Aller à l'étape 3

3. Recherche linéaire
- ▷ Choisir d_k tel que $\left(\frac{d_k}{\|d_k\|}\right)^\top \left(\frac{\nabla m_f^k(x_k)}{\|\nabla m_f^k(x_k)\|}\right) < -\epsilon_d$
- ▷ Chercher t_k tel que $f(x_k + t_k d_k) < f(x_k) + \eta t_k d_k^\top \nabla m_f^k(x_k)$

4. Mise à jour
- ▷ Si t_k est trouvé :
  - ▷ $x_{k+1} \leftarrow y$ où $f(y) \leq f(x_k + t_k d_k)$
  - ▷ $\mu_{k+1} \leftarrow \mu_k$
- ▷ Sinon :
  - ▷ $x_{k+1} \leftarrow x_k$
  - ▷ $\mu_{k+1} \leftarrow \mu_k/2$
- ▷ $\Delta_{k+1} \leftarrow \Delta_k$
- ▷ $k \leftarrow k+1$, aller à l'étape 1

est donc en général privilégié, de la forme :

$$m_f(x) = f(x_k) + g_k^\top (x - x_k) + (x - x_k)^\top H_k (x - x_k)^\top,$$

où $g_k \in \mathbb{R}^n$ et $H_k \in \mathbb{R}^{n \times n}$ symétrique. Ces paramètres correspondent au gradient et à la Hessienne du modèle en $x = 0$ et sont estimés en imposant que le modèle interpole un certain ensemble de points, $m_f(x^i) = f(x^i)$, $i = 0, 2, \ldots, p$.

Le modèle est supposé fidèle dans un voisinage du point x_k, en général ce voisinage est pris comme la boule de rayon Δ_k,

$$B(x_k, \Delta_k) = \{x \in \mathbb{R}^n : \|x - x_k\| \leq \Delta_k\}.$$

A chaque itération, ce modèle est minimisé dans la boule de rayon de Δ_k. Bien entendu,

le modèle n'est pas toujours très fidèle à l'objectif. C'est pourquoi le calcul d'un rapport r comparant la réduction prévue par le modèle et celle observée par l'objectif est d'un certain intérêt. Pour un point t, ce rapport est défini comme

$$r(t) = \frac{f(x_k) - f(t)}{m_f(x_k) - m_f(t)}.$$

Si ce rapport n'est pas suffisant, le modèle est considéré comme pas assez fidèle et soit la taille de la région de confiance est réduite, soit une procédure sensée améliorer la qualité du modèle est appliquée. L'algorithme complet est décrit à l'algorithme 4.

Algorithme 4 Algorithme de région de confiance sans dérivée

0. Initialisation
- ▷ Choisir une classe de modèle
- ▷ Choisir une méthode d'amélioration du modèle
- ▷ Données : x_0, Δ_{max}, $\Delta_0 \in]0; \Delta_{max}$
- ▷ Initialisation le modèle m_f
- ▷ $k \leftarrow 0$

1. Test critique
- ▷ Si $\|g_k\| \leq \epsilon$
  - ▷ Appliquer la méthode d'amélioration du modèle
  - ▷ Si le modèle n'est pas assez bon ou la région de confiance trop grande
    - ▷ Construction d'un nouveau modèle
  - ▷ Sinon
    - ▷ Vérifier le critère d'arrêt → STOP

2. Optimisation du sous-problème
- ▷ Trouver $t \in \operatorname{argmin}_{x \in B(x_k; \Delta_k)} m_f(x)$
- ▷ Evaluer $f(t)$ et calculer le rapport $r(t)$

3. Acceptation du candidat
- ▷ Si $r(t) \geq \eta_1$
  - ▷ $x_{k+1} \leftarrow t$, mise à jour du modèle avec le point t
- ▷ Sinon
  - ▷ $x_{k+1} \leftarrow x_k$

4. Amélioration du modèle
- ▷ Si $r(t) < \eta_1$
  - ▷ Appliquer la procédure d'amélioration du modèle

5. Mise à jour de la région de confiance

$$\Delta_{k+1} \in \begin{cases} [\Delta_k; \min\{\gamma_{inc}\Delta_k; \Delta_{max}\}] & \text{si } r(t) \geq \eta_1 \\ \{\gamma_{dec}\Delta_k\} & \text{si } r(t) < \eta_1 \text{ et } m_f \text{ est CL} \\ \{\Delta_k\} & \text{si } r(t) < \eta_1 \text{ et } m_f \text{ n'est pas CL} \end{cases}$$

- ▷ $k \leftarrow k + 1$

3.2.3 Optimisation Bayesienne

Les méthodes d'optimisation Bayesienne se basent elles aussi sur des modèles. Toutefois, il s'agit plutôt de modèles probabilistes.

En optimisation Bayesienne, la fonction objectif est considérée comme aléatoire, puisque celle-ci n'est pas connue. La fonction objectif suit une distribution *a priori*, qui représente ce qui est supposé de la fonction objectif. Une fois que la boîte noire est évaluée, la distribution de la fonction objectif est mise à jour pour former une distribution *a posteriori*. Les distributions *a priori* et *a posteriori* sont liées via la règle de Bayes.

En pratique, la fonction objectif est représentée par une variable aléatoire F avec une distribution *a priori* $p(F)$. Celle-ci représente notre croyance quant aux possibles valeurs que pourrait prendre F avant que celle-ci n'ait été observée. Ensuite, en ayant accès à des données D, des évaluations de l'objectif, et un modèle de vraisemblance $p(D|F)$, il est possible de construire la distribution *a posteriori* $p(F|D)$,

$$p(F|D) \propto p(D|F)p(F).$$

Cette distribution *a posteriori* est ensuite utilisée pour construire une fonction d'acquisition u qui déterminera de nouveaux points à évaluer. Ces nouveaux points seront utilisés pour mettre à jour les distributions à l'itération suivante. Une description de haut niveau de cette méthode est présentée à l'algorithme 5.

Il existe plusieurs manières de construire les distributions *a priori* et *a posteriori* ; les processus Gaussiens ou les processus de Wiener sont les plus connus. Plusieurs modèles sont présentés dans [63, section 2 et 3] et dans [20, section 2]. La nature de la fonction d'acquisition est également importante. En effet, les nouveaux points à évaluer sont déterminés par celle-ci, soit car la valeur de l'objectif est intéressante soit car l'incertitude quant à la fonction est importante. Cela permet une balance naturelle entre l'exploration de l'espace de recherche et l'intensification de la recherche à un endroit plus précis. Les exemples les plus connus consistent à maximiser la probabilité d'amélioration, l'espérance d'amélioration de la fonction ou une borne de confiance [52]. Une revue de possibles fonctions d'acquisition est présentée dans [63, section 4] et [20, section 2.3].

3.3 Méthodes de recherche directe

Les *méthodes de recherche directe* sont un ensemble d'algorithmes d'optimisation qui ont la structure suivante. Au début d'une itération, l'algorithme connaît une *solution courante*. Il

Algorithme 5 Optimisation Bayesienne

0. Initialisation
- ▷ Ensemble de données $D_0 = \emptyset$
- ▷ Fonction d'acquisition u
- ▷ $k \leftarrow 0$

1. Optimisation
- ▷ Trouver $x_k \in \text{argmax}_x\, u(x|D_k)$ à l'aide de la distribution *a priori*
- ▷ Evaluer l'objectif $y_k = f(x_k)$

2. Mise à jour
- ▷ $D_{k+1} = \{D_k, (x_k, y_k)\}$
- ▷ Mettre à jour

s'agit de la meilleure solution connue par la méthode. Un ensemble de points d'essais est évalué. Ensuite, des actions sont prises en fonction des évaluations de ces points ; si un de ces points améliore la solution courante, celle-ci est mise à jour, sinon un paramètre de taille de pas est réduit et un nouvel ensemble de points d'essais est généré. Il faut noter qu'aucun modèle de la fonction n'est construit pour générer l'ensemble de points à évaluer, ni aucune approximation des dérivées. Cette section présente plusieurs méthodes de recherche directe allant des plus simples, comme la recherche par coordonnées, jusqu'à des méthodes plus évoluées comme la recherche par treillis adaptatif.

3.3.1 Recherche par coordonnées

Une première méthode de recherche directe assez simple est la recherche par coordonnées [32]. La solution courante à l'itération k est notée x^k. Chaque itération est composée d'une étape de sonde, une étape lors de laquelle les points $x^k \pm \delta^k e_i$ sont évalués, où e_i est le vecteur de $\mathbb{R}^n$ composé uniquement de 0 et d'un 1 à la i^e composante. Si un point améliorant la solution courante est trouvé, celui-ci devient la nouvelle solution courante et l'itération est considérée comme un succès ; sinon le paramètre de pas δ^k est diminué et une nouvelle sonde est effectuée. Certaines variantes de cet algorithme existent. Par exemple, le paramètre de pas pourrait être augmenté lors des succès ou la sonde pourrait être interrompue dès qu'un point améliorant la solution est trouvé. Cette dernière variante est appelée stratégie opportuniste. Lorsque cette stratégie est appliquée, l'ordre d'évaluation des points de l'ensemble de sonde a une importance non négligeable. La recherche par coordonnées est décrite à l'algorithme 6.

Pour la recherche par coordonnées, il existe un résultat de convergence assez faible. Celui-ci est décrit au théorème 3.1.

Théorème 3.1 (Convergence de la recherche par coordonnées) *Soient $f : \mathbb{R}^n \mapsto \mathbb{R}$ une fonction $\mathcal{C}^0$ avec des ensembles de niveau bornés et $\{x^k\}$ la suite produite par la méthode*

Algorithme 6 Recherche par coordonnées

Données : objectif $f : \mathbb{R}^n \mapsto \mathbb{R}$ et un point de départ $x^0 \in \mathbb{R}^n$

0. Initialisation

$\delta^0 \in (0; \infty)$	Taille de pas initiale
$\epsilon_{stop} \in [0; \infty)$	Tolérance d'arrêt
$k \leftarrow 0$	Compteur d'itération

1. Sonde

▷ Si $f(t) < f(x^k)$ pour un certain $t \in P^k := \left\{x^k \pm \delta^k e_i : i = 1, 2, \ldots, n\right\}$

  ▷ $x^{k+1} \leftarrow t$ et $\delta^{k+1} \leftarrow \delta^k$

▷ Sinon

  ▷ $x^{k+1} \leftarrow x^k$ et $\delta^{k+1} \leftarrow \frac{\delta^k}{2}$

2. Terminaison

▷ Si $\delta^{k+1} \geq \epsilon_{stop}$

  ▷ Incrémenter $k \leftarrow k + 1$ et aller à 1

▷ Sinon

  ▷ Stop

de recherche par coordonnées avec $\epsilon_{stop} = 0$. Soit $\hat{x}$ le point d'accumulation des itérations échec de $\{x^k\}$.

Alors, pour tout $d \in \{\pm e_i : i = 1, 2, \ldots, n\}$, soit $f'(\hat{x}; d) \geq 0$, soit $f'(\hat{x}; d)$ n'existe pas. De plus, si $f \in \mathcal{C}^1$, alors $\hat{x}$ est un point critique de f, c'est-à-dire $\nabla f(\hat{x}) = 0$.

Ce résultat est intéressant mais relativement faible car il ne donne des garanties uniquement sur les dérivées dans les directions e_i.

Prenons par exemple la fonction convexe $f : \mathbb{R}^2 \mapsto \mathbb{R}, f(x) = \|x\|_\infty = \max(|x_1|, |x_2|)$, et le point $x^0 = [1, 1]^\top$. Quelle que soit la valeur de δ, $x^0 \pm e_i$ (pour $i = 1, 2$) n'améliore pas la solution courante $f(x^0) = 1$. En effet, les directions e_i ($i = 1, 2$) sont des directions de montée tandis que les directions $-e_i$ sont parallèles aux ensembles de niveau. La recherche par coordonnées ne peut donc pas trouver le minimum de la fonction, à savoir $[0, 0]^\top$. Le paramètre δ diminue à chaque nouvelle itération car celles-ci sont toujours des échecs.

La suite de cette section présente les méthodes de recherche par motif généralisée et de recherche directe par treillis adaptatif. Ces méthodes ont une structure similaire à la recherche par coordonnées mais possèdent des résultats de convergence plus intéressants.

3.3.2 Recherche par motif généralisée

La *recherche par motif généralisée* (Gps) [67] est une amélioration de la recherche par coordonnées. Concrètement, elle propose d'autres directions de recherche que celles alignées sur les axes, ainsi que la possibilité pour ces directions de changer en fonction des itérations. En plus de l'étape de sonde, qui est une recherche locale, la recherche par motif généralisée propose une étape de recherche plus globale appelée simplement étape de recherche. Cette recherche consiste à évaluer un nombre fini de points dans l'espace de recherche. Si l'étape de recherche génère un point améliorant la solution courante, il s'agit d'un succès, l'étape de sonde peut être omise. L'étape de sonde de Gps est assez similaire à l'étape de sonde de la recherche par coordonnées. Un ensemble de points autour de la solution courante est généré et évalué en espérant trouver une meilleure solution. La différence porte sur la façon de générer cet ensemble de sonde.

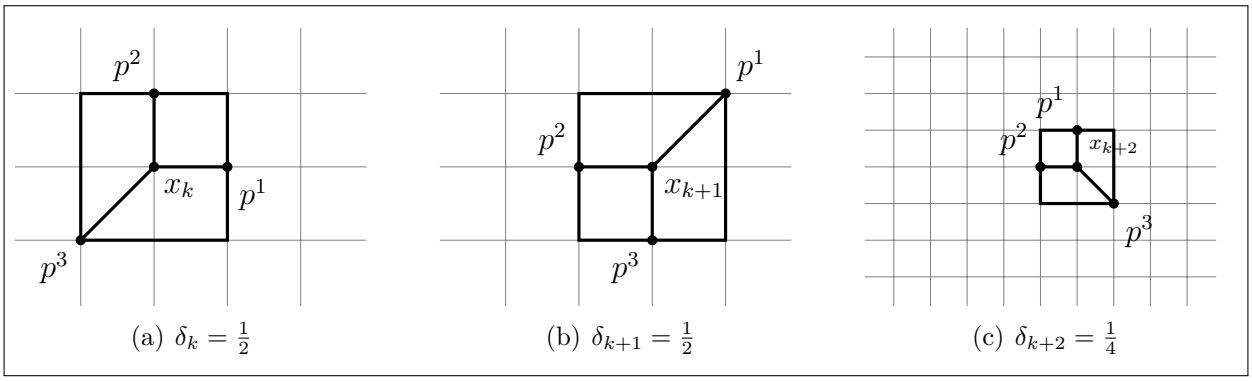
Pour assurer des résultats de convergence plus intéressants, il faut contrôler les étapes de sonde et de recherche. Cela est fait au travers d'un treillis, défini à la définition 3.1.

Définition 3.1 (Treillis ([10, chapitre 7], notre traduction) *Soit $G \in \mathbb{R}^{n\times n}$ une matrice inversible et $Z \in \mathbb{Z}^{n\times p}$ telle que les colonnes de Z forment un ensemble générateur positif de $\mathbb{R}^n$. On définit $D = GZ$. Le* treillis *généré par D centré sur la solution courante $x^k \in \mathbb{R}^n$ de taille de maille $\delta^k > 0$ est défini par*

$$M^k := \left\{x^k + \delta^k Dy : y \in \mathbb{N}^p\right\} \subset \mathbb{R}^n.$$

Puisque les colonnes de la matrice $Z \in \mathbb{Z}^{n\times p}$ forment un ensemble générateur positif et que la matrice G est inversible, les colonnes de la matrice D forment également un ensemble générateur positif. D est appelée une matrice générateur positif et l'ensemble de directions composé des colonnes de D est noté $\mathbb{D}$.

Tous les points évalués par l'algorithme doivent appartenir au treillis. Il faut noter que le treillis est un concept mais ne doit pas être construit en tant que tel au cours de l'exécution de l'algorithme. Une illustration du treillis lors de trois sondes successives est présentée à la figure 3.1. La solution courante et les points de sonde sont représentés par le symbole "•". Le cadre noir retient l'ensemble des points du treillis qui peuvent être évalués lors de la recherche locale. Lors de la première sonde sur la figure 3.1(a), le point p^1 améliore la solution. C'est pourquoi le cadre s'est déplacé sur la figure 3.1(b). Lors de la deuxième sonde, aucun point améliorant la solution n'est trouvé et le treillis est raffiné, comme cela est représenté sur la figure 3.1(c). La méthode de recherche par motif généralisée est décrite à l'algorithme 7.



(a) $\delta_k = \frac{1}{2}$ (b) $\delta_{k+1} = \frac{1}{2}$ (c) $\delta_{k+2} = \frac{1}{4}$

Figure 3.1 Représentation du treillis et cadre de sonde de GPS

Algorithme 7 Recherche par motif généralisée

Données : objectif $f : \mathbb{R}^n \mapsto \mathbb{R}$ et un point de départ $x^0 \in \mathbb{R}^n$

0. Initialisation

$\delta^0 \in (0; \infty)$	Paramètre de treillis intial
$D = GZ$	Matrice générateur positif
$\tau \in (0; 1), rationnel$	Paramètre d'ajustement du treillis
$\epsilon_{stop} \in [0; \infty)$	Tolérance d'arrêt
$k \leftarrow 0$	Compteur d'itération

1. Recherche

▷ Si $f(t) < f(x^k)$ pour un certain t d'un sous-ensemble fini du treillis M^k

  ▷ $x^{k+1} \leftarrow x^k$, $\delta^{k+1} \leftarrow \tau^{-1}\delta^k$, et aller à 3

▷ Sinon

  ▷ Aller à 2

2. Sonde

▷ Choisir un ensemble générateur positif $\mathbb{D}^k \subseteq \mathbb{D}$

▷ Si $f(t) < f(x^k)$ pour un certain $t \in P^k := \left\{x^k \pm \delta^k d : d \in \mathbb{D}^k\right\}$

  ▷ Alors $x^{k+1} \leftarrow t$ et $\delta^{k+1} \leftarrow \tau^{-1}\delta^k$

▷ Sinon x^k est un minimum local du treillis

  ▷ $x^{k+1} \leftarrow x^k$ et $\delta^{k+1} \leftarrow \tau\delta^k$

3. Terminaison

▷ Si $\delta^{k+1} \geq \epsilon_{stop}$

  ▷ incrémenter $k \leftarrow k + 1$ et aller à 1

▷ Sinon

  ▷ Stop

Pour analyser la convergence de cette méthode, il faut s'intéresser au comportement de l'algorithme lorsque le nombre d'itérations tend vers l'infini. La convergence de cette méthode

repose sur les arguments suivants. Supposons que l'ensemble de niveau $\{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ de la fonction à minimiser forme un ensemble compact. Alors, le nombre d'itérations où une meilleure solution courante est trouvée, un succès, est fini. En effet, puisque lors d'un succès, la fonction objectif est strictement réduite et que le treillis est une structure discrète, il n'y a qu'un nombre fini d'itérations formant un succès. Lorsque le nombre d'itérations tend vers l'infini, on peut en conclure que le nombre d'itérations formant des échecs tend également vers l'infini. Puisque la taille du maillage δ_k est réduite à chaque itération infructueuse, il est possible d'en déduire une limite quant à cette taille de maillage du treillis,

$$\lim_{k\to\infty} \inf \delta^k = 0. \tag{3.1}$$

De plus, il existe une sous-suite d'itérations infructueuses $\{k_i\}$ et un point x^* tels que

$$\lim_{i\to\infty} \delta^{k_i} = 0, \quad \lim_{i\to\infty} x^{k_i} = x^*.$$

Ceci vient du fait qu'il y a une infinité d'itérations où l'étape de sonde échoue et que la taille du maillage est réduite uniquement lors de telles itérations. Notons K_e^1 la sous-suite d'itérations échecs. Si la limite (3.1) est satisfaite, il existe une sous-suite $K_e^2 \subset K_e^1$ tel que δ_k tend vers 0, pour tout $k \in K_e^2$. La suite $\{x_k\}_{K_e^2}$ est donc bornée et contient une sous-suite convergente $\{x_k\}_{K_e^3}$. On note $x^* = \lim_{k\in K_e^3} x_k$.

