UNIVERSITÉ DE MONTRÉAL

# AFFINEMENT DE MODÈLES SUBSTITUTS EN OPTIMISATION DE BOÎTES NOIRES ET EN OPTIMISATION SANS DÉRIVÉES

JULIEN CÔTÉ-MASSICOTTE
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
AOÛT 2018

©Julien Côté-Massicotte, 2018.

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

AFFINEMENT DE MODÈLES SUBSTITUTS EN OPTIMISATION DE BOÎTES NOIRES ET EN OPTIMISATION SANS DÉRIVÉES

présenté par : CÔTÉ-MASSICOTTE Julien
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. ANJOS Miguel F., Ph. D., Président
M. AUDET Charles, Ph. D., membre et directeur de recherche
M. LE DIGABEL Sébastien, Ph. D., membre

## REMERCIEMENTS

Mille fois merci à mon directeur Charles Audet pour sa disponibilité, ses idées et son support au cours de mon passage à la Polytechnique. Un gros merci à Christophe Tribes pour ses suggestions et l'aide apportée à la programmation de l'algorithme. Merci aux membres du jury Sébastien Le Digabel et Miguel F. Anjos d'avoir pris le temps de lire et de réviser ce mémoire. Merci à Viviane Rochon Montplaisir pour ses conseils et sa bonne humeur.

Merci à mes collègues de bureaux Mathilde Kelly Bourque, Christian Bingane, Ilaria Salerno, Ludovic Salomon, Marie Pied, Joseph Dzahini, Dounia Lakhmiri, Loïc Anthony Sarrazin-Mc Cann, Catherine Poissant, Marie-Ange Dahito, Mathieu Tanneau, Émilie Chénier, Pierre-Yves Bouchet et tous les autres pour leur soutien moral.

Merci à ma famille et à mes amis, mais plus spécialement à ma mère Francine et à mon père Michel pour m'avoir soutenu moralement et financièrement tout au long de mes études universitaires.

# RÉSUMÉ

Ce mémoire se situe dans un contexte d'optimisation sans dérivées, et plus particulièrement dans un contexte d'optimisation de boîtes noires, où la fonction à minimiser et les fonctions relatives aux contraintes sont de type boîte noire. Celles-ci sont potentiellement bruitées, non différentiables, coûteuses en temps de calcul et leurs dérivées sont inaccessibles, inestimables ou inexistantes. Diverses méthodes permettent de résoudre des problèmes d'optimisation de boîtes noires, dont l'algorithme de recherche par treillis adaptatifs (MADS) qui ne dirige la recherche d'optimums qu'avec les valeurs des fonctions aux points explorés. Cet algorithme itératif sépare sa recherche en deux étapes : une recherche globale qui explore l'espace des solutions et une recherche locale qui sonde autour de la meilleure solution visitée.

Afin d'éviter d'évaluer inutilement la boîte noire, la recherche de MADS peut être guidée par des modèles du problème original qu'on nomme substituts. Ces modèles, moins lourds en temps de calcul, sont classés en deux catégories : les modèles statiques et les modèles dynamiques. Un modèle statique est une simplification de la boîte noire qui ne varie pas, tandis qu'un modèle dynamique est une approximation du problème mis à jour tout au long du déploiement de l'algorithme. Toutefois, les travaux précédents employant MADS n'exploitent pas simultanément les deux types de modèles.

Ce travail introduit un nouveau modèle substitut, le modèle hybride quadratique (MHQ), qui s'avère être un modèle dynamique quadratique qui corrige l'information du modèle statique. Au lieu d'apporter une correction additive ou multiplicative, le MHQ vient généraliser ces deux types de correction en considérant le modèle statique comme une variable du modèle quadratique. Ce nouveau modèle est accompagné d'une base théorique solide en plus d'une analyse de convergence basée sur le calcul des fonctions non lisses. MADS exploite les valeurs fournies par le MHQ afin d'ordonner les points candidats de la recherche locale du plus prometteur au moins prometteur. Les fonctions de la boîte noire sont par la suite évaluées aux points ordonnés avec une stratégie opportuniste. Cette approche permet généralement de réduire le nombre d'évaluations de la boîte noire afin d'atteindre la convergence.

Les tests numériques sont effectués sur trois problèmes analytiques et trois problèmes d'ingénierie sous forme de simulations. Les résultats obtenus montrent que l'apport du MHQ à l'algorithme MADS est de permettre une résolution des problèmes avec une plus grande précision. Le MHQ bénéficie donc du caractère global du modèle statique ainsi que de l'aspect local associé au modèle quadratique. Toutefois, l'approche par le MHQ possède des limitations, puisque son temps d'exécution est corrélé au nombre de contraintes du problème.

# ABSTRACT

The present work is in a context of derivative-free optimization, and more particularly in a context of blackbox optimization, where the function to be minimized and the functions related to the constraints are blackboxes. Those are potentially noisy, non-differentiable and computationally expensive. Furthermore, their derivatives are inaccessible or non-existent and they cannot be approximated. Various methods can be used to solve blackbox optimization problems, such as the Mesh Adaptive Direct Search (MADS) algorithm that directs the search only by using the values of the functions at the explored points. This iterative algorithm separates the search into two steps : a global search that explores the solution space and a local poll that explores near the best visited solution.

In order to avoid unnecessary blackbox evaluations, the MADS steps can be guided by models of the original problem called surrogates. These models, which are less time-consuming, are classified in two categories : static models and dynamic models. A static model is a simplification of the blackbox that does not vary, while a dynamic model is an approximation of the problem updated throughout the deployment of the algorithm. However, previous work using MADS do not simultaneously exploit both models.

This work introduces a new surrogate, the quadratic hybrid model (MHQ), a quadratic dynamic model that corrects information from the static model. Instead of bringing an additive or multiplicative correction, the MHQ generalizes these two types of correction by considering the static model as a variable of the quadratic model. This new model is accompanied by a solid theoretical basis in addition to a convergence analysis based on non-smooth calculus. MADS uses the values provided by the MHQ to order the candidate points of the local search from the most to the least promising. The blackbox functions are then evaluated at the ordered points with an opportunistic strategy. This approach generally reduces the number of blackbox evaluations to achieve convergence.

The numerical tests are performed on three analytical problems and three simulation-based engineering problems. The obtained results show that the contribution of the MHQ to the MADS algorithm is to solve problems with a greater precision. The MHQ thus benefits from the global feature of the static model as well as from the local aspect of the quadratic model. However, the MHQ approach has limitations, since its execution time is correlated with the number of problem constraints.

# TABLE DES MATIÈRES

REMERCIEMENTS . . . . . iii

RÉSUMÉ . . . . . iv

ABSTRACT . . . . . v

TABLE DES MATIÈRES . . . . . vi

LISTE DES TABLEAUX . . . . . viii

LISTE DES FIGURES . . . . . ix

LISTE DES SIGLES, ABRÉVIATIONS ET NOTATION . . . . . xi

CHAPITRE 1 INTRODUCTION . . . . . 1
- 1.1 Motivation et objectif de la recherche . . . . . 2
- 1.2 Plan du mémoire . . . . . 2

CHAPITRE 2 DÉVELOPPEMENT DES MÉTHODES DE RECHERCHE DIRECTE ET DES FONCTIONS SUBSTITUTS . . . . . 4
- 2.1 Recherche par coordonnées (Cs) . . . . . 5
- 2.2 Recherche par motifs (Gps) . . . . . 6
- 2.3 Recherche par treillis adaptatifs (Mads) . . . . . 9
- 2.4 Optimisation à l'aide de fonctions substituts et de modèles . . . . . 13
- 2.5 La gestion des contraintes . . . . . 19
- 2.6 Nomad . . . . . 22

CHAPITRE 3 INTÉGRATION D'UN SUBSTITUT STATIQUE AU MODÈLE QUADRATIQUE . . . . . 25
- 3.1 Modèle hybride quadratique . . . . . 25
- 3.2 Approche algorithmique . . . . . 30
- 3.3 Géométrie des ensembles de points d'interpolation . . . . . 36

CHAPITRE 4 RÉSULTATS NUMÉRIQUES . . . . . 40
- 4.1 Un problème de Hock et Schittkowski . . . . . 42
- 4.2 Quatre problèmes de type MDO . . . . . 44

4.3 Problème LOCKWOOD . . . . . 53
4.4 Discussion . . . . . 56

CHAPITRE 5 CONCLUSION . . . . . 59
5.1 Synthèse des travaux . . . . . 59
5.2 Travaux futurs . . . . . 60

RÉFÉRENCES . . . . . 62

# LISTE DES TABLEAUX

Tableau 3.1 Valeur de la fonction $g(x)$ selon la coordonnée $x$. . . . . 29
Tableau 3.2 Exemple de points à ordonner $\mathscr{P} = \{x^1, x^2, x^3, x^4, x^5, x^6\}$. . . . . 34

Tableau 4.1 Distinctions entre le vrai problème et le substitut statique pour le problème Simple_MDO_2n_Variables. . . . . 45
Tableau 4.2 Distinctions entre le vrai problème et le substitut statique pour le problème *simplified wing*. . . . . 48
Tableau 4.3 Distinctions entre le vrai problème et le substitut statique pour le problème *supersonic business jet design*. . . . . 50

# LISTE DES FIGURES


Figure 2.1 Algorithme de recherche par coordonnées (Cs). . . . . 5
Figure 2.2 Exemples d'une base, de bases positives et d'ensembles générateurs positifs de $\mathbb{R}^2$. . . . . 7
Figure 2.3 Algorithme Gps pour un problème d'optimisation sans contrainte. . . . 8
Figure 2.4 Exemples de cadres de sonde pour différentes valeurs de $\delta^k$ et $\Delta^k$. . . . 10
Figure 2.5 Algorithme de recherche par treillis adaptatifs (Mads). . . . . 11
Figure 2.6 Structure de recherche directe assistée d'un modèle substitut. . . . . 15
Figure 2.7 Les points dominés et non dominés sont respectivement symbolisés par des cercles vides et des cercles pleins. Ils sont représentés dans l'espace $(h, f)$. . . . . 21
Figure 2.8 Structure de PB dans un contexte d'optimisation sans dérivées. . . . . 22

Figure 3.1 Différents modèles substituts de la fonction $g(x)$. . . . . 30
Figure 3.2 Exemple dans $\mathbb{R}^2$ d'une région de confiance $\mathcal{B}_\infty(\mu; \rho r)$ avec $\rho = 2$. . . . 33
Figure 3.3 Mads assisté d'un MHQ à l'étape de sonde locale. . . . . 35

Figure 4.1 Profils de données obtenus avec une tolérance $\tau$ sur 10 instances d'exécution de 100 problèmes numériques Hock et Schittkowski numéro 100. 43
Figure 4.2 Exemple de schéma d'optimisation d'un problème de type MDO. . . . . 44
Figure 4.3 Profils de données obtenus avec une tolérance $\tau$ sur 10 instances d'exécution de 100 problèmes numériques Simple_MDO_10_Variables. . . . 46
Figure 4.4 Profils de données obtenus avec une tolérance $\tau$ sur 10 instances d'exécution de 100 problèmes numériques Simple_MDO_16_Variables. . . . 47
Figure 4.5 Profils de données obtenus avec une tolérance $\tau$ sur 10 instances d'exécution de 100 problèmes numériques MDO. . . . . 49
Figure 4.6 Profils de données obtenus avec une tolérance $\tau$ sur 1 instance d'exécution de 25 problèmes numériques *supersonic business jet design*. . . . . 51
Figure 4.7 Profils de données obtenus avec une tolérance $\tau$ sur 1 instance d'exécution de 100 problèmes numériques *supersonic business jet design*. . . . . 53
Figure 4.8 Profils de données obtenus avec une tolérance $\tau$ sur 1 instance d'exécution de 100 problèmes numériques LOCKWOOD. . . . . 54

Figure 4.9 Proportion de problèmes résolus sous une tolérance $\tau$ en fonction du temps sur 1 instance d'exécution de 100 problèmes numériques LOCKWOOD. . . . . 55
Figure 4.10 Profils de données comparant msc et mhq obtenus avec une tolérance $\tau$ sur 10 instances d'exécution de 100 problèmes numériques *simplified wing*. 57
Figure 4.11 Profils de données comparant msc, stat et mhq obtenus avec une tolérance $\tau$ sur 1 instance d'exécution de 100 problèmes numériques LOCKWOOD. 57

# LISTE DES SIGLES, ABRÉVIATIONS ET NOTATION

| Sigles et abréviations | |
|---|---|
| BBO | Optimisation de boîtes noires (*Blackbox Optimization*) |
| Cs | Recherche par coordonnées (*Coordinate Search*) |
| DFO | Optimisation sans dérivées (*Derivative-Free Optimization*) |
| EB | Barrière extrême (*Extreme Barrier*) |
| Gps | Recherche par motifs généralisée (*Generalised Pattern Search*) |
| LtMads | Mads avec directions générées par des matrices triangulaires inférieures aléatoires (*Lower Triangular* Mads) |
| Mads | Recherche directe sur treillis adaptatif (*Mesh Adaptive Direct Search*) |
| MDO | Optimisation multidisciplinaire (*Multi-disciplinary Design Optimization*) |
| MHQ | Modèle hybride quadratique |
| MSC | Modèle statique corrigé |
| Nomad | Programme d'optimisation non linéaire utilisant l'algorithme Mads (*Nonlinear Optimization by Mesh Adaptive Direct Search*) |
| OrthoMads | Mads déterministe avec directions orthogonales (*Orthogonal* Mads) |
| PB | Barrière progressive (*Progressive Barrier*) |

| Notation | |
|---|---|
| $f(x)$ | Fonction objectif à optimiser |
| $\tilde{f}(x)$ | Substitut statique de $f(x)$ |
| $\hat{f}^k(x)$ | Modèle hybride de $f(x)$ à l'itération $k$ |
| $c_j(x)$ | Membre de gauche des contraintes relaxables et quantifiables |
| $\tilde{c}_j(x)$ | Substitut statique de $c_j(x)$ |
| $\hat{c}_j^k(x)$ | Modèle hybride de $c_j(x)$ à l'itération $k$ |
| $c(x)$ | Vecteur des contraintes $c_j(x)$ |
| $\tilde{c}(x)$ | Vecteur des contraintes $\tilde{c}_j(x)$ |
| $h(x)$ | Fonction de violation de contraintes associée aux contraintes $c_j(x)$ |
| $\hat{h}^k(x)$ | Fonction de violation de contraintes associée aux contraintes $\hat{c}_j^k(x)$ |
| $\delta^k$ | Taille du treillis à l'itération $k$ |
| $\Delta^k$ | Taille du pas de sonde à l'itération $k$ |
| $D^k$ | Ensemble des directions de sonde à l'itération $k$ |
| $M^k$ | Ensemble des points du treillis à l'itération $k$ |
| $P^k$ | Ensemble des points de sonde à l'itération $k$ |
| $V^k$ | L'ensemble des points visités possédant une valeur de $f(x)$ et $c(x)$, i.e., la cache, à l'itération $k$ |
| $\tilde{V}^k$ | L'ensemble des points visités possédant une valeur de $\tilde{f}(x)$ et $\tilde{c}(x)$, i.e., la cache du statique, à l'itération $k$ |
| $\Omega$ | Ensemble des points réalisables |
| $X$ | Sous-ensemble de $\mathbb{R}^n$ comprenant les points respectant les contraintes non relaxables et non quantifiables |
| $x^0$ | Point de départ |
| $x^k$ | Meilleure solution à l'itération $k$ |
| $x^F$ | Meilleure solution réalisable à l'itération $k$ (MADS avec PB) |
| $x^I$ | Meilleure solution irréalisable à l'itération $k$ (MADS avec PB) |
| $m$ | Nombre de contraintes relaxables et quantifiables |
| $n$ | Nombre de variables |
| $\mathbb{Y}$ | Ensemble de points d'interpolation |
| $\lvert D \rvert$ | Cardinalité de l'ensemble $D$ |
| $\lVert x \rVert$ | Norme euclidienne de $x$ |
| $\lVert x \rVert_\infty$ | Norme infinie de $x$ |

# CHAPITRE 1 INTRODUCTION

L'optimisation est la branche des mathématiques qui cherche à résoudre les problèmes caractérisés par une fonction à minimiser ou maximiser sous certaines contraintes à respecter. Lorsque la fonction à optimiser et/ou les fonctions relatives aux contraintes sont complexes, il advient que les dérivées soient inaccessibles, inestimables ou même inexistantes. C'est le cas pour les problèmes de type «boîte noire». Une boîte noire est un processus auquel on donne des paramètres en entrée et qui retourne les valeurs des fonctions en sortie, mais la structure interne du processus n'est pas disponible analytiquement. En plus de n'avoir aucune hypothèse sur l'existence de leurs dérivées, les fonctions associées à ces problèmes sont généralement coûteuses en temps de calcul. Les problèmes de ce type sont typiquement des simulations par ordinateur de problèmes d'ingénierie. On peut prendre par exemple un problème de prévention de tsunami où on cherche à placer un certain nombre de bouées qui captent des ondes autour d'une région côtière afin de maximiser le temps de possible évacuation. Un autre exemple serait un problème industriel où l'objectif est d'identifier certains paramètres d'une chaîne de production afin de minimiser le temps d'exécution de celle-ci.

L'optimisation de boîtes noires (BBO) est une famille de l'optimisation qui se focalise sur la résolution des problèmes de type boîte noire. L'optimisation sans dérivées (DFO), quant à elle, est l'étude mathématique des algorithmes d'optimisation qui ne nécessitent aucune information sur les dérivées du problème. Contrairement à la BBO, la DFO considère que les dérivées du problème peuvent exister. Cependant, on ne cherche pas à les identifier explicitement, car les obtenir ou les estimer serait trop coûteux. Les méthodes de DFO n'exploitent donc que les valeurs de la fonction objectif et des fonctions reliées aux contraintes afin de guider la recherche d'un optimum. Les algorithmes de DFO les plus populaires sont l'algorithme Nelder-Mead [71], les méthodes à régions de confiance [34] et les méthodes de recherche directe. Cette dernière famille s'applique bien en BBO puisque, en plus de fournir des preuves de convergences théoriques, ses algorithmes ne supposent aucune condition sur la convexité, la différentiabilité, la continuité et toutes autres propriétés de la fonction objectif et des contraintes. Parmi les méthodes de recherche directe, on retrouve l'algorithme MADS (*Mesh Adaptive Direct Search*) [14] qui a été développé spécifiquement pour résoudre les problèmes de type boîte noire. Une description en détail de cet algorithme est fournie à la section 2.3, puisque ce mémoire s'intéresse particulièrement à celui-ci.

## 1.1 Motivation et objectif de la recherche

Quoiqu'il doive respecter un certain cadre algorithmique, MADS permet beaucoup de variations dans son implémentation. Parmi celles-ci, on retrouve l'usage de simplifications du problème pour guider la recherche. L'utilité d'un modèle simplifié du problème, qu'on définira à la section 2.4 comme un modèle substitut, est qu'il donne un aperçu du comportement du vrai problème tout en étant peu dispendieux en temps de calcul, ce qui n'est habituellement pas le cas pour une boîte noire. Il est montré dans [9, 18, 19, 25, 35, 80] que l'algorithme MADS est généralement plus robuste lorsqu'il est assisté de modèles substituts.

Actuellement, les modèles substituts exploités par MADS sont classés en deux catégories : les modèles statiques du problème original, qui ne varient pas, et les modèles dynamiques qui se mettent à jour tout au long du déploiement de l'algorithme. Ces deux types de modèles sont toutefois exploités de manière indépendante. En effet, si on possède un modèle statique, les travaux précédents n'utilisent que l'information de l'une des deux catégories de modèles substituts pour orienter la recherche de MADS. Bien que différentes, ces deux approches pour guider la recherche apportent généralement de bons résultats. On peut donc supposer qu'une fusion de ces deux approches augmenterait davantage la robustesse de MADS.

Le but de ce projet de recherche est de proposer un nouveau type de modèle substitut afin de guider la recherche de l'algorithme MADS : les modèles «hybrides». Ceux-ci amènent aux modèles statiques une correction mise à jour tout au long de l'algorithme. La correction expérimentée dans ce mémoire est essentiellement un modèle quadratique auquel on apporte une légère modification. Le choix de ce modèle dynamique est qu'il possède de bonnes propriétés de recherche locale. On s'attend donc à ce que le modèle hybride proposé ici soit un compromis bénéfique entre les propriétés locales du modèle quadratique et le caractère global du modèle statique.

## 1.2 Plan du mémoire

Ce mémoire est organisé de la façon suivante. Le chapitre 2 est voué à une revue de la littérature sur le développement des méthodes de recherche directe ainsi que sur les modèles substituts. On y trouve d'abord l'historique des algorithmes de recherche par motifs qui ont mené à la conceptualisation de l'algorithme MADS. Par la suite, on introduit différents types de modèles substituts et comment utiliser ces dernières dans un algorithme de recherche directe tel que MADS. Ensuite, puisqu'on travaille dans un contexte d'optimisation sous contraintes, on montre comment celles-ci sont gérées. Enfin, on présente le logiciel NOMAD

ainsi que ses particularités pratiques pour ce mémoire.

Le chapitre 3 se veut être le cœur de ce mémoire. L'objectif ici est de proposer une nouvelle forme de modèle substitut et d'exposer l'approche algorithmique permettant d'exploiter l'information de ce modèle pour orienter la recherche locale de MADS. L'approche proposée comprend une analyse de convergence basée sur le calcul des fonctions non lisses et une analyse de la géométrie des ensembles de points d'interpolation.

Le chapitre 4 est consacré aux résultats numériques. On teste premièrement l'algorithme sur trois problèmes analytiques pour ensuite l'exécuter sur des problèmes de type boîte noire. Pour tester son efficacité, l'algorithme proposé est comparé aux autres méthodes qui guident la recherche locale déjà présentes dans MADS. Finalement, une synthèse des travaux et des résultats est faite au chapitre 5, accompagnée de certaines pistes pour des travaux futurs.

# CHAPITRE 2 DÉVELOPPEMENT DES MÉTHODES DE RECHERCHE DIRECTE ET DES FONCTIONS SUBSTITUTS

Ce travail considère le problème d'optimisation de la forme suivante :

$$\min_{x \in \Omega} f(x), \tag{2.1}$$

où $\Omega = \{x \in X : c_j(x) \leq 0, j \in J\} \subset \mathbb{R}^n$ représente l'espace des solutions réalisables, $f, c_j : X \to \mathbb{R} \cup \{\infty\}$ pour tout $j \in J = \{1, 2, \ldots, m\}$, et $X$ est un sous-ensemble de $\mathbb{R}^n$. Typiquement, $X$ est un ensemble de la forme $X = \{x : L \leq x \leq U\}$ avec $L \in (\mathbb{R} \cup \{-\infty\})^n$ et $U \in (\mathbb{R} \cup \{+\infty\})^n$. L'ensemble $X$ borne l'espace des solutions possibles et ne peut être violé, *i.e.*, aucun point à l'extérieur de $X$ ne peut être exploré lors de la recherche. Les contraintes $c_j$, $j \in J = \{1, 2, \ldots, m\}$, quant à elles, délimitent l'espace des solutions réalisables sur $\mathbb{R}^n$ et peuvent être violées au cours de la recherche. Elles doivent toutefois être respectées à la solution finale.

On s'intéresse au cas où les fonctions $f, c_j, j \in J$, sont de type boîte noire et qu'aucune information sur les dérivées n'est disponible. C'est le cas, par exemple, lorsque l'évaluation de la fonction objectif et/ou des contraintes requiert une simulation numérique. De plus, puisque ces fonctions sont coûteuses à évaluer et possiblement bruitées, les dérivées sont considérées inestimables ou même inexistantes. Il est donc préférable d'aborder ces problèmes avec des méthodes de recherche directe. Ces méthodes, contrairement à celles basées sur les dérivées, ne nécessitent aucun renseignement sur le gradient de la fonction à optimiser. Elles nécessitent seulement des valeurs de $f, c_j, j \in J$, pour guider la recherche d'optimum. Les méthodes de recherche directe ont été élaborées pour les problèmes d'optimisation de type boîte noire, mais elles sont pratiques pour un grand nombre de problèmes d'optimisation de fonctions non linéaires bruitées, non lisses ou non différentiables. Une description détaillée des problèmes visés par ces méthodes se trouve dans les livres [16, 36].

C'est en 1952 que Fermi et Metropolis [45] présentent le premier algorithme de recherche directe, soit la recherche par coordonnées (Cs : *Coordinate Search*). Cette méthode utilise la base canonique $\{e_i\}_{i=1}^n$ de $\mathbb{R}^n$, et son opposé $\{-e_i\}_{i=1}^n$, comme directions de recherche. Plus tard, en 1997, Torczon [82] développe l'algorithme de recherche par motifs (Gps : *Generalized Pattern Search*). Cette méthode, qui s'avère être une généralisation de l'algorithme Cs et d'autres algorithmes (*evolutionary operation* [30], *pattern search algorithm* [53], *multidirectional search algorithm* [81]), utilise des directions de recherche plus raffinées que ces derniers.

Finalement, en 2006, Audet et Dennis [14] généralisent l'algorithme GPS en élaborant l'algorithme de recherche par treillis adaptatifs (MADS). En plus d'être soutenu par une analyse de convergence rigoureuse basée sur le calcul de fonctions non lisses [32], cet algorithme propose une recherche d'un optimum plus performante que ses prédécesseurs. Les sections suivantes présentent en détail ces trois algorithmes de recherche directe.