Cela implique que l'algorithme raffine le maillage du treillis dans le voisinage de la solution courante. En supposant que la fonction objectif est localement Lipschiptz autour du point x^*, on peut conclure d'un résultat quant aux dérivées directionnelles de Clarke [22],

$$f^\circ(x^*; d) \geq 0, \quad \forall d \in \mathbb{D}^*, \tag{3.2}$$

où $\mathbb{D}^*$ est un ensemble générateur positif de D et

$$f^\circ(x; d) = \limsup_{y\to x, t\downarrow 0} \frac{f(y + td) - f(y)}{t}.$$

En imposant que la fonction objectif est strictement dérivable au point x^*, on obtient

$$\nabla f(x^*) = 0.$$

Ce résultat de convergence est plus général que celui de la recherche par coordonnées mais souffre d'un défaut majeur. La matrice générateur positif D est de dimension finie. Il existe donc un ensemble fini de générateurs positifs et donc de directions explorées autour de la

solution x^*.

Une étude plus complète de la convergence de Gps peut être trouvée dans [7, 10, 24, chapitre 7].

3.3.3 Recherche directe sur treillis adaptatif

Bien que la Gps généralise la recherche par coordonnées en sondant dans une plus grande variété de directions, les résultats de convergence ne sont pas encore à la hauteur de nos espérances. En effet, l'initialisation de Gps requiert une matrice générateur positif. Toutes les directions de sonde vont être sélectionnées parmi les colonnes de cette matrice, ce qui implique un nombre fini de directions de sonde dans Gps.

Pour généraliser la recherche par motif généralisée, la recherche directe par treillis adaptatif (Mads) [8] propose un moyen d'avoir une infinité de directions de sonde possibles. Cela permet de contrecarrer les limites de convergence de Gps. Un autre avantage de Mads est sa gestion des contraintes notamment via la barrière progressive. La gestion des contraintes n'est pas au cœur de ce projet mais cela reste tout de même un avantage de cet algorithme.

Contrairement à Gps, la recherche directe par treillis adaptatif possède un paramètre pour la taille du treillis δ^k et un paramètre pour la taille du cadre de sonde Δ^k. Celui-ci n'était pas nécessaire dans Gps car cette dernière peut être vue comme un cas particulier de Mads où ces deux paramètres ont la même valeur.

Mads introduit donc les notions de cadre de sonde et de taille du cadre de sonde reprises dans la définition 3.2.

Définition 3.2 (Cadre de sonde [10, section 8.1], notre traduction) *Soit $G \in \mathbb{R}^{n\times n}$ une matrice inversible et $Z \in \mathbb{Z}^{n\times p}$ telle que les colonnes de Z forment un ensemble générateur positif de $\mathbb{R}^n$. On définit $D = GZ$. On choisit un paramètre de taille de treillis $\delta^k > 0$ et on définit Δ^k tel que $\delta^k \leq \Delta^k$. Le* cadre *de taille Δ^k généré par D centré sur la solution courante $x^k \in \mathbb{R}^n$ est défini par*

$$F^k := \left\{x \in M^k : \|x - x^k\|_\infty \leq \Delta^k b\right\}$$

avec $b = \max\{\|d'\|_\infty : d' \in \mathbb{D}\}$ et où Δ^k est appelé paramètre de taille du cadre.

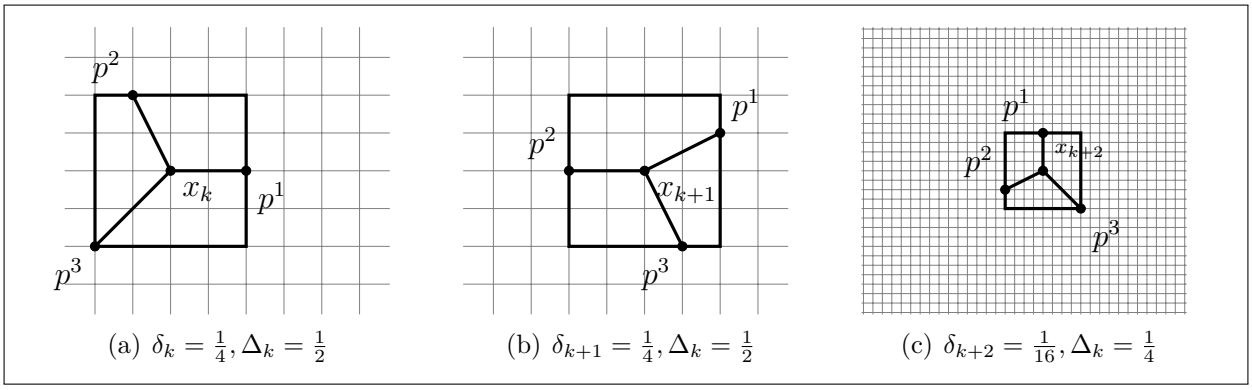
L'ensemble de sonde de Gps est inclus dans le cadre de sonde de Mads, qui est lui-même est sous-ensemble du treillis, pour un treillis et une solution courante donnés. En réduisant plus rapidement le paramètre de taille du treillis que celui de taille du cadre, l'ensemble de directions de sonde pouvant être sélectionnées est plus grand, de manière à créer un ensemble

de directions de sonde dense dans la sphère unité. Les paramètres de taille du treillis et de sonde doivent respecter $0 < \delta^k \leq \Delta^k$ à chaque itération et

$$\lim_{k \in K} \delta_k = 0 \iff \lim_{k \in K} \Delta_k = 0, \quad \text{pour tout sous-ensemble d'indices } K.$$

Pour cela, la stratégie $\delta^k = \min\{\Delta^k, (\Delta^k)^2\}$ est proposée, bien que d'autres stratégies sont possibles [11].

L'augmentation du nombre de directions de sonde possibles est illustrée à la figure 3.2. Il s'agit d'une situation similaire à celle de la figure 3.1, mais adaptée à la situation de Mads. Il est possible d'y voir qu'il y a une augmentation de points du treillis dans le cadre lorsque le maillage est raffiné. En effet, lorsque le treillis est raffiné à la figure 3.2(c), on passe de $25 = (4+1)^2$ à $81 = (8+1)^2$ points du treillis qui appartiennent dans le cadre de sonde. Ceci illustre la densité croissante des directions de sonde lorsque Δ_k tend vers 0. La méthode de recherche directe par treillis adaptatif est présentée à l'algorithme 8.



(a) $\delta_k = \frac{1}{4}, \Delta_k = \frac{1}{2}$ (b) $\delta_{k+1} = \frac{1}{4}, \Delta_k = \frac{1}{2}$ (c) $\delta_{k+2} = \frac{1}{16}, \Delta_k = \frac{1}{4}$

Figure 3.2 Représentation du treillis et cadre de sonde de Mads

L'analyse de convergence de Mads est similaire à celle de Gps. La différence la plus importante repose sur le fait qu'il y a une infinité de directions de sonde possibles. De plus, la différentiation du cadre de sonde et du treillis permet d'avoir un ensemble de directions de sonde plus riche que Gps, et donc un résultat de convergence plus fort. En effet, puisque le paramètre de taille du treillis doit toujours être plus petit que le paramètre de taille du cadre de sonde, plus de directions peuvent être utilisées pour créer un ensemble générateur positif. Lorsque le nombre d'itérations tend vers l'infini, cet ensemble de directions de sonde possibles devient dense dans la sphère unité. L'analyse de convergence de Gps tient également pour Mads. Puisqu'il y a un plus grand ensemble de directions de sonde possible, le résultat de

Algorithme 8 Recherche directe par treillis adaptatif

Données : objectif $f : \mathbb{R}^n \mapsto \mathbb{R}$ et un point de départ $x^0 \in \mathbb{R}^n$

0. Initialisation

$\Delta^0 \in (0; \infty)$	Paramètre de cadre initial
$D = GZ$	Matrice générateur positif
$\tau \in (1; \infty), rationnel$	Paramètre d'ajustement du treillis
$\omega^+ \geq 0, \omega^- \leq -1$	Paramètres d'ajustement du treillis
$\epsilon_{stop} \in [0; \infty)$	Tolérance d'arrêt
$k \leftarrow 0$	Compteur d'itération

1. Mise à jour du paramètre

▷ Définir le paramètre de taille du treillis $\delta^k = \min\{\Delta^k, (\Delta^k)^2\}$

2. Recherche

▷ Si $f(t) < f(x^k)$ pour un certain t d'un sous-ensemble fini S^k du treillis M^k
  ▷ $x^{k+1} \leftarrow x^k$, $\Delta^{k+1} \leftarrow \tau^{\omega^+}\Delta^k$, et aller à 4

▷ Sinon
  ▷ Aller à 3

3. Sonde

▷ Choisir un ensemble générateur positif $\mathbb{D}^k_\Delta$ tel que $P^k = \{x^k + \delta^k d : d \in \mathbb{D}^k_\Delta\}$ soit un sous-ensemble du cadre F^k de taille Δ

▷ Si $f(t) < f(x^k)$ pour un certain $t \in P^k$
  ▷ Alors $x^{k+1} \leftarrow t$ et $\Delta^{k+1} \leftarrow \tau^{\omega^+}\Delta^k$

▷ Sinon x^k est un minimum local du treillis
  ▷ $x^{k+1} \leftarrow x^k$ et $\Delta^{k+1} \leftarrow \tau^{\omega^-}\Delta^k$

4. Terminaison

▷ Si $\Delta^{k+1} \geq \epsilon_{stop}$
  ▷ incrémenter $k \leftarrow k + 1$ et aller à 1

▷ Sinon
  ▷ Stop

convergence de Gps (3.2) peut être étendu pour Mads en

$$f^\circ(x^*; d) \geq 0, \quad \forall d \in \mathbb{R}^n.$$

Une analyse de convergence complète de Mads est reprise dans [10, 24, chapitre 7] ou [8]. La qualité de Mads dépend de la manière de générer les ensembles générateur positif aux étapes de sonde. Différentes implémentations sont présentées dans [1, 8, 45, 69].

Il est important de noter que l'analyse de convergence de Gps et de Mads se base essentiellement sur l'étape de sonde. L'étape de recherche doit uniquement satisfaire les deux conditions suivantes :

1. Evaluer un nombre fini de points,
2. Tous les points évalués doivent appartenir au treillis.

Le cadre algorithmique de MADS laisse donc de grandes libertés permettant d'intégrer des stratégies de recherche diverses et variées comme une recherche basée sur des modèles ou sur une heuristique, tant que celles-ci satisfont ces deux conditions.

Par exemple, il est possible d'intégrer la méthode de Nelder-Mead dans un algorithme MADS, comme présenté dans [13]. Une autre possibilité, présentée dans [23], est d'appliquer une étape de recherche basée sur des modèles quadratiques.

3.4 Optimisation sans dérivée en grande dimension

Cette section se concentre sur les méthodes d'optimisation sans dérivée qui s'appliquent plus spécifiquement aux problèmes en grande dimension. Une approche assez simple serait d'utiliser le parallélisme. Cela permet d'effectuer plusieurs évaluations de la boîte noire simultanément et limiter le temps d'exécution des algorithmes.

Ce travail s'intéresse plus particulièrement aux stratégies de réduction de dimension. Cela permet de résoudre des problèmes de grande de taille, sans exploiter le parallélisme. Par exemple, une adaptation de la méthode de descente basée sur des modèles en grande dimension est présentée dans [70]. Les algorithmes supportés par une analyse de convergence nous semblent préférables, c'est pourquoi des stratégies de réduction de dimension pour les algorithmes d'optimisation Bayesienne et MADS sont présentées aux sections 3.4.2 et 3.4.3 respectivement. Toutefois, une version de l'algorithme de Nelder-Mead adapté pour des problèmes de grande dimension est également décrite.

3.4.1 Algorithme de Nelder-Mead en grande dimension

Dans [49], Mehta explique que l'algorithme de Nelder-Mead, avec ses paramètres mis à leur valeur par défaut, a de mauvaises performances sur les problèmes en grande dimension. Les auteurs de [33] expliquent que les opérations d'expansion et de contraction ont une certaine propriété de descente mais que l'efficacité de celle-ci diminue lorsque la taille du problème augmente. Dans ce cas, l'algorithme est alors dominé par l'opération de réflexion. Plusieurs stratégies ont alors été proposées pour fixer les paramètres de l'algorithme. Celles-ci se basent principalement sur des paramètres qui s'adaptent à la taille du problème.

Les auteurs de [30] ont appliqué un algorithme génétique pour faire évoluer l'algorithme de Nelder-Mead tandis que dans [43], les résultats d'une analyse de sensibilité sont exploités

pour fixer ces paramètres. Dans [49], Mehta présente une manière de fixer ces paramètres en se basant sur les points de Chebyshev. Une autre approche pour améliorer les performances de l'algorithme est de perturber le centroïde aléatoirement ce qui améliore sa convergence, comme présenté dans [31]. La méthode de Nelder-Mead n'est pas fondamentalement transformée lors de ces adaptations, car celles-ci reposent plus sur le fait de trouver de bonnes valeurs pour les paramètres inhérents de la méthode.

3.4.2 Réduction de dimension en optimisation Bayesienne

En plus des versions exploitant le parallélisme, il existe également des méthodes d'optimisation Bayesienne appliquant des stratégies de réduction de dimension.

Celles-ci supposent que l'objectif a une dimension effective petite comparée à sa dimension. Concrètement, une fonction objectif $f : \mathbb{R}^n \mapsto \mathbb{R}$ possède une dimension effective $n_e \leq n$ s'il existe un sous-espace linéaire T de dimension n_e tel que pour tous vecteurs $x \in T$ et $x_\top \in T_\top$,

$$f(x + x_\top) = f(x).$$

La dimension effective n_e est la plus petite dimension satisfaisant cette égalité. On parle également de fonctions avec un sous-espace actif T.

Au lieu d'essayer de résoudre le problème original

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{sujet à} \quad & x \in \mathcal{X} \end{aligned}$$

où $\mathcal{X}$ désigne des bornes sur les variables, l'idée est de résoudre le problème

$$\begin{aligned} \min_{y \in \mathbb{R}^{n_e}} \quad & f(Ay) \\ \text{sujet à} \quad & y \in \mathcal{Y} \end{aligned}$$

où $A \in \mathbb{R}^{n \times n_e}$ et $\mathcal{Y}$ représente le nouvel espace réalisable.

La difficulté de cette stratégie vient du fait que la matrice A ainsi que la dimension effective ne sont pas connues. La matrice A est souvent choisie comme une matrice aléatoire [21, 54]. Le choix du domaine de petite dimension $\mathcal{Y}$ peut aussi avoir un impact sur les performances de l'algorithme [17].

3.4.3 Algorithme STATS-MADS

Comme précisé précédemment, le cadre algorithmique de MADS offre une certaine flexibilité quant à l'étape de recherche. En effet, celle-ci n'est pas indispensable et doit respecter certaines conditions pour pouvoir préserver l'analyse de convergence de la méthode.

L'algorithme STATS-MADS, décrit dans [2, 15], est une instance de MADS développée pour résoudre des problèmes d'optimisation sans dérivée de grande dimension. STATS-MADS se base sur l'idée d'identifier les variables les plus importantes et d'alterner entre une optimisation en n dimensions et une optimisation sur un sous-espace dans lequel certaines variables sont fixées.

L'ensemble des indices des n variables du problème est noté $I = \{1, 2, \ldots, n\}$. $J_l \subset I$ désigne un sous-ensemble de variables. Au cours de l'optimisation, lancer une instance de MADS sur J_l signifie que toutes les variables $j \subset I \setminus J_l$, les variables dont l'indice n'appartient pas à J_l, sont fixées à $\hat{x}_j$, la solution courante. La nouvelle instance de MADS est donc une instance en dimension $|J_l|$.

L'identification de ces variables prépondérantes se fait en appliquant une méthode d'analyse de sensibilité basée sur une analyse de variance. Pour chaque variable, le rapport de corrélation, décrit à la section 2.3.1, permet de mesurer la sensibilité de la boîte noire par rapport à cette variable. Ensuite, les variables sont classées par ordre croissant de sensibilité et les $\lceil pn \rceil$ premières sont fixées, où p est un paramètre fixé par l'utilisateur qui définit la proportion de variables devant être fixées. Deux autres paramètres doivent être définis ; n^I désigne le nombre maximal d'évaluations successives de la boîte noire dans l'espace entier et n^J désigne le nombre maximal d'évaluations successives dans le sous-espace. L'algorithme STATS-MADS est décrit à l'algorithme 9.

Les travaux présentés dans [2,15] montrent que l'algorithme STATS-MADS améliore effectivement les performances de MADS sur un ensemble de problèmes d'optimisation non contraints. Cette amélioration est principalement due à l'optimisation en petite dimension.

Une autre approche pour s'attaquer aux problèmes en grande dimension avec MADS est d'utiliser le parallélisme. De plus amples informations sur le parallélisme sont disponibles dans [6,9]. Dans la version parallèle de MADS, l'espace de recherche est divisé de telle manière à ce qu'un processeur n'ait qu'un petit nombre de variables à traiter. Une version de cet algorithme applique une méthode d'analyse de sensibilité pour déterminer l'attribution des variables aux processeurs disponibles ; cet algorithme est décrit dans [3, 4].

Algorithme 9 Algorithme STATS-MADS

0. Initialisation

$\Delta^0 \in (0; \infty)$	Paramètre de cadre initial
$x^0 \in \mathcal{X}$	point initial
n^I	nombre maximum d'évaluations successives dans l'espace entier
n^J	nombre maximum d'évaluations successives dans le sous-espace

1. Boucle : Pour $l = 0, 1, 2, \ldots,$

- ▷ Lancer une instance de MADS à partir de x^l avec taille de treillis initial Δ^l et budget n^I ;
- ▷ soit $\hat{x}^l$ la meilleure solution trouvée et Δ_{l+1} la taille de treillis final ;
- ▷ Calculer les indices de sensibilité et définir $J_l \subset I$;
- ▷ Lancer MADS sur J_l à partir de $\hat{x}^l$ avec taille de pas initial Δ_l et budget n^J ;
- ▷ soit x_{l+1} la meilleure solution trouvée ;

CHAPITRE 4 Algorithme PCA-MADS

Afin d'améliorer les performances de l'algorithme MADS en grande dimension, il est intéressant de développer une nouvelle stratégie pour l'étape de recherche. Celle-ci consiste à utiliser une méthode d'analyse de sensibilité afin de définir un nouveau problème d'optimisation de plus petite taille. Cela a déjà été fait dans certains travaux comme ceux décrits à la section 3.4. La méthode STATS-MADS identifie puis fixe, momentanément, les variables les moins influentes. A la différence de cette méthode, l'algorithme proposé cherche à fixer des combinaisons de variables. L'étape de recherche applique une analyse en composante principale, décrite à la section 2.6, pour construire un problème de plus petite dimension. Nous appelons cet algorithme PCA-MADS.

L'analyse et la construction du problème en petite dimension sont décrites dans les sections 4.1, 4.2 et 4.3. La section 4.4 décrit le nouvel algorithme proposé tandis que la section 4.6 reprend les paramètres principaux de celui-ci. L'influence de ses paramètres sera analysée au chapitre 5.

4.1 Analyse en composante principale dans PCA-MADS

L'analyse en composante principale est une méthode statistique qui permet, à partir de $(n+1)$ variables aléatoires corrélées, de définir $(n+1)$ variables non corrélées. Celles-ci sont des combinaisons linéaires des variables originales. En pratique, la première composante principale correspond à la direction qui reprend la plus grande variabilité des variables aléatoires. La deuxième composante principale est la direction qui reprend le plus de variabilité tout en étant orthogonale à la première, et ainsi de suite. L'optimisation de boîte noire a, en général, comme facteur critique le nombre d'évaluations du problème. C'est pourquoi l'analyse de sensibilité de la boîte noire se fait à partir de l'ensemble des évaluations déjà effectuées, contenues dans la *cache*, pour déterminer des directions intéressantes.

Supposons donc que l'on dispose de $(n+1)$ variables aléatoires, notées X, qui correspondent aux n variables du problème d'optimisation et à la valeur correspondante à la fonction objectif. A partir des évaluations déjà effectuées, il est possible d'estimer une matrice de covariance et d'effectuer une analyse en composante principale à partir de cette matrice. Cette analyse permet de fournir $(n+1)$ directions ; celles-ci seront écrites dans une matrice $M_+ \in \mathbb{R}^{(n+1)\times(n+1)}$. Les nouvelles variables non corrélées $\bar{X} \in \mathbb{R}^{n+1}$ sont obtenues grâce à la

transformation des variables originales X et les directions obtenues par l'analyse :

$$\bar{X} = M_+^\top X.$$

L'analyse en composante principale est utilisée pour repérer des corrélations entre des variables aléatoires à partir d'un nuage de points. Or, le but de cette analyse est de déterminer des directions en dimension n selon lesquelles la fonction objectif a le plus de variabilité. Les directions issues de l'analyse en composante principale seront ordonnées en fonction de leur alignement avec la dimension de l'objectif. De cette manière, les premières directions sont celles qui influent le plus sur la fonction objectif.

Soit $M \in \mathbb{R}^{n\times n}$ une matrice qui correspond aux n premières lignes des n premières colonnes de la matrice M_+ réordonnée. Celles-ci donnent les directions de $\mathbb{R}^n$ qui ont le plus d'influence sur l'objectif. La direction ayant le moins d'impact sur l'objectif est négligée afin d'obtenir une matrice carrée. Il existe donc une transformation qui permet de transformer un vecteur $x \in \mathbb{R}^n$ en un vecteur $\bar{x} \in \mathbb{R}^n$,

$$\bar{x} = M^\top x \tag{4.1}$$

où les premières variables ont plus d'influence sur l'objectif que les dernières.

L'algorithme proposé cherche à réduire la dimension de l'espace de recherche en ne gardant que des dimensions qui ont une grande influence sur l'objectif. La matrice M peut être divisée en deux matrices de projection $P \in \mathbb{R}^{n\times p}$ et $Q \in \mathbb{R}^{n\times(n-p)}$,

$$M = [P \quad Q]\,. \tag{4.2}$$

La matrice P contient les p directions qui ont le plus d'impact sur l'objectif ; la matrice Q, quant à elle, contient les $(n-p)$ directions qui ont peu ou pas d'influence sur la fonction objectif. D'une manière similaire à (4.1), il est possible de définir les vecteurs $y \in \mathbb{R}^p$ et $z \in \mathbb{R}^{n-p}$ comme

$$\bar{x} = \begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} P^\top \\ Q^\top \end{bmatrix} x \tag{4.3}$$

Exemple 4.1 *Par exemple, considérons la fonction* $f_1(x_1, x_2) = 10x_1 + x_1x_2$ *et le nuage de points en deux dimensions* $\{(-5,-5);(-5,5);(5,-5);(5,5);(0,5);(5,0);(1,5);(5,1);(-1,5);(-5,1)\}$. *L'ensemble des points évalués est repris dans le tableau 4.1. A partir de ce nuage*

de points, nous pouvons construire une matrice de covariance S,

$$S = \begin{bmatrix} 19.389 & -0.944 & 200.556 \\ -0.944 & 16.456 & -9.444 \\ 200.556 & -9.444 & 2361.111 \end{bmatrix}.$$

Les vecteurs propres de cette matrice de covariance sont les composantes principales du nuage de points. Ceux-ci sont les colonnes de la matrice M_+ ci-dessous,

$$M_+ = \begin{bmatrix} -0.9964 & -0.0096 & 0.0847 \\ -0.0100 & 0.9999 & -0.0040 \\ 0.0847 & 0.0049 & 0.9964 \end{bmatrix}.$$

Dans notre situation, nous cherchons à minimiser la fonction objectif, on s'intéresse donc aux composantes principales étant les plus alignées sur la direction de l'objectif, c'est-à-dire le vecteur ayant la plus grande $(n+1)^e$ composante en valeur absolue. En réordonnant les colonnes de la matrice M_+ selon leur valeur dans la $(n+1)^e$ ligne, par ordre décroissant, on obtient

$$M'_+ = \begin{bmatrix} 0.0847 & -0.9964 & -0.0096 \\ -0.0049 & -0.0100 & 0.9999 \\ 0.9964 & 0.0847 & 0.0049 \end{bmatrix}.$$

Puisque l'espace de recherche est un sous-ensemble de $\mathbb{R}^2$, la matrice M_+ est amputée de sa dernière ligne et de sa dernière colonne pour construire une matrice M',

$$M' = \begin{bmatrix} 0.0847 & -0.9964 \\ -0.0049 & -0.0100 \end{bmatrix}.$$

Les colonnes de cette matrice M' forment les directions de $\mathbb{R}^n$ qui nous intéressent. En normant ces directions, on obtient la matrice M,

$$M = \begin{bmatrix} 0.9983 & -0.9999 \\ -0.0578 & -0.0101 \end{bmatrix}.$$

Cette matrice M permet de définir les matrices de projection P et Q de l'équation (4.2)*, en fonction de la dimension p qui nous intéresse. Dans ce cas-ci, la direction* $(0.9983, -0.0578)^\top$ *indique que la variable x_1 a plus d'importance que la variable x_2 sur l'objectif* $f_1(x_1, x_2) = 10x_1 + x_1x_2$.

Exemple 4.2 *Considérons la fonction $f_2(x_1, x_2) = x_1+x_2$ et le même ensemble de points que l'exemple précédent. Suite à une procédure similaire à l'exemple précédent, les composantes principales du nuage de points sont*

$$M_+ = \begin{bmatrix} 0.5774 & 0.6729 & 0.4625 \\ 0.5774 & -0.7370 & 0.3515 \\ -0.5774 & -0.0641 & 0.8140 \end{bmatrix}.$$

En réordonnant les colonnes de la matrice en fonction de la dernière ligne, on obtient

$$M'_+ = \begin{bmatrix} 0.4625 & 0.5774 & 0.6729 \\ 0.3515 & 0.5774 & -0.7370 \\ 0.8140 & -0.5774 & -0.0641 \end{bmatrix}.$$

Ensuite, en tronquant la dernière ligne et la dernière colonne et en normant les vecteurs restant, il est possible d'écrire la matrice M,

$$M = \begin{bmatrix} 0.7962 & 0.7071 \\ 0.6051 & 0.7071 \end{bmatrix}.$$

Dans ce cas, les directions proposées sont assez proches de $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})^\top$*, ce qui correspond à la direction attendue, au vu de la fonction objectif $f_2(x_1, x_2) = x_1 + x_2$.*

4.2 Changement de variables

Le sous-problème d'optimisation est défini à partir des variables $y \in \mathbb{R}^p$. Celles-ci correspondent aux $p < n$ combinaisons linéaires des variables x qui ont le plus d'impact sur l'objectif. Puisque la dimension p est plus petite que la dimension n, plusieurs points en dimension n peuvent être projetés sur un même point y en dimension p.

Tableau 4.1 Ensemble d'évaluations pour les fonctions $f_1(x_1, x_2) = 10x_1 + x_1x_2$ et $f_2(x_1, x_2) = x_1 + x_2$

x	$f_1(x)$	$f_2(x)$
$(-5, -5)$	-25	-10
$(-5, 5)$	-75	0
$(5, -5)$	25	0
$(5, 5)$	75	10
$(0, 5)$	0	5
$(5, 0)$	50	5
$(1, 5)$	15	6
$(5, 1)$	55	6
$(-1, 5)$	-15	4
$(-5, 1)$	-55	-4

Lors de la recherche dans le sous-espace, les points sont de dimension p. A partir de ces points, il faut pouvoir retrouver les points correspondants en dimension n afin d'évaluer la boîte noire. On propose ici deux stratégies afin de pouvoir construire un point en dimension n correspondant à $y \in \mathbb{R}^p$. Les différences de performance entre ces deux stratégies sont étudiées à la section 5.5.5.

La première solution consiste à compléter les informations manquantes par des informations disponibles. Lorsqu'une instance de MADS est lancée sur le sous-problème, il faut lui fournir un point de départ. Celui-ci va être défini comme l'origine du sous-espace de recherche. En pratique, ce point de départ correspond au meilleur point trouvé jusqu'à présent, la solution courante à l'itération k, $x^k \in \mathbb{R}^n$. Au moment d'évaluer un point $y \in \mathbb{R}^p$, le point

$$x = x^k + Py \tag{4.4}$$

est construit. Puisque la matrice P est de dimension $n \times p$, le point $x^k + Py$ est bien de dimension n. De plus, lorsque $y = 0$, le point construit correspond effectivement à la solution courante x^k. Des illustrations de cette transformation sont présentées aux figures 4.1 et 4.2.

La deuxième méthode se base sur l'opérateur de la pseudo-inverse. Soit une matrice $A \in \mathbb{R}^{m \times n}$, avec $m \leq n$. La pseudo-inverse de Moore-Penrose [57] de A est une matrice $A^\dagger \in \mathbb{R}^{n \times n}$

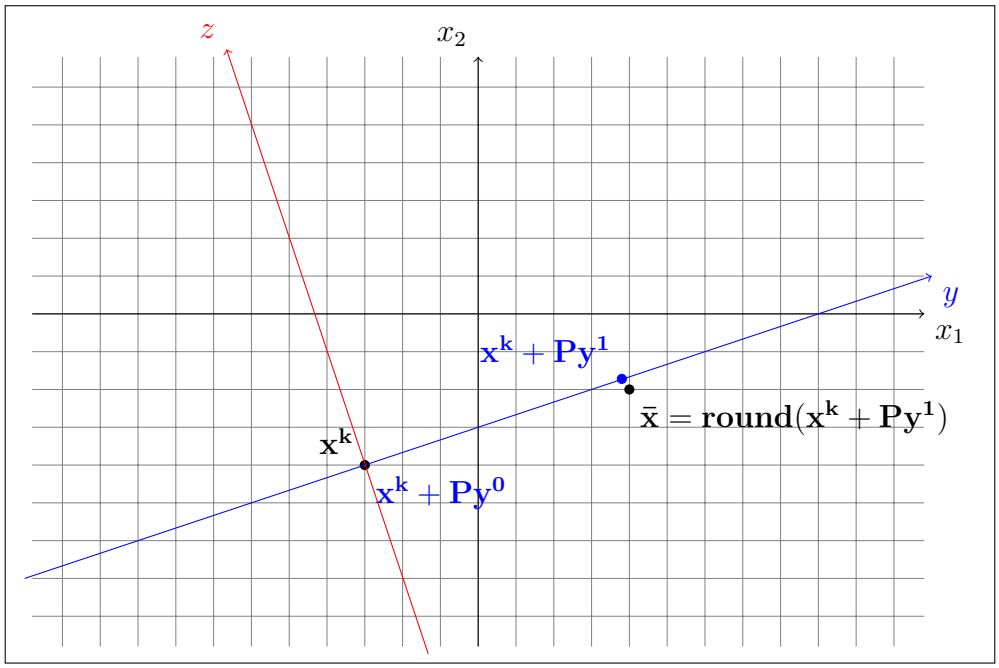


Figure 4.1 Illustration de la transformation (4.4) en deux dimensions

telle que

$$AA^\dagger A = A; \quad A^\dagger AA^\dagger = A^+; \quad (AA^\dagger)^\top = AA^\dagger; \quad (A^\dagger A)^\top = A^\dagger A.$$

Si A a des colonnes linéairement indépendantes, alors $A^\dagger = (A^\top A)^{-1}A^\top$. La pseudo-inverse de Moore-Penrose de la transposée de la matrice P est notée $(P^\top)^\dagger$. Si cette matrice est réelle et a des colonnes linéairement indépendantes, alors

$$(P^\top)^\dagger = (PP^\top)^{-1}P.$$

En multipliant la relation $y = P^\top x$, définie à l'équation (4.3), par la pseudo-inverse de $P^\top$, un point de dimension n est construit,

$$x = (P^\top)^\dagger y.$$

Cette matrice pseudo-inverse peut donc être utilisée pour passer d'un espace à l'autre. Afin que l'origine du sous-espace corresponde à la meilleure solution trouvée x^k, la transformation

$$x = x^k + (P^\top)^\dagger y \tag{4.5}$$

est privilégiée.

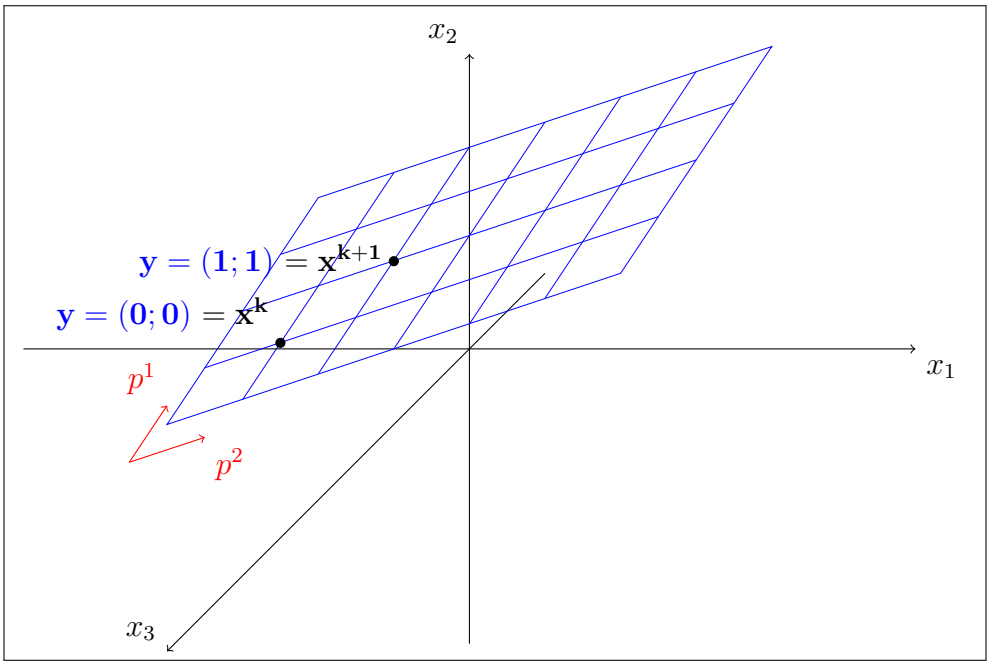


Figure 4.2 Illustration de la transformation (4.4) en trois dimensions. Les vecteur $p^1, p^2 \in \mathbb{R}^3$ sont issus de l'analyse en composante principale et définissent le nouvel espace de recherche de dimension 2. Dans cet espace, le point x^k correspond au point $(0,0)^\top$ de ce sous-espace et x^{k+1} au point $(1,1)^\top$

4.3 Evaluation de la boîte noire

Dans le cadre de l'algorithme MADS, tous les points sont évalués sur un treillis. Dans l'algorithme PCA-MADS proposé, des points en dimension p sont projetés sur des points en dimension n afin d'être évalués. Pour préserver les résultats de convergence de l'algorithme original, ceux-ci seront alors arrondis afin d'appartenir au treillis en n dimensions. Les points évalués n'appartiendront pas toujours au sous-espace défini et cela permettra de récupérer un peu d'information hors de l'espace de recherche.