## 2.1 Recherche par coordonnées (CS)

On considère un problème de la forme (2.1) où $\Omega = \mathbb{R}^n$, *i.e.*, sans contrainte. Le principe de CS à l'itération $k$ est d'évaluer $f$ sur l'ensemble des $2n$ points de sonde défini par

$$P^k = \{x^k \pm \delta^k e_i : i \in \{1, 2, \ldots, n\}\} \tag{2.2}$$

où, à l'itération $k$, $x^k$ est la meilleure solution connue et $\delta^k \in \mathbb{R}_+^*$ est la longueur des pas. Dans le cas où $f(x^k) \leq f(x)$ pour tout $x \in P^k$, l'itération est un échec et la longueur des pas est réduite de moitié pour l'itération suivante. Dans le cas contraire, l'itération est un succès et on prend un élément de $\underset{x}{\operatorname{argmin}}\{f(x) : x \in P^k\}$ comme meilleure solution à l'itération suivante. L'algorithme CS se résume comme suit :

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

0. <u>Initialisation</u>
   - $\delta^0 \in (0, \infty)$ longueur de pas initiale
   - $\epsilon_{stop} \in (0, \infty)$ critère d'arrêt
   - $k \leftarrow 0$ compteur d'itération
1. <u>Sonde</u>
   - si $f(y) < f(x^k)$ pour un $y \in \underset{x}{\operatorname{argmin}}\{f(x) : x \in P^k\}$
     - poser $x^{k+1} \leftarrow y$ et $\delta^{k+1} \leftarrow \delta^k$
   - sinon
     - poser $x^{k+1} \leftarrow x^k$ et $\delta^{k+1} \leftarrow \frac{1}{2}\delta^k$
2. <u>Terminaison</u>
   - si $\delta^k \geq \epsilon_{stop}$
     - $k \leftarrow k + 1$ et retourner à 1
   - sinon arrêter

Figure 2.1 Algorithme de recherche par coordonnées (CS).

Cet algorithme peut être inefficace. En effet, l'algorithme a tendance à converger vers le premier minimum local qu'il approche. De plus, selon le point de départ $x^0$ et la longueur de

pas initiale $\delta^0$, l'algorithme peut converger très lentement vers un optimum local ou même arrêter en un point sans intérêt. Par exemple, lorsqu'on cherche à minimiser la fonction $f(x) = -(x_1 x_2)^2$ à partir du point de départ $x^0 = (0, 0)$, l'algorithme stagne en $x^0$, qui est un maximum global.

Cette méthode de recherche effectue plusieurs évaluations inutiles de $f$ à chaque itération. Pour éviter ces évaluations superflues, la recherche opportuniste [13] peut être implémentée dans cet algorithme. Au lieu de prendre un élément de $\underset{x}{\operatorname{argmin}}\{f(x) : x \in P^k\}$, cette recherche interrompt l'itération $k$ dès qu'un élément $x \in P^k$ satisfaisant $f(x) < f(x^k)$ est identifié. Une recherche récente [79] démontre l'efficacité de l'opportunisme pour les méthodes de recherche directe.

## 2.2 Recherche par motifs (Gps)

L'algorithme suivant, Gps [82], reprend le principe de Cs. Toutefois, il est développé afin d'éluder les inconvénients de ce dernier. En effet, il introduit une étape de recherche globale optionnelle qui permet d'explorer davantage l'espace des solutions, évitant ainsi une convergence hâtive vers un minimum local. De plus, contrairement à Cs, l'algorithme Gps permet d'augmenter la longueur de pas lors d'un succès. Ainsi, la convergence vers un minimum est généralement accélérée. La recherche par motifs utilise un ensemble générateur positif pour déterminer ses directions de sondes locales. Une introduction à cette notion définie en 1954 par Davis [40] est présentée ici.

### Base positive et ensemble générateur positif

**Définition 2.1 (Ensemble générateur positif)** *Un ensemble fini de vecteurs* $\mathcal{D} = \{d_1, d_2, \ldots, d_{|\mathcal{D}|}\}$ *est un* ensemble générateur positif *de* $\mathbb{R}^n$ *si chaque vecteur* $v \in \mathbb{R}^n$ *peut s'exprimer sous la forme*

$$v = \sum_{i=1}^{|\mathcal{D}|} \lambda_i d_i, \ \ \textit{avec } \lambda_i \geq 0 \textit{ pour } i = 1, 2, \ldots, |\mathcal{D}|.$$

**Définition 2.2 (Base positive)** *Un ensemble générateur positif* $\mathcal{D}$ *est une* base positive *de* $\mathbb{R}^n$ *si aucun de ses sous-ensembles stricts n'est un ensemble générateur positif.*

Dans la figure 2.2, (b), (c) et (d) illustrent des ensembles générateurs positifs de $\mathbb{R}^2$, (b) et (c) sont des bases positives et seul (a) est une base de $\mathbb{R}^2$. Ces exemples sont tirés de [16].

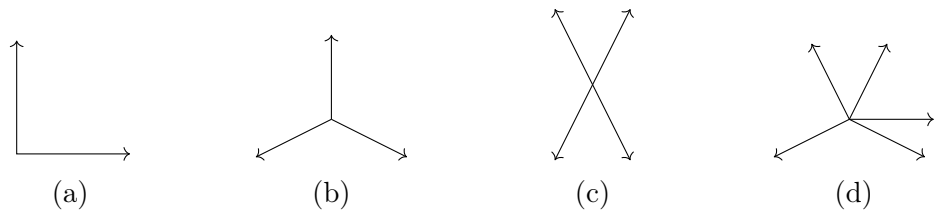



Figure 2.2 Exemples d'une base, de bases positives et d'ensembles générateurs positifs de $\mathbb{R}^2$.

### L'algorithme Gps

L'ensemble des directions possibles $D = \{d_1, d_2, \ldots, d_{|D|}\}$ pour l'algorithme Gps doit être un ensemble générateur positif de la forme $D = GZ$, où $G \in \mathbb{R}^{n\times n}$ est une matrice inversible et $Z \in \mathbb{Z}^{n\times|D|}$ est un ensemble générateur positif. Ces restrictions sont requises pour l'analyse de convergence. À l'itération $k$, la recherche par motifs s'effectue sur un sous-ensemble fini du treillis $M^k$ défini par

$$M^k := \{x^k + \delta^k Dy : y \in \mathbb{N}^{|D|}\} \tag{2.3}$$

où $x^k$ est la meilleure solution connue et $\delta^k \in \mathbb{R}_+^*$ est la longueur de pas, désormais appelée la taille du treillis. La première version de l'algorithme de recherche par motifs a été conçue pour résoudre des problèmes de la forme (2.1) sans contrainte. Cette version est résumée à la figure 2.3. On y introduit les paramètres $w_+$, $w_-$ et $\tau$ qui offrent une mise à jour de $\delta^k$ plus riche et généralement plus performante que celle de Cs.

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

0. Initialisation

| | |
|---|---|
| $\delta^0 \in (0, \infty)$ | taille du treillis initial |
| $D = GZ$ | ensemble générateur positif |
| $\tau \in (1, \infty) \cap \mathbb{Q}$ | coefficient d'ajustement de la taille du treillis |
| $w_+ \in \mathbb{N}, w_- \in \mathbb{Z}^- \backslash \{0\}$ | seuils pour la mise à jour de la taille du treillis |
| $\epsilon_{stop} \in (0, \infty)$ | critère d'arrêt |
| $k \leftarrow 0$ | compteur d'itération |

1. Recherche globale (optionnelle)

   si $f(y) < f(x^k)$ pour un $y \in S^k$, un sous-ensemble fini de $M^k$
   poser $x^{k+1} \leftarrow y$ et aller à l'étape 3
   sinon aller à l'étape 2

2. Sonde locale

   choisir un ensemble générateur positif $D^k \subseteq D$
   si $f(y) < f(x^k)$ pour un $y \in P^k := \{x^k + \delta^k d : d \in D^k\}$, poser $x^{k+1} \leftarrow y$
   sinon poser $x^{k+1} \leftarrow x^k$

3. mise à jour de $\delta^k$ et terminaison

   si $\delta^k \geq \epsilon_{stop}$
   $\delta^{k+1} \leftarrow \tau^{w^k} \delta^k$, où $w^k \in \begin{cases} \{0, 1, \ldots, w_+\} & \text{si l'itération est un succès} \\ \{w_-, w_- + 1, \ldots, -1\} & \text{sinon} \end{cases}$
   $k \leftarrow k + 1$ et retourner à 1
   sinon arrêter

Figure 2.3 Algorithme Gps pour un problème d'optimisation sans contrainte.

En posant $D = [I_n \ \ -I_n]$, $w_+ = 0$, $w_- = -1$, $\tau = 2$, $S^k = \emptyset$ pour tout $k$ et en désactivant l'interruption opportuniste, on retrouve l'algorithme de recherche par coordonnées.

La recherche globale de l'algorithme permet d'évaluer la fonction objectif à des points sur le treillis qui n'interviennent pas dans la sonde locale. Le principal avantage de cette recherche est d'éviter la convergence vers des minima locaux non globaux. Le sous-ensemble fini des points à évaluer $S^k \subset M^k$ est à la discrétion de l'utilisateur. Ce dernier peut générer $S^k$ par ses propres stratégies de recherche ou en utilisant des stratégies plus génériques, comme l'échantillonnage par hypercubes latins (Lhs) [67], la recherche à voisinage variable (Vns) [68], la recherche aléatoire ou l'utilisation de modèles substituts (voir la section 2.4).

L'algorithme Gps a été amélioré au fil des ans afin de pouvoir gérer les contraintes de bornes [61], les contraintes linéaires [62], les contraintes générales [13] ainsi que les problèmes à variables de catégories [12].

L'analyse de convergence de l'algorithme GPS n'est pas présentée ici, puisqu'elle n'est pas aussi riche que celle de MADS exposée à la section suivante. Elle est toutefois présentée dans [82]. Même si l'algorithme assure l'existence d'un point d'accumulation $\hat{x}$, rien ne garantit que ce point est un minimum local. En effet, puisque la recherche s'effectue seulement sur un ensemble fini de directions $D$, il est possible qu'il existe un seuil $\epsilon > 0$ et une direction $d \notin D$ tels que $f(\hat{x}+\beta d) < f(\hat{x})$ pour tout $\beta \in (0, \epsilon]$. L'algorithme suivant, MADS, contourne ce problème en proposant des directions de recherche plus riches que celles de GPS.

## 2.3 Recherche par treillis adaptatifs (MADS)

L'algorithme MADS est décrit ici comme il a été présenté par Audet et Dennis [14]. Il a été initialement conçu pour résoudre les problèmes d'optimisation contraints de la forme (2.1). La motivation principale derrière le développement de MADS était de généraliser l'algorithme GPS sans restreindre la sonde locale à un nombre fini de directions.

MADS reprend une bonne partie de la terminologie de GPS. Soit $D = \{d_1, d_2, \ldots, d_{|D|}\}$ un ensemble générateur positif de la forme $D = GZ$, où $G \in \mathbb{R}^{n\times n}$ est une matrice inversible et $Z \in \mathbb{Z}^{n\times|D|}$ est un ensemble générateur positif. À l'itération $k$, la recherche par treillis adaptatifs s'effectue sur un sous-ensemble fini du treillis $M^k$ défini par

$$M^k := \{x + \delta^k Dz : z \in \mathbb{N}^{|D|},\ x \in V^k\} \tag{2.4}$$

où $\delta^k \in \mathbb{R}_+^*$ est le paramètre de taille du treillis. L'ensemble des points visités $V^k \subset X$ représente les points où, avant l'itération $k$, la fonction objectif $f$ et les contraintes $c_j, j \in \{1, 2, \ldots, m\}$, ont été évaluées. On introduit $V^k$ pour assurer que l'ensemble des points évalués antérieurement soit sur le treillis.

L'étape de recherche globale de MADS n'est pas présentée dans cette section, puisqu'elle est identique à celle de l'algorithme GPS. La grande différence entre ces deux algorithmes intervient dans la sonde locale. En effet, MADS introduit le paramètre $\Delta^k \in \mathbb{R}_+^*$, le pas de sonde à l'itération $k$, dans l'optique de ne plus être restreint à un nombre fini de directions de recherche locale. Ce paramètre doit respecter les conditions suivantes :

$$\delta^k \leq \Delta^k \text{ pour tout } k; \tag{2.5}$$

$$\lim_{k\in K} \delta^k = 0 \iff \lim_{k\in K} \Delta^k = 0 \text{ pour tout sous-ensemble infini d'indices } K. \tag{2.6}$$

Au lieu d'être constant tout au long de l'algorithme, l'ensemble des directions possibles est

mis à jour à chaque itération $k$. La notation suivante est introduite pour définir cet ensemble :

$$\mathbb{D}^k := \{d : d = Dy \neq 0,\ y \in \mathbb{N}^{|D|},\ \delta^k \|d\|_\infty \leq \Delta^k \max\{\|d'\|_\infty : d' \in D\}\}.$$

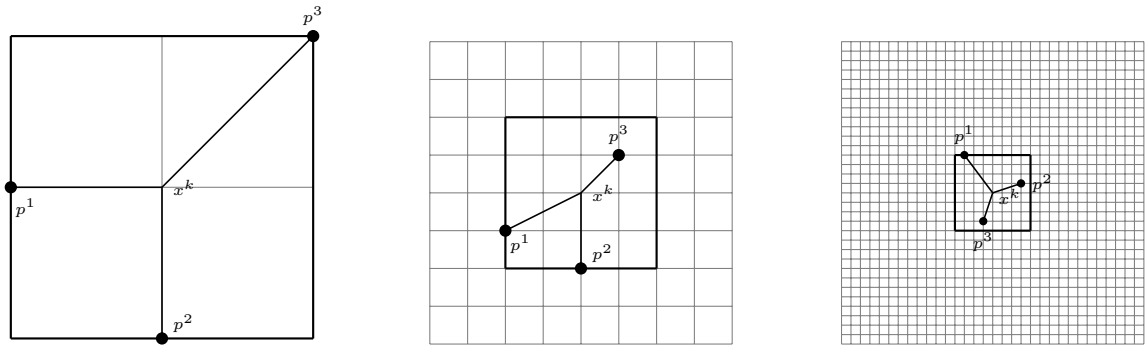
Ces directions, formées à partir des éléments de $D$, permettent de former l'ensemble des points du treillis délimités par le cadre de sonde autour de $x^k$, la meilleure solution à l'itération $k$. Ce cadre est défini par

$$F^k := \{x^k\} \cup \{x^k + \delta^k d : d \in \mathbb{D}^k\}$$

et, par définition, $F^k \subset M^k$. La recherche locale de MADS s'effectue sur un sous-ensemble de $F^k$ défini par

$$P^k := \{x^k + \delta^k d : d \in D^k\} \tag{2.7}$$

où $D^k \subseteq \mathbb{D}^k$ est un ensemble générateur positif. La figure 2.4, extraite de [16], présente des ensembles $P^k = \{p^1, p^2, p^3\}$ tirés de différents cadres de sonde $F^k$ dans $\mathbb{R}^2$. Les traits plus opaques délimitent le cadre $F^k$. Ainsi, les points de $P^k$ se trouvent sur la grille fine à l'intérieur des lignes opaques.




$\delta^k = 1$, $\Delta^k = 1$ $\qquad$ $\delta^k = \frac{1}{4}$, $\Delta^k = \frac{1}{2}$ $\qquad$ $\delta^k = \frac{1}{16}$, $\Delta^k = \frac{1}{4}$

Figure 2.4 Exemples de cadres de sonde pour différentes valeurs de $\delta^k$ et $\Delta^k$.

Les équations (2.5) et (2.6) offrent la possibilité de faire tendre la taille du treillis $\delta^k$ plus rapidement vers zéro que la taille du pas de sonde $\Delta^k$. Ainsi, l'algorithme MADS permet de produire un ensemble de directions de sonde plus riche que ses prédécesseurs. La figure 2.4 illustre un cas où $\Delta^k = \min\{\delta^k, \sqrt{\delta^k}\}$. On aperçoit que le nombre de directions possibles augmente lorsque $\delta^k$ diminue. Tout dépendant des paramètres d'initialisation de l'algorithme, l'union pour tout $k$ des directions normalisées peut être dense dans l'hypersphère unité.

La figure 2.5 résume la recherche par treillis adaptatifs tel que présentée dans [14]. On y

introduit la fonction barrière

$$f_\Omega(x) := \begin{cases} f(x) \text{ si } x \in \Omega \\ +\infty \text{ si } x \notin \Omega \end{cases} \tag{2.8}$$

qui permet d'éviter l'évaluation de la fonction objectif lorsqu'un point n'est pas dans l'ensemble des solutions réalisables. L'algorithme minimise la fonction $f_\Omega$ sans contrainte. Cette gestion des contraintes ainsi que d'autres gestions plus sophistiquées sont abordées en détail à la section 2.5.

Étant donnés $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$ et un point de départ $x^0 \in \Omega$

0. Initialisation

| | |
|---|---|
| $\Delta^0 \in (0, \infty)$ | taille du pas de sonde initiale |
| $\delta^0 \in (0, \Delta^0]$ | taille du treillis initial |
| $\tau \in (0,1) \cap \mathbb{Q}$ | coefficient d'ajustement de la taille du treillis |
| $w_+ \in \mathbb{N}, w_- \in \mathbb{Z}^- \backslash \{0\}$ | seuils pour la mise à jour de la taille du treillis |
| $\epsilon_{stop} \in (0, \infty)$ | critère d'arrêt |
| $k \leftarrow 0$ | compteur d'itération |

1. Recherche globale (optionnelle)

   si $f_\Omega(y) < f_\Omega(x^k)$ pour un $y \in S^k$, un sous-ensemble fini de $M^k$
   poser $x^{k+1} \leftarrow y$ et aller à l'étape 3
   sinon aller à l'étape 2

2. Sonde locale

   choisir un ensemble générateur positif $D^k$ tel que $P^k \subseteq F^k$
   si $f_\Omega(y) < f_\Omega(x^k)$ pour un $y \in P^k$, poser $x^{k+1} \leftarrow y$
   sinon poser $x^{k+1} \leftarrow x^k$

3. Mise à jour des paramètres et terminaison

   si $\delta^k \geq \epsilon_{stop}$
   $\delta^{k+1} \leftarrow \tau^{w^k} \delta^k$, où $w^k \in \begin{cases} \{0, 1, \ldots, w_+\} & \text{si l'itération est un succès} \\ \{w_-, w_- + 1, \ldots, -1\} & \text{sinon} \end{cases}$
   $\Delta^{k+1}$ est mis à jour selon (2.5) et (2.6)
   $k \leftarrow k + 1$ et retourner à 1
   sinon arrêter

Figure 2.5 Algorithme de recherche par treillis adaptatifs (MADS).

L'algorithme MADS a été amélioré au fil du temps avec, par exemple, la gestion des problèmes d'optimisation bi-objectif [24] (BIMADS) et la gestion des problèmes à plus de 50 variables à l'aide d'une décomposition parallèle de l'espace [7, 10] (PSD-MADS). De plus, des méthodes faisant appel aux modèles substituts présentés à la section 2.4 ont été insérées à l'étape de sonde locale afin de réduire le nombre d'évaluations des fonctions du

problème (2.1) [18, 35, 80].

### Analyse de convergence

L'analyse de convergence présentée ici est une version abrégée. La version complète est disponible dans l'article [14]. Celle-ci est basée sur le calcul de fonctions non lisses de Clarke [32] et la généralisation des notions de cônes tangents de Jahn [54]. Les définitions suivantes sont nécessaires afin d'énoncer le théorème principal de l'analyse de convergence de MADS.

**Définition 2.3 (Dérivée généralisée de Clarke)** *Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction Lipschitz près de $x \in \mathbb{R}^n$. La* dérivée généralisée de Clarke *définie en $x$ dans la direction $d \in \mathbb{R}^n$ est*

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

Cette dérivée vient généraliser la notion de dérivée directionnelle. Elle est principalement utile dans les cas où $f$ n'est ni différentiable ni convexe. Le concept suivant généralise la notion de cône tangent lorsque les fonctions définissant les contraintes ne sont pas différentiables.

**Définition 2.4 (Cône hypertangent)** *Un vecteur $d \in \mathbb{R}^n$ est dit* hypertangent *à l'ensemble $\Omega \subset \mathbb{R}^n$ au point $x \in \Omega$ si et seulement si il existe $\epsilon \in \mathbb{R}^*_+$ tel que*

$$y + tw \in \Omega \text{ pour tout } y \in \Omega \cap \mathcal{B}_\epsilon(x),\ w \in \mathcal{B}_\epsilon(d) \text{ et } 0 < t < \epsilon.$$

*L'ensemble de tous les vecteurs hypertangents à $\Omega$ en $x$ est nommé le* cône hypertangent *à $\Omega$ au point $x$ et est noté $T^H_\Omega(x)$.*

$T^H_\Omega(x)$ est toujours un ensemble ouvert et convexe. Il prend toute son importance en DFO, puisque, contrairement au cône tangent, aucun gradient n'intervient dans sa définition. Le concept suivant définit le point d'accumulation vers lequel une sous-suite des centres de sondes produits par MADS converge.

**Définition 2.5 (Sous-suite raffinante et point raffiné)** *Soit $K$ un sous-ensemble d'indices. Une sous-suite convergente de centres de sonde $\{x^k\}_{k \in K}$ est dite une* sous-suite raffinante *si et seulement si chaque itération $k \in K$ est un échec et si $\lim_{k \in K} \delta^k = 0$. La limite $\hat{x}$ de $\{x^k\}_{k \in K}$ est dite un* point raffiné.

Avec ces notions, on peut désormais définir une direction raffinante.

**Définition 2.6 (Direction raffinante)** *Soit $\{x^k\}_{k \in K}$ une sous-suite raffinante et $\hat{x}$ son*

*point raffiné. Une direction $d$ est dite raffinante si et seulement si il existe un sous-ensemble infini d'indices $L \subseteq K$ avec des directions de sonde $d^k \in D^k$ tel que* $\lim_{k \in L} \frac{d^k}{\|d^k\|} = \frac{d}{\|d\|}$.

L'analyse de convergence de MADS repose sur les hypothèses suivantes :

— un point de départ est connu, *i.e.*, $x^0 \in \Omega$ pour MADS sous barrière extrême ou $x^0 \in X$ pour MADS sous barrière progressive (voir la section 2.5 pour plus de détails) ;

— toutes les solutions générées par MADS appartiennent à un ensemble compact.

La deuxième hypothèse est satisfaite, par exemple, lorsque les ensembles de niveaux de la fonction objectif $f$ sont bornés. Sous cette hypothèse, la proposition suivante assure l'existence d'au moins une sous-suite raffinante.

**Proposition 2.7** *Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction avec des ensembles de niveaux bornés. En appliquant l'algorithme* MADS *sur $f$, les paramètres de taille du treillis et du pas de sonde satisfont la condition suivante :*

$$\liminf_{k \to \infty} \Delta^k = \liminf_{k \to \infty} \delta^k = 0.$$

Celle-ci garantit l'existence d'un point raffiné, qui est essentielle au théorème suivant.

**Théorème 2.8** *Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction Lipschitz près d'un point raffiné $\hat{x} \in \Omega$, et $\hat{d} \in T_\Omega^H(\hat{x})$ une direction raffinante. Alors, la dérivée généralisée de Clarke de $f$ en $\hat{x}$ dans la direction hypertangente $\hat{d}$ est non négative, i.e., $f^\circ(\hat{x}; \hat{d}) \geq 0$.*

Le théorème 2.8 est démontré en détail dans [14]. Il généralise, dans le cas lisse, la condition nécessaire stipulant qu'il n'y a pas de direction de descente dans le cône des solutions réalisables. La convergence de l'algorithme MADS découle directement de celui-ci.

## 2.4 Optimisation à l'aide de fonctions substituts et de modèles

En DFO et en BBO, il est fréquent d'évaluer des fonctions qui nécessitent des secondes, des minutes, des heures ou même des jours pour évaluer un point [8]. Dans ces cas, il est recommandé d'utiliser un modèle substitut du problème pour orienter la recherche dans l'espace des solutions [19, 29]. On donne ci-dessous une définition informelle inspirée de [16] d'un modèle substitut. Des exemples suivront pour illustrer ce concept.

**Définition 2.9 (Problème, fonctions et modèle substituts (*Surrogates*))** *Soit $\tilde{c}$ le vecteur des contraintes $\tilde{c}_j$, $j \in J = \{1, 2, \ldots, m\}$, et $c$ le vecteur des contraintes $c_j$, $j \in J$. Le*

*problème d'optimisation*

$$\begin{aligned} &\min_{x \in X \subset \mathbb{R}^n} \tilde{f}(x) \\ &s.c. \quad \tilde{c}(x) \leq 0, \end{aligned} \tag{2.9}$$

*est dit un* problème substitut *au problème* (2.1) *si* $\tilde{f} : X \subset \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ *et* $\tilde{c} : X \subset \mathbb{R}^n \mapsto (\mathbb{R} \cup \{\infty\})^m$*, partagent certaines similarités avec* $f$ *et* $c$*, mais qu'elles sont moins coûteuses en temps de calcul.* $\tilde{f}$ *et* $\tilde{c}$ *sont dites les* fonctions substituts *aux fonctions* $f$ *et* $c$ *et l'ensemble des fonctions substituts s'appelle le* modèle substitut*. Le problème* (2.1) *est alors appelé le «vrai» problème.*

La définition de modèle substitut est volontairement vague afin d'englober l'ensemble des modèles du problème (2.1). Il existe deux types de modèles substituts : les modèles dynamiques et les modèles statiques [8]. Toutefois, la terminologie pour exprimer ces deux familles de modèles varie selon les ouvrages. En effet, les termes statique et dynamique sont respectivement remplacés par non adaptatif et adaptatif dans [58], par physique et fonctionnel dans [36] ainsi que par mécaniste et empirique dans [70]. L'expression modèle substitut varie elle aussi dans la littérature, changeant pour un modèle à basse fidélité [48, 64, 73]. Dans ces cas, le problème (2.1) est appelé un modèle à haute fidélité.