Considérons que le treillis de l'algorithme MADS est défini par la matrice $D = [I_n - I_n]$ où I_n est la matrice identité en n dimensions. Soit Δ^m_{min} la valeur du plus petit paramètre de taille du treillis au cours de l'exécution de l'algorithme. Un point $x \in \mathbb{R}^n$ sera projeté sur le point

$$\bar{x} = x^k + \Delta^m_{min} \times round\left(\frac{x - x^k}{\Delta^m_{min}}\right), \tag{4.6}$$

où x^k est un point sur le treillis et l'opérateur $round(\cdot)$ est tel que $round(0.5) = 1$ et $round(-0.5) = -1$.

Lors de l'optimisation du problème en dimension p, un point $y \in \mathbb{R}^p$ est évalué de la façon suivante. Dans un premier temps, l'une des méthodes (4.4) ou (4.5) est utilisée pour en construire un nouveau. Cela permet de définir un point $x \in \mathbb{R}^n$ correspondant au point $y \in \mathbb{R}^p$ à évaluer. Ensuite, le point $x \in \mathbb{R}^n$ est projeté sur un point $\bar{x}$ du treillis de MADS en dimension n selon (4.6). La boîte noire est évaluée au point $\bar{x}$ et sa sortie est utilisée dans l'optimisation du sous-problème comme si cette valeur correspondait à l'évaluation de la boîte noire réduite au point $y \in \mathbb{R}^p$.

4.4 Algorithme PCA-MADS

Comme précisé à la section 3.3.3, le cadre algorithmique de MADS laisse la possibilité d'ajouter une étape de recherche qui peut évaluer un nombre fini de points sur le treillis. L'algorithme PCA-MADS propose d'optimiser un problème en dimension réduite comme étape de recherche. La construction de ce problème se base sur une analyse en composantes principales et évalue des points sur le treillis. En donnant un budget fini pour l'optimisation de ce problème en petite dimension et en évaluant la boîte noire sur le treillis, cette étape de recherche satisfait donc bien les conditions d'une étape de recherche dans MADS. Idéalement, l'étape de recherche en petite dimension devrait se poursuivre tant que celle-ci trouve des solutions plus intéressantes que celles déjà connues. Si la recherche n'améliore pas la solution courante, alors il faut effectuer une étape de sonde en grande dimension, selon l'algorithme MADS classique.

Une description de l'algorithme PCA-MADS est proposée à l'algorithme 10.

Algorithme 10 PCA-MADS

0. Initialisation
1. Evaluer N points sur le treillis
2. Boucle :

2.1 Définir un sous-problème de dimension p à partir des N derniers points évalués et d'une analyse en composante principale ;

2.2 Lancer une instance de MADS sur le sous-problème avec un budget B fini ;

2.3 Si aucun point améliorant la solution n'a été trouvé, effectuer une étape de sonde en n dimension ;

2.4 Evaluer les critères de terminaison de l'algorithme en dimension n et mettre à jour les paramètres en conséquence

4.5 Analyse de convergence

L'algorithme Pca-Mads proposé fait partie de la classe d'algorithmes Mads décrits à la section 3.3.3. Son analyse de convergence sera donc similaire à l'analyse de convergence de cette classe d'algorithmes.

A chaque itération de l'algorithme Pca-Mads, deux étapes successives peuvent être effectuées. La première est l'étape de recherche, qui consiste en la construction d'un sous-problème et l'optimisation de celui-ci avec un certain budget d'évaluations fini. La deuxième est l'étape de sonde qui n'est appliquée que si l'étape de recherche n'a pas généré de nouvelle solution améliorante. De plus, aussi bien à l'étape de recherche qu'à l'étape de sonde, tous les points évalués appartiennent à une structure finie appelée treillis, définie à la définition 3.1.

En appliquant les règles de mise à jour des paramètres de taille du treillis δ^k et de taille de cadre Δ^k,

$$\delta^k = \min(\Delta^k, (\Delta^k)^2), \quad \Delta^{k+1} = \tau^{\omega_k}\Delta^k,$$
$$\text{pour } \omega_k \in \begin{cases} \{0, 1, 2, \ldots, \omega^+\} & \text{si un point améliorant la solution est trouvé,} \\ \{\omega^-, \omega^- - 1, \ldots, -1\} & \text{sinon,} \end{cases}$$

le maillage du treillis est raffiné uniquement lors des itérations infructueuses, c'est-à-dire les itérations où aucun point améliorant la solution n'a été trouvé.

Pour analyser la convergence de l'algorithme, il faut s'intéresser à la situation où le nombre d'itérations tend vers l'infini. Prenons une fonction objectif dont l'ensemble de niveau $\{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ forme un compact. Puisque tous les points évalués appartiennent au treillis et que ce dernier est une structure finie, il n'y a qu'un nombre fini d'itérations réussies, c'est-à-dire une itération où une nouvelle solution améliorante a été générée. Donc, il y a un nombre infini d'itérations infructueuses. Le cadre de sonde sera raffiné infiniment ; le paramètre définissant sa taille suit donc la limite

$$\lim_{k\to\infty} \inf \Delta^k = 0;$$

et puisque $\delta^k \leq \Delta^k$,

$$\lim_{k\to\infty} \inf \delta^k = 0.$$

Par un raisonnement similaire que pour l'analyse de convergence de Gps à la section 3.3.2,

il existe une sous-suite d'itérations infructueuses $\{k_i\}$ et un point x^* tels

$$\lim_{i\to\infty} \delta^{k_i} = 0, \quad \lim_{i\to\infty} x^{k_i} = x^*.$$

Cela implique que l'algorithme raffine le treillis autour de la solution courante x^*. En supposant que la fonction est localement Lipschitz autour de x^*, on en conclut que

$$f^\circ(x^*; d) \geq 0, \quad \forall d \in \mathbb{R}^n.$$

Les résultats de convergence sont donc similaires à la classe d'algorithme MADS originale.

4.6 Paramètres

Le cadre algorithmique de MADS laisse des libertés quant à la valeur de certains paramètres, ainsi que la présence et les caractéristiques de l'étape de recherche. La variante proposée PCA-MADS profite notamment de ces libertés. Une liste des paramètres et des choix à faire pour un algorithme MADS, ainsi que ceux supplémentaires issus de la variante PCA-MADS est reprise ci-dessous.

Les paramètres et les choix d'implémentation pour l'algorithme MADS général sont les suivants.

1. Initialisation :
   — paramètre de taille de cadre initial Δ^0,
   — matrice générateur positif D,
   — paramètres d'ajustement du treillis τ, ω^+ et ω^-,
   — tolérance d'arrêt ϵ
2. Etape de recherche : nombre de points à évaluer et façon de les générer ;
3. Etape de sonde : nombre de directions ($(n+1)$ ou $2n$ directions) et façon de les générer.

Lors de l'initialisation, des valeurs des paramètres sont proposées dans [8]. Pour l'étape de sonde, plusieurs implémentations sont proposées dans [1, 8, 69].

Pour l'algorithme PCA-MADS, la façon de générer des points lors de l'étape de recherche repose sur une analyse en composante principale et une méthode d'optimisation. Les paramètres liés à cette étape sont repris dans la liste suivante :

— budget d'évaluations pour l'optimisation du sous-problème B ;
— choix de la dimension du sous-problème p ;

— nombre de points nécessaires pour effectuer l'étape de recherche N_{min}^{search} et la possibilité d'en générer pour compléter la *cache* ;
— nombre de points utilisés dans l'analyse en composante principale N_{pca}^{search} et la façon de les sélectionner ;
— méthode de construction du sous-problème à partir de l'analyse en composante principale et la façon d'évaluer la boîte noire ;
— paramètres inhérents à la méthode d'optimisation du sous-problème.

Une méthode de construction du sous-problème et d'évaluations de la boîte a été présentée aux sections précédentes.

CHAPITRE 5 Tests et résultats

Ce chapitre vise à comparer les performances de l'algorithme proposé Pca-Mads avec celles d'autres méthodes d'optimisation sans dérivée. Dans un premier temps, quelques tests exploratoires sont effectués sur des fonctions simples pour observer le comportement de l'algorithme Pca-Mads, ensuite celui-ci est lancé sur des problèmes plus difficiles. Les performances des algorithmes d'optimisation sans dérivée sont comparées à l'aide de profils de performance et de données. La construction de ces derniers est décrite à la section 5.1. La plateforme *COCO*, qui permet de fournir des suites de problèmes, est décrite à la section 5.2 et les résultats de tests effectués sur celle-ci sont présentés à la section 5.3. Par la suite, le comportement de l'algorithme Pca-Mads en petite et grande dimension est étudié à la section 5.4. Dans la section 5.5, l'influence des principaux paramètres de l'algorithme est étudiée. La section 5.6 compare Pca-Mads avec d'autres algorithmes d'optimisation, sur une suite de tests et sur des problèmes issus de la littérature.

5.1 Profils de performances et profils de données

Les performances de différents algorithmes peuvent être comparées au moyen de profils de performances et de données. Ceux-ci ont été introduits par Dolan, Moré et Wild [29, 51] et sont construits de la façon suivante.

Considérons un ensemble de problèmes $\mathcal{P}$, un ensemble d'algorithmes ou méthodes $\mathcal{S}$ et une mesure de performance $t_{p,s}$. Cette dernière est choisie comme le nombre d'évaluations nécessaires pour satisfaire un test de convergence,

$$f(x^0) - f(x^k) \geq (1 - \tau)(f(x^0) - f_L)$$

où x^i représente la i^e évaluation, τ est une tolérance et f_L la valeur de la meilleure solution trouvée pour un problème donné. A partir de cette mesure de performance, un ratio de performance $r_{p,s}$ est obtenu comme

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,a} : a \in \mathcal{S}\}}.$$

Le profil de performance d'un algorithme $s \in \mathcal{S}$ est la proportion de problèmes dont le ratio

de performance est au plus α, avec $\alpha \geq 1$:

$$\rho_s(\alpha) = \frac{|\{p \in \mathcal{P} : r_{p,s} \leq \alpha\}|}{|\mathcal{P}|}.$$

La valeur de $\rho_s(1)$ représente la proportion de problèmes pour lesquels l'algorithme s a trouvé la meilleure solution et la valeur de $\rho_s(\alpha)$ représente la proportion de problèmes pour lesquels l'algorithme s a satisfait le test de convergence en un nombre d'évaluations inférieur à α multiplié par le nombre d'évaluations nécessaires au meilleur algorithme pour trouver la solution d'un problème donné. Pour α suffisamment grand, $\rho_s(\alpha)$ donne la proportion de problèmes pour lesquels l'algorithme s a satisfait le critère de convergence.

Le profil de données d'un algorithme s est

$$d_s(\kappa) = \frac{\left|\left\{p \in \mathcal{P} : \frac{t_{p,s}}{n_p+1} \leq \kappa\right\}\right|}{|\mathcal{P}|},$$

où n_p est le nombre de variables du problème p. Le profil de données d'un algorithme représente la proportion de problèmes pour lesquels la méthode a satisfait le test de convergence avec au plus κ, $\kappa \geq 0$, groupes de $n_p + 1$ évaluations.

Ces profils donnent une performance relative de chaque algorithme par rapport aux autres, sur un ensemble de problèmes donnés. L'algorithme présentant la courbe au-dessus des autres a de meilleures performances, car il résout une plus grande proportion de problèmes que les autres pour un budget donné.

5.2 Plateforme *COCO* et suite de fonctions *bbob*

COCO [35] est une plateforme visant à comparer des méthodes d'optimisation de boîtes noires continues. Celle-ci fournit plusieurs suites de problèmes à minimiser, de différentes dimensions, et ils peuvent être bruités ou contraints. Les fonctions sont implémentées en C mais la plateforme offre des interfaces en C/C++, Java, Matlab/Octave ou Python. Le but de cette plateforme est d'automatiser la procédure de comparaison d'algorithme d'optimisation sans dérivée et de traitements des résultats.

Toutes ces suites se basent sur un ensemble de 24 fonctions définies dans [36]. Chacune des fonctions peut être utilisée pour créer plusieurs instances. Toutes les fonctions peuvent être définies en plusieurs dimensions et les variables sont bornées par $[-5; 5]^n$. L'ensemble de fonctions peut être divisé en plusieurs sous-ensembles avec certaines caractéristiques, comme cela est décrit dans [36]. Ceux-ci sont

— 5 fonctions séparables ;
— 4 fonctions avec un conditionnement faible à modéré ;
— 5 fonctions avec un haut conditionnement et unimodales ;
— 5 fonctions multimodales avec une structure globale adéquate ;
— 5 fonctions multimodales avec une faible structure globale.

De plus amples informations sur les fonctions sont disponibles dans [36]. Les solutions optimales sont connues pour chacun de ces problèmes. La plateforme crée plusieurs instances de problème à partir de chaque fonction et fournit les bornes et un point de départ pour chaque instance. Le budget d'évaluations total est défini par l'utilisateur via un facteur multipliant la dimension du problème.

La plateforme *COCO* traite également les résultats issus des expériences sur leur suite de fonctions. Elle utilise comme élément central pour comparer les différents algorithmes le nombre d'évaluations des fonctions pour atteindre un certain seuil, une certaine valeur de l'objectif. Nous préférons comparer les performances des algorithmes via des profils de performances et de données décrits à la section 5.1.

5.3 Tests sur la suite *COCO*

Les premiers tests sur la plateforme *COCO* ont pour but de comparer les performances de Pca-Mads et d'une version plus standard de Mads. Pour ce faire, une implémentation de Pca-Mads est comparée à la même implémentation avec l'étape de recherche désactivée. Cette dernière sera nommée simplement Mads. Les différences entre les deux méthodes comparées portent sur l'étape de recherche de Mads. Dans sa version basique, aucune étape de recherche n'est effectuée tandis que dans la version Pca-Mads, une étape de recherche basée sur une analyse en composante principale telle que décrite au chapitre 4 est effectuée.

Une étape importante de l'algorithme Mads ainsi que Pca-Mads est l'étape de sonde. Lors de celle-ci, la méthode génère des directions de sonde et plusieurs méthodes pour générer ces directions ont été décrites dans la littérature. La première, nommée LTMads, présentée dans [8], se base sur des matrices triangulaires inférieures pour générer les directions de sonde. Une deuxième méthode, nommée OrthoMads et décrite dans [1], génère des directions de sonde orthogonales. Un avantage de cette méthode est son déterminisme. Van Dyke et Asaki décrivent dans [69] une méthode permettant de générer des directions de sonde uniformément dans l'espace. Cette méthode se base sur une décomposition QR. Le choix de l'une ou l'autre de ces méthodes a été guidé par leur rapidité à implémenter et à s'exécuter. La méthode retenue est donc la méthode LTMads.

Les deux méthodes sont initialisées avec

- $\Delta^0 = 1$;
- $D = [I_n - I_n]$;
- $\tau = 2$, $\omega^- = -1$ et $\omega^+ = 1$;
- $\epsilon_{stop} = 10^{-15}$;
- Sonde de $2n$ points générés à la manière de LTMADS, avec stratégie opportuniste.

Pour l'étape de recherche de Pca-Mads, les paramètres suivants sont utilisés,

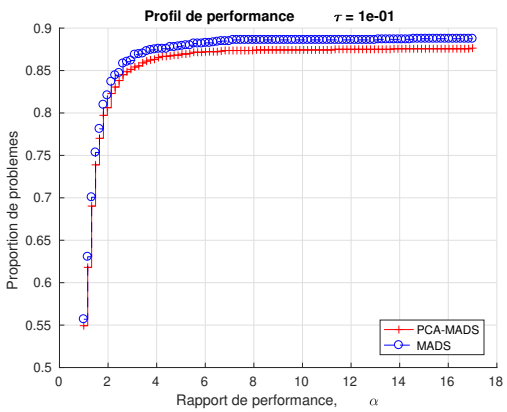
- dimension $p = \lceil \frac{n}{5} \rceil$;
- nombre d'évaluations minimum pour effectuer une recherche $N_{min}^{search} = 2n$, généré par échantillonnage par hypercube latin ;
- nombre de points utilisés dans l'analyse en composante principale N_{pca}^{search} est l'ensemble des points évalués ;
- budget d'évaluations de l'optimisation du sous-problème égal à un vingtième du budget total ;
- transformation $x = x^k + Py$ (4.4) ;
- critère d'arrêt pour l'optimisation du sous-problème $\epsilon_{stop} = \frac{\Delta^k}{2}$;
- les autres paramètres de la méthode Mads optimisant le sous-problème sont les mêmes que pour la méthode Mads du problème original en dimension n.

Cet ensemble de paramètres est utilisé pour tous les tests présentés dans la suite de ce chapitre, sauf si précisé autrement.

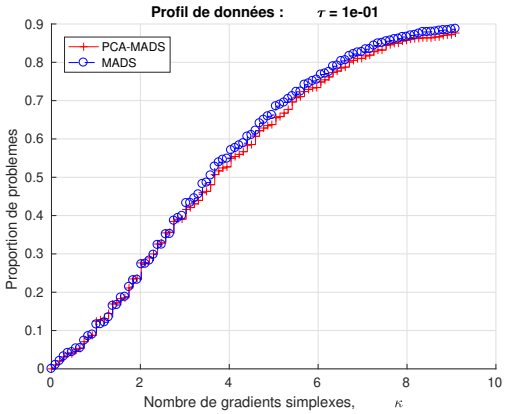
L'ensemble de problèmes sur lesquels ces algorithmes seront comparés vient de la plateforme *COCO*. La suite de fonctions *bbob* offre un ensemble de problèmes non bruités et non contraints. Cette suite est composée des 24 fonctions en dimension $2, 3, 5, 10$. Pour chaque fonction et dans chaque dimension, 16 instances de problèmes sont créées. Cela donne un total de 1440 problèmes. Le budget d'évaluations est fixé à $10n$ pour chaque algorithme, où n est la dimension du problème.

La figure 5.1 présente des profils de performance et des profils de données pour ces tests. Les méthodes Mads et Pca-Mads sont assez proches, avec un léger avantage à Mads tout de même. Ces tests ont été effectués sur une suite de fonctions de dimension petite et avec un budget assez restreint et ne permet pas de généraliser les résultats obtenus.

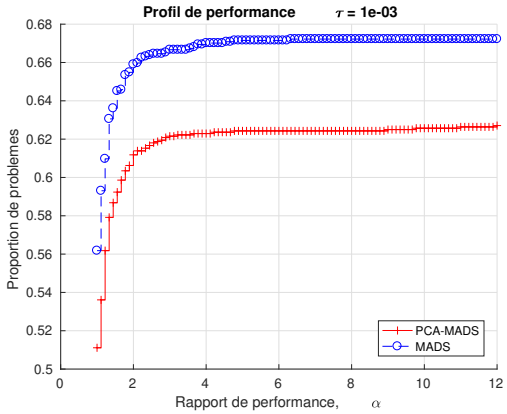
Sur cette figure, les profils de performance stagnent assez vite et n'apportent pas de nouvelles informations par rapport aux profils de données construits avec le même critère de convergence. Les prochains résultats seront donc présentés avec des profils de données uniquement si les profils de performance n'apportent pas d'information complémentaire.



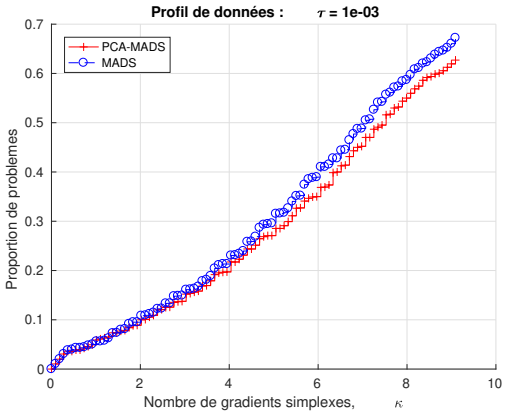
(a) Profil de performance pour une tolérance $\tau = 10^{-1}$



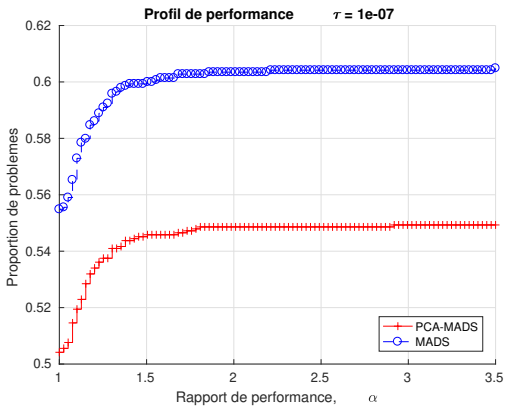
(b) Profil de données pour une tolérance $\tau = 10^{-1}$



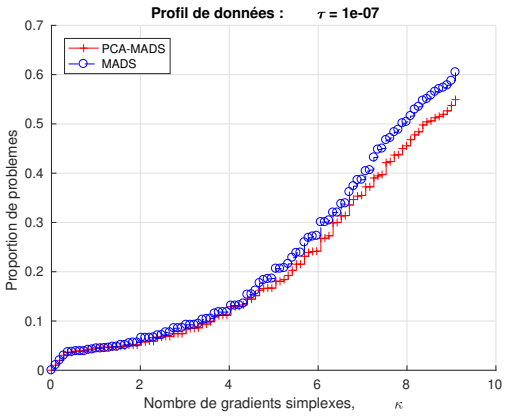
(c) Profil de performance pour une tolérance $\tau = 10^{-3}$



(d) Profil de données pour une tolérance $\tau = 10^{-3}$



(e) Profil de performance pour une tolérance $\tau = 10^{-7}$



(f) Profil de données pour une tolérance $\tau = 10^{-7}$

Figure 5.1 Comparaison de deux implémentations de MADS et de PCA-MADS similaires, sur la suite de fonctions *bbob* de *coco* avec un budget de $10n$

5.4 Comportement de PCA-MADS en petite et grande dimension

Les premiers tests exploratoires en grande dimension permettent de comparer les performances de l'algorithme Pca-Mads proposé avec une implémentation similaire de Mads sur des problèmes simples. Nous considérons la fonction de Rosenbrock à dimension variable suivante :

$$f(x) = \sum_{i=1}^{n-1} b(x_{i+1} - x_i^2)^2 + (a - x_i)^2, \tag{5.1}$$

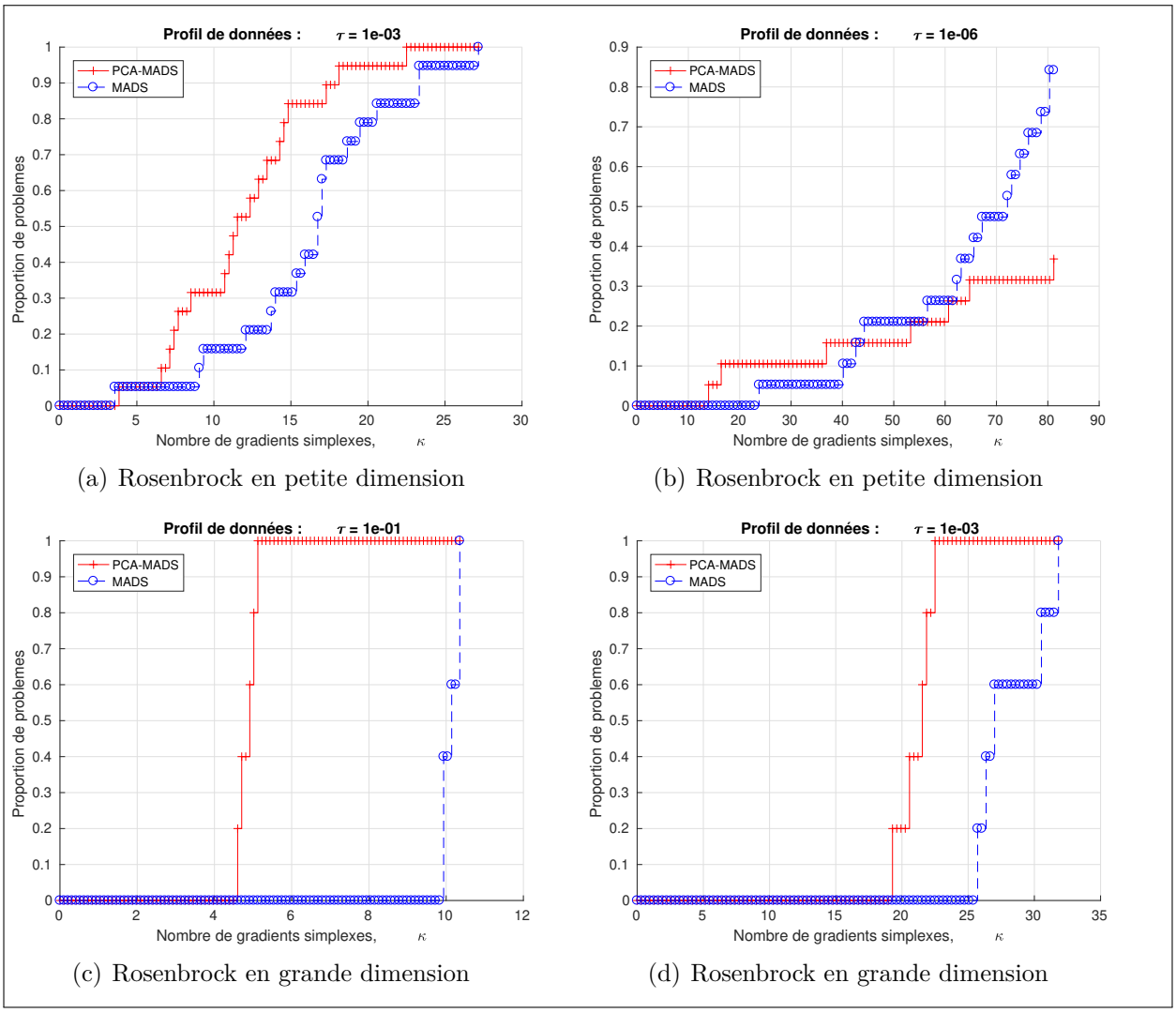
où $a = 1$ et $b = 100$. Pour ces tests, deux ensembles d'instances de la fonction de Rosenbrock sont considérés. Le premier ensemble reprend cette fonction avec $n = 2, 3, \ldots, 20$ variables, tandis que le second reprend la fonction de Rosenbrock à $n = 100, 200, \ldots, 500$ variables. Cela permet de mettre en évidence le comportement de l'algorithme aussi bien en petite qu'en grande dimension.

Ces tests comparent également une version simple de Mads et le nouvel algorithme Pca-Mads. Ces deux méthodes ont été initialisées de la même manière que les tests de la section précédente. Des profils de données ont été tracés pour une implémentation de Mads et de Pca-Mads. Ceux-ci sont représentés sur la figure 5.2.

Lorsque la tolérance du critère de convergence est assez faible, c'est-à-dire τ relativement grand, $\tau = 10^{-3}$ par exemple, l'algorithme Pca-Mads semble avoir de meilleures performances que Mads, comme l'indique la figure 5.2(a). Avec une tolérance plus petite, le comportement des algorithmes semble s'inverser. Cela est visible sur la figure 5.2(b). Cela indique que Mads a tendance à trouver de meilleures solutions que Pca-Mads en petite dimension, mais ce dernier trouve des solutions proches de la solution de Mads plus rapidement.

En regardant les profils pour la même fonction en plus grande dimension, visibles sur les figures 5.2(c) et 5.2(d), une situation différente est à noter. En effet, en plus grande dimension, l'algorithme Pca-Mads semble avoir de meilleures performances que l'implémentation Mads. Pour la plupart des problèmes, Pca-Mads trouve de meilleures solutions et plus rapidement que Mads. Cela indique que la stratégie de réduction de dimension semble être plus efficace pour des problèmes de grande dimension, aux alentours de quelques centaines de variables, par rapport aux problèmes en petite dimension, soit quelques dizaines de variables sur cette fonction particulière.

Le nombre et la variété des problèmes testés étant restreints, aucune conclusion définitive quant aux performances de chacun des algorithmes proposés n'est réellement possible. De plus, il y a de nombreux paramètres et stratégies qui peuvent jouer sur la performance des algorithmes.



(a) Rosenbrock en petite dimension

(b) Rosenbrock en petite dimension

(c) Rosenbrock en grande dimension

(d) Rosenbrock en grande dimension

Figure 5.2 Comparaison des méthodes MADS et PCA-MADS sur deux ensembles de fonctions de Rosenbrock en petites et grandes dimensions avec des budgets de $100n$ et $50n$ respectivement

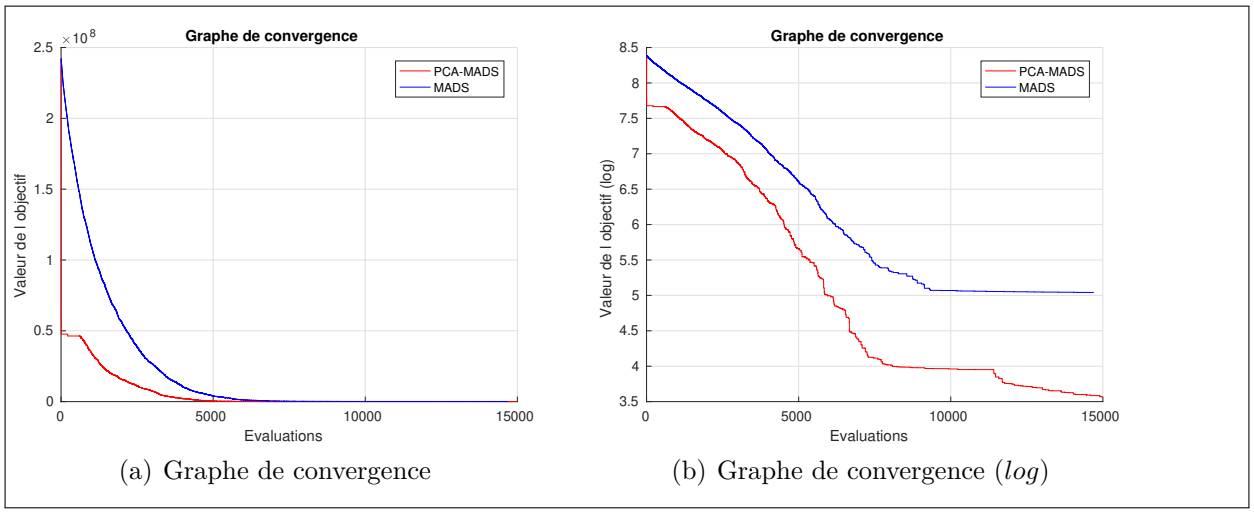
Il est possible de regarder plus en détails les différences de comportement de ces deux méthodes. La figure 5.3 présente des graphes de convergence pour les deux algorithmes sur une fonction de Rosenbrock en dimension 300. La figure 5.3(a) indique la valeur de l'objectif de la meilleure solution connue au cours de l'exécution de l'algorithme et la figure 5.3(b) donne le logarithme de cette valeur.

Ces figures permettent de mettre en avant un comportement particulier de PCA-MADS. Au tout début de l'exécution de la méthode, une diminution très rapide de l'objectif est observée. Ceci est probablement dû à la stratégie de remplissage de la *cache*. En effet, lors de la première itération de l'algorithme, la *cache* ne contient qu'un seul point, le point le départ. Il n'est donc

pas pertinent d'effectuer une analyse de sensibilité avec un seul point. Un échantillonnage par hypercube latin permet d'évaluer des points répartis dans l'espace de recherche afin de pouvoir appliquer une analyse de sensibilité à la deuxième itération.

Après cette phase de diminution très rapide vient une phase de diminution très faible. A ce moment-là, la stratégie de recherche n'est pas très efficace. Cela vient probablement du fait que les points contenus dans la *cache* ne permettent pas de récupérer des directions exploitables et du fait que le treillis est encore assez grossier. En effet, celui-ci est raffiné lors des itérations formant des échecs et au début de l'exécution, on n'observe pas ou peu d'itérations infructueuses. Lors de l'étape de recherche, les points à évaluer sont projetés sur le treillis ce qui se traduit par un nombre important de ré-évaluations de points déjà présents dans la *cache*.

Par la suite, la recherche semble devenir efficace et la diminution de l'objectif reprend. La figure 5.3(b) permet de voir que PCA-MADS trouve de meilleures solutions que MADS, et ce plus rapidement. Au total, il apparaît qu'environ une moitié des itérations bénéficient d'une étape de recherche améliorant la solution objectif.



(a) Graphe de convergence (b) Graphe de convergence (*log*)

Figure 5.3 Graphe de convergence des algorithmes MADS et PCA-MADS sur une fonction de Rosenbrock en dimension 300

5.5 Influence des paramètres

Il existe plusieurs paramètres et stratégies pouvant influencer les performances de PCA-MADS. Afin de pouvoir recommander des valeurs par défaut à un utilisateur de PCA-MADS, étudier l'influence de celles-ci est primordial. De plus, cela permettra de comparer l'algorithme PCA-MADS avec ses meilleures performances à d'autres méthodes connues. Pour ce faire, les

principaux paramètres de l'algorithme sont identifiés et plusieurs valeurs ou stratégies sont comparées au moyen de profils de données.

L'ensemble de fonctions utilisé pour ces tests est basé sur la suite de fonctions disponibles dans *COCO*. Puisque l'algorithme a été conçu pour résoudre des problèmes en relativement grande dimension, sans bruit et sans contrainte, la suite *bbob-largescale* de la plateforme *COCO* sera utilisée. Afin de limiter les temps d'exécution de ces tests, la première instance de 10 fonctions représentatives de la suite sera utilisée, et ce en dimension 80, 160, 320, 640. Cela donne un total de 40 problèmes. Nous avons sélectionné les fonctions $2, 4, 7, 9, 11, 13, 16, 18, 21$ et 23. De cette façon, deux fonctions avec chaque caractéristique se retrouvent dans l'ensemble de problèmes. Le budget d'évaluations est fixé à $50n$. A la suite de l'étude d'un paramètre, celui-ci est fixé à sa valeur la plus performante pour l'analyse de l'influence des paramètres suivants.

5.5.1 Dimension du sous-problème p

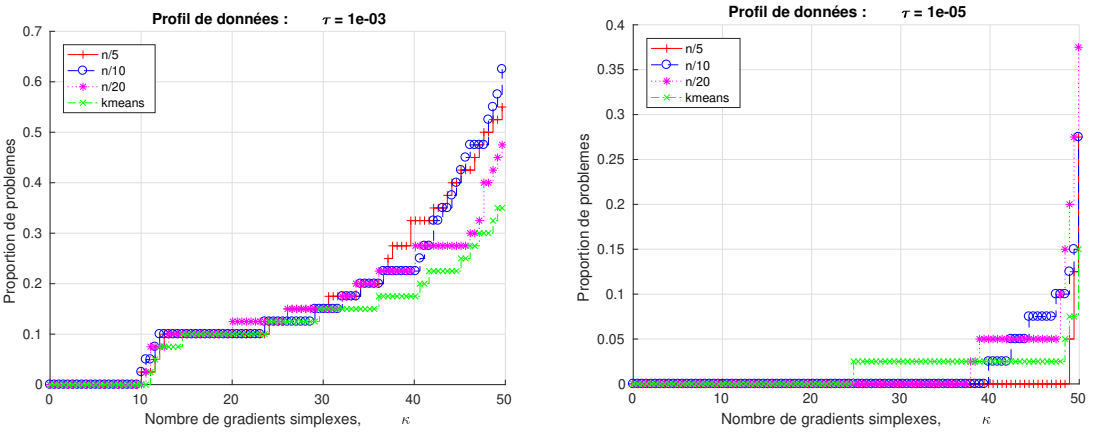
Au cours de l'étape de recherche, l'algorithme PCA-MADS construit un sous-problème de plus faible dimension et cherche à l'optimiser. La dimension de ce sous-problème semble être un paramètre critique, qui peut influencer les performances de l'algorithme. Plusieurs stratégies peuvent donc être comparées. La dimension du problème original est notée n et la dimension du sous-problème est notée p. Les premières stratégies consistent simplement à diviser la dimension du problème original par un facteur constant. Chaque sous-problème construit aura donc la même dimension. Une alternative à cette stratégie est d'utiliser un algorithme de classement pour séparer les directions issues de l'analyse en composante principale et de les diviser en deux groupes. L'algorithme $k-means$ est utilisé pour sa simplicité d'utilisation. En mettant $k = 2$, celui-ci permet de séparer les $(n+1)$ directions en deux groupes. La dimension du problème p correspondra à la taille de l'ensemble des directions ayant les directions les plus influentes sur l'objectif.

Toutes les stratégies comparées sont reprises dans la liste ci-dessous.

1. Stratégie $p = \lceil \frac{n}{5} \rceil$;
2. Stratégie $p = \lceil \frac{n}{10} \rceil$;
3. Stratégie $p = \lceil \frac{n}{20} \rceil$;
4. Stratégie $k - means$.