Les modèles dynamiques sont mis à jour à chaque itération, tandis que les modèles statiques restent constants tout au long de la recherche. Toutefois, la recherche directe assistée d'un modèle statique est similaire à celle assistée de modèles dynamiques. Les modèles substituts peuvent intervenir de différentes façons dans les algorithmes de recherche directe. Ils sont principalement utilisés pour orienter un algorithme de recherche globale ou pour ordonner la liste de points d'essai qui intervient dans la sonde locale. La figure 2.6, inspirée de [29], généralise les méthodes de recherche directe assistées d'un modèle substitut.

Étant données $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$, $c_j : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$, $j \in \{1, 2, \ldots, m\}$, et leurs fonctions substituts $\tilde{f} : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$, $\tilde{c}_j : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$, $j \in \{1, 2, \ldots, m\}$.

0. Initialisation des paramètres algorithmiques
1. Recherche globale à l'aide du modèle substitut (optionnelle)
   - générer une liste $\mathcal{L}$ de points d'essai en travaillant avec le modèle substitut
   - évaluer le vrai problème aux points de $\mathcal{L}$ de manière opportuniste
   - aller à l'étape 3 si une meilleure solution est rencontrée
   - sinon, aller à l'étape 2
2. Recherche locale
   - générer une liste de points d'essai $\mathcal{P}$ dans le voisinage de la meilleure solution
   - utiliser le modèle substitut pour ordonner les points de $\mathcal{P}$, du plus prometteur au moins prometteur
   - évaluer le vrai problème aux points de $\mathcal{P}$ de manière opportuniste
3. Mise à jour du modèle substitut (cas modèle dynamique)
   - mettre à jour le modèle substitut en tenant compte des valeurs $f, c_1, \ldots, c_m$ des points évalués aux étapes 1 et 2
4. Mise à jour des paramètres
   - si le critère d'arrêt n'est pas atteint
     - mettre à jour les paramètres algorithmiques et retourner à 1
   - sinon arrêter

Figure 2.6 Structure de recherche directe assistée d'un modèle substitut.

### Modèles dynamiques

Les modèles dynamiques sont mis à jour à chaque itération en considérant l'information recueillie aux itérations précédentes. Ils sont construits dans le but de fournir une approximation, soit locale ou globale, peu coûteuse du vrai problème, *i.e.*, $\tilde{f} \approx f$, $\tilde{c} \approx c$. Une vue d'ensemble des modèles dynamiques populaires en BBO et en DFO non lisse sont disponibles dans les articles [56] et [17] respectivement. Les approximations locales sont généralement des modèles de régression polynomiale [70], dont le modèle quadratique [6, 35, 37, 42], et des régressions multivariées par splines adaptatives [50]. Les approximations globales, quant à elles, sont principalement des processus gaussiens [74, 76], du krigeage [28, 29, 31, 41, 48, 55] et des fonctions de base radiale [28, 57, 60, 72, 77, 78]. Ces modèles dynamiques sont comparés avec MADS dans [19]. On y présente aussi comment exploiter l'information de plus d'un modèle pour une recherche directe.

On introduit ci-dessous le modèle quadratique, puisque le modèle proposé à la section 3.1 s'inspire fortement de celui-ci. En fait, le modèle proposé est un modèle quadratique auquel

on ajoute une variable. Pour cette raison, le modèle quadratique est présenté ici de manière abrégée, tandis que le modèle proposé est présenté plus formellement à la section 3.1.

### Modèles quadratiques

Le modèle quadratique est fréquemment utilisé pour approximer localement une fonction $g : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$. On utilise ici la notation $g$ pour parler d'une fonction quelconque, puisqu'un modèle quadratique peut être construit pour chacune des fonctions du problème (2.1), *i.e.*, $f,\ c_j,\ j \in \{1, 2, \ldots, m\}$. Les points candidats à sa construction dépendent d'une région de confiance [4, 33, 34, 46]. La forme généralisée d'un modèle quadratique dans $\mathbb{R}^n$ est

$$Q(x) = \alpha_0 + \alpha^\top x + \frac{1}{2} x^\top H x, \tag{2.10}$$

où $H = H^\top$ et $x = [x_1, x_2, \ldots, x_n]^\top$. Ce modèle possède $q+1 = (n+1)(n+2)/2$ scalaires, soit 1 pour $\alpha_0$, $n$ pour $\alpha$ et $n(n+1)/2$ pour $H$. En utilisant une base de l'espace des polynômes de degré 2, l'équation (2.10) peut se réécrire sous la forme $Q(x) = \alpha^\top \rho(x)$ , où

$$\begin{aligned} \rho(x) &= (\rho_0(x), \rho_1(x), \ldots, \rho_q(x))^\top \\ &= \left(1, x_1, x_2, \ldots, x_n, \frac{x_1^2}{2}, \frac{x_2^2}{2}, \ldots, \frac{x_n^2}{2},\ x_1x_2,\ x_1x_3, \ldots,\ x_{n-1}x_n\right)^\top \end{aligned} \tag{2.11}$$

et $\alpha \in \mathbb{R}^{q+1}$. Les définitions suivantes sont nécessaires pour caractériser le modèle quadratique. Il s'agit de conditions à respecter afin d'assurer l'existence et l'unicité d'un modèle quadratique.

**Définition 2.10 (Ensemble propice à la régression quadratique)** *Soit un ensemble de points* $\mathbb{Y} = \{y^0, y^1, \ldots, y^p\} \subset \mathbb{R}^n$ *avec* $p \geq q$ , $q + 1 = (n + 1)(n + 2)/2$ *et* $g(\mathbb{Y}) = (g(y^0), g(y^1), \ldots, g(y^p))^\top$, *où* $g : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$. *L'ensemble* $\mathbb{Y}$ *est* propice à la régression quadratique *si la matrice*

$$M(\rho, \mathbb{Y}) = \begin{bmatrix} \rho_0(y^0) & \rho_1(y^0) & \ldots & \rho_q(y^0) \\ \rho_0(y^1) & \rho_1(y^1) & \ldots & \rho_q(y^1) \\ \vdots & \vdots & \vdots & \vdots \\ \rho_0(y^p) & \rho_1(y^p) & \ldots & \rho_q(y^p) \end{bmatrix} \in \mathbb{R}^{(p+1)\times(q+1)} \tag{2.12}$$

*est de rang* $q + 1$ *et que* $g(\mathbb{Y}) \in \mathbb{R}^{p+1}$.

Le modèle quadratique de Frobenius [39, 75] est une alternative à la régression quadratique

dans les cas où l'on possède moins de $q+1$ points pour construire un modèle.

**Définition 2.11 (Ensemble propice au modèle quadratique de Frobenius)** *Soit* $g : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$, $\rho_L(x) = (1, x_1, x_2, \ldots, x_n)^\top$ *et* $\rho_Q(x) = (\frac{x_1^2}{2}, \frac{x_2^2}{2}, \ldots, \frac{x_n^2}{2},\ x_1x_2,\ x_1x_3, \ldots, x_{n-1}x_n)^\top$. *Un ensemble de points* $\mathbb{Y} = \{y^0, y^1, \ldots, y^p\} \subset \mathbb{R}^n$, $n+1 < p+1 < (n+1)(n+2)/2$, *est* propice au modèle quadratique de Frobenius *si la solution du problème suivant est unique :*

$$\begin{aligned} &\min_{\alpha_L, \alpha_Q} \frac{1}{2} \|\alpha_Q\|^2 \\ &s.c.\quad g(y^i) = \rho_L(y^i)\alpha_L + \rho_Q(y^i)\alpha_Q \quad \textit{pour } i = 0, 1, \ldots, p\ , \end{aligned} \tag{2.13}$$

*où* $\alpha_L \in \mathbb{R}^{n+1}$, $\alpha_Q \in \mathbb{R}^N$ *et* $N = n(n+1)/2$.

On note que le problème (2.13) est équivalent à minimiser la norme de Frobenius $\|\cdot\|_F$ de la matrice Hessienne $H$ du modèle $Q(x)$ sous la forme (2.10). En effet, puisque $H$ est symétrique, le coefficient devant $x_ix_j$, $i \neq j$, est le même que celui devant $x_jx_i$. Ainsi, si $H = \{h_{i,j}\}_{1 \leq i,j \leq n}$, on a

$$\frac{1}{2}x^\top Hx = \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n x_i h_{i,j} x_j = \frac{1}{2}\sum_{i=1}^n x_i^2 h_{i,i} + \sum_{i=1}^n \sum_{j \neq i}^n x_i x_j h_{i,j} = \rho_Q(x)\alpha_Q.$$

Minimiser $\|\alpha_Q\|^2$ est donc équivalent à minimiser la somme des coefficients au carré de la matrice $H$, *i.e.*, minimiser $\|H\|_F$.

Pour construire $Q(x)$, il suffit d'identifier le paramètre $\alpha = [\alpha_L\ \alpha_Q]^\top$. Pour ce faire, on nécessite un ensemble de $p+1$ points $\mathbb{Y} = \{y^0, y^1, \ldots, y^p\}$, $p > n$. On distingue quatre cas :

i. Si $n < p < q$ et $\mathbb{Y}$ est propice au modèle quadratique de Frobenius, $\alpha$ est l'unique solution du système (2.13).

ii. Si $p \geq q$ et $\mathbb{Y}$ est propice à la régression quadratique, alors $\alpha$ est la solution de

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

où $M(\rho, \mathbb{Y})$ est la matrice (2.12) et $g(\mathbb{Y}) = (g(y^0), g(y^1), \ldots, g(y^p))^\top$.

iii. Si $n < p$ mais $\mathbb{Y}$ n'est ni propice au modèle quadratique de Frobenius et ni propice à la régression quadratique, on identifie un modèle quadratique unique à l'aide de la méthode présentée à la section 3.3.

iv. Si $p \leq n$, aucun modèle quadratique n'est construit, puisqu'on manque d'information pour construire un modèle quadratique qui couvre bien l'espace. Il serait toutefois possible de construire un modèle linéaire si $p = n$ et $\mathbb{Y}$ forme un simplexe. Ce modèle n'est toutefois pas présenté dans ce mémoire, puisque le cas $p = n$ ne survient que très

rarement dans le contexte où on utilise ces modèles dynamiques.

### Modèles statiques

Un modèle statique est habituellement une simplification du vrai problème. Cette simplification peut être obtenue de plusieurs façons. Par exemple,

— Négliger la friction d'un problème physique ;

— Réduire le critère d'arrêt d'une méthode numérique pour un arrêt prématuré ;

— Utiliser un maillage plus grossier pour la méthode des éléments finis ;

— En mécanique des fluides, les équations de Navier-Stokes peuvent être remplacées par les équations d'Euler.

L'objectif du modèle statique est d'orienter la recherche vers les points les plus prometteurs. Ainsi, les fonctions statiques ne sont pas nécessairement des approximations des fonctions du vrai problème. Par exemple, dans le cas non contraint, la fonction $\tilde{f} \approx \frac{f}{100}$ n'approxime pas $f$, mais elle est excellente pour orienter la recherche, car les optimums locaux de ces deux fonctions sont très près l'un de l'autre. Une fonction statique de ce genre est utilisée dans [23], où la vraie fonction est le temps nécessaire pour effectuer une série de tâches et la fonction statique est le temps nécessaire pour effectuer une portion de ces tâches.

### Corrections aux modèles statiques

Une alternative aux modèles statiques est d'apporter une correction à ceux-ci à chaque itération. Ces nouveaux modèles substituts sont alors de nature dynamique, mais ils considèrent l'information fournie par les modèles statiques. Les corrections les plus communes sont additives ($g(x) \approx \tilde{g}(x) + A(x)$, $A : \mathbb{R}^n \mapsto \mathbb{R}$) ou multiplicatives ($g(x) \approx B(x)\tilde{g}(x)$, $B : \mathbb{R}^n \mapsto \mathbb{R}$) [5, 43, 64]. Une autre méthode populaire est celle du *Space Mapping* [26, 27], qui consiste à corriger l'espace des solutions du modèle statique afin que cette dernière se rapproche de celle du vrai problème. D'autres techniques de correction de modèles statiques sont présentées dans [73]. On note que ces procédés sont généralement utilisés dans des contextes où les dérivées des modèles statiques sont connues ou estimables. Ces méthodes ne s'appliquent donc pas tous quand le modèle statique est une boîte noire.

## 2.5 La gestion des contraintes

Habituellement, trois types de contraintes interviennent dans un problème de DFO/BBO : les contraintes non relaxables, les contraintes relaxables et quantifiables et les contraintes «cachées». On utilise ici la taxonomie KARQ proposée par Le Digabel et Wild [59] pour classer les types de contraintes en DFO et en BBO. Elles sont classées en fonction de si elles sont connues (K) ou non (H), vérifiable à priori (A) ou à l'aide d'une simulation (S), relaxables (R) ou non (U), et si elles sont quantifiables (Q) ou non (N). On utilise la notation «*» lorsqu'une contrainte peut autant être caractérisée par un critère que par son inverse. Par exemple, une contrainte KAU* représente une contrainte connue, vérifiable à priori, non relaxable et quantifiable ou non.

On considère ici un problème de la forme (2.1).

i. L'ensemble $X$ représente les contraintes non relaxables. Elles doivent être respectées à chaque itération. Il s'agit des contraintes de types KAU*.

ii. Les contraintes relaxables et quantifiables $c_j$, $j \in \{1, 2, \ldots, m\}$, doivent être respectées à la solution finale. Toutefois, durant la recherche, une boîte noire peut être évaluée même si ces contraintes sont violées. Il s'agit des contraintes de types K*RQ. Dans certains cas, une contrainte $c_j$ peut être non quantifiable (K*RN). C'est le cas avec le problème STYRENE [9], où quatre des contraintes sont binaires. Cependant, les contraintes $c_j$ des problèmes tests de ce mémoire sont uniquement de types K*RQ.

iii. Les contraintes «cachées» apparaissent lorsque la boîte noire échoue, même pour un point $x \in X$. Par exemple, si la valeur qu'on cherche à minimiser est un réel et qu'à l'insu de l'utilisateur, la simulation fait intervenir une racine carrée, alors le terme sous la racine carrée doit être supérieur ou égal à zéro. Il s'agit des contraintes de type HSUN.

En DFO/BBO, deux méthodes sont principalement utilisées pour la gestion des contraintes K*RQ, soit la barrière extrême (EB) et la barrière progressive (PB). La fonction de violation de contraintes intervient dans ces deux méthodes.

**Définition 2.12** La fonction de violation de contraintes $h : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$ *associée aux contraintes* $c_j(x) \leq 0, j \in \{1, 2, \ldots, m\}$ *de type* K*RQ *est définie par*

$$h(x) := \begin{cases} \displaystyle\sum_{j=1}^{m} (\max(0, c_j(x)))^2 & \textit{si } x \in X, \\ +\infty & \textit{si } x \notin X. \end{cases} \tag{2.14}$$

### Barrière extrême

L'approche par EB, telle que présentée dans la première version de MADS [14], gère les contraintes relaxables comme des contraintes non relaxables. C'est également avec cette méthode qu'on gère les contraintes K*RN. Le principe de cette approche est simplement de minimiser la fonction barrière $f_\Omega(x)$ (voir (2.8)) sans contrainte au lieu de $f(x)$ sous $x \in \Omega$. Ainsi, l'algorithme n'explore que les solutions réalisables. Cette gestion de contraintes ne peut être utilisée qu'en présence d'une solution initiale réalisable. Toutefois, si aucun point de départ réalisable n'est disponible, l'utilisateur peut appliquer EB en deux phases. Cette alternative consiste d'abord à minimiser la fonction $h$ à partir d'un point de départ irréalisable $x^0 \in X \backslash \Omega$ jusqu'à l'obtention d'un point $\bar{x}$ tel que $h(\bar{x}) = 0$, *i.e.*, $\bar{x} \in \Omega$. On minimise ensuite la fonction $f_\Omega(x)$ à partir du nouveau point de départ $\bar{x}$.

### Barrière progressive

L'approche par PB [15], inspirée par la méthode du filtre [13, 47], gère les contraintes K*RQ à l'aide de la fonction $h$ et d'un seuil $h^k_{\max} \in \mathbb{R}_+$ mis à jour à la fin de chaque itération. On introduit la «barrière» $h^k_{\max}$ afin de rejeter l'évaluation de la fonction objectif aux points où la violation des contraintes est trop importante. De manière générale, on initialise $h^0_{\max} = \infty$. Afin d'évaluer la qualité d'une solution, on utilise le principe de dominance.

**Définition 2.13** *Un point réalisable* $x \in \Omega$ domine $y \in \Omega$, *noté* $x \prec_f y$, *si* $0 = h(x) = h(y)$ *et* $f(x) < f(y)$.

*Un point irréalisable* $x \in X \backslash \Omega$ domine $y \in X$, *noté* $x \prec_h y$, *si* $h(x) \leq h(y)$ *et* $f(x) \leq f(y)$ *avec au moins une inégalité stricte.*

*Un point* $x$ *d'un ensemble* $S \subset X$ *est* non dominé *s'il n'existe aucun* $y \in S$ *qui domine* $x$.

La figure 2.7 représente 12 points de $\mathbb{R}^n$ selon leurs valeurs $f$ et $h$. On y aperçoit 2 points réalisables, situés sur l'axe $h = 0$, et 10 points irréalisables, dont 4 non dominés. La meilleure solution réalisable est notée $x^F$, tandis que la meilleure solution irréalisable est notée $x^I$. Cette dernière est la solution avec la meilleure fonction objectif parmi les solutions irréalisables non dominées avec un $h$ plus petit ou égal à $h^k_{\max}$. La région grisâtre représente l'ensemble des points dominés et l'ensemble des points avec une valeur de $h$ supérieure à $h^k_{\max}$.

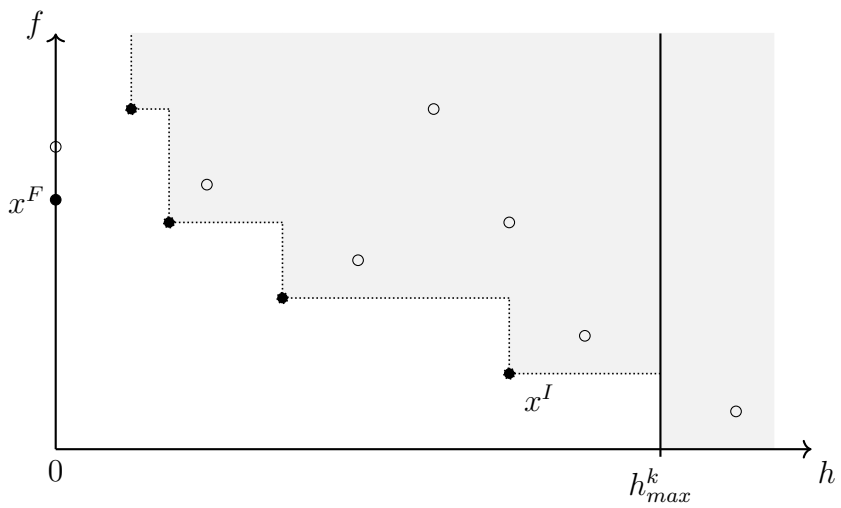



Figure 2.7 Les points dominés et non dominés sont respectivement symbolisés par des cercles vides et des cercles pleins. Ils sont représentés dans l'espace $(h,\ f)$.

La figure 2.8 donne un aperçu général de l'utilisation de cette gestion de contraintes pour un problème de DFO de la forme (2.1). $V^k$, tout comme à la section 2.3, représente l'ensemble des points visités au tout début de l'itération $k$. Un algorithme qui utilise PB donne généralement $x^F$ et $x^I$ comme valeurs en sortie. Cette information permet à l'utilisateur de juger si une légère relaxation de contraintes pourrait être favorable au problème.

Étant donnés $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$, $h : \mathbb{R}^n \mapsto \mathbb{R}_+ \cup \{\infty\}$ et un point de départ $x^0 \in X$

0. Initialisation
   - $h^0_{\max} = \infty$ seuil initial de la barrière progressive
   - $k \leftarrow 0$ compteur d'itération
1. Mise à jour
   - $x^F \in \underset{x}{\operatorname{argmin}}\{f(x) : x \in V^k, \quad x \in \Omega\}$
   - $x^I \in \underset{x}{\operatorname{argmin}}\{f(x) : x \in V^k, \quad x \in X \backslash \Omega, \quad h(x) \leq h^k_{\max}\}$
2. Recherche
   - si $y \prec_f x^F$ ou $y \prec_h x^I$ pour un $y \in S^k$, un sous-ensemble fini de l'espace
     poser $h^{k+1}_{\max} \leftarrow h(x^I)$
   - sinon, si $0 < h(y) < h(x^I)$ pour au moins un $y \in V^k \cup S^k$
     poser $h^{k+1}_{\max} \leftarrow \max\{h(v) : h(v) < h(x^I), v \in V^k \cup S^k\}$
   - sinon, poser $h^{k+1}_{\max} \leftarrow h(x^I)$ et perturber les paramètres de recherche de manière stratégique
3. Terminaison
   - arrêter si le critère d'arrêt est atteint
   - sinon, incrémenter $k \leftarrow k + 1$ et aller à 1

Figure 2.8 Structure de PB dans un contexte d'optimisation sans dérivées.

On compte une troisième approche pour gérer les contraintes, soit la barrière progressive-extrême [11]. Celle-ci consiste à gérer en premier temps les contraintes K*RQ par PB et ensuite par EB. Plus précisément, la gestion d'une contrainte $c_j \leq 0$, $j \in \{1, 2, \ldots, m\}$, passe de PB à EB lorsque $c_j$ est satisfaite en au moins un point. Finalement, lorsqu'il y a plusieurs contraintes $c_j$, il est possible de gérer une partie de celles-ci avec EB et l'autre partie avec PB. Cela peut être utile dans certaines situations, mais particulièrement dans les cas où certaines contraintes sont binaires.

## 2.6 Nomad

Le logiciel Nomad (*Nonlinear Optimization by Mesh Adaptive Direct Search*) [1, 20, 58] est une implémentation de l'algorithme Mads codée en C++. Il est conçu pour résoudre tout problème de la forme (2.1), mais spécifiquement pour les problèmes de DFO et de BBO. Cette section présente certaines caractéristiques de Nomad profitables pour ce mémoire. Les fonctionnalités et caractéristiques de Nomad sont détaillées dans [20, 58].

Pour générer l'ensemble des points d'essai dans le voisinage de la meilleure solution à l'itération $k$, MADS utilise un ensemble de directions $D^k$. NOMAD peut construire cet ensemble de différentes façons. On considère ici trois familles de directions de recherche : GPS, LTMADS et ORTHOMADS. GPS génère les directions avec la stratégie élaborée à la figure 2.3. L'algorithme GPS est donc intégré dans NOMAD. LTMADS (*Lower Triangular* MADS) [14] permet quant à lui de générer un ensemble de directions de sonde dense dans l'espace. Cependant, LTMADS est probabiliste. Ainsi, les directions qu'il génère d'une optimisation à l'autre ne sont pas les mêmes. Finalement, ORTHOMADS [2] offre les mêmes avantages que LTMADS, mais sans l'aspect aléatoire. En effet, les directions de sonde de ORTHOMADS sont générées de manière déterministe, permettant ainsi de reproduire les résultats d'une optimisation. De plus, les directions générées par ORTHOMADS recouvrent plus efficacement l'espace que celles produites par LTMADS.

NOMAD utilise les méthodes présentées à la section 2.5 pour gérer les contraintes *R*K. Il offre aussi la possibilité d'orienter la recherche à l'aide de modèles substituts. Ces modèles, lorsqu'ils sont dynamiques, sont construits à partir de l'information fournie par la cache. Celle-ci correspond à l'ensemble $V^k$ introduit à la section 2.3. Tous les points qui ont été évalués sont stockés dans la cache avec leurs valeurs $f$ et $c_j$, $j \in \{1, 2, \ldots, m\}$. La cache est consultée avant chaque nouvelle évaluation afin d'éviter d'évaluer deux fois le même point. Lorsqu'un modèle statique est utilisé pour orienter la recherche, une deuxième cache notée $\tilde{V}^k$ est créée. Celle-ci contient tous les points qui ont été évalués avec le modèle statique du problème. $\tilde{V}^k$ est consultée avant chaque nouvelle évaluation du modèle statique afin de ne pas évaluer deux fois le modèle statique au même point.

NOMAD propose plusieurs méthodes de recherche globale. Parmi celles-ci, on retrouve la recherche spéculative, LHS, VNS, la recherche par modèle quadratique et la recherche par Nelder-Mead. La recherche spéculative [14] consiste à regarder plus loin dans la direction du succès après une amélioration de la solution. Cette recherche est habituellement activée, puisqu'elle est peu coûteuse en temps de calcul et accélère généralement la convergence pratique de l'algorithme.

LHS est utilisée au début de la recherche directe. Cette méthode consiste à explorer une liste de points quasi-aléatoire sur l'espace des solutions réalisables. LHS est généralement utile lorsqu'aucun point réalisable n'est disponible. La méthode pour générer les points est présentée dans [67].