Les performances de l'algorithme, en fonction de ces stratégies, sont comparées à l'aide de profils de données tels que décrits à la section 5.1. Ceux-ci sont repris à la figure 5.4.

Lorsque le critère de convergence est relativement précis, par exemple $\tau = 10^{-3}$ sur le profil



(a) Profil de données pour une tolérance $\tau = 10^{-3}$

(b) Profil de données pour une tolérance $\tau = 10^{-5}$

Figure 5.4 Comparaison des performances de PCA-MADS avec différentes dimensions du sous-problème (paramètre p), sur une partie de la suite de fonctions *bbob-largescale* de *COCO*

5.4(a), deux stratégies se distinguent. Il s'agit des stratégies $p = \lceil \frac{n}{5} \rceil$ et $p = \lceil \frac{n}{10} \rceil$. Cela peut s'expliquer par les dimensions des problèmes utilisés pour la comparaison. En effet, avec la stratégie $p = \lceil \frac{n}{5} \rceil$ (resp. $p = \lceil \frac{n}{10} \rceil$), les sous-problèmes construits sont de taille 16 à 128 (resp. 8 à 64). Il s'agit de tailles pour lesquelles l'algorithme MADS a de bonnes performances. Lorsque le critère de convergence pour la création des profils est encore plus strict, avec $\tau = 10^{-5}$ par exemple, la stratégie $p = \lceil \frac{n}{20} \rceil$ est la meilleure des stratégies comparées. Sur le profil de performance 5.4(b), aucune des autres stratégies ne parvient à s'approcher de la stratégie $p = \lceil \frac{n}{20} \rceil$. La stratégie basée sur l'algorithme $k-means$ ne semble pas être un choix judicieux par rapport aux autres stratégies étudiées.

5.5.2 Budget d'évaluations pour l'optimisation du sous-problème

L'étape de recherche consiste à optimiser un nouveau problème d'optimisation de petite dimension avec un algorithme MADS. Le budget alloué à cette optimisation aura une influence sur la qualité de celle-ci. Dans le cadre de l'algorithme PCA-MADS, il y a une alternance entre cette minimisation et une étape de sonde en grande dimension. Le budget maximum d'évaluations pour l'optimisation de ce sous-problème devrait avoir une influence sur les performances de PCA-MADS. n étant la dimension du problème original, le budget total d'évaluations est noté B et le nombre d'évaluations déjà effectuées est noté ev.

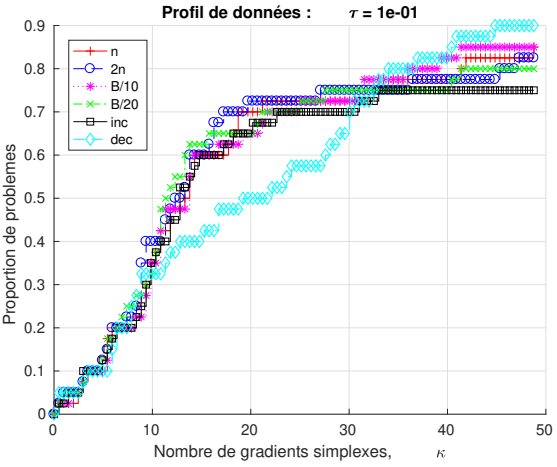
Les stratégies suivantes sont comparées :

1. Stratégie $\lceil \frac{B}{10} \rceil$: le budget maximum d'évaluations pour le sous-problème correspond

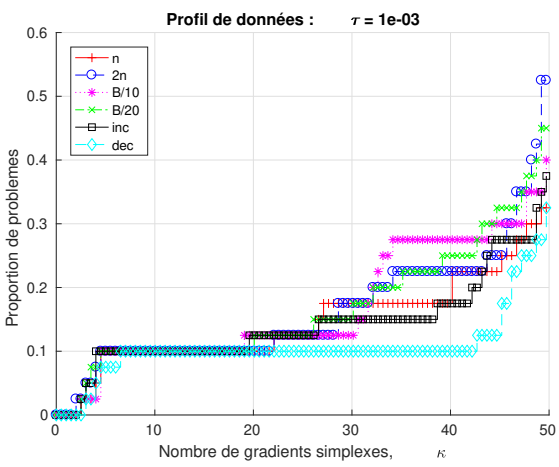
à un dixième du budget total d'évaluations.

2. Stratégie $\lceil \frac{B}{20} \rceil$: le budget maximum d'évaluations pour le sous-problème correspond à un vingtième du budget total d'évaluations.
3. Stratégie n : le budget maximum d'évaluations pour le sous-problème correspond au nombre de variables du problème original.
4. Stratégie $2n$: le budget maximum d'évaluations pour le sous-problème correspond à deux fois le nombre de variables du problème original.
5. Stratégie *inc* : à l'étape de recherche, on donne un budget de $\lceil (1 + \frac{5ev}{B})\frac{n}{2} \rceil$ évaluations. Cela donne un budget minimum de $\frac{n}{2}$ et maximum de $3n$ évaluations et le budget augmente à chaque nouvelle étape de recherche.
6. Stratégie *dec* : à l'étape de recherche, on donne un budget de $\lceil (1 - \frac{5ev}{6B})3n \rceil$ évaluations. Cela donne un budget minimum de $\frac{n}{2}$ et maximum de $3n$ évaluations et le budget diminue à chaque nouvelle étape de recherche.

Plusieurs profils de données ont été tracés pour comparer ces stratégies. Ceux-ci sont repris à la figure 5.5. Les profils avec une faible précision, visibles sur la figure 5.5(a), indiquent que la stratégie visant à diminuer le budget au fur et à mesure des itérations semble un peu plus intéressante que les autres, bien que ces dernières soient relativement proches. Lorsque la précision augmente, la stratégie $2n$ s'isole comme une meilleure stratégie que les autres. Cela semble donc être le choix le plus approprié parmi les stratégies proposées.



(a) Profil de données pour une tolérance $\tau = 10^{-1}$



(b) Profil de données pour une tolérance $\tau = 10^{-3}$

Figure 5.5 Comparaison des performances de Pca-Mads avec différents budgets d'évaluations pour l'optimisation du sous-problème, sur une partie de la suite de fonctions *bbob-largescale* de *COCO*

5.5.3 Ensemble de points utilisés pour l'analyse de sensibilité

L'algorithme Pca-Mads repose principalement sur la construction d'un sous-problème de dimension inférieure au problème original. La construction de ce sous-problème est donc une étape critique de l'algorithme. Cette construction se fait à l'aide d'une analyse en composante principale à partir d'un nuage de points. Les combinaisons de variables qui auraient le plus d'influence sur la valeur de l'objectif sont déduites du résultat de cette analyse. Le choix de l'ensemble de points utilisés pour l'analyse en composante principale peut donc avoir une certaine influence sur les performances de l'algorithme.

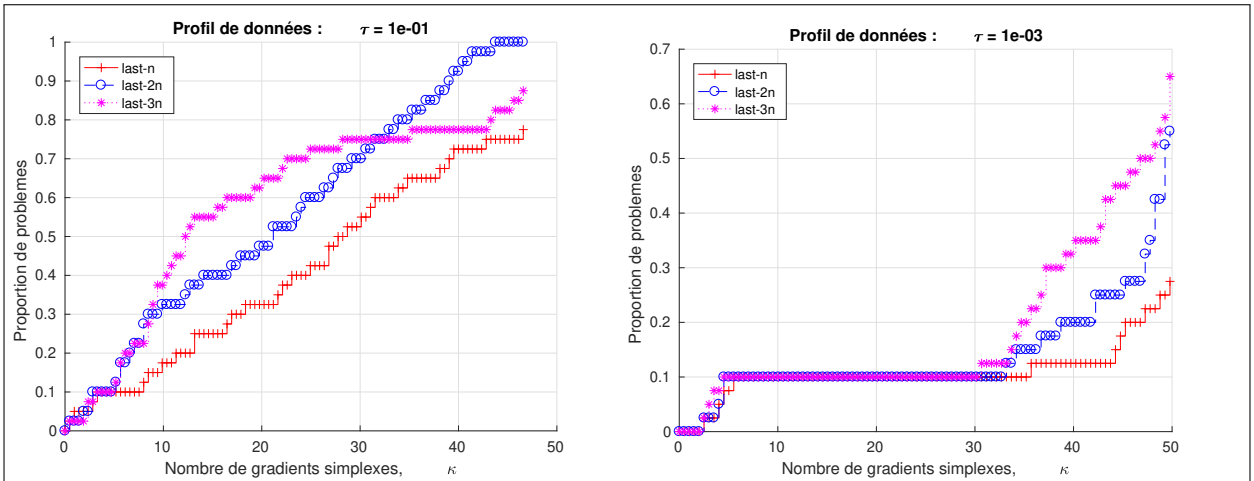
Tous les points utilisés lors de l'analyse de sensibilité sont des points qui appartiennent déjà à la *cache* et ont donc déjà été évalués lors des itérations précédentes de l'algorithme. Il faut donc définir différentes stratégies de sélection des points dans la *cache*, ainsi que le nombre de ces points.

Pour le nombre de points sélectionnés, les trois possibilités suivantes sont considérées : $n, 2n, 3n$ points, où n est la dimension du problème original. Cela nous paraît un bon compromis entre ne pas sélectionner trop de points pour garder un temps d'exécution raisonnable et garder suffisamment de points pour avoir une analyse de sensibilité qui donne des résultats pertinents. Les différentes stratégies de sélection des points sont les suivantes :

1. $last - x$: les x derniers points de la *cache* sont sélectionnés.
2. $closest - x$: les points sont triés en fonction de leur distance avec la solution courante au moment de l'analyse de sensibilité et les x points les plus proches sont sélectionnés pour l'analyse en composante principale.
3. $dist - \epsilon$: les points sont triés en fonction de leur distance avec la solution courante au moment de l'analyse de sensibilité et les points à une distance inférieure à $\epsilon\Delta^m_{min}$ sont sélectionnés pour l'analyse en composante principale. Δ^m_{min} désigne la plus petite valeur que le paramètre de taille du maillage Δ^m a pris au cours de l'exécution de l'algorithme. Cela permet de lier la distance considérée avec la taille du treillis.

Ces différentes stratégies ont été comparées sur le même ensemble de problèmes que pour les paramètres précédents. Pour plus de clarté, des profils de performance et de données ont été tracés pour chaque stratégie aux figures 5.6, 5.7 et 5.8. Chacune de ces figures reprend une stratégie (resp. *last*, *closest* et *dist*) avec plusieurs valeurs pour leur paramètre. La figure 5.9 compare les trois stratégies avec leur meilleur paramètre.

La figure 5.6 montre que, parmi les stratégies de sélection des derniers points évalués, la stratégie $last - 2n$ semble avoir de meilleures performances que les autres pour une faible précision, tandis que la stratégie $last - 3n$ prend le dessus avec une plus forte précision pour

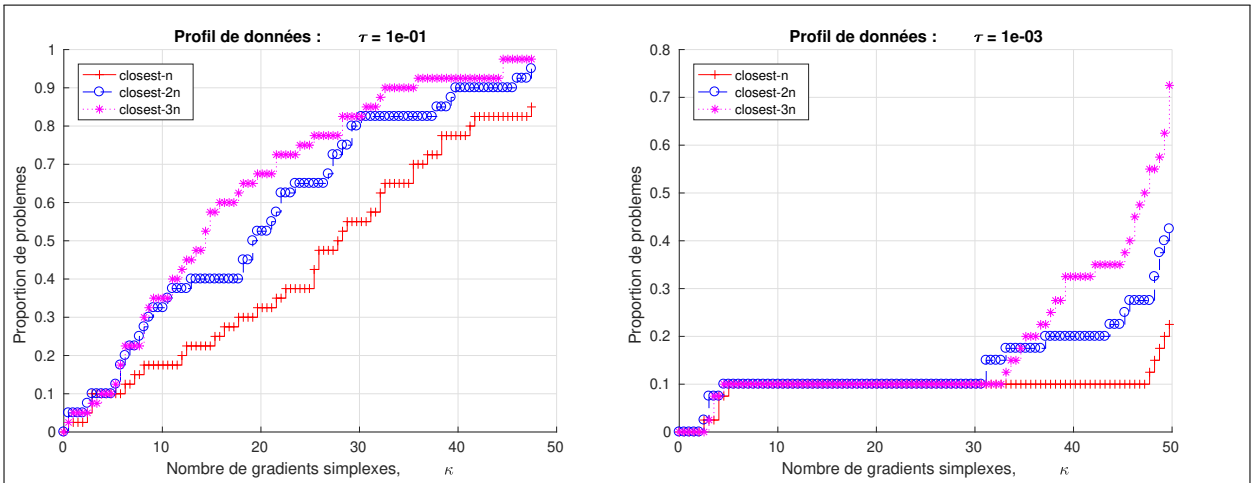


(a) Profil de données pour une tolérance $\tau = 10^{-1}$

(b) Profil de données pour une tolérance $\tau = 10^{-3}$

Figure 5.6 Comparaison des performances de PCA-MADS avec différents nombres des derniers points sélectionnés pour l'analyse de sensibilité (stratégies $last-n$), sur une partie de la suite de fonctions *bbob-largescale* de *COCO*

le critère de convergence des profils.



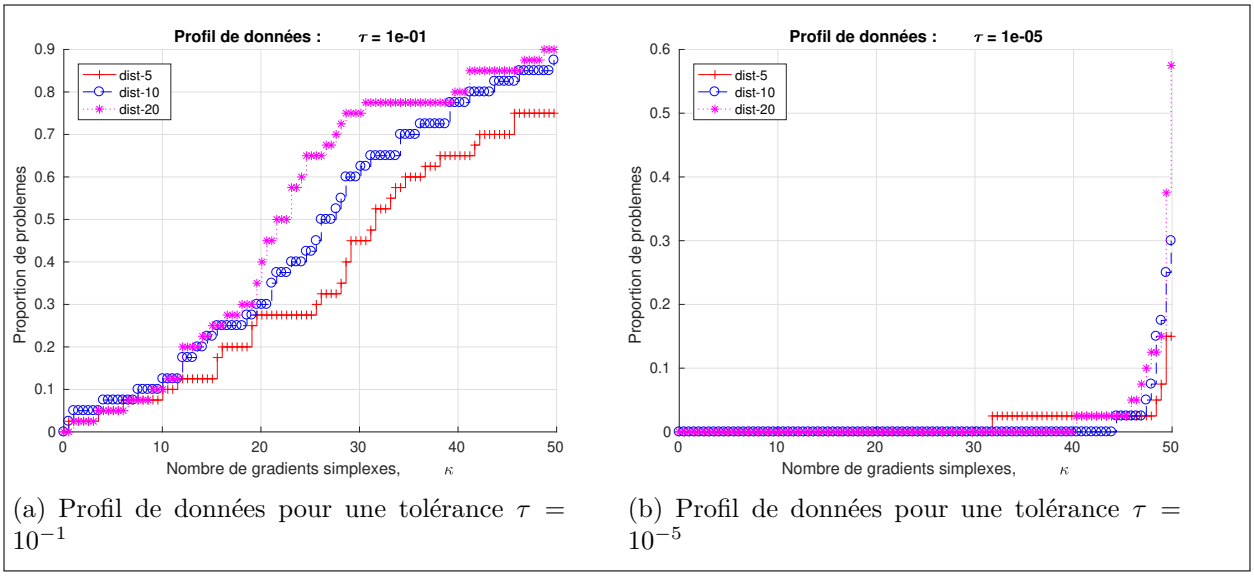
(a) Profil de données pour une tolérance $\tau = 10^{-1}$

(b) Profil de données pour une tolérance $\tau = 10^{-3}$

Figure 5.7 Comparaison des performances de PCA-MADS avec différents nombres des plus proches points sélectionnés pour l'analyse de sensibilité (stratégie $closest-n$), sur une partie de la suite de fonctions *bbob-largescale* de *COCO*

La figure 5.7 indique que lorsqu'il s'agit de sélectionner les points les plus proches de la solution courante pour l'analyse en composante principale, la stratégie qui en sélectionne $3n$

est plus intéressante que les autres.



(a) Profil de données pour une tolérance $\tau = 10^{-1}$

(b) Profil de données pour une tolérance $\tau = 10^{-5}$

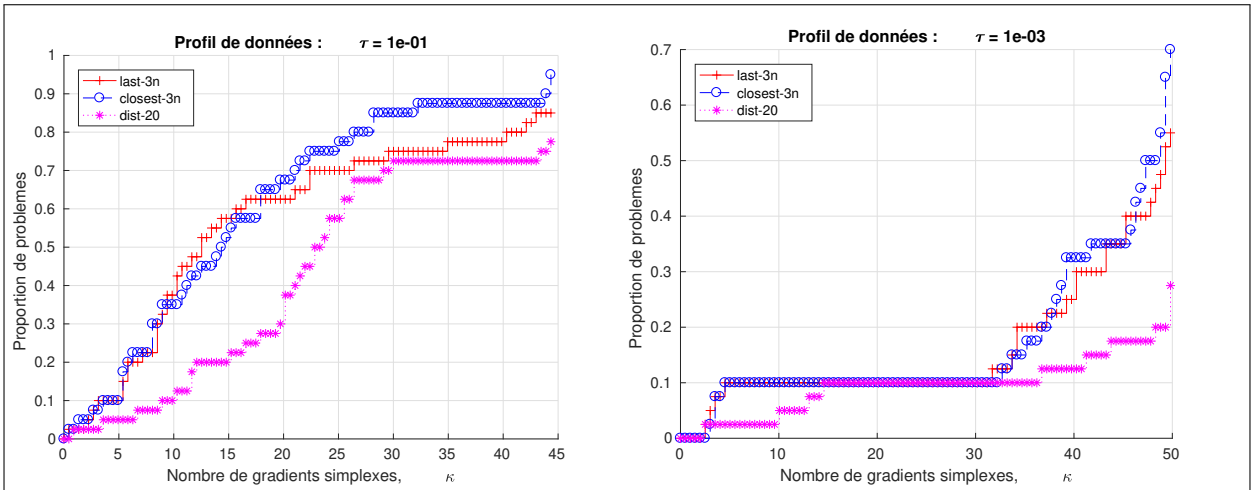
Figure 5.8 Comparaison des performances de PCA-MADS avec une stratégie de sélection de points pour l'analyse de sensibilité basée sur la distance autour de la solution courante (stratégie $dist - \epsilon$), sur une partie de la suite de fonctions *bbob-largescale* de *COCO*

Les stratégies les plus performantes sont celles avec un plus grand nombre de points sélectionnés pour l'analyse de sensibilité de la boîte noire. Cela se confirme également pour la stratégie $dist - \epsilon$, dont les comparaisons sont visibles sur la figure 5.8.