VNS [9, 51, 68] perturbe la meilleure solution connue à l'itération $k$ afin d'explorer une liste

de $\xi$ points situés à l'extérieur de la sonde locale. Ensuite, à partir des $\nu$ meilleurs points, $\nu \leq \xi$, on effectue une descente afin d'identifier de nouveaux minima locaux. Généralement, ces descentes se terminent prématurément pour éviter d'évaluer trop de fois le vrai problème. Lorsque VNS est utilisé avec un modèle statique, l'exploration et la descente sont effectuées avec le modèle statique. Seul le point le plus prometteur est évalué au vrai problème.

La recherche par modèle quadratique [35] consiste à construire, à l'itération $k$, un modèle aux alentours de $x^F$ et un aux alentours de $x^I$, où $x^F$ et $x^I$ sont définies comme à la figure 2.8. L'algorithme MADS est ensuite lancé sur les deux modèles. Cette méthode génère au plus quatre points, soit la meilleure solution réalisable et la solution irréalisable avec la plus petite valeur de $h$ de chaque modèle. Ces points sont par la suite évalués au vrai problème de manière opportuniste.

La recherche par Nelder-Mead [71] est un algorithme d'optimisation sans dérivées introduit en 1965 pour résoudre des problèmes d'optimisations sans contrainte sur $\mathbb{R}^n$. La recherche globale par Nelder-Mead adaptée à MADS, MADS-NM, ajuste cette méthode pour tout problème de la forme (2.1). MADS-NM explore globalement le problème à l'aide de simplexes et d'une alternative à l'algorithme Nelder-Mead qui permet de gérer les contraintes. Cette méthode est présentée en détail par Audet et Tribes dans [25].

Lorsqu'un substitut statique est fourni par l'utilisateur, NOMAD offre les deux possibilités suivantes pour orienter la sonde locale : utiliser l'information du modèle statique ou ignorer ce dernier et utiliser à la place l'information d'un modèle dynamique mis à jour à chaque itération. Toutefois, NOMAD ne propose aucun modèle substitut dynamique qui utilise l'information fournie par le modèle statique. Le chapitre suivant présente une nouvelle possibilité pour orienter la sonde locale de NOMAD, soit un modèle quadratique qui tient compte du modèle substitut statique fourni par l'utilisateur.

# CHAPITRE 3 INTÉGRATION D'UN SUBSTITUT STATIQUE AU MODÈLE QUADRATIQUE

Ce chapitre décrit la méthode de recherche locale proposée dans ce projet, soit la sonde locale orientée par un modèle hybride quadratique (MHQ). Avant de présenter ce modèle et d'expliquer en quoi il est un bon candidat, on définit ce qu'est un modèle hybride.

**Définition 3.1 (Modèle hybride)** *Soit* $\tilde{g} : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$, $\tilde{g} \neq 0$, *un modèle statique de* $g : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$. *Un* modèle hybride $\hat{g} : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$ *de* $g$ *est un modèle dynamique auquel l'information de* $\tilde{g}$ *est ajoutée, i.e.,* $\hat{g}$ *dépend de* $x = [x_1, x_2, \ldots, x_n]^\top$, *mais aussi de* $\tilde{g}(x) \in \mathbb{R}$.

Au lieu d'apporter une correction au modèle statique comme le font les techniques de correction additive et multiplicative mentionnées à la section 2.4, le modèle hybride considère le modèle statique comme une variable. Ainsi, aucune information ou approximation des dérivées du modèle statique n'est nécessaire à la construction du modèle, ce qui convient bien dans un contexte de DFO/BBO. On note qu'un modèle hybride ne peut être construit que si le modèle statique $\tilde{g}$ est non trivial. En effet, si on avait $\tilde{g} = 0$, ce modèle statique n'apporterait aucune nouvelle information sur $g$ et serait nuisible à la construction du modèle. Ainsi, pour cette section, lorsqu'on mentionne un modèle statique, celui-ci n'est pas nul.

L'objectif du MHQ est de tirer profit du meilleur des deux mondes. En effet, le modèle statique à tendance à donner une idée générale de la fonction tout en captant potentiellement des éléments portant sur la nature de celle-ci. Le modèle quadratique, quant à lui, possède de bonnes propriétés de recherche locale, mais perd de son importance lorsque la région de confiance prend trop d'ampleur. Le MHQ vise à utiliser les propriétés des modèles quadratiques afin d'améliorer la précision du modèle statique. En fait, le MHQ vient corriger ce dernier à l'aide de sa propre information et d'un modèle quadratique pour qu'il se rapproche du vrai problème.

## 3.1 Modèle hybride quadratique

On reprend ici le principe du modèle quadratique présenté à la section 2.4, mais on l'augmente d'une variable $x_0$ qui servira à représenter la valeur du modèle statique. Le MHQ sera donc construit pour un espace de dimension $n+1$, où $n$ est le nombre de variables du problème (2.1).

La dimension supplémentaire est associée à la variable dépendante $x_0 = \tilde{g}(x)$. Formellement, on considère $\tilde{g} : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$ un modèle statique de $g : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$ et

$$\begin{aligned} \phi(x) &= (\phi_0(x), \phi_1(x), \ldots, \phi_t(x))^\top \\ &= \left(1, x_0, x_1, \ldots, x_n, \frac{x_0^2}{2}, \frac{x_1^2}{2}, \ldots, \frac{x_n^2}{2},\ x_0x_1,\ x_0x_2, \ldots,\ x_{n-1}x_n\right)^\top, \end{aligned} \tag{3.1}$$

où $x_0 = \tilde{g}(x)$ et $t + 1 = (n + 2)(n + 3)/2$. On cherche à construire un MHQ de la fonction $g(x)$, soit

$$\hat{g}_\beta(x) = \beta^\top \phi(x), \tag{3.2}$$

où $\beta \in \mathbb{R}^{t+1}$. Pour ce faire, on nécessite un ensemble de points d'interpolation $\mathbb{Y} = \{y^0, y^1, \ldots, y^p\} \subset \mathbb{R}^n$, $p \in \mathbb{N}$. La construction du modèle consiste à identifier la solution du problème

$$\min_{\beta \in \mathbb{R}^{t+1}} \sum_{y \in \mathbb{Y}} (g(y) - \hat{g}_\beta(y))^2. \tag{3.3}$$

Les définitions suivantes sont nécessaires pour caractériser le MHQ. Il s'agit de conditions à respecter afin d'assurer l'existence et l'unicité d'un MHQ. Celles-ci sont similaires à celles présentées à la section 2.4.

**Définition 3.2 (Ensemble propice à la régression quadratique hybride)** *Soit un ensemble de points* $\mathbb{Y} = \{y^0, y^1, \ldots, y^p\} \subset \mathbb{R}^n$ *avec* $p \geq t$ , $t + 1 = (n + 2)(n + 3)/2$ *et* $g(\mathbb{Y}) = (g(y^0), g(y^1), \ldots, g(y^p))^\top$, *où* $g : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$. *L'ensemble* $\mathbb{Y}$ *est* propice à la régression quadratique hybride *si la matrice*

$$M(\phi, \mathbb{Y}) = \begin{bmatrix} \phi_0(y^0) & \phi_1(y^0) & \ldots & \phi_t(y^0) \\ \phi_0(y^1) & \phi_1(y^1) & \ldots & \phi_t(y^1) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_0(y^p) & \phi_1(y^p) & \ldots & \phi_t(y^p) \end{bmatrix} \in \mathbb{R}^{(p+1)\times(t+1)} \tag{3.4}$$

*est de rang* $t + 1$ *et que* $g(\mathbb{Y}) \in \mathbb{R}^{p+1}$.

Tout comme pour le modèle quadratique, on peut utiliser la théorie de Frobenius afin de construire un MHQ lorsqu'on possède moins de $t + 1$ points.

**Définition 3.3 (Ensemble propice au modèle hybride quadratique de Frobenius)** *Soit* $\tilde{g} : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$ *un modèle statique de* $g : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$, $\phi_L(x) = (1, x_0, x_1, \ldots, x_n)^\top$ *et* $\phi_Q(x) = (\frac{x_0^2}{2}, \frac{x_1^2}{2}, \ldots, \frac{x_n^2}{2},\ x_0x_1,\ x_0x_2, \ldots,\ x_{n-1}x_n)^\top$, *où* $x_0 = \tilde{g}(x)$. *Un ensemble de points* $\mathbb{Y} = \{y^0, y^1, \ldots, y^p\} \subset \mathbb{R}^n$, $n + 2 < p + 1 < (n + 2)(n + 3)/2$, *est* propice au modèle hybride

quadratique de Frobenius *si la solution du système suivant est unique :*

$$\begin{aligned} &\min_{\beta_Q \in \mathbb{R}^N} \frac{1}{2} \|\beta_Q\|^2 \\ &s.c. \quad g(y^i) = \phi_L(y^i)\beta_L + \phi_Q(y^i)\beta_Q \quad \textit{pour } i = 0, 1, \ldots, p, \end{aligned} \tag{3.5}$$

*où* $\beta_L \in \mathbb{R}^{n+2}$ *et* $N = (n+1)(n+2)/2$.

On considère un ensemble de $p+1$ points d'interpolation $\mathbb{Y} = \{y^0, y^1, \ldots, y^p\}$. On distingue cinq cas possibles qui dépendent de la valeur de $p$ afin d'identifier $\beta = [\beta_L \ \beta_Q]^\top$.

Cas I : $p = t$ et l'ensemble $\mathbb{Y}$ est propice à la régression quadratique hybride.
Le problème (3.3) se résume à résoudre le système

$$M(\phi, \mathbb{Y})\beta = g(\mathbb{Y}), \tag{3.6}$$

où $g(\mathbb{Y}) = (g(y^0), g(y^1), \ldots, g(y^p))$. Le système (3.6) possède une solution unique, puisque $M(\phi, \mathbb{Y})$ est inversible par l'hypothèse sur $\mathbb{Y}$.

Cas II : $p > t$ et l'ensemble $\mathbb{Y}$ est propice à la régression quadratique hybride.
Le système (3.6) est surdéterminé. Toutefois, on peut identifier la solution $\beta$ avec l'égalité suivante :

$$\beta = [M(\phi, \mathbb{Y})^\top M(\phi, \mathbb{Y})]^{-1} M(\phi, \mathbb{Y})^\top g(\mathbb{Y})$$

Cette égalité découle directement du théorème suivant.

**Théorème 3.4** *Soit un système d'équations* $Ax = b$, *où* $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ *et* $n \leq m$. *Si* $A$ *est de rang* $n$, *alors il existe un seul vecteur* $x$ *minimisant* $\|Ax - b\|$ *et sa solution est* $x = [A^\top A]^{-1} A^\top b$.

*Démonstration.* En multipliant par $A^\top$ chaque côté de l'égalité $Ax = b$, on obtient l'égalité $A^\top Ax = A^\top b$. La matrice $A^\top A$ est inversible puisque, par hypothèse, la matrice $A$ est de rang $n$. On a donc les équivalences suivantes :

$$\begin{aligned} A^\top Ax = A^\top b &\iff [A^\top A]^{-1}[A^\top A]x = [A^\top A]^{-1} A^\top b \\ &\iff x = [A^\top A]^{-1} A^\top b. \end{aligned}$$

□

Cas III : $n + 1 < p < t$ et l'ensemble $\mathbb{Y}$ est propice au MHQ de Frobenius.
Le système (3.6) est sous-déterminé. Il est toutefois possible d'identifier une solution $\beta$ unique

en utilisant le modèle quadratique de Frobenius appliqué au MHQ. Cette méthode consiste à minimiser la norme des coefficients des termes quadratiques de l'équation (3.2). On peut écrire $\beta$ sous la forme $\beta = [\beta_L \ \beta_Q]^\top$ avec $\beta_L \in \mathbb{R}^{n+2}$, $\beta_Q \in \mathbb{R}^N$ et $N = (n+1)(n+2)/2$. Avec cette formulation, l'équation (3.2) devient

$$\hat{g}_\beta(x) = \phi_L(x)\beta_L + \phi_Q(x)\beta_Q,$$

où $\phi_L(x) = (1, x_0, x_1, x_2, \ldots, x_n)^\top$, $\phi_Q(x) = (\frac{x_0^2}{2}, \frac{x_1^2}{2}, \frac{x_2^2}{2}, \ldots, \frac{x_n^2}{2},\ x_0x_1,\ x_0x_2, \ldots,\ x_{n-1}x_n)^\top$ et $x_0 = \tilde{g}(x)$. On peut alors identifier la solution $\beta$ en résolvant le problème

$$\begin{aligned} \min_{\beta_Q \in \mathbb{R}^N} \ & \frac{1}{2} \|\beta_Q\|^2 \\ s.c. \ \ & g(\mathbb{Y}) = M(\phi_L, \mathbb{Y})\beta_L + M(\phi_Q, \mathbb{Y})\beta_Q, \end{aligned} \tag{3.7}$$

où $M(\phi_L, \mathbb{Y})$ et $M(\phi_Q, \mathbb{Y})$ sont des matrices de la forme (3.4), mais associées aux bases $\phi_L$ et $\phi_Q$ respectivement. Ce problème possède une solution unique sous l'hypothèse sur $\mathbb{Y}$. On peut alors résoudre le problème (3.7) à l'aide des multiplicateurs de Lagrange. En posant $\mathcal{L}(\beta, \lambda) = \frac{1}{2}\|\beta_Q\|^2 - \lambda^\top(M(\phi_L, \mathbb{Y})\beta_L + M(\phi_Q, \mathbb{Y})\beta_Q - g(\mathbb{Y}))$ comme lagrangien, où $\lambda = [\lambda_0, \lambda_1, \ldots, \lambda_p]^\top$, on obtient le système d'équations suivant :

$$\begin{aligned} g(\mathbb{Y}) &= M(\phi_L, \mathbb{Y})\beta_L + M(\phi_Q, \mathbb{Y})\beta_Q, \\ 0 &= \lambda^\top M(\phi_L, \mathbb{Y}), \\ \beta_Q &= \lambda^\top M(\phi_Q, \mathbb{Y}). \end{aligned} \tag{3.8}$$

Avec l'information de (3.8), on peut construire le système

$$F(\phi, \mathbb{Y}) \begin{bmatrix} \lambda \\ \beta_L \end{bmatrix} = \begin{bmatrix} M(\phi_Q, \mathbb{Y})M(\phi_Q, \mathbb{Y})^\top & M(\phi_L, \mathbb{Y}) \\ M(\phi_L, \mathbb{Y})^\top & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ \beta_L \end{bmatrix} = \begin{bmatrix} g(\mathbb{Y}) \\ 0 \end{bmatrix}, \tag{3.9}$$

où $F(\phi, \mathbb{Y}) \in \mathbb{R}^{(p+n+3)\times(p+n+3)}$, afin d'identifier les valeurs de $\beta_L$ et $\lambda$. La proposition suivante simplifie l'identification d'un ensemble propice au modèle hybride quadratique de Frobenius.

**Proposition 3.5** *Un ensemble de points $\mathbb{Y} = \{y^0, y^1, \ldots, y^p\} \subset \mathbb{R}^n$, $n+2 < p+1 < (n+2)(n+3)/2$, est propice au modèle hybride quadratique de Frobenius si et seulement si le système (3.9) possède une solution unique, i.e., $F(\phi, \mathbb{Y})$ est inversible.*

*Démonstration.* Soit $\mathbb{Y} = \{y^0, y^1, \ldots, y^p\} \subset \mathbb{R}^n$, $n+2 < p+1 < (n+2)(n+3)/2$ propice au

MHQ de Frobenius. On a la suite d'équivalences suivantes :

$$\begin{aligned}
\mathbb{Y} \text{ est propice au MHQ de Frobenius} &\iff (3.5) \text{ possède une solution unique} \\
&\iff \beta_L \text{ et } \beta_Q \text{ de } (3.8) \text{ sont uniques} \\
&\iff \beta_L \text{ et } \lambda \text{ de } (3.9) \text{ sont uniques} \\
&\iff (3.9) \text{ possède une solution unique.}
\end{aligned}$$

□

Ainsi, par la proposition 3.5, la matrice $F(\phi, \mathbb{Y})$ est inversible, puisque $\mathbb{Y}$ est propice au modèle hybride quadratique de Frobenius par hypothèse. Avec la troisième égalité de (3.8), on obtient la solution $\beta_Q$, soit

$$\beta_Q = \lambda^\top M(\phi_Q, \mathbb{Y}).$$

Cas IV : $p > n + 1$ mais l'ensemble $\mathbb{Y}$ n'est propice à aucun MHQ.
Il existe une infinité de solutions au système (3.6) ou au système (3.9). On identifie toutefois un MHQ unique à l'aide de la méthode présentée à la section 3.3.

Cas V : $p \leq n + 1$ et, par définition, l'ensemble $\mathbb{Y}$ n'est propice à aucun MHQ.
Aucun MHQ n'est construit, puisqu'on manque d'information pour construire un MHQ qui couvre bien l'espace. Cependant, il serait possible de construire un modèle hybride linéaire si $p = n + 1$ et $\mathbb{Y}$ forme un simplexe dans $\mathbb{R}^{n+1}$, où la dimension supplémentaire est associée à la variable $\tilde{g}(x)$. Ce modèle n'est toutefois pas présenté dans ce mémoire, puisque le cas $p = n + 1$ ne survient que très rarement dans le contexte où on utilise ces modèles hybrides.

### Exemple de modèle hybride quadratique

Soit $g : \mathbb{R} \mapsto \mathbb{R}$ une fonction de type boîte noire et $\tilde{g} : \mathbb{R} \mapsto \mathbb{R}$ un modèle statique défini par

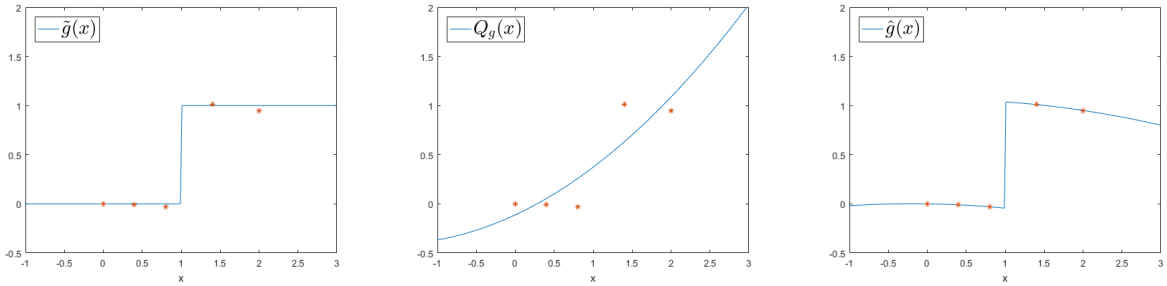
$$\tilde{g}(x) = \begin{cases} 0 & \text{si } x < 1, \\ 1 & \text{sinon.} \end{cases}$$

Le tableau suivant montre la valeur de la fonction $g(x)$ en 5 points de $\mathbb{R}$.

Tableau 3.1 Valeur de la fonction $g(x)$ selon la coordonnée $x$.

| $x$ | 0.0 | 0.4 | 0.8 | 1.4 | 2.0 |
|---|---|---|---|---|---|
| $g(x)$ | 0 | $-0.01$ | $-0.03$ | 1.01 | 0.95 |

La figure suivante illustre le modèle statique $\tilde{g}$, le modèle quadratique de la fonction $g$ noté $Q_g$ et le MHQ $\hat{g}$ de la fonction $g$. Les étoiles représentent les points du tableau 3.1.




(a) Modèle statique $\tilde{g}$ (b) Modèle quadratique $Q_g$ (c) MHQ $\hat{g}$

Figure 3.1 Différents modèles substituts de la fonction $g(x)$.

Comme le montre la figure 3.1, $\tilde{g}$ capte le saut, mais ne retourne que deux valeurs. $Q_g$, quant à lui, ne capte ni le saut, ni les variations décroissantes des points avant et après $x = 1$. Pour sa part, $\hat{g}$ capte le saut et les variations.

## 3.2 Approche algorithmique

On considère ici un problème d'optimisation contraint de la forme générale suivante :

$$\min_{x \in \Omega} f(x), \tag{3.10}$$

où $\Omega = \{x \in X : c_j(x) \leq 0, j \in J\} \subset \mathbb{R}^n, f, c_j : X \to \mathbb{R} \cup \{\infty\}$ pour tout $j \in J = \{1, 2, \ldots, m\}$, et $X$ est un sous-ensemble de $\mathbb{R}^n$. On considère la situation où le problème (3.10) est accompagné d'un problème substitut statique sous la forme du problème suivant :

$$\min_{x \in \tilde{\Omega}} \tilde{f}(x), \tag{3.11}$$

où $\tilde{\Omega} = \{x \in X : \tilde{c}_j(x) \leq 0, j \in J\} \subset \mathbb{R}^n$ et $\tilde{f}, \tilde{c}_j : X \to \mathbb{R} \cup \{\infty\}$ pour tout $j \in J = \{1, 2, \ldots, m\}$. Dans un tel cas, le problème (3.10) réfère au vrai problème.

L'objectif ici est de construire un MHQ du problème (3.10) à l'aide des fonctions relatives à (3.11). On suppose ici qu'on résout (3.10) avec le logiciel NOMAD et qu'à l'itération $k$, l'algorithme possède suffisamment de points pour former un MHQ du vrai problème. À

l'aide de la théorie décrite à la section 3.1, on peut créer les $m+1$ modèles

$$\hat{f}^k \approx f, \hat{c}_1^k \approx c_1, \ldots, \hat{c}_{m-1}^k \approx c_{m-1} \text{ et } \hat{c}_m^k \approx c_m.$$

Les paragraphes suivants décrivent comment ces MHQ peuvent interagir avec le logiciel NOMAD.

### Recherche globale

Comme mentionné à la section 2.6, NOMAD offre la possibilité d'utiliser des modèles substituts pour effectuer une recherche globale. Toutefois, le MHQ n'est pas adéquat pour les méthodes globales impliquant un modèle substitut. On explique ci-dessous pourquoi on rejette d'implémenter le MHQ dans la recherche globale.

La recherche VNS explore des points généralement loin de la majorité des solutions présentes dans la cache. Cette recherche peut être orientée à l'aide d'un modèle statique, puisque celui-ci fournit une bonne idée du comportement global de la fonction. Cependant, le MHQ n'est pas idéal pour approximer globalement le vrai problème, puisque l'aspect quadratique du modèle n'est plus fiable lorsque la région de confiance est trop grande. Il serait toutefois possible de construire des MHQ pour orienter les descentes de VNS. Or, pour construire un MHQ, il faut à la fois les valeurs du modèle statique et les valeurs du vrai problème. Il faudrait alors évaluer le modèle statique et le vrai problème aux points explorés lors des descentes. Ceci est absurde, puisqu'on utilise le modèle statique afin d'éviter d'évaluer trop souvent le vrai problème.

La recherche globale par modèles dynamiques construit des modèles pour approximer le problème à l'intérieur d'une certaine région de confiance. Ces modèles dynamiques sont par la suite minimisés à l'aide de l'algorithme MADS. Puisque MADS est un algorithme de recherche directe, le modèle dynamique est évalué maintes fois avant d'atteindre le critère d'arrêt. Si on remplaçait le modèle dynamique par un modèle hybride, il faudrait alors évaluer le modèle statique à chaque point exploré pendant la recherche. Or, un modèle statique est généralement plus coûteux à évaluer qu'un modèle dynamique. Ainsi, même si l'information du modèle statique peut améliorer la fiabilité du modèle à optimiser, le temps de calcul supplémentaire à prévoir est trop grand pour envisager d'implémenter le modèle hybride dans cette méthode de recherche globale. D'autres méthodes, présentées dans

La recherche Nelder-Mead de [25], pour sa part, ne tire aucun profit des modèles substituts. En effet, cette approche s'effectue habituellement en peu d'évaluations. Pour cette raison, la

recherche s'exécute seulement sur le problème original.

Ces méthodes de recherche globale sont toutefois indépendantes à l'étape de recherche locale. Ainsi, même si elles n'utilisent pas l'information d'un MHQ dans leurs algorithmes, elles peuvent être employées avec un MHQ opérant dans la sonde locale.

### Sonde locale

La recherche locale assistée d'un MHQ reprend le principe de celle assistée d'un modèle quadratique présentée dans [35] et implémentée dans NOMAD. Tout comme cette dernière, la recherche locale assistée d'un MHQ consiste à construire, à l'itération $k$, un modèle local du vrai problème afin d'ordonner la liste des points d'essai $P^k$ selon leurs valeurs $\hat{f}^k(x)$ et $\hat{c}^k = [\hat{c}_1^k(x), \hat{c}_2^k(x), \ldots, \hat{c}_m^k(x)]^\top$. Les points ordonnés sont ensuite évalués par les fonctions du vrai problème de manière opportuniste.

À l'itération $k$, la recherche locale assistée d'un MHQ débute après que MADS ait généré la liste des points d'essai $P^k$ autour de la meilleure solution connue $x^k$ ($x^F$ ou $x^I$ dans le cas PB). Les points de $P^k$ sont d'abord évalués avec le substitut statique (3.11). On calcule ensuite les valeurs $\max_i$ et $\min_i$, $i \in \{1, 2, \ldots, n\}$, selon les équations suivantes :

$$\max_i = \max\{y_i : y \in P^k\}, \qquad \min_i = \min\{y_i : y \in P^k\}.$$

Ces valeurs permettent d'identifier le centre $\mu$ de $P^k$ défini par

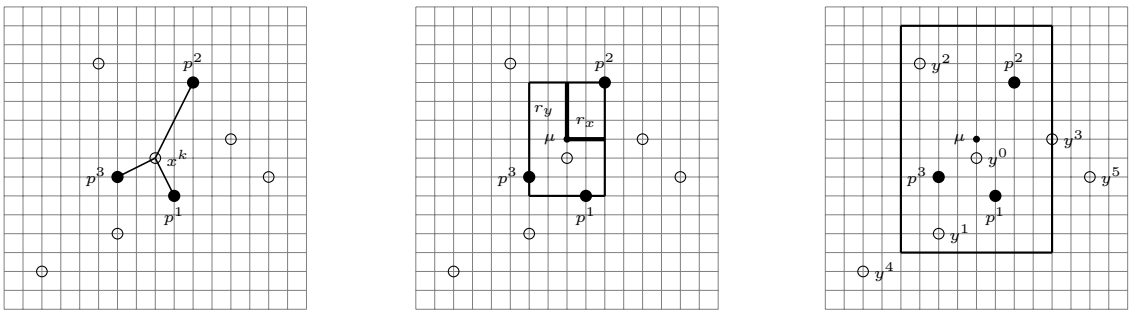
$$\mu = [\mu_1, \mu_2, \ldots, \mu_n]^\top, \text{où } \mu_i = \frac{1}{2}(\max_i + \min_i), \quad i \in \{1, 2, \ldots, n\}, \tag{3.12}$$

ainsi que le rayon $r$ de la plus petite boîte $\mathcal{B}_\infty(\mu; r) = \{x \in \mathbb{R}^n \ : \ \|x - \mu\|_\infty \leq r\}$ comprenant $P^k$ défini par

$$r = [r_1, r_2, \ldots, r_n]^\top, \text{où } r_i = \frac{1}{2}(\max_i - \min_i), \quad i \in \{1, 2, \ldots, n\}. \tag{3.13}$$

Ces valeurs sont utilisées afin de construire une région de confiance $\mathcal{B}_\infty(\mu; \rho r)$, où le scalaire $\rho \in (0, \infty)$ est le facteur du rayon. Ce facteur permet d'ajuster la région de confiance selon les préférences de l'utilisateur. Les points d'interpolation $\mathbb{Y} = \{y^0, y^1, \ldots, y^p\}$ choisis pour créer le MHQ doivent faire partie de $\mathcal{B}_\infty(\mu; \rho r)$ et avoir été antérieurement évalués aux problèmes (3.10) et (3.11). Ainsi, à l'itération $k$, $\mathbb{Y} = \mathcal{B}_\infty(\mu; \rho r) \cap V^k \cap \tilde{V}^k$, où $V^k$ est la cache du vrai problème et $\tilde{V}^k$ est la cache du modèle statique.

La figure 3.2 illustre les étapes précédentes avec un exemple dans $\mathbb{R}^2$. Les cercles pleins représentent l'ensemble de sonde $P^k$ et les cercles vides représentent l'ensemble $V^k \cap \tilde{V}^k$. On génère le centre $\mu$ et le rayon $r$ avec l'information de $P^k$ pour ensuite former la région de confiance $\mathcal{B}_\infty(\mu; \rho r)$. Les points $y^0, y^1, y^2$ et $y^3$ forment l'ensemble des points d'interpolation $\mathbb{Y}$, tandis que les points $y^4$ et $y^5$ ne sont pas pris en considération pour la construction du MHQ.




Création de $P^k = \{p^1, p^2, p^3\}$ $\quad$ $\mu, r = [r_x, r_y]$ et $\mathcal{B}_\infty(\mu; r)$ $\quad$ $\mathcal{B}_\infty(\mu; 2r)$ et $\mathbb{Y} = \{y^0, y^1, y^2, y^3\}$

Figure 3.2 Exemple dans $\mathbb{R}^2$ d'une région de confiance $\mathcal{B}_\infty(\mu; \rho r)$ avec $\rho = 2$.

On considère ici que si $|\mathbb{Y}| = p + 1$ est supérieur ou égal à $n + 2$, un MHQ unique peut être construit à partir de $\mathbb{Y}$, même si cet ensemble n'est pas nécessairement propice au modèle hybride quadratique de Frobenius ou propice à la régression quadratique hybride. Cette remarque est expliquée en détail à la section 3.3. Ainsi, si $p \geq n + 1$, les MHQ $\hat{f}^k, \hat{c}^k$ sont construits. Dans le cas contraire, aucun MHQ n'est construit et on tient seulement compte de l'information du modèle statique. Afin d'éviter de toujours séparer ces deux cas, on pose $\hat{f}^k = \tilde{f}, \hat{c}^k_j = \tilde{c}_j$, $j \in \{1, 2, \ldots, m\}$, lorsqu'aucun MHQ n'est construit.

On pose $\mathscr{P}$ comme la liste de points d'essai à ordonner. Si les contraintes sont gérées avec EB, $\mathscr{P}$ est tout simplement $P^k$. Cependant, ce n'est pas le cas lorsqu'on gère les contraintes avec PB. En effet, avec cette dernière, on possède un centre de sonde secondaire $x^I$, soit la solution irréalisable telle que définie à la section 2.5. On répète alors les étapes précédentes, mais avec $x^I$ au lieu de $x^F$. La nouvelle liste de points d'essai s'ajoute donc à $P^k$ pour former $\mathscr{P}$. Les MHQ formés autour de $x^I$ ne sont pas les mêmes que ceux formés autour de $x^F$. Toutefois, afin de ne pas alourdir le texte, on garde la notation $\hat{f}^k, \hat{c}^k$ pour définir ces MHQ.

L'ordonnancement des points de $\mathscr{P}$ est effectué en fonction des valeurs des modèles $\hat{f}^k$ et $\hat{c}^k$. Soit $\mathscr{L}$ la liste des points ordonnés et $\hat{h}^k$ la fonction de violation de contrainte (voir définition 2.12) construite à partir des contraintes $\hat{c}^k$. Pour tout couple de points $x^i, x^j \in \mathscr{P}$, $i, j \in \{1, 2, \ldots, |\mathscr{P}|\}$, $i \neq j$, $x^i$ précède $x^j$ dans $\mathscr{L}$ si

i. $x^i$ est non dominé dans $\mathscr{P}$ et $x^j$ est dominé par au moins un point de $\mathscr{P}$ ;

ii. $\hat{h}^k(x^i) = \hat{h}^k(x^j) = 0$ et $x^i \prec_{\hat{f}^k} x^j$ ;

iii. $\hat{h}^k(x^i) > 0, \hat{h}^k(x^j) > 0$ et $x^i \prec_{\hat{h}^k} x^j$ ;

iv. $x^i$ ne domine pas $x^j$, $x^j$ ne domine pas $x^i$, mais $\hat{h}^k(x^i) < \hat{h}^k(x^j)$.

Les conditions iii. et iv. n'interviennent que lorsque les contraintes sont gérées par PB. Lorsqu'elles sont gérées par EB, $x^i = x^k + \delta^k d^i$ est placé devant $x^j = x^k + \delta^k d^j$ si $\hat{h}^k(x^i) > 0$, $\hat{h}^k(x^j) > 0$ et si l'angle entre $d^i$ et $d^s$ est plus petit que l'angle entre $d^j$ et $d^s$, où $d^s$ est la direction de descente du dernier succès.

À titre illustratif, le tableau 3.2 présente un exemple d'une liste de points d'essai $\mathscr{P}$ à ordonner sous PB. La liste de points ordonnés correspondante est $\mathscr{L} = \{x^1, x^5, x^3, x^6, x^2, x^4\}$.

Tableau 3.2 Exemple de points à ordonner $\mathscr{P} = \{x^1, x^2, x^3, x^4, x^5, x^6\}$.

| $i$ | Centre de sonde | $\hat{f}^k(x^i)$ | $\hat{h}^k(x^i)$ |
|---|---|---|---|
| 1 | $x^F$ | 14 | 0 |
| 2 | $x^F$ | 12 | 0.7 |
| 3 | $x^F$ | 16 | 0 |
| 4 | $x^I$ | 9 | 1.5 |
| 5 | $x^I$ | 15 | 0 |
| 6 | $x^I$ | 10 | 0.7 |

Finalement, la liste de points $\mathscr{L}$ est évaluée de manière opportuniste. Ainsi, lorsqu'on utilise EB, on interrompt l'itération $k$ dès qu'un point $x \in \mathscr{L}$ satisfait $x \prec_f x^k$. Si on utilise PB, on interrompt l'itération $k$ dès qu'un point $x \in \mathscr{L}$ satisfait $x \prec_f x^F$. Toutefois, si on identifie un point $x \in \mathscr{L}$ tel que $h(x) > 0$ et $x \prec_h x^I$, l'itération $k$ n'est pas interrompue et $x^I$ est mise à jour qu'à la fin de l'itération. La figure 3.3 résume les étapes de recherche locale assistée d'un MHQ.

Étant donnés $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$, $\tilde{f} : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$, $c_j : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$, $\tilde{c}_j : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$, $j \in \{1, 2, \ldots, m\}$, et un point de départ $x^0 \in X$

0. Initialisation
   - $\rho \in (0, \infty)$ facteur de rayon
1. Recherche globale (optionnelle)
2. Sonde locale
   - 2.1. Création des MHQ de $f, c_1, \ldots, c_m$
     - choisir un ensemble générateur positif $D^k$ tel que $P^k \subseteq F^k$
     - calculer $\tilde{f}(y)$ pour tout $y \in P^k$
     - poser $\mu$ et $r$ selon les équations (3.12) et (3.13) respectivement
     - poser $\mathbb{Y} = \mathcal{B}_\infty(\mu; \rho r) \cap V^k \cap \tilde{V}^k$
     - si $|\mathbb{Y}| \geq n + 2$
       - construire les $m + 1$ MHQ $\hat{f}^k(x), \hat{c}_1^k(x), \ldots, \hat{c}_m^k(x)$
     - si $|\mathbb{Y}| < n + 2$
       - poser $\hat{f}^k(x) \leftarrow \tilde{f}(x), \hat{c}_1^k(x) \leftarrow \tilde{c}_1(x), \ldots, \hat{c}_m^k(x) \leftarrow \tilde{c}_m(x)$
   - 2.2. Ordonnancement
     - poser la liste des points à ordonner $\mathscr{P} \leftarrow P^k$
     - si les contraintes sont gérées par PB
       - répéter l'étape 2.1 avec $x^I$ (la liste de points d'essai est notée $P_I^k$)
       - $\mathscr{P} \leftarrow \mathscr{P} + P_I^k$
     - $\mathscr{L} \leftarrow \mathscr{P}$, avec les points ordonnés selon $\hat{f}^k(x)$ et $\hat{h}^k(x)$
     - évaluer de manière opportuniste le problème (3.10) aux points de $\mathscr{L}$
3. Mise à jour des paramètres et terminaison

Figure 3.3 MADS assisté d'un MHQ à l'étape de sonde locale.

### Analyse de convergence

L'utilisation d'un MHQ afin d'ordonner les points à évaluer à l'étape de sonde locale n'affecte pas l'analyse de convergence de MADS présenté à la section 2.3. En effet, l'ordre dans lequel une liste de points d'essai est évaluée n'a aucun impact sur les résultats théoriques. Ainsi, les propriétés de convergence de MADS, y compris le théorème fondamental 2.8, demeurent valides.

### Alternative au modèle proposé

On propose ici une autre manière d'approximer le vrai problème à l'aide d'un MHQ. Celle-ci consiste à construire un MHQ de la différence entre le vrai problème et le modèle statique.

Pour illustrer l'idée, on considère un problème de la forme (3.10) sans contrainte. Soit $E(x) = f(x) - \tilde{f}(x)$ l'erreur entre $f$ et $\tilde{f}$. Afin de fournir une approximation de $f$, on construit un MHQ de l'erreur $E$. En notant ce modèle $\hat{E}$, on peut approximer la fonction objectif par $f(x) \approx \hat{E}(x) + \tilde{f}(x)$. Cette méthode empêche de générer un terme d'erreur lorsqu'une ou plusieurs fonctions $\tilde{f}, \tilde{c}_1, \ldots, \tilde{c}_m$ sont égales à $f, c_1, \ldots, c_m$ respectivement. En effet, si une fonction $g(x)$ était égale à une fonction statique $\tilde{g}(x)$, l'erreur $E(x)$ serait nulle et donc $\hat{E}$ aussi. Or, dans une telle situation, la construction d'un MHQ de $g(x)$ n'égalerait pas forcément $\tilde{g}(x)$. Toutefois, en pratique, les deux méthodes donnent généralement la même approximation du vrai problème. C'est pour cette raison et pour des raisons d'élégance qu'on préfère construire un MHQ du vrai problème et non un MHQ de l'erreur.

Cette approche est semblable aux méthodes de correction abordées à la section 2.4. En effet, on obtient une correction additive $g(x) \approx \tilde{g}(x) + A(x)$, où $A(x)$ correspond au MHQ $\hat{E}(x)$. L'aspect intéressant de cette méthode est que la correction additive tient compte de l'information du modèle statique.

## 3.3 Géométrie des ensembles de points d'interpolation

D'après la théorie, afin de construire un MHQ, un ensemble de points d'interpolation $\mathbb{Y}$ doit être propice au modèle hybride quadratique. Cette propriété est directement reliée avec la géométrie de $\mathbb{Y}$ et aux fonctions statiques $\tilde{f}, \tilde{c}_j$, $j \in \{1, 2, \ldots, m\}$. En pratique, les méthodes de recherche directe ont tendance à produire implicitement des ensembles de points d'essai avec de bonnes propriétés géométriques [35, 39, 44]. Or, rien n'assure que l'ajout de l'information du modèle statique ne vient pas briser les propriétés géométriques de l'ensemble $\mathbb{Y}$. Par exemple, la proposition suivante présente une famille de fonctions pour laquelle un ensemble de points $\mathbb{Y}$ ne peut être propice à la régression quadratique hybride.

**Proposition 3.6** *Un ensemble de points $\mathbb{Y} = \{y^0, y^1, \ldots, y^p\} \subset \mathbb{R}^n$ n'est pas propice à la régression quadratique hybride si la fonction statique $\tilde{g} : \mathbb{R}^n \mapsto \mathbb{R}$ est un polynôme de degré inférieur ou égal à 2.*

*Démonstration.* Soit un ensemble $\mathbb{Y} = \{y^0, y^1, \ldots, y^p\}$, avec $p \geq t$, $t+1 = (n+2)(n+3)/2$ et un modèle statique $\tilde{g} : \mathbb{R}^n \mapsto \mathbb{R}$, $\tilde{g}(x) = \xi^\top \rho(x)$, où $\rho(x)$ est la base de l'espace des polynômes de degré 2 définie en (2.11) et $\xi \in \mathbb{R}^{q+1}$, où $q + 1 = (n + 1)(n + 2)/2$. On considère la base

$\phi(x)$ telle que définie en (3.1). Pour identifier le MHQ, on construit la matrice

$$M(\phi, \mathbb{Y}) = \begin{bmatrix} \phi_0(y^0) & \phi_1(y^0) & \dots & \phi_t(y^0) \\ \phi_0(y^1) & \phi_1(y^1) & \dots & \phi_t(y^1) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_0(y^p) & \phi_1(y^p) & \dots & \phi_t(y^p) \end{bmatrix} \in \mathbb{R}^{(p+1)\times(t+1)}.$$

$M(\phi, \mathbb{Y})$ doit être de rang $t+1$ afin que l'ensemble $\mathbb{Y}$ soit propice à la régression quadratique hybride. Or,

$$x_0 = \tilde{g}(x) \subset \text{Vect}(\rho(x)) = \text{Vect}\left(\phi(x)\backslash[x_0, \frac{x_0^2}{2}, x_0x_1, x_0x_2, \dots, x_0x_n]\right) \subset \text{Vect}\left(\phi(x)\backslash[x_0]\right).$$

Ainsi, $x_0 \subset \text{Vect}\left(\phi(x)\backslash[x_0]\right)$ et donc $M(\phi, \mathbb{Y})$ est au plus de rang $t$, ce qui implique que $\mathbb{Y}$ n'est pas propice à la régression quadratique hybride. □

En BBO, les modèles statiques sont habituellement issus de simulations. Ainsi, leur complexité n'affecte généralement pas les propriétés géométriques d'un ensemble de points $\mathbb{Y}$ bien réparti dans l'espace. Toutefois, en pratique, il arrive qu'un ensemble de points $\mathbb{Y}$ ne soit pas propice au modèle hybride quadratique. Dans une telle situation, l'unicité de $\beta$ est perdue, et donc aussi l'unicité du MHQ $\hat{g} = \beta^\top \phi(x)$. Il existe toutefois des méthodes qui permettent d'identifier une solution unique aux systèmes (3.6) et (3.9). L'une d'entre elles consiste à identifier une solution unique au sens des moindres carrés à l'aide d'une décomposition en valeurs singulières [49]. La solution unique est celle de plus petite norme. La notion suivante intervient dans l'identification de cette solution.

**Définition 3.7 (Pseudo-inverse d'une matrice à coefficients réels)** *Soit $A$ une matrice de dimension $i \times j$ à coefficients réels. Le* pseudo-inverse *de $A$, notée $A^\dagger$, est l'unique matrice respectant les conditions suivantes :*

1. $AA^\dagger A = A$ ;
2. $A^\dagger AA^\dagger = A^\dagger$ ;
3. $(AA^\dagger)^\top = AA^\dagger$ ;
4. $(A^\dagger A)^\top = A^\dagger A$.

Le pseudo-inverse d'une matrice $A$ peut être déterminé à l'aide d'une décomposition en valeurs singulières.

**Proposition 3.8** *Soit $A$ une matrice de dimension $i \times j$. Il existe une factorisation de $A$ de la forme $A = U\Sigma V^\top$, où*

— *$U$ est une matrice $i \times i$ formée de bases orthonormées,*

— *$V$ est une matrice $j \times j$ formée de bases orthonormées,*

— *$\Sigma$ est une matrice $i \times j$ telle que $\Sigma = diag(\sigma_1, \sigma_2, \ldots, \sigma_j)$, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_s > 0$, $\sigma_{s+1} = \ldots = \sigma_j = 0$, où $\sigma_k$, $k \in \{1, \ldots, j\}$, sont les valeurs singulières de $A$.*

*Le pseudo-inverse de $A$ est $A^\dagger = V\Sigma^\dagger U^\top$, où $\Sigma^\dagger = diag(1/\sigma_1, 1/\sigma_2, \ldots, 1/\sigma_s, 0, \ldots, 0)$.*

Le théorème suivant donne une solution unique à un système d'équations linéaires possédant au moins une solution.

**Théorème 3.9** *Soit un système d'équations $Ax = b$, où $A$ est une matrice $i \times j$, $i \geq j$. La solution du problème*

$$\begin{aligned} &\min_x \|x\| \\ &s.c. \quad x \in \operatorname*{argmin}_{\bar{x} \in \mathbb{R}^j} \|A\bar{x} - b\|^2 . \end{aligned} \tag{3.14}$$

*est donnée par $x^\dagger = A^\dagger b$, où $A^\dagger$ est le pseudo-inverse de $A$ .*

*Démonstration.* Soit le système

$$Ax = \begin{bmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,j} \\ a_{2,1} & a_{2,2} & \ldots & a_{2,j} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i,1} & a_{i,2} & \ldots & a_{i,j} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_j \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_i \end{bmatrix} = b,$$

où $rang(A) = s$. $A$ possède une décomposition en valeurs singulières de la forme $A = U\Sigma V^\top$, où $U$, $V$ et $\Sigma$ sont définies à la proposition 3.8. On utilise les propriétés des matrices orthonormées afin d'obtenir les égalités suivantes :

$$\begin{aligned} \|Ax - b\|^2 &= \left\|U\Sigma V^\top x - b\right\|^2 && (A = U\Sigma V^\top) \\ &= \left\|U^\top \left(U\Sigma V^\top x - b\right)\right\|^2 && (U^\top \text{ est une isométrie}) \\ &= \left\|\Sigma V^\top x - U^\top b\right\|^2 && (U^\top = U^{-1}) \\ &= \|\Sigma y - z\|^2 && (y = V^\top x \text{ et } z = U^\top b) \\ &= \left(\sum_{k=1}^{s} (\sigma_k y_k - z_k)^2 + \sum_{k=s+1}^{i} z_k^2\right). \end{aligned}$$

Puisque $y = V^\top x$ et $V$ est une isométrie, on a $\|y\| = \|x\|$. De plus, par la suite d'égalités précédente, on a que $\|Ax - b\|^2$ est minimisée si et seulement si $\|\Sigma y - z\|^2$ est minimisée. Les

solutions minimisant $\|\Sigma y - z\|^2$ sont de la forme

$$\{y \in \mathbb{R}^j : y_k = z_k/\sigma_k, k = 1, 2, \ldots, s\}$$

et celle qui possède la plus petite norme est

$$y^\dagger = [z_1/\sigma_1 \ z_2/\sigma_2 \ \ldots \ z_s/\sigma_s \ 0 \ \ldots \ 0]^\top = \Sigma^\dagger U^T b,$$

où $\Sigma^\dagger = diag(1/\sigma_1, 1/\sigma_2, \ldots, 1/\sigma_s, 0, \ldots, 0)$. Puisque $x = Vy$ et $\|x\| = \|y\|$, la solution recherchée du système (3.14) est $x^\dagger = V\Sigma^\dagger U^T b = A^\dagger b$. $\square$

Le théorème 3.9 permet donc d'identifier une solution unique aux systèmes (3.6) et (3.9), même si les ensembles de points d'interpolations $\mathbb{Y}$, $|\mathbb{Y}| \geq n + 2$, ne sont pas nécessairement propices au modèle hybride quadratique. On note que le théorème 3.9 s'applique aussi pour la construction d'un modèle quadratique lorsqu'un ensemble de points d'interpolation $\mathbb{Y}$, $|\mathbb{Y}| \geq n + 1$, n'est pas nécessairement propice au modèle quadratique.

# CHAPITRE 4 RÉSULTATS NUMÉRIQUES

Ce chapitre présente des résultats numériques de six problèmes tests sur lesquels l'algorithme de la section 3.2 a été expérimenté. Le premier est un problème analytique contraint à sept variables, les quatre suivants sont des problèmes d'optimisation multidisciplinaire (MDO) et le dernier est un problème d'assainissement des eaux souterraines par pompage et traitement. Les résultats sont obtenus à l'aide de la version 3.8.0 du logiciel NOMAD.

Pour l'ensemble des problèmes, on compare quatre stratégies d'ordonnancement des points d'essai à l'étape de sonde locale de l'algorithme MADS, soit celle accompagnée d'un MHQ («mhq»), celle accompagnée d'un modèle statique («stat»), celle accompagnée d'un modèle quadratique («quad») et celle sans modèle qui ordonne selon l'angle avec le dernier succès («succès»). Afin de comparer adéquatement ces stratégies, on désactive les méthodes de recherche globale à l'exception de la recherche spéculative. À moins d'apporter une réelle différence dans le temps d'exécution, on néglige le coût du modèle statique. Les directions pour l'ensemble des tests sont générées par l'algorithme ORTHOMADS avec $2n$ directions à chaque étape de sonde locale. Afin de comparer les différents algorithmes, on utilise des profils de données [69].

## Profils de données

Ces profils fournissent des renseignements sur l'efficacité et la robustesse d'un algorithme. Les définitions suivantes, basées sur les définitions 4.1 et 4.2 de [22], sont utiles afin de bien définir le test de convergence à la base de ces profils.

**Définition 4.1 (Problème numérique)** *Pour un algorithme d'optimisation nécessitant un point initial en entrée, un* problème numérique *est caractérisé par un unique problème $\mathcal{P}$ de la forme* (2.1) *auquel on associe un point initial $x_0 \in X$. On note le problème numérique $\mathcal{P}_{x_0}$.*

Pour un problème d'optimisation, on peut avoir autant de problèmes numériques que de points initiaux. Ceci est utile lorsque le nombre de problèmes tests est limité. C'est le cas ici, puisque peu de problèmes de la littérature disponibles possèdent un modèle statique cohérent avec la définition de modèle substitut.

**Définition 4.2 (Instance d'exécution)** *On note* instance d'exécution *l'exécution d'une optimisation sur un problème numérique. Reproduire l'optimisation sur le même problème*

*numérique constitue une différente instance d'exécution.*

Lorsque le comportement d'un algorithme est dépendant d'une graine aléatoire, il est possible d'obtenir différentes instances d'exécution pour un même problème numérique en variant la valeur de la graine aléatoire. C'est le cas pour l'algorithme OrthoMads, où le choix des directions est directement lié à une graine aléatoire. Pour un algorithme de ce type, la variation de la graine aléatoire permet d'obtenir davantage d'instances d'exécution, et donc davantage de résultats numériques pour analyser plus rigoureusement son efficacité et sa robustesse.

Les profils de données tracent, selon une tolérance $\tau > 0$, la proportion de problèmes numériques résolus par toutes les instances d'exécution de chaque algorithme en fonction du nombre d'évaluations. Habituellement, on classe le nombre d'évaluations en groupes de $n_{\mathcal{P}}+1$ évaluations, où $n_{\mathcal{P}}$ est la dimension d'un problème $\mathcal{P}$. Les grouper selon la taille d'un simplexe permet une mise à l'échelle des profils de données, puisque le nombre d'évaluations requises pour résoudre un problème est généralement corrélé avec la dimension du problème. Toutefois, lorsque les résultats sont tous issus du même problème $\mathcal{P}$, on ne groupe pas le nombre d'évaluations, car $n_{\mathcal{P}}$ est invariable. C'est le cas pour l'ensemble des profils de données exposés dans ce mémoire.