Les profils de la figure 5.9 permettent de comparer les trois stratégies avec leur meilleur nombre de points. Une stratégie se distingue sur ces profils ; il s'agit de la stratégies $closest - 3n$, qui sélectionne les $3n$ points les plus proches de la solution courante pour appliquer l'analyse de sensibilité. La figure 5.7 semble indiquer que les performances de cette stratégie sont améliorées lorsqu'un plus grand ensemble de points est sélectionné. Il serait donc intéressant d'essayer de trouver s'il y a un nombre de points optimum ou si la sélection de l'entièreté de la *cache* est en fait le meilleure stratégie possible.

5.5.4 Evaluation initiale de points

Lors de la première itération, la *cache* ne contient qu'un seul point, le point de départ de l'algorithme. Comme vu précédemment, les performances de l'algorithme sont améliorées lorsqu'un relativement grand nombre de points sont sélectionnés pour l'analyse en composante principale. Afin de pouvoir profiter de l'étape de recherche dès la première itération, il serait intéressant d'évaluer un certain nombre de points avant d'effectuer cette étape. Un nombre



(a) Profil de données pour une tolérance $\tau = 10^{-1}$

(b) Profil de données pour une tolérance $\tau = 10^{-3}$

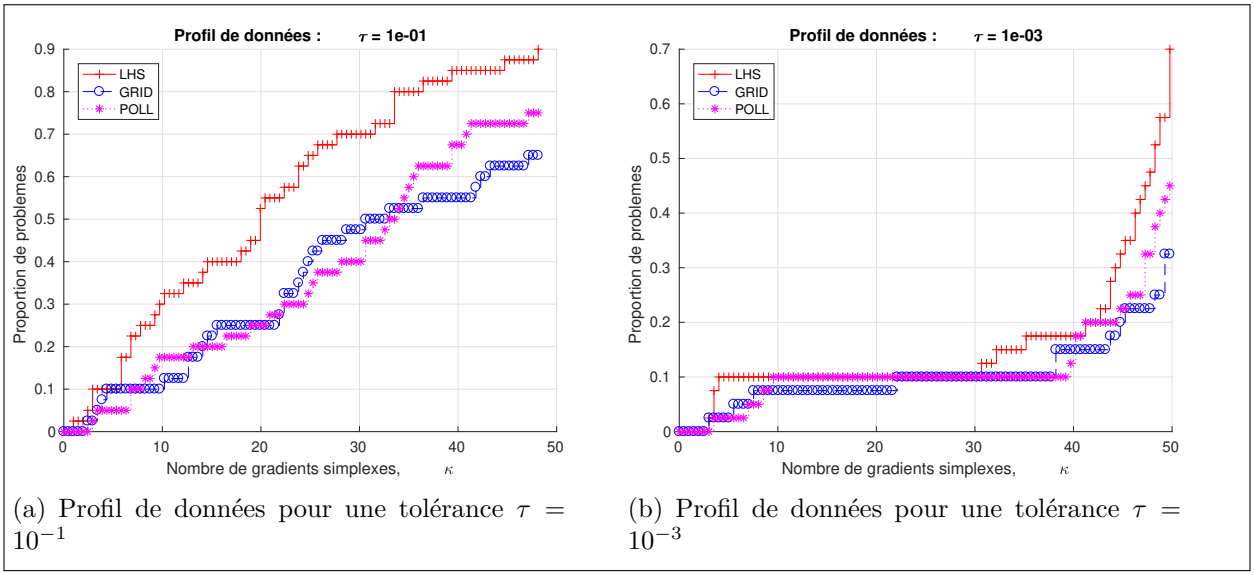
Figure 5.9 Comparaison des performances de PCA-MADS avec différentes stratégies de sélection de points pour l'analyse de sensibilité, sur une partie de la suite de fonctions *bbob-largescale* de *COCO*

de $2n$ points semble approprié ; suffisamment de points pour avoir des résultats intéressants mais pas trop pour préserver le budget d'évaluations pour les itérations ultérieures.

Les trois stratégies suivantes sont proposées pour évaluer des points à la première itération.

- Echantillonnage par hypercube latin (LHS) : il s'agit d'une méthode d'échantillonnage aléatoire qui permet de bien répartir les points échantillonnés dans l'espace tout entier.
- GRID : une méthode d'échantillonnage déterministe qui répartit les points sélectionnés sur une grille dans l'espace.
- POLL : une stratégie qui consiste à effectuer une sonde pour remplir la *cache*. Cela revient à ignorer l'étape de recherche lors de la première itération.

Les performances des différentes stratégies sont comparées à la figure 5.10. Les profils indiquent clairement que la stratégie d'échantillonnage par hypercube latin a des meilleures performances que les autres stratégies, aussi bien à faible qu'à plus forte précision. Cela n'est pas surprenant, une étape de recherche basée sur un échantillonnage par hypercube latin a déjà été appliquée avec l'algorithme MADS ou GPS, comme cela est précisé dans [8].



(a) Profil de données pour une tolérance $\tau = 10^{-1}$

(b) Profil de données pour une tolérance $\tau = 10^{-3}$

Figure 5.10 Comparaison des performances de Pca-Mads avec différentes stratégies de remplissage initial de la *cache* pour la première étape de recherche, sur une partie de la suite de fonctions *bbob-largescale* de *COCO*

5.5.5 Stratégie de construction et d'évaluation du sous-problème

Dans la section 4.2, deux stratégies sont proposées pour la transformation d'un point en dimension p en un point en dimension n. Celles-ci sont

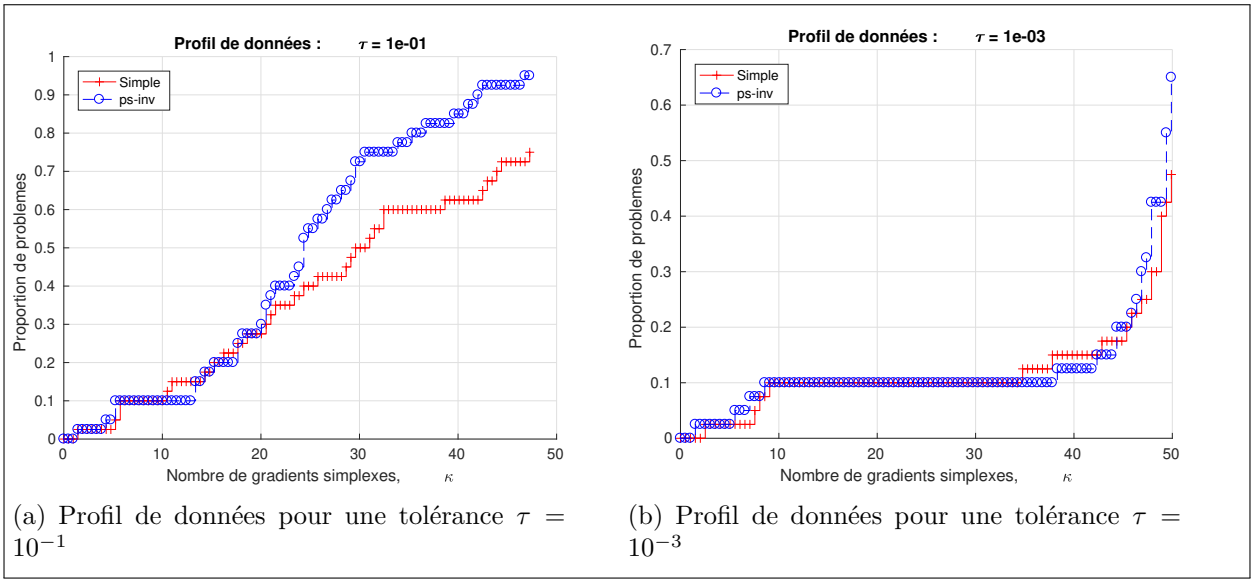
$$x = x^k + Py, \qquad \text{éq (4.4) et} \qquad x = x^k + (P^\top)^\dagger y, \qquad \text{éq. (4.5)}.$$

Etant donné que ces transformations sont au cœur de l'algorithme Pca-Mads, celles-ci devraient avoir une certaine influence sur les performances de la méthode. Les deux stratégies sont comparées à l'aide de profils de performance et de données, visibles à la figure 5.11.

Ces profils indiquent clairement que la stratégie utilisant l'opérateur de pseudo-inverse, légendée *ps-inv*, a de meilleures performances que la transformation plus simple, et ce aussi bien avec une faible qu'une forte précision pour le critère de convergence des profils. La transformation de l'équation (4.5) sera donc privilégiée.

5.6 Comparaison avec d'autres méthodes et algorithmes

Cette section permet de comparer l'algorithme proposé Pca-Mads avec d'autres algorithmes d'optimisation. Ceux-ci sont l'algorithme Stats-Mads, décrit à la section 3.4.3 et l'algorithme CMA-ES [34], un algorithme génétique similaire à ceux décrits à la section 3.1.2.



(a) Profil de données pour une tolérance $\tau = 10^{-1}$

(b) Profil de données pour une tolérance $\tau = 10^{-3}$

Figure 5.11 Comparaison des performances de Pca-Mads avec les deux stratégies d'évaluation du sous-problème présentées aux équations (4.4) et (4.5), sur une partie de la suite de fonctions *bbob-largescale* de *COCO*

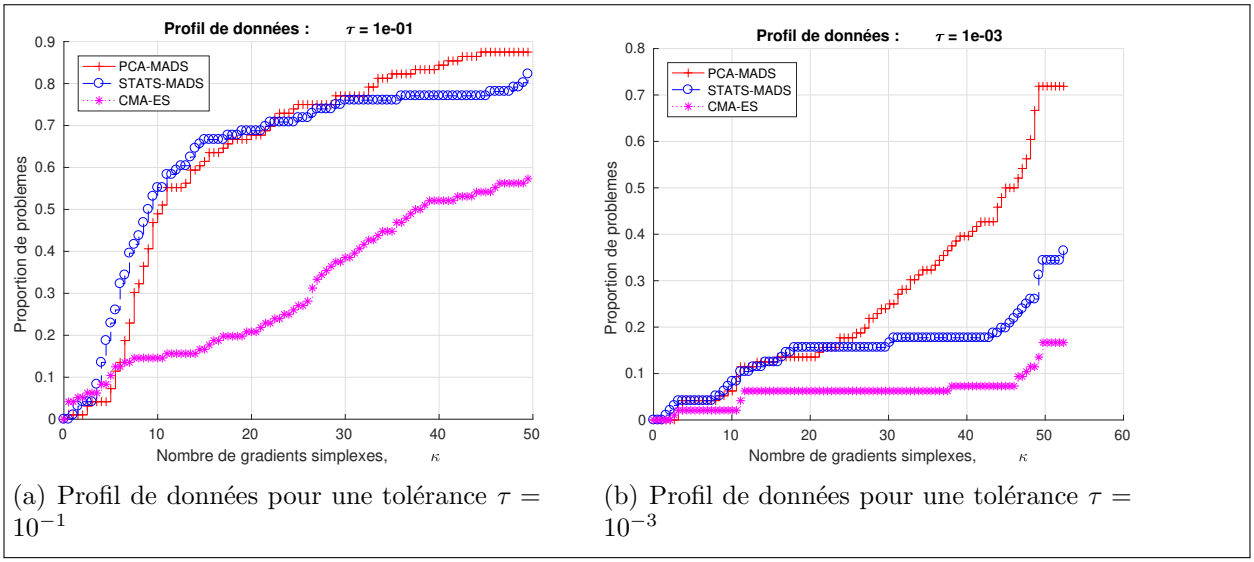
Dans un premier temps, ces algorithmes seront comparés sur la suite *bbob-largescale* de la plateforme *COCO* et ensuite sur un nouvel ensemble de problèmes issus de la littérature.

5.6.1 Comparaison sur la suite *COCO*

Une nouvelle fois, la plateforme *COCO* est utilisée pour comparer des algorithmes. La suite de fonctions *bbob-largescale* est utilisée pour ces tests. Les dimensions des problèmes considérés sont 80, 160, 320 et 640. Cela permet d'avoir un échantillon de problèmes de grande dimension. L'ensemble de problèmes est composé de la première instance de chaque fonction dans chaque dimension, ce qui donne un total de 96 problèmes.

Les profils de données de la figure 5.12 permettent de comparer les performances des trois algorithmes. Ceux-ci montrent que l'algorithme Stats-Mads semble avoir des performances similaires, voire meilleures que l'algorithme Pca-Mads jusqu'à $25n$ à $30n$ évaluations. Ensuite, Pca-Mads prend clairement l'avantage.

Il faut noter que Pca-Mads possède un certain avantage par rapport aux autres algorithmes. En effet, les paramètres de celui-ci ont été fixés en fonction des résultats obtenus sur une partie de la suite considérée. C'est pourquoi des profils ont également été tracés en gardant uniquement les problèmes qui n'ont pas été utilisés lors des tests de la section 5.5. Ces profils sont visibles à la figure 5.13. A faible précision, bien que leurs performances soient assez



(a) Profil de données pour une tolérance $\tau = 10^{-1}$

(b) Profil de données pour une tolérance $\tau = 10^{-3}$

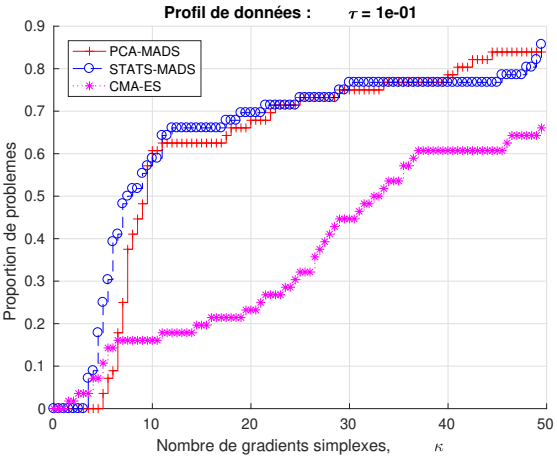
Figure 5.12 Comparaison des algorithmes PCA-MADS, STATS-MADS et CMA-ES au moyen de profils de performance et de données sur la suite de fonctions *BBOB-largescale* en dimension $80, 160, 320$ de *COCO* avec un budget de $50n$

proches, STATS-MADS semble avoir un avantage par rapport à PCA-MADS. Ce comportement n'est pas visible à plus forte précision. En effet PCA-MADS a clairement un avantage par rapport à STATS-MADS lorsque le critère de convergence est plus strict, ce qui est visible sur le profil de la figure 5.13(b). L'algorithme génétique CMA-ES a de moins bonnes performances que les deux autres algorithmes dans tous les cas.

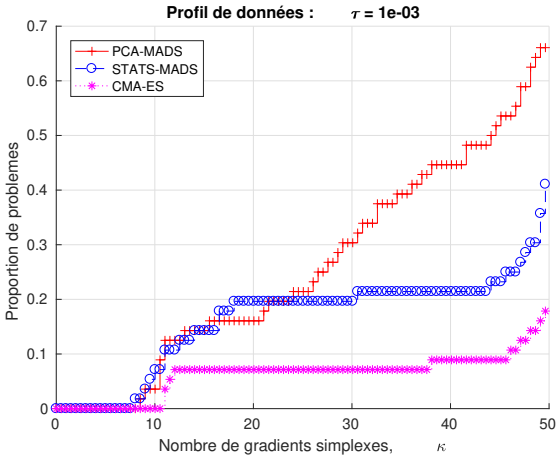
5.6.2 Comparaison sur des problèmes issus de la littérature

Pour la suite de la comparaison, nous considérons un ensemble de problèmes de dimension variable issus de la littérature. Ceux-ci sont repris dans le tableau 5.1. Ce tableau contient les meilleures solutions, pour chaque problème et chaque dimension, obtenues par les deux meilleurs algorithmes des tests précédents, à savoir PCA-MADS et STATS-MADS. Pour chaque problème, la meilleure solution obtenue par l'un ou l'autre algorithme est indiquée en gras.

L'algorithme PCA-MADS trouve de meilleures solutions que STATS-MADS pour certains des problèmes et pour la plupart des dimensions. Néanmoins, STATS-MADS domine clairement PCA-MADS pour le problème PENALTY1. Le problème G2 est un problème dont les contraintes sont difficiles à satisfaire. Les deux algorithmes ont donc du mal à trouver des solutions réalisables différentes même si les meilleures solutions trouvées ne sont pas les points de départ. Néanmoins, cela confirme les résultats de la section précédente, l'approche de PCA-MADS est intéressante lors de l'optimisation de boîtes noires de grande dimension



(a) Profil de données pour une tolérance $\tau = 10^{-1}$



(b) Profil de données pour une tolérance $\tau = 10^{-3}$

Figure 5.13 Comparaison des algorithmes PCA-MADS, STATS-MADS et CMA-ES au moyen de profils de performance et de données sur une partie de la suite de fonctions *BBOB-largescale* en dimension $80, 160, 320$ de *COCO* avec un budget de $50n$

avec l'algorithme MADS.

Tableau 5.1 Comparaison des solutions de différents algorithmes sur un ensemble de problèmes issus de la littérature

Problème	**Dimension**	**Pca-Mads**	**Stats-Mads**
BROWNAL	$n = 600$	$\mathbf{-216305309.391029}$	-216197339.471389
	$n = 800$	$\mathbf{-512459636.278400}$	-512348752.593470
	$n = 1000$	$\mathbf{-1000756562.998527}$	-1000537007.708609
[50, Problème 27]	$n = 1500$	$\mathbf{-3376668083.684607}$	-3376197820.744400
G2 [9]	$n = 600$	-51.334076	-51.334076
	$n = 800$	-68.378302	-68.378302
	$n = 1000$	-85.422528	-85.422528
	$n = 1500$	-128.033092	-128.033092
L1HILB [47]	$n = 600$	$\mathbf{0.509424}$	1.896469
	$n = 800$	$\mathbf{0.616835}$	2.392375
	$n = 1000$	4.490846	$\mathbf{2.515649}$
	$n = 1500$	$\mathbf{2.433164}$	3.902208
PENALTY1	$n = 600$	-383.202462	$\mathbf{-603.041571}$
	$n = 800$	-499.452491	$\mathbf{-805.309644}$
	$n = 1000$	-599.723565	$\mathbf{-1003.565068}$
[50, Problème 23]	$n = 1500$	-891.330350	$\mathbf{-1509.736833}$
VARDIM	$n = 600$	$\mathbf{8932322761.775856}$	11317779579.835079
	$n = 800$	$\mathbf{33427194649.599163}$	38882984612.787933
	$n = 1000$	$\mathbf{77736754861.641235}$	98219498344.552536
[50, Problème 25]	$n = 1500$	$\mathbf{463537077338.545654}$	519702720318.423035

CHAPITRE 6 Conclusion

L'optimisation de boîtes noires est le domaine des mathématiques appliquées qui se concentre sur la recherche d'extrema d'une fonction objectif dont les dérivées, ou celles des contraintes, ne sont pas accessibles ou n'existent pas. Il s'agit d'un domaine qui regroupe n'importe quel problème qui, pour une entrée donnée, peut retourner des valeurs pour l'objectif et les contraintes. Ces problèmes prennent généralement la forme d'une simulation ou d'un code informatique dont les fonctionnements internes ne sont pas connus de l'optimiseur ; c'est la raison pour laquelle ils sont appelés boîtes noires.