Le test de convergence suivant caractérise une instance d'exécution comme étant résolue dans un contexte de profils de données. Soit $\mathcal{P}_{x_0}$ un problème numérique, où $x_0 \in \Omega$ est un point de départ réalisable. On note $x_b$ comme étant le meilleur point réalisable obtenu par tous les algorithmes testés sur toutes les instances d'exécution de $\mathcal{P}_{x_0}$. Une instance d'exécution de $\mathcal{P}_{x_0}$ est dite résolue par l'algorithme $\mathcal{A}$ sous la tolérance $\tau > 0$ si

$$f(x_0) - f(x_e) \geq (1-\tau)(f(x_0) - f(x_b)), \tag{4.1}$$

où $x_e$ est la meilleure solution réalisable obtenue après $e$ évaluations de l'instance d'exécution par l'algorithme $\mathcal{A}$.

À l'exception d'un problème, les problèmes tests sont tous lancés à partir de points de départ réalisables. Une modification de la condition (4.1) est apportée plus tard dans ce chapitre pour les cas où les points de départ sont non réalisables. Ce mémoire ne présente toutefois aucune analyse de convergence de la violation de contraintes $h$ lorsque les points de départ sont irréalisables. En effet, comme les ensembles $\Omega$ des problèmes tests sont facilement atteignables et que la minimisation de $h$ afin d'atteindre un point réalisable est effectuée en début d'optimisation, le nombre de points d'interpolation dans la région de confiance est

généralement insuffisant pour construire un MHQ. Ainsi, puisque mhq utilise l'information du modèle statique lorsque qu'il n'a pas assez de points d'interpolation pour construire des MHQ, la convergence de $h$ pour mhq serait sensiblement la même que celle de stat.

## 4.1 Un problème de Hock et Schittkowski

Le problème de Hock et Schittkowski [52] numéro 100 possède quatre contraintes d'inégalité sur $\mathbb{R}^7$. Soit $X = [-10; 10]^7$. La formulation du problème est

$$\begin{aligned}
\min_{x \in X} \quad f(x) = \; & (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 \\
& + 10x_5^6 + 7x_6^4 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 \\
s.c. \quad c_1(x) = \; & 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 - 127 \leq 0 \\
c_2(x) = \; & 7x_1 + 3x_2 + 10x_3^2 + x^4 - x_5 - 282 \leq 0 \\
c_3(x) = \; & 23x_1 + x_2^2 + 6x_6^2 - 8x_7 - 196 \leq 0 \\
c_4(x) = \; & 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0
\end{aligned} \tag{4.2}$$

et son minimum global est situé près de $(2.33, 1.95, -0.48, 4.37, -0.62, 1.04, 1.59)$. Le modèle suivant, proposé par Marduel [63], est un modèle statique du problème (4.2) obtenu en perturbant ce dernier. La formulation du problème statique est

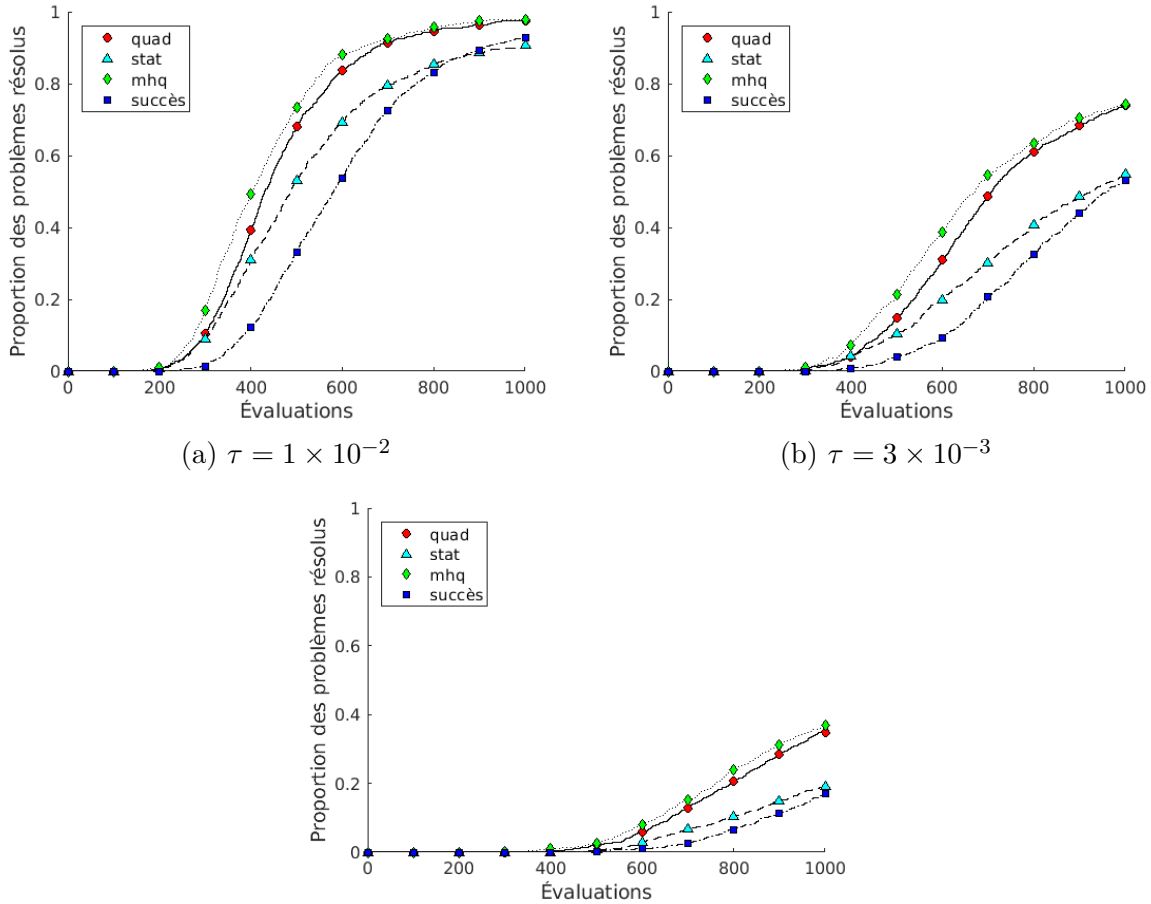
$$\begin{aligned}
\min_{x \in X} \quad \tilde{f}(x) = \; & 7(x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 \\
& + 10x_5^6 + 7x_6^4 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 - 4x_1x_3x_7 \\
s.c. \quad \tilde{c}_1(x) = \; & 2x_1^2 - 5x_2^4 + x_3 + 4x_4^2 + 5x_5 - 127 \leq 0 \\
\tilde{c}_2(x) = \; & 7x_1 + 3x_2 + 10x_3^2 + x^4 - x_5 - 282 + 5x_2x_4x_6 \leq 0 \\
\tilde{c}_3(x) = \; & 23x_1 + x_2^2 + 6x_6^2 - 8x_7 - 196 - 4(1 - x_4)(3 - 7x_5) \leq 0 \\
\tilde{c}_4(x) = \; & 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0
\end{aligned} \tag{4.3}$$

et son minimum global est situé près de $(3.33, 5, 1.60, 5, 0, 0.52, 2.47)$.

Le problème (4.2) et son substitut statique (4.3) ont tous les deux une fonction objectif et deux contraintes polynomiales quadriques ainsi que deux contraintes quadratiques. Le modèle statique est intentionnellement de mauvaise qualité. En effet, on cherche à voir si le MHQ est en mesure de capter l'information intéressante de (4.3) tout en lui apportant une correction quadratique pour se rapprocher de (4.2). La formulation de (4.3) complique toutefois cette

correction, puisque la perturbation apportée au vrai modèle est en partie cubique.

Les résultats de la figure 4.1 sont issus de 100 problèmes numériques générés à l'aide de 100 points de départ réalisables dans l'hypercube $[-1; 1]^7$. Pour chacun des problèmes numériques, on effectue 10 instances d'exécution. Les contraintes sont gérées par PB et le nombre maximal d'évaluations du vrai problème est fixé à 1000.




(a) $\tau = 1 \times 10^{-2}$

(b) $\tau = 3 \times 10^{-3}$

(c) $\tau = 1 \times 10^{-3}$

Figure 4.1 Profils de données obtenus avec une tolérance $\tau$ sur 10 instances d'exécution de 100 problèmes numériques Hock et Schittkowski numéro 100.

Les trois profils de données montrent des résultats similaires. En effet, on aperçoit pour les trois valeurs de $\tau$ que quad domine sur stat et sur succès. On note aussi que les problèmes se résolvent en un peu moins d'évaluations avec mhq qu'avec quad. Cependant, les deux modèles sont quasi égaux pour le nombre de problèmes résolus après 1000 évaluations, sauf pour $\tau = 1 \times 10^{-3}$ où mhq est légèrement supérieur. Ces minces avantages sont toutefois importants, puisqu'ils montrent que les MHQ parviennent à corriger le piètre modèle statique

afin de fournir un meilleur ordonnancement que les modèles quadratiques.

## 4.2 Quatre problèmes de type MDO

Un problème de type MDO est un problème d'optimisation dans lequel plusieurs disciplines interagissent entre elles. En effet, la donnée en entrée de l'une des disciplines est la donnée en sortie de l'autre. Les valeurs en sortie sont obtenues lorsque l'analyse multidisciplinaire du problème est en équilibre. Celle-ci est dite en équilibre après un certain nombre d'itérations si, pour chaque discipline, la variation entre deux valeurs consécutives est inférieure à un critère de convergence $\epsilon > 0$. On note que les itérations qu'on mentionne ici (et tout au long de cette section) sont des itérations d'analyse multidisciplinaire. Le schéma de la figure 4.2 illustre un exemple d'optimisation d'un problème de type MDO. Dans un contexte de conception de pare-chocs de voiture, la première discipline du schéma pourrait être l'aérodynamique, la deuxième la mécanique des structures et la troisième l'économie.

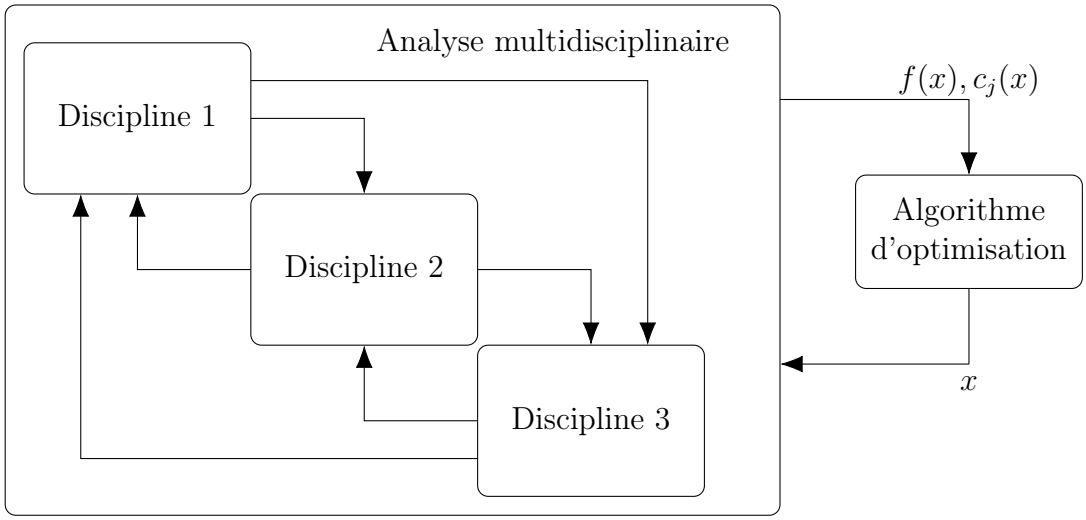



Figure 4.2 Exemple de schéma d'optimisation d'un problème de type MDO.

### Deux problèmes analytiques sans contrainte

Le problème d'optimisation Simple_MDO_2n_Variables [85] est

$$\min_{x \in X} a_1^*(x) + a_2^*(x), \tag{4.4}$$

où $X = [-100; 100]^{2n}$, $2n$ représente la dimension du problème et

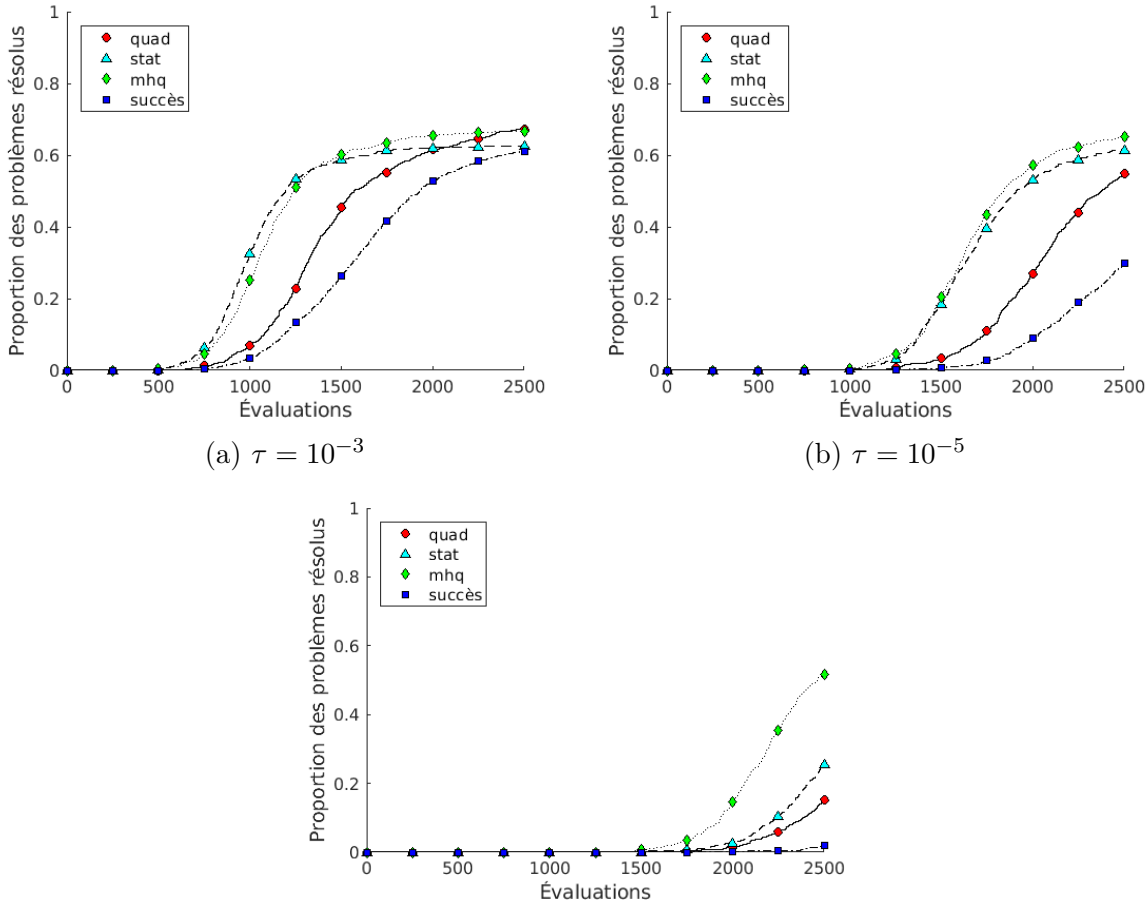
$$a_1 = \frac{x_1^2 - 2x_1 + 1 + \sum_{i=2}^{n}(2x_i^2 - 2x_{i-1}x_i + 1)}{1 + 0.5a_2},$$
$$a_2 = \left(\sum_{i=n+1}^{2n}(2x_i^2 - 2x_{i-1}x_i + 1)\right)\sqrt{1 + 0.5a_1}.$$

Les termes $a_1^*$ et $a_2^*$ de (4.4) sont étoilés pour dire qu'il s'agit des valeurs $a_1$ et $a_2$ lorsque l'analyse multidisciplinaire est en équilibre. Le tableau 4.1 montre les distinctions entre le vrai problème et le problème substitut. L'analyse multidisciplinaire se termine prématurément si le critère de convergence n'a pas été atteint avant le nombre maximal d'itérations. Ce deuxième critère d'arrêt est utilisé pour tous les problèmes de cette section.

Tableau 4.1 Distinctions entre le vrai problème et le substitut statique pour le problème Simple_MDO_2n_Variables.

| Nature du problème | Critère de convergence $\epsilon$ | Nombre maximal d'itérations |
|---|---|---|
| Vrai | $10^{-6}$ | 10000 |
| Substitut | 1 | 5 |

Le problème est testé ici avec $n = 5$ et $n = 8$. Pour chacun des problèmes, les résultats des figures 4.3 et 4.4 sont issus de 100 problèmes numériques générés à l'aide de 100 points de départ réalisables dans l'hypercube $[-2; 2]^{2n}$. Pour chacun des problèmes numériques, on effectue 10 instances d'exécution. Le nombre maximal d'évaluations du vrai problème est fixé à 2500 pour $n = 5$ et à 10000 pour $n = 8$.



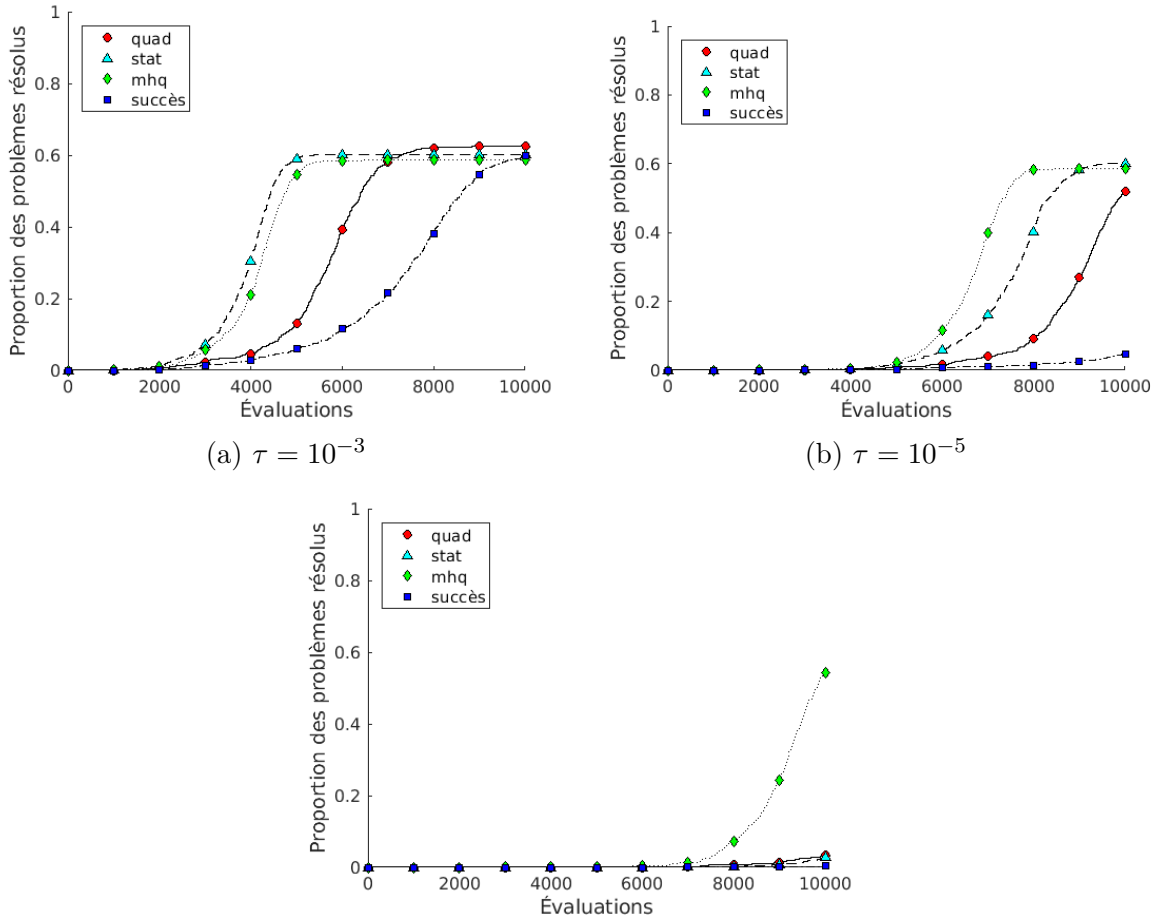

(a) $\tau = 10^{-3}$

(b) $\tau = 10^{-5}$

(c) $\tau = 10^{-7}$

Figure 4.3 Profils de données obtenus avec une tolérance $\tau$ sur 10 instances d'exécution de 100 problèmes numériques Simple_MDO_10_Variables.

On note tout d'abord que pour toutes les valeurs de $\tau$ présentées à la figure 4.3, les ordonnancements à l'aide d'un modèle surpassent succès. Pour $\tau = 10^{-3}$, on aperçoit que mhq et stat convergent plus rapidement que quad, mais que quad résout tout de même plus de problèmes que stat après $\sim$ 2100 évaluations. mhq et stat se démarquent davantage de quad lorsque $\tau = 10^{-5}$. mhq se distingue quelque peu stat pour cette tolérance, mais cette dominance se fait surtout voir lorsque $\tau = 10^{-7}$. En effet, sous cette tolérance, mhq résout approximativement 25% plus de problèmes que stat. Pour un même pourcentage de problèmes résolus, stat nécessite $\sim$ 300 évaluations de plus que mhq.




(a) $\tau = 10^{-3}$

(b) $\tau = 10^{-5}$

(c) $\tau = 10^{-7}$

Figure 4.4 Profils de données obtenus avec une tolérance $\tau$ sur 10 instances d'exécution de 100 problèmes numériques Simple_MDO_16_Variables.

La figure 4.4 montre encore une fois que succès est la stratégie la plus faible. Pour $\tau = 10^{-3}$, on aperçoit que stat et mhq convergent beaucoup plus rapidement que quad, mais ce dernier résout néanmoins plus de problèmes que ces compétiteurs passé $\sim$ 7300 évaluations. Avec $\tau = 10^{-5}$, mhq nécessite $\sim$ 1000 évaluations de moins que stat pour atteindre un pourcentage de problèmes résolus supérieur à 15%. Les deux atteignent sensiblement le même pourcentage de problèmes résolus après 10000 évaluations, supérieur à celui de quad. mhq domine tous les autres méthodes d'ordonnancement lorsque $\tau = 10^{-7}$. En effet, sous cette tolérance, il résout environ 55% des problèmes, tandis qu'aucune des autres méthodes ne réussit à atteindre 5%.

À la figure 4.3, près de 75% des problèmes résolus sous $\tau = 10^{-3}$ avec mhq sont résolus sous $\tau = 10^{-7}$. À la figure 4.4, il s'agit d'environ 95% des problèmes résolus sous $\tau = 10^{-3}$ avec mhq qui sont résolus sous $\tau = 10^{-7}$. mhq construit donc, pour les deux problèmes analytiques précédents, des modèles fidèles lorsque les régions de confiance sont petites.

### Un problème de conception d'aile d'avion

Le problème analysé ici, noté le problème *simplified wing*, est une simplification d'un problème de conception d'aile d'avion [84]. Les disciplines impliquées dans ce problème sont l'aérodynamique, la mécanique des structures et la performance. Résoudre le problème *simplified wing* consiste à maximiser la distance parcourue par un avion tout en respectant 10 contraintes d'inégalité. Il est en fonction de 10 variables mises à l'échelle et bornées par l'hypercube $[0; 100]^{10}$. Le problème statique, tout comme celui de Simple_MDO_2n_Variables, est construit en relaxant les critères d'arrêt de l'analyse multidisciplinaire. Le tableau 4.2 montre les distinctions entre ce dernier et le vrai problème.

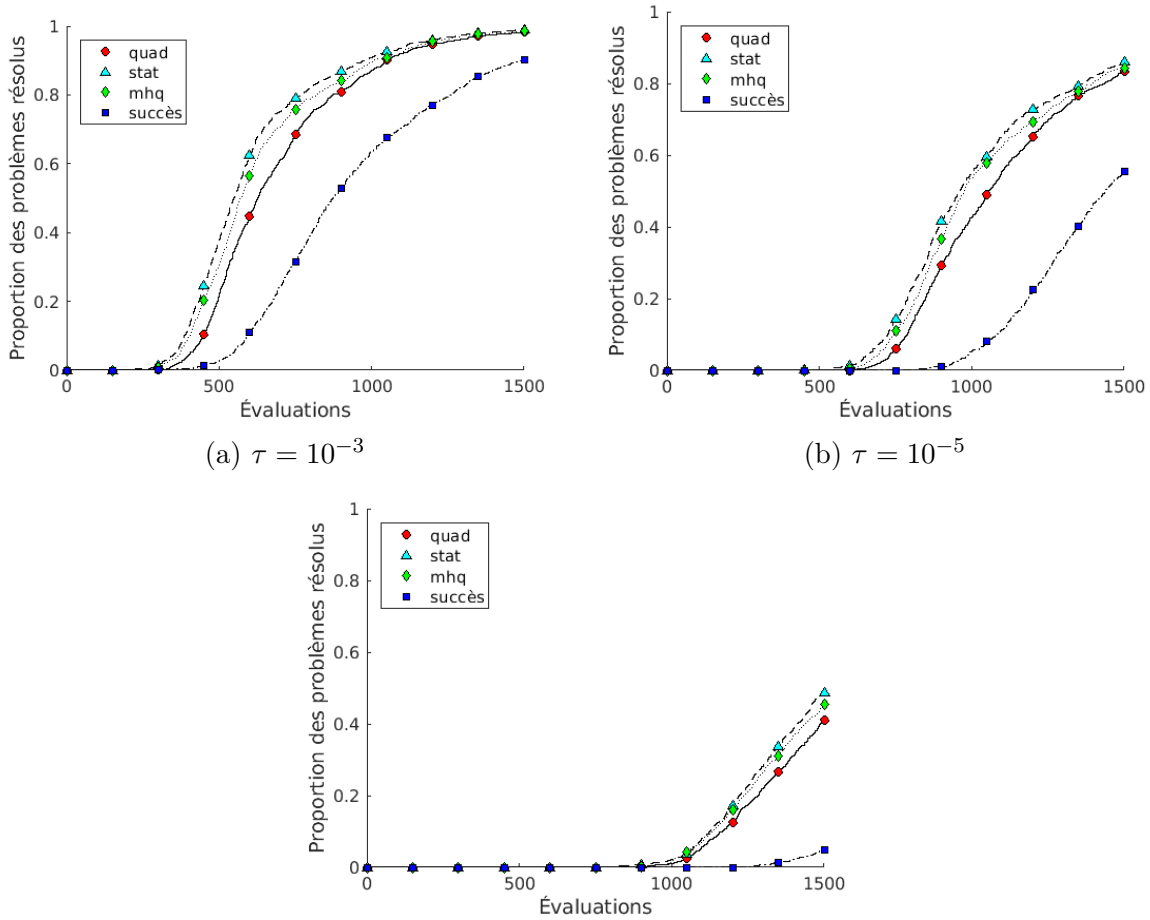
Tableau 4.2 Distinctions entre le vrai problème et le substitut statique pour le problème *simplified wing*.

| Nature du problème | Critère de convergence $\epsilon$ | Nombre maximal d'itérations |
|---|---|---|
| Vrai | $10^{-12}$ | 100 |
| Substitut | 0.1 | 5 |

Les résultats de la figure 4.5 sont issus de 100 problèmes numériques générés à l'aide de 100 points de départ réalisables dans l'hyperrectangle

$$\{x : [98\ 48\ 23\ 36\ 48\ 48\ 80\ 10\ 74\ 48]^\top \leq x \leq [100\ 52\ 26\ 40\ 52\ 52\ 84\ 13\ 77\ 52]^\top\}.$$

Pour chacun des problèmes numériques, on effectue 10 instances d'exécution. Les contraintes sont gérées par EB et le nombre maximal d'évaluations du vrai problème est fixé à 1500.




(a) $\tau = 10^{-3}$

(b) $\tau = 10^{-5}$

(c) $\tau = 10^{-7}$

Figure 4.5 Profils de données obtenus avec une tolérance $\tau$ sur 10 instances d'exécution de 100 problèmes numériques MDO.

Les trois graphiques de la figure 4.5 dégagent des résultats similaires. On y voit que succès est encore une fois pire méthode d'ordonnancement. Pour ce qui est des trois autres méthodes, leurs comportements sont semblables. stat est toutefois légèrement supérieur à quad et mhq. Ces résultats montrent que la correction apportée au problème statique par le MHQ n'est pas toujours avantageuse. En effet, pour le problème *simplified wing*, le MHQ semble détériorer l'information fournie par le modèle statique.

### Le problème *supersonic business jet design*

Le problème *supersonic business jet design* est présenté dans [3] et le problème d'optimisation est défini dans [83]. Les disciplines impliquées dans ce problème de MDO sont l'aérodynamique, la mécanique des structures et la propulsion. Ces disciplines sont toutes en interaction

avec un sous-problème noté "*Aircraft*" qui calcule le poids total de l'avion. Résoudre le problème consiste à minimiser le poids d'un avion tout en respectant 78 contraintes d'inégalité. Puisque le problème original est en fonction de 29 variables et qu'il possède un grand nombre de minima locaux, une simplification du problème est effectuée ici pour obtenir des résultats comparables. La simplification considérée dans ce mémoire est la même que celle utilisée dans [21]. Celle-ci consiste essentiellement à fixer aux valeurs optimales les 18 variables reliées aux épaisseurs des éléments structuraux. Le problème *supersonic business jet design* simplifié est donc en fonction de 11 variables et possède 78 contraintes d'inégalité, dont 12 qui deviennent invariantes et réalisables avec les variables fixées. Les bornes sur les variables sont les mêmes que celles considérées dans [83]. Le problème statique est encore une fois construit en relaxant les critères d'arrêt de l'analyse multidisciplinaire. Le tableau 4.3 montre les distinctions entre le vrai problème et le problème statique.

Tableau 4.3 Distinctions entre le vrai problème et le substitut statique pour le problème *supersonic business jet design*.

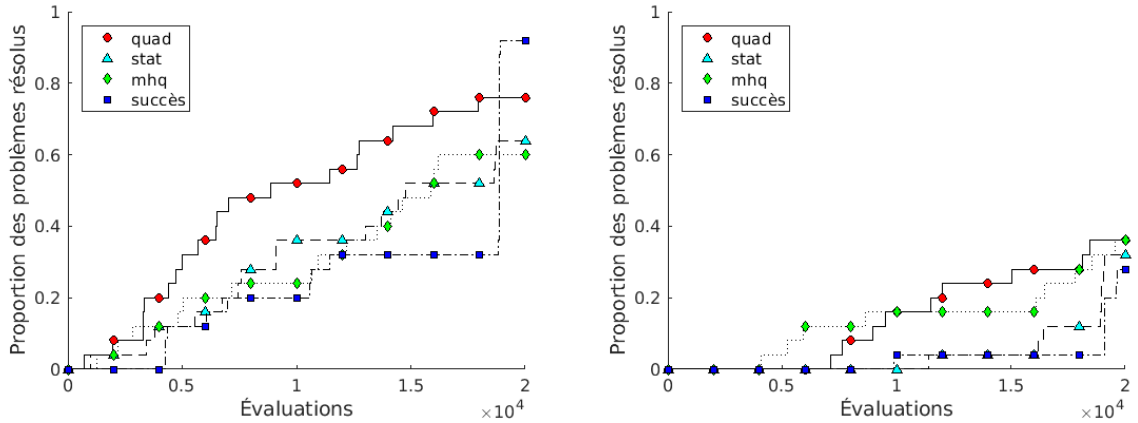
| Nature du problème | Critère de convergence $\epsilon$ | Nombre maximal d'itérations |
|---|---|---|
| Vrai | $10^{-5}$ | 50 |
| Substitut | 0.1 | 5 |

Les résultats de la figure 4.6 proviennent de 25 problèmes numériques générés à l'aide de 25 points de départ irréalisables tirés de [21]. Pour chacun des problèmes numériques, on effectue 1 instance d'exécution. Les contraintes sont gérées par EB et le nombre maximal d'itérations est fixé à 20000.

Puisque les points de départ sont irréalisables, la condition (4.1) doit être légèrement modifiée pour pouvoir tracer des profils de données. La formulation de la condition (4.1) dans le cas où le point de départ $x_0$ est irréalisable est

$$\bar{f}_{fea} - f(x_e) \geq (1 - \tau)(\bar{f}_{fea} - f(x_b)),$$

où $\bar{f}_{fea}$ est la moyenne des valeurs de la première fonction objectif réalisable obtenue par chaque instance d'exécution de $\mathcal{P}_{x_0}$ sur chaque algorithme.




(a) $\tau = 10^{-2}$ (b) $\tau = 10^{-3}$

Figure 4.6 Profils de données obtenus avec une tolérance $\tau$ sur 1 instance d'exécution de 25 problèmes numériques *supersonic business jet design*.

L'objectif des graphiques de la figure 4.6 est de montrer qu'il n'y a pas de réel intérêt à utiliser mhq dans le cas présent. En effet, on obtient avec quad des résultats supérieurs sous $\tau = 10^{-2}$ et similaires pour $\tau = 10^{-3}$. Le nombre de problèmes est limité à 25, car le coût de mhq est excessivement élevé comparé aux autres algorithmes. On parle ici d'un temps d'exécution 10 fois plus élevé que ces compétiteurs. De plus, avec 25 problèmes numériques, mhq ne semble pas plus avantageux que quad. Il serait donc peu probable qu'il se démarque pour 100 problèmes numériques.

L'approche par le MHQ n'est pas conseillée pour un tel type de problème. En effet, la matrice (3.4) utilisée pour la construction d'un MHQ d'une fonction $g$ est dépendante à la fonction statique $\tilde{g}$ associée. Ainsi, pour un problème de la forme (2.1) à $m$ contraintes, on doit construire $m+1$ matrices $M(\phi, \mathbb{Y})$ (voir (3.4)) ou $F(\phi, \mathbb{Y})$ (voir (3.9)) à chaque itération. Dans le cas présent, $m = 78$ et $n = 11$. Ainsi, à chaque itération, mhq évalue le problème statique aux $2n = 22$ points d'essai avant de faire une décomposition en valeurs singulières de 79 matrices de dimension $n^* \times n^*$, où $n^* \in \{2n+6, 2n+7, \ldots, (n+2)(n+5)/2\} = \{28, 29, \ldots, 104\}$. Les bornes ici sont les dimensions minimales et maximales de la matrice $F(\phi, \mathbb{Y})$ du système (3.9). En résumé, en plus de ne pas se démarquer à la figure 4.6, mhq est très coûteux en temps de calcul comparé aux trois autres algorithmes.

### Alternative pour le problème *supersonic business jet design*

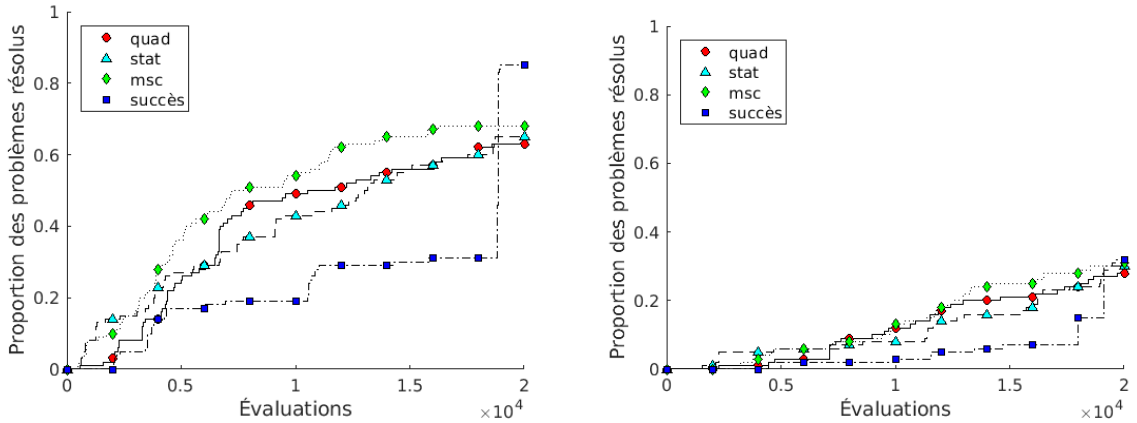
On présente ici une alternative au MHQ qui évite de décomposer en valeurs singulières 79 matrices à chaque itération. Le modèle proposé, qu'on nomme modèle statique corrigé

(MSC), est essentiellement le modèle statique auquel on apporte une correction additive à l'aide d'un modèle quadratique. Ainsi, pour une fonction substitut statique $\tilde{g}$ associée à une fonction $g$, le MSC $g_{cor} : \mathbb{R} \mapsto \mathbb{R}$ de la fonction $g$ est $g_{cor}(x) = \tilde{g}(x) + Q_E(x)$, où $Q_E(x)$ est un modèle quadratique de l'erreur $E(x) = g(x) - \tilde{g}(x)$.

Le MSC n'est pas formellement défini dans ce mémoire puisqu'il n'est pas adéquat pour tous les types de modèles statiques. En effet, puisqu'il ne peut pas modifier le coefficient devant le modèle statique, le MSC endommagerait probablement l'information d'un modèle statique qui n'approxime pas le vrai problème, mais dont ses optima locaux sont près de ceux du vrai problème. Toutefois, cette correction est convenable lorsque le modèle substitut est une approximation du vrai problème, ce qui est le cas avec le problème *supersonic business jet design*.

L'algorithme pour ordonnancer les points d'essai à l'étape de sonde locale de l'algorithme MADS à l'aide d'un MSC («msc») est sensiblement le même que celui avec un MHQ présenté à la figure 3.3. Les seules différences sont que l'on construit des MSC au lieu des MHQ et que le nombre minimal d'éléments dans l'ensemble des points d'interpolation diminue à $n+1$. En effet, puisque le modèle statique n'est plus considéré comme variable du modèle dynamique, on exige moins de points d'interpolation. L'avantage ici est qu'on nécessite qu'une seule matrice $M(\phi, \mathbb{Y})$ ou $F(\phi, \mathbb{Y})$ pour construire un MSC du problème. Le temps de calcul de cette méthode est donc comparable aux algorithmes quad, stat et succès.

Les résultats de la figure 4.7 proviennent de 100 problèmes numériques générés à l'aide des 100 points de départ irréalisables utilisés dans [21]. Pour chacun des problèmes numériques, on effectue 1 instance d'exécution. Les contraintes sont gérées par EB et le nombre maximal d'itérations est fixé à 20000.




(a) $\tau = 10^{-2}$ (b) $\tau = 10^{-3}$

Figure 4.7 Profils de données obtenus avec une tolérance $\tau$ sur 1 instance d'exécution de 100 problèmes numériques *supersonic business jet design*.

Lorsque $\tau = 10^{-2}$, on aperçoit qu'il est avantageux d'utiliser msc. En effet, si on oublie le coup de chance de succès, msc est plus efficace et plus robuste que ces compétiteurs. Pour $\tau = 10^{-3}$, les 4 méthodes résolvent approximativement le même nombre de problèmes après 20000 évaluations. Toutefois, msc se démarque légèrement après $\sim$ 13000 évaluations en résolvant ces problèmes plus rapidement que les trois autres algorithmes.

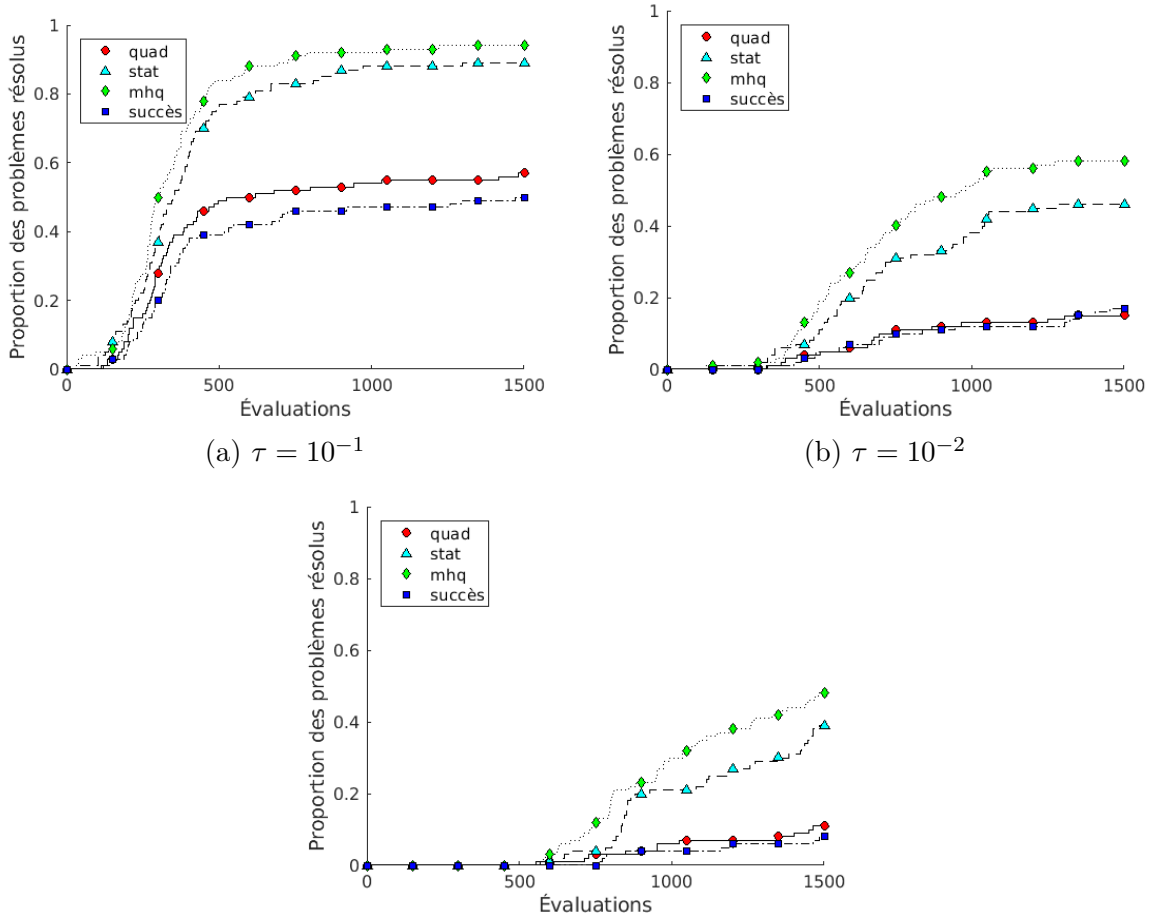
## 4.3 Problème LOCKWOOD

Le problème LOCKWOOD [65, 66] est un problème d'assainissement des eaux souterraines par pompage et traitement basé sur le *Lockwood Solvent Groundwater Plume Site* situé au Montana. Résoudre le problème consiste à minimiser la quantité de pompage effectué par six puits (les 6 paramètres en entrée du problème) afin de stabiliser deux panaches de pollutions (2 contraintes d'égalités) situés sur le site. Les flux des deux panaches de pollutions sont obtenus par le simulateur Bluebird [38]. Les variables en entrée sont bornées par l'hypercube $[0; 20000]^6$. Afin que le problème soit de la forme (2.1), les contraintes d'égalités $c_1$ et $c_2$ sont gérées comme 4 contraintes d'inégalités. En effet, si $c_3 = -c_1$ et $c_4 = -c_2$, on a

$$\{x : c_1(x) = 0,\ c_2(x) = 0\} \iff \{x : c_1(x) \leq 0,\ c_2(x) \leq 0,\ c_3(x) \leq 0,\ c_4(x) \leq 0\}.$$

Deux paramètres internes de la boîte noire ont été modifiés pour créer un modèle substitut statique du problème LOCKWOOD. Ce modèle s'évalue 5 fois plus rapidement que le vrai problème.

Les résultats de la figure 4.8 proviennent de 100 problèmes numériques générés par 100 points de départ réalisables dans l'hypercube $[9800; 10200]^6$. Les contraintes sont gérées par PB et le nombre maximal d'évaluations du vrai problème est fixé à 1500.




(a) $\tau = 10^{-1}$

(b) $\tau = 10^{-2}$

(c) $\tau = 10^{-3}$

Figure 4.8 Profils de données obtenus avec une tolérance $\tau$ sur 1 instance d'exécution de 100 problèmes numériques LOCKWOOD.

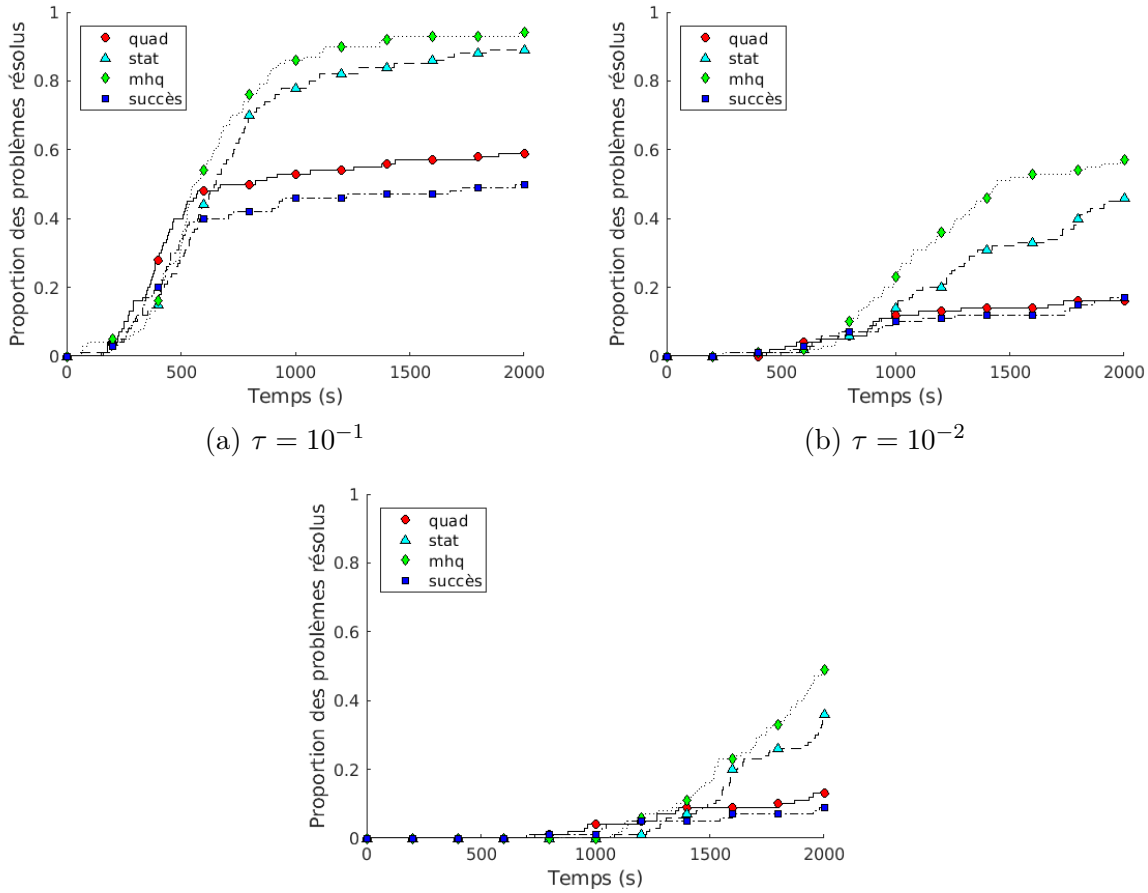
On observe à la figure 4.8 un réel avantage à utiliser un problème substitut afin d'orienter la recherche. En effet, on voit une véritable dominance de la part de stat et mhq. Or, puisque la boîte noire LOCKWOOD prend environ 1.5 secondes à être évaluée et que le modèle statique prend 5 fois moins de temps à être évalué, il est plus approprié de comparer les algorithmes en fonction du temps d'exécution qu'en fonction du nombre d'évaluations.

La figure 4.9 reprend les 100 problèmes numériques de la figure 4.8, mais compare les algorithmes en fonction du temps de calcul. Le temps de calcul maximal alloué est 2000 secondes. Les instances d'exécutions ont été effectuées sur un PC (Intel(R) Core(TM) i7-6700 CPU @

3.40GHz 8.00 GB RAM) avec un système d'exploitation Linux. La condition (4.1) doit être légèrement modifiée pour tracer la proportion de problèmes résolus par l'algorithme $\mathcal{A}$ sous une tolérance $\tau$ en fonction du temps. La formulation de la condition (4.1) devient

$$f(x_0) - f(x_t) \geq (1 - \tau)(f(x_0) - f(x_b)),$$

où $x_t$ est la meilleure solution réalisable obtenue après $t$ secondes de l'instance d'exécution par l'algorithme $\mathcal{A}$.




(a) $\tau = 10^{-1}$

(b) $\tau = 10^{-2}$

(c) $\tau = 10^{-3}$

Figure 4.9 Proportion de problèmes résolus sous une tolérance $\tau$ en fonction du temps sur 1 instance d'exécution de 100 problèmes numériques LOCKWOOD.

Comme à la figure 4.8, on aperçoit à la figure 4.9 une dominance de la part des algorithmes qui exploitent l'information du modèle statique. Avec $\tau = 10^{-1}$, mhq atteint le pourcentage de problèmes résolus par quad après 2000 secondes en seulement $\sim$ 650 secondes. stat suit une tendance semblable à celle de mhq, mais ces résultats sont légèrement inférieurs. Pour

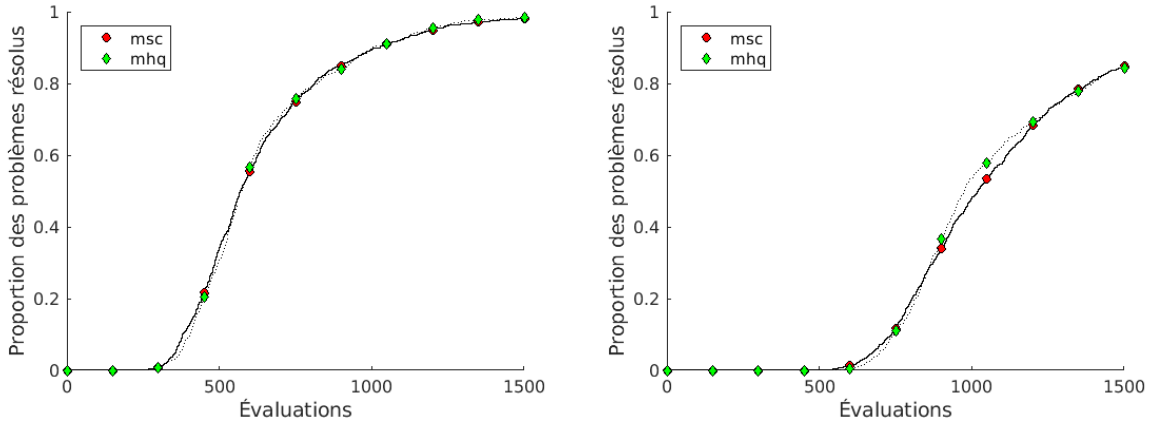
$\tau = 10^{-2}$, mhq se démarque de stat autant pour la vitesse de convergence que pour le nombre de problèmes résolus. En effet, il nécessite $\sim$ 600 évaluations de moins pour atteindre le pourcentage de problèmes résolus par stat après 2000 secondes. Après ce temps, mhq résout 11 problèmes de plus que stat et 40 de plus que succès. Lorsque $\tau = 10^{-3}$, mhq devient le meilleur algorithme après $\sim$ 1200 secondes. Après 2000 secondes, il résout 49 problèmes, soit 13 de plus que stat et 36 de plus que quad.