6.1 Synthèse des travaux

Le travail présenté dans ce document se concentre sur une catégorie spécifique de problèmes d'optimisation de boîtes noires, il s'agit de problèmes d'optimisation sans dérivée de grande dimension et non contraints. Dans le contexte de boîtes noires, un problème est considéré de grande dimension lorsqu'il possède de quelques centaines à quelques milliers de variables.

Parmi les caractéristiques principales des problèmes d'optimisation sans dérivée, outre l'absence de dérivées, on retrouve un long temps d'exécution et donc un budget total d'évaluations limité. De plus, étant donné le manque d'information disponible, un algorithme pertinent se doit de sonder l'espace de recherche autour de la solution trouvée afin de s'assurer de la qualité de celle-ci. En grande dimension, cela se traduit par un très grand nombre d'évaluations nécessaires, et donc une exécution très longue.

L'algorithme Mads permet de pallier ces problèmes. Il s'agit d'une méthode bien connue et flexible qui permet de résoudre des problèmes d'optimisation sans dérivée de quelques dizaines de variables. L'algorithme Stats-Mads, basé sur Mads, applique une analyse de sensibilité au moyen d'une méthode statistique afin d'identifier les variables prépondérantes, c'est-à-dire celles qui ont le plus d'influence sur l'objectif. Ensuite, l'algorithme alterne entre une instance de Mads en grande dimension et une instance de Mads dont les variables les moins influentes du problème ont été fixées. Les conclusions des travaux de Stats-Mads [15] indiquent que la plupart de la diminution de l'objectif s'effectue lors de l'optimisation du problème en plus petite dimension.

Ce travail propose un nouvel algorithme d'optimisation de la classe Mads, inspiré de Stats-Mads, qui fait appel à des analyses en composante principale. L'algorithme proposé est donc appelé Pca-Mads. Celui-ci cherche à réduire, momentanément, la dimension du problème

au moyen d'une analyse en composante principale. En permettant de passer l'étape de sonde en grande dimension si l'étape de recherche en petite dimension a permis de diminuer la fonction objectif, l'algorithme Pca-Mads évite l'étape la plus coûteuse en évaluations. Cela permet notamment de poursuivre la recherche en petite dimension tant que celle-ci génère des solutions améliorantes, contrairement à Stats-Mads. De plus, de par sa structure, Pca-Mads hérite des propriétés de convergence de l'algorithme Mads sur lequel il est basé. Cela permet notamment d'assurer de trouver un point stationnaire après une infinité d'itérations et ce résultat est obtenu grâce à la sonde en grande dimension. L'algorithme Pca-Mads hérite également de la flexibilité de Mads via son étape de recherche.

6.2 Discussion et limitations de la solution proposée

Les résultats obtenus indiquent clairement l'intérêt de l'approche de Pca-Mads pour des problèmes en grande dimension. L'algorithme génétique CMA-ES ne rivalise pas avec les performances de Pca-Mads. Stats-Mads, quant à lui, a des performances qui s'approchent de celles de Pca-Mads et trouve même de meilleures solutions sur certains problèmes. Toutefois, sur l'ensemble des problèmes de quelques centaines de variables considérés lors de ce travail, Pca-Mads reste supérieur à Stats-Mads. En petite dimension, Pca-Mads ne semble pas avoir un comportement réellement intéressant, bien que celui-ci n'ait pas été étudié en profondeur.

La principale limitation de l'algorithme est sa sensibilité à ses nombreux paramètres. Bien que l'influence de certains d'entre eux ait été étudiée, il en existe d'autres qui pourraient jouer sur son comportement. Une autre limitation, certes inhérente au domaine, est son long temps d'exécution. Le choix de la valeur de certains paramètres comme ceux liés à l'analyse en composante principale revient à faire un compromis entre la qualité de l'analyse et une demande en ressources, notamment au niveau du temps et de la mémoire. Il faut également noter que, lors de l'étape de recherche, un nouveau problème est construit basé sur des combinaisons linéaires de variables. Celles-ci, en fonction du problème étudié, pourraient n'avoir aucun sens physique.

6.3 Améliorations possibles

La première amélioration vient en réponse aux limitations de l'algorithme. L'utilisation de méthodes d'optimisation pour choisir les valeurs des paramètres d'un algorithme est une approche à envisager. Cela a déjà été fait dans d'autres travaux, notamment [5,12,44]. L'étude des paramètres de ce travail n'a été faite que sur un ensemble restreint de problèmes tests et les

valeurs comparées ont été choisies arbitrairement. Une approche algorithmique permettrait d'envisager d'autres combinaisons de paramètres et d'améliorer l'algorithme.

Une autre perspective de recherche serait d'inclure l'approche de Pca-Mads à l'algorithme Psd-Mads [9], une version parallèle de Mads. Cet algorithme cherche à diviser l'espace de recherche et d'en attribuer une partie à chaque processeur. Cela se fait en ne laissant libres que quelques variables par processeur. Une possibilité serait d'attribuer des combinaisons de variables issues d'une analyse en composante principale similaire à Pca-Mads. Cela permettrait de faire évoluer certains processeurs dans de nouveaux sous-espaces de recherche.

RÉFÉRENCES

[1] M.A. Abramson, C. Audet, J.E. Dennis, Jr., and S. Le Digabel. OrthoMADS : A Deterministic MADS Instance with Orthogonal Directions. *SIAM Journal on Optimization*, 20(2) :948–966, 2009.

[2] L. Adjengue, C. Audet, and I. Ben Yahia. A variance-based method to rank input variables of the Mesh Adaptive Direct Search algorithm. *Optimization Letters*, 8(5) :1599–1610, 2014.

[3] S. Alarie, N. Amaioua, C. Audet, S. Le Digabel, and L.-A. Leclaire. Selection of variables in parallel space decomposition for the mesh adaptive direct search algorithm. Technical Report G-2018-38, Les cahiers du GERAD, 2018.

[4] N. Amaioua. *Modèles quadratiques et décomposition parallèle pour l'optimisation sans dérivées*. PhD thesis, Polytechnique Montréal, 2018.

[5] C. Audet, C.-K. Dang, and D. Orban. Algorithmic Parameter Optimization of the DFO Method with the OPAL Framework. In *Software Automatic Tuning : From Concepts to State-of-the-Art Results*, chapter 15, pages 255–274. Springer, K. Naono and K. Teranishi and J. Cavazos and R. Suda edition, 2010.

[6] C. Audet, C.-K. Dang, and D. Orban. Efficient use of parallelism in algorithmic parameter optimization applications. *Optimization Letters*, 7(3) :421–433, 2013.

[7] C. Audet and J.E. Dennis, Jr. Analysis of generalized pattern searches. *SIAM Journal on Optimization*, 13(3) :889–903, 2003.

[8] C. Audet and J.E. Dennis, Jr. Mesh Adaptive Direct Search Algorithms for Constrained Optimization. *SIAM Journal on Optimization*, 17(1) :188–217, 2006.

[9] C. Audet, J.E. Dennis, Jr., and S. Le Digabel. Parallel Space Decomposition of the Mesh Adaptive Direct Search Algorithm. *SIAM Journal on Optimization*, 19(3) :1150–1170, 2008.

[10] C. Audet and W. Hare. *Derivative-Free and Blackbox Optimization*. Springer Series in Operations Research and Financial Engineering. Springer International Publishing, Cham, Switzerland, 2017.

[11] C. Audet, S. Le Digabel, and C. Tribes. The Mesh Adaptive Direct Search Algorithm for Granular and Discrete Variables. *SIAM Journal on Optimization*, 29(2) :1164–1189, 2019.

[12] C. Audet and D. Orban. Finding optimal algorithmic parameters using derivative-free optimization. *SIAM Journal on Optimization*, 17(3) :642–664, 2006.

[13] C. Audet and C. Tribes. Mesh-based Nelder-Mead algorithm for inequality constrained optimization. *Computational Optimization and Applications*, 71(2) :331–352, 2018.

[14] C. Bélisle, H.E. Romeijn, and R.L. Smith. Hit-and-run algorithms for generating multivariate distributions. Technical Report 2, 1993.

[15] I. Ben Yahia. Identification statistique de variables importantes pour l'optimisation de boîtes noires. Master's thesis, École Polytechnique de Montréal, 2012.

[16] B. Bettonvil and J.P.C. Kleijnen. Searching for important factors in simulation models with many factors : Sequential bifurcation. *European Journal of Operational Research*, 96(1) :180–194, 1997.

[17] M. Binois, D. Ginsbourger, and O.Roustant. On the choice of the low-dimensional domain for global optimization via random embeddings. *Journal of global optimization*, 76(1) :69–90, 2020.

[18] A. Boneh and A. Golan. Constraints' redundancy and feasible region boundedness by random feasible point generator (RFPG). In *Third European congress on operations research (EURO III), Amsterdam*, 1979.

[19] E. Borgonovo and E. Plischke. Sensitivity analysis : a review of recent advances. *European Journal of Operational Research*, 248(3) :869–887, 2016.

[20] E. Brochu, V.M. Cora, and N. De Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv :1012.2599*, 2010.

[21] C. Cartis and A. Otemissov. A dimensionality reduction technique for unconstrained global optimization of functions with low effective dimensionality. Technical report, 2020.

[22] F.H. Clarke. *Optimization and Nonsmooth Analysis*. John Wiley & Sons, New York, 1983. Reissued in 1990 by SIAM Publications, Philadelphia, as Vol. 5 in the series Classics in Applied Mathematics.

[23] A.R. Conn and S. Le Digabel. Use of quadratic models with mesh-adaptive direct search for constrained black box optimization. *Optimization Methods and Software*, 28(1) :139–158, 2013.

[24] A.R. Conn, K. Scheinberg, and L.N. Vicente. *Introduction to Derivative-Free Optimization*. MOS-SIAM Series on Optimization. SIAM, Philadelphia, 2009.

[25] D.D. Cox and S. John. A statistical method for global optimization. In *[Proceedings] 1992 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1241–1246. IEEE, 1992.

[26] R.I. Cuckier, C.M. Fortuin, K.E. Shuler, and A.G. Petschek. Study of sensitivity of coupled reaction systems to uncertainties in rate coefficients. I. Theory. *J. of Chem. Phys*, 59(8) :3873–3878, 1973.

[27] R. I. Cukier, J. H. Schaibly, and K. E. Shuler. Study of the sensitivity of coupled reaction systems to uncertainties in rate coefficients. III. Analysis of the approximations. *The Journal of Chemical Physics*, 63(3) :1140–1149, 1975.

[28] A. Dean and S. Lewis. *Screening : methods for experimentation in industry, drug discovery, and genetics*. Springer Science & Business Media, 2006.

[29] E.D. Dolan and J.J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2) :201–213, 2002.

[30] I. Fajfar, J. Puhan, and Á. Bűrmen. Evolving a Nelder–Mead algorithm for optimization with genetic programming. *Evolutionary computation*, 25(3) :351–373, 2017.

[31] I. Fajfar, J. Puhan, and Á. Bűrmen. The Nelder–Mead simplex algorithm with perturbed centroid for high-dimensional function optimization. *Optimization Letters*, 13(5) :1011–1025, 2019.

[32] E. Fermi and N. Metropolis. Numerical solution of a minimum problem. Los Alamos Unclassified Report LA–1492, Los Alamos National Laboratory, Los Alamos, USA, 1952.

[33] F. Gao and L. Han. Implementing the Nelder-Mead simplex algorithm with adaptive parameters. *Computational Optimization and Applications*, 51(1) :259–277, 2012.

[34] N. Hansen. The CMA Evolution Strategy : A Comparing Review. In J. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, editors, *Towards a New Evolutionary Computation*, volume 192 of *Studies in Fuzziness and Soft Computing*, pages 75–102. Springer Berlin Heidelberg, 2006.

[35] N. Hansen, A. Auger, O. Mersmann, T. Tusar, and D. Brockhoff. COCO : A platform for comparing continuous optimizers in a black-box setting. *arXiv preprint arXiv :1603.08785*, 2016.

[36] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009 : Noiseless functions definitions. Technical Report RR-6829, INRIA, 2009.

[37] T. Homma and A. Saltelli. Importance measures in global sensitivity analysis of nonlinear models. *Reliability Engineering & System Safety*, 52(1) :1–17, 1996.

[38] B. Iooss. Revue sur l'analyse de sensibilité globale de modèles numériques. *Journal de la Société Française de Statistique*, 152(1) :1–23, 2011.

[39] B. Iooss and P. Lemaître. A review on global sensitivity analysis methods. In *Uncertainty management in simulation-optimization of complex systems*, pages 101–122. Springer, Boston, MA, 2015.

[40] I.T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer-Verlag, New York, 1986.

[41] C. T. Kelley. Detection and remediation of stagnation in the Nelder–Mead algorithm using a sufficient decrease condition. *SIAM journal on optimization*, 10(1) :43–55, 1999.

[42] M. Koda, G. J. Mcrae, and J. H. Seinfeld. Automatic sensitivity analysis of kinetic mechanisms. *International Journal of Chemical Kinetics*, 11(4) :427–444, 1979.

[43] G.N. Sesh Kumar and V.K. Suri. Multilevel Neider Mead's simplex method. In *2014 9th International Conference on Industrial and Information Systems (ICIIS)*, pages 1–6. IEEE, 2014.

[44] D. Lakhmiri, S. Le Digabel, and C. Tribes. HyperNOMAD : Hyperparameter optimization of deep neural networks using mesh adaptive direct search. Technical Report G-2019-46, Les cahiers du GERAD, 2019.

[45] S. Le Digabel. Algorithm 909 : NOMAD : Nonlinear Optimization with the MADS algorithm. *ACM Transactions on Mathematical Software*, 37(4) :44 :1–44 :15, 2011.

[46] D.K.J. Lin. A new class of supersaturated designs. *Technometrics*, 35(1) :28–31, 1993.

[47] L. Lukšan and J. Vlcek. Test problems for nonsmooth unconstrained and linearly constrained optimization. Technical Report 798, Institute of Computer Science, Academy of Sciences of the Czech Republic, 2000.

[48] K.I.M. McKinnon. Convergence of the Nelder-Mead simplex method to a nonstationary point. *SIAM Journal on Optimization*, 9(1) :148–158, 1998.

[49] V.K. Mehta. Improved Nelder–Mead algorithm in high dimensions with adaptive parameters based on Chebyshev spacing points. *Engineering Optimization*, pages 1–15, 2019.

[50] J.J. Moré, B.S. Garbow, and Kenneth E. Hillstrom. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software*, 7(1) :17–41, 1981.

[51] J.J. Moré and S.M. Wild. Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20(1) :172–191, 2009.

[52] R. Moriconi, K.S. Sesh Kumar, and M.P. Deisenroth. High-dimensional Bayesian optimization with projections using quantile Gaussian processes. *Optimization Letters*, 14(1) :51–64, 2020.

[53] M.D. Morris. Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33(2) :161–174, 1991.

[54] A. Nayebi, A. Munteanu, and A. Poloczek. A Framework for Bayesian Optimization in Embedded Subspaces. In *International Conference on Machine Learning*, pages 4752–4761, 2019.

[55] J.A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4) :308–313, 1965.

[56] Y. Nesterov. *Introductory lectures on convex optimization : A basic course*. Springer Science & Business Media, Cham, 2013.

[57] R. Penrose. A generalized inverse for matrices. In *Mathematical proceedings of the Cambridge philosophical society*, volume 51, pages 406–413. Cambridge University Press, 1955.

[58] L.M. Rios and N.V. Sahinidis. Derivative-free optimization : a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3) :1247–1293, 2013.

[59] A. Saltelli. Making best use of model evaluations to compute sensitivity indices. *Computer physics communications*, 145(2) :280–297, 2002.

[60] A. Saltelli, K. Chan, M. Scott, et al. *Sensitivity analysis. Probability and statistics series*. New York : John Wiley & Sons, 2000.

[61] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola. *Global sensitivity analysis : the primer*, volume 1. John Wiley & Sons, 2008.

[62] A. Saltelli, S. Tarantola, and KP-S. Chan. A quantitative model-independent method for global sensitivity analysis of model output. *Technometrics*, 41(1) :39–56, 1999.

[63] B. Shahriari, K. Swersky, Z. Wang, R.P. Adams, and N. De Freitas. Taking the human out of the loop : A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1) :148–175, 2015.

[64] R.L. Smith. Efficient monte carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32 :1296–1308, 1984.

[65] I.M. Sobol. Sensitivity estimates for nonlinear mathematical models. *Mathematical modelling and computational experiments*, 1(4) :407–414, 1993.

[66] J.-Y. Tissot and C. Prieur. Bias correction for the estimation of sensitivity indices based on random balance designs. *Reliability Engineering & System Safety*, 107 :205–213, 2012.

[67] V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7(1) :1–25, 1997.

[68] P. Tseng. Fortified-Descent Simplicial Search Method : A General Approach. *SIAM Journal on Optimization*, 10(1) :269–288, 1999.

[69] B. Van Dyke and T.J. Asaki. Using QR Decomposition to Obtain a New Instance of Mesh Adaptive Direct Search with Uniformly Distributed Polling Directions. *Journal of Optimization Theory and Applications*, 159(3) :805–821, 2013.

[70] Y. Wang, S. Du, S. Balakrishnan, and A. Singh. Stochastic zeroth-order optimization in high dimensions. *arXiv preprint arXiv :1710.10551*, 2017.

[71] C. Xu and G. Z. Gertner. Uncertainty and sensitivity analysis for models with correlated parameters. *Reliability Engineering & System Safety*, 93(10) :1563–1573, 2008.