## 4.4 Discussion

L'apport du MHQ à l'étape de sonde locale de MADS est de permettre une résolution avec une plus grande précision. En effet, dans la majorité des cas, ce sont les graphes avec les plus petites valeurs du facteur de tolérance $\tau$ qui se démarquent. Cette observation suggère que l'information du modèle statique guide la recherche en début d'optimisation et permet d'identifier rapidement un bon bassin. Par la suite, l'ajout des termes quadratiques permet d'accélérer la convergence vers la solution locale, prenant de plus en plus d'importance au fur et à mesure que le nombre de points d'interpolation augmente. Autrement dit, le MHQ bénéficie du caractère global du modèle statique ainsi que de l'aspect local associé au modèle quadratique.

### Retour sur le modèle statique corrigé

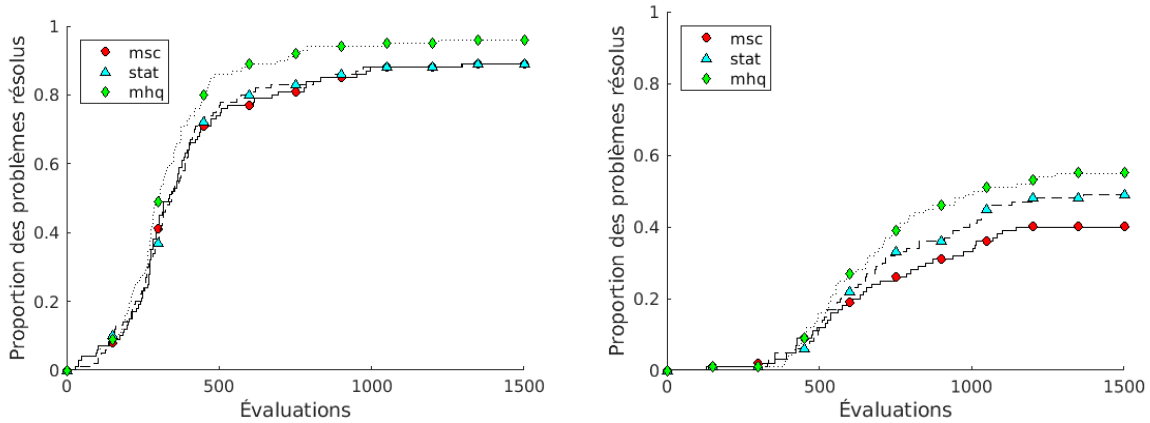
On prend un peu de recul ici pour revenir sur l'alternative au MHQ proposée à la section 4.2 pour le problème *supersonic business jet design*. Le MSC s'applique bien lorsque le modèle statique est une approximation du problème original. Dans les problèmes de type MDO, le relâchement des critères de convergence permet la création d'un modèle simplifié qui approxime le vrai problème. Cette propriété explique donc pourquoi la résolution du problème *supersonic business jet design* est avantagée par le MSC. C'est le cas aussi pour le problème d'ingénierie *simplified wing*. La figure 4.10 reprend les paramètres d'entrée de la figure 4.5, mais ne compare que les méthodes msc et mhq.




(a) $\tau = 10^{-3}$ (b) $\tau = 10^{-5}$

Figure 4.10 Profils de données comparant msc et mhq obtenus avec une tolérance $\tau$ sur 10 instances d'exécution de 100 problèmes numériques *simplified wing*.

On aperçoit pour les deux valeurs de $\tau$ que les méthodes donnent sensiblement les mêmes résultats. Les deux algorithmes semblent donc avoir des comportements semblables lorsque le modèle statique fourni par l'utilisateur est une approximation. Toutefois, ce n'est pas le cas lorsque le modèle statique n'est pas une approximation, comme au problème LOCKWOOD. La figure 4.11 reprend les paramètres d'entrée de la figure 4.8, mais compare les méthodes msc, stat et mhq. On regarde ici le nombre d'évaluations et non le temps de calcul, car les trois méthodes ont des temps d'exécution semblables.




(a) $\tau = 10^{-1}$ (b) $\tau = 10^{-2}$

Figure 4.11 Profils de données comparant msc, stat et mhq obtenus avec une tolérance $\tau$ sur 1 instance d'exécution de 100 problèmes numériques LOCKWOOD.

Les résultats de la figure 4.11 montrent qu'en plus de donner un moins bon ordonnancement

que le MHQ, le MSC détériore l'information du modèle statique. Cette observation montre que le MSC n'est pas adéquat pour tout type de modèle substitut statique. En effet, cette approche ne peut rectifier l'information du modèle statique que par une correction quadratique, ce qui peut être dommageable dans les cas où le modèle statique n'approxime pas le vrai problème. Le MHQ, quant à lui, contrôle l'impact du modèle statique en le considérant comme une variable. C'est pour cette raison que le MHQ s'adapte bien à tout modèle statique et qu'il est préférable au modèle additif proposé.

# CHAPITRE 5 CONCLUSION

## 5.1 Synthèse des travaux

Ce mémoire introduit un nouveau modèle substitut afin d'ordonnancer les points d'essai à l'étape de sonde locale de l'algorithme MADS. Ce modèle, nommé modèle hybride quadratique, est un modèle quadratique légèrement modifié afin qu'il puisse exploiter l'information d'un modèle statique du vrai problème. Au lieu d'apporter une correction additive ou multiplicative au modèle statique, le MHQ généralise ces deux types de corrections en travaillant dans un espace de dimension supérieure, où la dimension supplémentaire correspond à la valeur du modèle statique. Tout comme le modèle quadratique, le MHQ nécessite un nombre minimum de points d'interpolation dans une région de confiance afin d'être construit. Si ce nombre minimum de points est respecté, l'unicité et l'existence du MHQ sont garanties, même si l'ensemble des points d'interpolation n'est pas nécessairement bien distribué dans l'espace.

La nouvelle méthode d'ordonnancement a été testée sur trois problèmes analytiques et trois problèmes contraints d'ingénierie, soit les problèmes *simplified wing*, *supersonic business jet design* et LOCKWOOD. Pour chacun d'entre eux, l'ordonnancement assisté d'un MHQ est comparé avec les méthodes d'ordonnancement les plus communes, soit celle assistée d'un modèle quadratique, celle assistée d'un modèle statique et celle qui ordonne selon l'angle avec le dernier succès. Pour l'ensemble des tests, l'étape de recherche globale a été désactivée et les directions qui interviennent dans la sonde locale de MADS ont été générées par l'algorithme ORTHOMADS avec $2n$ directions.

Les résultats numériques pour les trois problèmes analytiques montrent que l'ordonnancement assisté par un MHQ donne généralement de meilleurs résultats que ces compétiteurs les plus communs. Cette dominance se fait surtout voir pour les deux problèmes Simple_MDO_2n_Variables, où mhq résout plus de 50% des problèmes lorsque $\tau = 10^{-7}$. Toutefois, ceux-ci ne sont pas représentatifs des familles de problèmes qu'on cible, car ils n'ont pas de contraintes. Il est donc plus pertinent d'interpréter les résultats des autres problèmes. Le problème *supersonic business jet design* démontre par son temps d'exécution que mhq n'est pas adéquat pour tout problème de type boîte noire. Les résultats du problème *simplified wing*, quant à eux, ne permettent pas de conclure sur la pertinence de mhq, puisque son comportement est comparable à celui de stat. Finalement, le MHQ semble être le meilleur modèle substitut pour guider la recherche locale de MADS pour le problème LOCKWOOD.

L'ensemble des résultats de ce mémoire suggèrent que la recherche locale de MADS tire profit de l'information du MHQ. En effet, lorsque le nombre de contraintes n'est pas trop élevé, l'ordonnancement assisté d'un MHQ des points d'essai de la sonde locale de MADS donne des résultats comparables ou meilleurs que lorsque les points sont ordonnancés selon un modèle quadratique ou un modèle statique. Il est donc recommandé d'utiliser le MHQ à l'étape de sonde locale lorsqu'un modèle statique du problème est accessible et que le nombre de contraintes de ce problème n'est pas trop grand. Il serait toutefois intéressant de tester cette nouvelle approche sur davantage de problèmes tests pour valider sa pertinence.

## 5.2 Travaux futurs

Trois avenues de recherche pourraient être explorées lors de travaux futurs. Premièrement, au chapitre 4, les directions de sonde locale de chaque méthode sont générées par l'algorithme ORTHOMADS avec $2n$ directions, où $n$ est la dimension du problème. Toutefois, il est montré dans [18] que l'algorithme MADS donne généralement de meilleurs résultats lorsque les directions de sonde locale sont générées par ORTHOMADS avec $n+1$ directions. Cet algorithme génère et ordonne $2n$ directions $\{d_1, -d_1\} \cup \{d_2, -d_2\} \cup \ldots \cup \{d_n, -d_n\}$, puis garde les $n$ directions les plus prometteuses, soit une parmi chaque sous-ensemble $\{d_i, -d_i\}, i \in \{1, \ldots, n\}$. Un deuxième ordonnancement est alors effectué sur les $n$ directions et si aucune d'entre elles n'apporte un succès, on génère une $(n+1)^{\text{ème}}$ direction. Il serait intéressant d'adapter la recherche locale assistée d'un MHQ à l'algorithme ORTHOMADS avec $n+1$ directions. La difficulté ici est que la région de confiance du deuxième ordonnancement est $2^n$ fois plus petite. Ainsi, le MHQ manque souvent de points d'interpolation pour être construit et le deuxième ordonnancement est effectué avec la valeur du modèle statique. Il serait donc pertinent d'ajuster la région de confiance en conséquence afin de permettre une construction plus fréquente de MHQ.

Deuxièmement, comme aperçu à la section 4.2 avec le problème *supersonic business jet design*, la construction des MHQ vient lourdement ralentir l'algorithme MADS lorsque le nombre de contraintes d'un problème est élevé. Pour remédier à la situation, on pourrait calculer les écarts entre les $m$ contraintes du vrai problème et celles du modèle statique pour ensuite modéliser à l'aide d'un MHQ les $m^*$ contraintes avec les plus grands écarts, où $m^*$ est un nombre relativement petit par rapport à $m$. Ainsi, le MHQ pourrait corriger une partie du modèle statique en un temps raisonnable. Une autre possibilité serait de construire un MHQ pour la fonction objectif et des MSC pour les contraintes. En effet, dans la plupart des cas, la fonction objectif est celle dont la valeur de l'approximation est la plus pertinente

dans l'ordonnancement des points. Les contraintes seraient aussi approximées, mais leurs estimations seraient moins fiables que si elles étaient faites avec un MHQ.

Finalement, on définit au chapitre 3 ce qu'est un modèle hybride, mais ce mémoire ne présente que le modèle hybride quadratique. Avec les modèles dynamiques mentionnés à la section 2.4, on pourrait créer d'autres types de modèles hybrides et exploiter leurs informations afin d'orienter la recherche dans un algorithme de recherche directe. Les modèles dynamiques proposés par Talgorn et al. [80] pour guider la recherche globale ou l'ordonnancement de la sonde locale de MADS pourraient être transformés en modèles hybrides afin de bénéficier de l'information d'un modèle statique.

# RÉFÉRENCES


[1] Abramson, M., Audet, C., Couture, G., Dennis, Jr., J., Le Digabel, S., et Tribes, C. The NOMAD project. Logiciel disponible sur https://www.gerad.ca/nomad/.

[2] Abramson, M., Audet, C., Dennis, Jr., J., et Le Digabel, S. OrthoMADS : A deterministic MADS instance with orthogonal directions. *SIAM Journal on Optimization 20*, 2 (2009), 948–966.

[3] Agte, J., Sobieszczanski-Sobieski, J., et Sandusky, R. Supersonic business jet design through bilevel integrated system synthesis. In *Proceedings of the World Aviation Conference* (San Francisco, CA, 1999), vol. SAE Paper No. 1999-01-5622, MCB University Press, Bradford, UK.

[4] Alexandrov, N., Dennis, Jr., J., Lewis, R., et Torczon, V. A trust-region framework for managing the use of approximation models in optimization. *Structural and Multidisciplinary Optimization 15*, 1 (1998), 16–23.

[5] Alexandrov, N., Lewis, R., Gumbert, C., Green, L., et Newman, P. Approximation and model management in aerodynamic optimization with variable-fidelity models. *Journal of Aircraft 38*, 6 (2001), 1093–1101.

[6] Amaioua, N., Audet, C., Conn, A. R., et Digabel, S. L. Efficient solution of quadratically constrained quadratic subproblems within the mesh adaptive direct search algorithm. *European Journal of Operational Research 268*, 1 (2018), 13 – 24.

[7] Amaioua, N., Audet, C., Le Digabel, S., Alarie, S., et Leclaire, L. Selection of variables in parallel space decomposition for the mesh adaptive. Tech. Rep. G-2018-38, Les cahiers du GERAD, 2018.

[8] Audet, C. A survey on direct search methods for blackbox optimization and their applications. In *Mathematics without boundaries : Surveys in interdisciplinary research*, P. Pardalos et T. Rassias, Eds. Springer, 2014, ch. 2, pp. 31–56.

[9] Audet, C., Béchard, V., et Digabel, S. L. Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. *Journal of Global Optimization 41*, 2 (2008), 299–318.

[10] Audet, C., Dennis, J., et Digabel, S. Parallel space decomposition of the mesh adaptive direct search algorithm. *SIAM Journal on Optimization 19*, 3 (2008), 1150–1170.

[11] Audet, C., Dennis, J. E., et Le Digabel, S. Globalization strategies for mesh adaptive direct search. *Computational Optimization and Applications 46*, 2 (2010), 193–215.

[12] Audet, C., et Dennis, Jr., J. Pattern search algorithms for mixed variable programming. *SIAM Journal on Optimization 11*, 3 (2001), 573–594.

[13] Audet, C., et Dennis, Jr., J. A pattern search filter method for nonlinear programming without derivatives. *SIAM Journal on Optimization 14*, 4 (2004), 980–1010.

[14] Audet, C., et Dennis, Jr., J. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization 17*, 1 (2006), 188–217.

[15] Audet, C., et Dennis, Jr., J. A progressive barrier for derivative-free nonlinear programming. *SIAM Journal on Optimization 20*, 1 (2009), 445–472.

[16] Audet, C., et Hare, W. *Derivative-Free and Blackbox Optimization*. Springer Series in Operations Research and Financial Engineering. Springer international Publishing, 2017.

[17] Audet, C., et Hare, W. Model-based methods in derivative-free nonsmooth optimization. Tech. Rep. G-2018-34, Les cahiers du GERAD, 2018.

[18] Audet, C., Ianni, A., Le Digabel, S., et Tribes, C. Reducing the number of function evaluations in mesh adaptive direct search algorithms. *SIAM Journal on Optimization 24*, 2 (2014), 621–642.

[19] Audet, C., Kokkolaras, M., Le Digabel, S., et Talgorn, B. Order-based error for managing ensembles of surrogates in mesh adaptive direct search. *Journal of Global Optimization 70*, 3 (2018), 645–675.

[20] Audet, C., Le Digabel, S., et Tribes, C. NOMAD user guide. Tech. Rep. G-2009-37, Les cahiers du GERAD, 2009.

[21] Audet, C., Le Digabel, S., et Tribes, C. Dynamic scaling in the mesh adaptive direct search algorithm for blackbox optimization. *Optimization and Engineering 17*, 2 (2016), 333–358.

[22] Audet, C., Le Digabel, S., et Tribes, C. The mesh adaptive direct search algorithm for granular and discrete variables. Tech. Rep. G-2018-16, Les cahiers du GERAD, 2018.

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

[24] Audet, C., Savard, G., et Zghal, W. Multiobjective optimization through a series of single-objective formulations. *SIAM Journal on Optimization 19*, 1 (2008), 188–210.

[25] Audet, C., et Tribes, C. Mesh-based Nelder-Mead algorithm for inequality constrained optimization. *Computational Optimization and Applications* (2018).

[26] Bakr, M. H., Bandler, J. W., Madsen, K., et Søndergaard, J. An introduction to the space mapping technique. *Optimization and Engineering 2*, 4 (2001), 369–384.

[27] Bandler, J. W., Cheng, Q. S., Dakroury, S. A., Mohamed, A. S., Bakr, M. H., Madsen, K., et Sondergaard, J. Space mapping : the state of the art. *IEEE Transactions on Microwave Theory and Techniques 52*, 1 (2004), 337–361.

[28] Bhosekar, A., et Ierapetritou, M. Advances in surrogate based modeling, feasibility analysis, and optimization : A review. *Computers and Chemical Engineering 108* (2018), 250 – 267.

[29] Booker, A. J., Dennis, J. E., Frank, P. D., Serafini, D. B., Torczon, V., et Trosset, M. W. A rigorous framework for optimization of expensive functions by surrogates. *Structural optimization 17*, 1 (1999), 1–13.

[30] Box, G. Evolutionary operation : A method for increasing industrial productivity. *Appl. Statist. 6* (1957), 81–101.

[31] Chen, Y.-C., Wei, C., et Yeh, H.-C. Rainfall network design using kriging and entropy. *Hydrological Processes 22*, 3 (2008), 340–346.

[32] Clarke, F. *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.

[33] Colson, B. *Trust-Region Algorithms for Derivative-Free Optimization and Nonlinear Bilevel Programming*. PhD thesis, Département de Mathématique, FUNDP, Namur, Belgium, 2003.

[34] Conn, A., Gould, N., et Toint, P. *Trust region methods*. SIAM, 2000.

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

[36] Conn, A., Scheinberg, K., et Vicente, L. *Introduction to Derivative-Free Optimization*. MOS-SIAM Series on Optimization. SIAM, Philadelphia, 2009.

[37] Conn, A., et Toint, P. An algorithm using quadratic interpolation for unconstrained derivative free optimization. In *Nonlinear Optimization and Applications*, G. D. Pillo et F. Gianessi, Eds. Plenum Publishing, New York, 1996, pp. 27–47.

[38] Craig, J. Bluebird developer manual, 2002. Available at http://www.civil.uwaterloo.ca/jrcraig/pdf/bluebird_developers_manual.pdf.

[39] Custódio, A. L., Rocha, H., et Vicente, L. N. Incorporating minimum frobenius norm models in direct search. *Computational Optimization and Applications 46*, 2 (2010), 265–278.

[40] Davis, C. Theory of positive linear dependence. *American Journal of Mathematics 76* (1954), 733–746.

[41] Dhar, A., et Datta, B. Global optimal design of ground water monitoring network using embedded kriging. *Ground Water 47*, 6 (2009), 806–815.

[42] Eaves, B. On quadratic programming. *Management Science 17*, 11 (1971), 698–711.

[43] Eldred, M., Giunta, A., et Collis, S. Second-order corrections for surrogate-based optimization with model hierarchies. *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference* (2004).

[44] Fasano, G., Morales, J. L., et Nocedal, J. On the geometry phase in model-based algorithms for derivative-free optimization. *Optimization Methods and Software 24*, 1 (2009), 145–154.

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

[46] Fletcher, R., Gould, N., Leyffer, S., Toint, P., et Wächter, A. On the global convergence of trust-region SQP-filter algorithms for general nonlinear programming. *SIAM Journal on Optimization 13*, 3 (2002), 635–659.

[47] Fletcher, R., et Leyffer, S. Nonlinear programming without a penalty function. *Mathematical Programming 91*, 2 (2002), 239–269.

[48] Forrester, A. I., Sóbester, A., et Keane, A. J. Multi-fidelity optimization via surrogate modelling. *Proceedings of the Royal Society of London A : Mathematical, Physical and Engineering Sciences 463*, 2088 (2007), 3251–3269.

[49] Forsythe, G., Malcolm, M., et Moler, C. *Computer methods for mathematical computations*. Prentice-Hall series in automatic computation. Prentice-Hall, 1977.

[50] Friedman, J. H. Multivariate adaptive regression splines. *Ann. Statist. 19*, 1 (1991), 1–67.

[51] Hansen, P., et Mladenović, N. Variable neighborhood search : principles and applications. *European Journal of Operational Research 130*, 3 (2001), 449–467.

[52] Hock, W., et Schittkowski, K. *Test Examples for Nonlinear Programming Codes*. Springer-Verlag, Berlin, Heidelberg, 1981.

[53] Hooke, R., et Jeeves, T. "Direct Search" solution of numerical and statistical problems. *Journal of the Association for Computing Machinery 8*, 2 (1961), 212–229.

[54] Jahn, J. *Tangent Cones*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994, pp. 81–107.

[55] Jones, D., Schonlau, M., et Welch, W. Efficient global optimization of expensive black box functions. *Journal of Global Optimization 13*, 4 (1998), 455–492.

[56] Khac, V. K., Claudia, D., Youssef, H., et Leo, L. Surrogate-based methods for black-box optimization. *International Transactions in Operational Research 24*, 3 (2015), 393–424.

[57] Kitayama, S., Arakawa, M., et Yamazaki, K. Sequential approximate optimization using radial basis function network for engineering optimization. *Optimization and Engineering 12*, 4 (2011), 535–557.

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

[59] Le Digabel, S., et Wild, S. A taxonomy of constraints in simulation-based optimization. Tech. Rep. G-2015-57, Les cahiers du GERAD, 2015.

[60] Le Thi, H., Vaz, A., et Vicente, L. Optimizing radial basis functions by d.c. programming and its use in direct search for global derivative-free optimization. *TOP* (2011), 1–25.

[61] Lewis, R., et Torczon, V. Pattern search algorithms for bound constrained minimization. *SIAM Journal on Optimization 9*, 4 (1999), 1082–1099.

[62] Lewis, R., et Torczon, V. Pattern search methods for linearly constrained minimization. *SIAM Journal on Optimization 10*, 3 (2000), 917–941.

[63] Marduel, X. Optimisation à fidélité variable. Master's thesis, École Polytechnique de Montréal, 2002.

[64] Marduel, X., Tribes, C., et Trépanier, J.-Y. Variable-fidelity optimization : Efficiency and robustness. *Optimization and Engineering 7*, 4 (2006), 479–500.

[65] Matott, L., Leung, K., et Sim, J. Application of MATLAB and Python optimizers to two case studies involving groundwater flow and contaminant transport modeling. *Computers & Geosciences 37*, 11 (2011), 1894–1899.

[66] Matott, L., Rabideau, A., et Craig, J. Pump-and-treat optimization using analytic element method flow models. *Advances in Water Resources 29*, 5 (2006), 760–775.

[67] McKay, M., Beckman, R., et Conover, W. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics 21*, 2 (1979), 239–245.

[68] Mladenović, N., et Hansen, P. Variable neighborhood search. *Computers and Operations Research 24*, 11 (1997), 1097–1100.

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

[70] Myers, R. H., Montgomery, D. C., et Anderson-Cook, C. M. *Response surface methodology : process and product optimization using designed experiments*, 3 ed. Wiley, Hoboken, N.J., 2009.

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

[72] Nuñez, L., Regis, R. G., et Varela, K. Accelerated random search for constrained global optimization assisted by radial basis function surrogates. *Journal of Computational and Applied Mathematics 340* (2018), 276 – 295.

[73] Peherstorfer, B., et Willcox, K. Survey of multifidelity methods in uncertainty propagation, inference, and optimization.

[74] Peremezhney, N., Hines, E., Lapkin, A., et Connaughton, C. Combining gaussian processes, mutual information and a genetic algorithm for multi-target optimization of expensive-to-evaluate functions. *Engineering Optimization 46*, 11 (2014), 1593–1607.

[75] Powell, M. Least frobenius norm updating of quadratic models that satisfy interpolation conditions. *Mathematical Programming 100*, 1 (2004), 183–215.

[76] Rasmussen, C., et Williams, C. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[77] Regis, R. Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions. *Computers and Operations Research 38*, 5 (2011), 837–853.

[78] Regis, R. Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. *Engineering Optimization 46*, 2 (2014), 218–243.

[79] Sarrazin-Mc Cann, L. Opportunisme et ordonnancement en optimisation sans dérivées. Master's thesis, École Polytechnique de Montréal, 2018.

[80] Talgorn, B., Audet, C., Le Digabel, S., et Kokkolaras, M. Locally weighted regression models for surrogate-assisted design optimization. *Optimization and Engineering 19*, 1 (2018), 213–238.

[81] Torczon, V. On the convergence of the multidirectional search algorithm. *SIAM Journal on Optimization 1* (1991), 123–145.

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

[83] Tosserams, S., Kokkolaras, M., Etman, L., et Rooda, J. A non-hierarchical formulation of analytical target cascading. *Journal of Mechanical Design 132*, 5 (2010), 051002–1–051002–13.

[84] Tribes, C., Dubé, J.-F., et Trépanier, J.-Y. Decomposition of multidisciplinary optimization problems : formulations and application to a simplified wing design. *Engineering Optimization 37*, 8 (2005), 775–796.

[85] Tribes, C., et Trépanier, J.-Y. Coordination of multidisciplinary distributed analyses and optimizations. In *Canadian Aeronautics and Space Institute AERO'09 Conference* (2009).