Titre: Title:	Optimisation de boîtes noires à précision variable
Auteur: Author:	Pierre-Yves Bouchet
Date:	2019
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Bouchet, P.-Y. (2019). Optimisation de boîtes noires à précision variable [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/3840/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Optimisation de boîtes noires à précision variable

PIERRE-YVES BOUCHET
Département de mathématiques et de génie industriel

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

Mai 2019

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Optimisation de boîtes noires à précision variable

présenté par **Pierre-Yves BOUCHET**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Dominique ORBAN, président
Charles AUDET, membre et directeur de recherche
Richard LABIB, membre

DÉDICACE

À ma famille, pour son soutien inconditionnel,
ainsi qu'à quelques proches, qui le seront toujours
malgré la distance qui pourra nous séparer.
Merci.
Nous sommes faits de l'étoffe dont sont tissés les vents...

REMERCIEMENTS

Le dépôt de ce mémoire, qui marque la fin d'un projet conséquent, m'offre la possibilité de remercier comme il se doit de nombreuses personnes l'ayant rendu possible.

En premier lieu, je souhaite remercier les membres du jury présents lors de ma soutenance : MM. Dominique Orban, Richard Labib et Charles Audet, pour l'appréciation qu'ils portent à mon travail.

Cette maîtrise recherche marque pour moi un nouveau pas vers la carrière académique que j'essaie de construire, et qui sera embellie des noms des institutions m'ayant hébergé : L'École Polytechnique de Montréal et le Groupe d'Études et de Recherche en Analyse des Décisions (GERAD).

La rédaction de ce document a requis beaucoup d'efforts, et les relectures par d'autres personnes m'ont été utiles. Un grand merci à MM. Charles Audet et Sébastien Le Digabel pour leur suivi régulier de l'avancement de mon mémoire.

Merci également à M. Stéphane Alarie pour l'opportunité de confronter mes algorithmes à un problème réel. La boîte noire sur laquelle nous travaillions était bien plus complexe que les problèmes de test que j'utilisais de mon côté, et a mis en évidence le bien-fondé de ma méthode ainsi que certaines possibilités de l'améliorer.

Je pense également à mes collègues au GERAD, grâce à qui chaque «pause café» pouvait se transformer en terreau fertile pour la croissance de nouvelles idées liées à nos recherches, riche d'une excellente ambiance et des domaines d'expertise de chacun.

Enfin, j'aimerais remercier tout particulièrement mon directeur, M. Charles Audet. Les recherches présentées dans ce document n'ont pu aboutir que grâce à la confiance qu'il m'a accordé en acceptant ma candidature à faire une maîtrise sous sa direction. Ce projet a été, à titre personnel, particulièrement enrichissant. Qu'il soit remercié pour sa disponibilité et ses nombreux commentaires durant l'ensemble du projet, son immense expertise en optimisation, et le soutien financier qu'il m'a apporté pour que je puisse me concentrer sur ma recherche. Ce fut un réel plaisir de travailler sous sa direction, dans un cadre aussi studieux qu'agréable, pour produire un résultat que j'estime intéressant et dont je suis personnellement fier.

RÉSUMÉ

Ce projet s'insère dans le cadre d'un domaine de mathématiques appliquées qu'on appelle optimisation. Il s'agit de l'étude de problèmes dans lesquels on souhaite maximiser ou minimiser une certaine quantité d'intérêt qu'on nommera *fonction objectif*, tout en respectant certaines limitations éventuelles qu'on désignera par *contraintes*.

Dans notre cadre d'étude, la fonction objectif est une *boîte noire*, terme désignant une fonction dont la forme analytique est inconnue. La seule information qu'on puisse obtenir est la valeur qu'elle prend lorsqu'on l'évalue en un point, sans qu'on sache comment cette valeur a été calculée. Ainsi, il faut trouver un point minimisant la valeur renvoyée, et ce sans connaître le comportement de la fonction.

Enfin, on supposera dans ce projet que les valeurs renvoyées par la fonction objectif ne sont pas exactes. L'objectif de ce travail est d'optimiser des fonction dites *bruitées*, et dont l'amplitude du bruit est réglable. Toute évaluation de la fonction objectif renvoie la véritable valeur de la fonction plus une erreur aléatoire. L'*amplitude du bruit* sera alors l'écart-type de cette erreur.

Ainsi, une *boîte noire à précision variable* est une fonction inconnue (dont on ne peut pas prédire le comportement) qui ne peut jamais être évaluée de façon certaine, à cause d'une erreur aléatoire apparaissant à chaque évaluation, mais dont on peut limiter l'amplitude en réglant la précision du calcul. On dispose ici d'un «levier» de précision, offrant un compromis entre garantie de qualité de la valeur observée (à haute précision) au prix d'un grand temps de calcul, et incertitude entourant la valeur renvoyée (à faible précision) mais obtenue rapidement.

Les recherches résumées dans ce document ont pour but de proposer une façon d'optimiser de telles fonctions, c'est-à-dire un moyen de trouver un point minimisant la valeur exacte qu'elles devraient renvoyer en l'absence d'erreur. Un tel point devrait également pouvoir être calculé de façon efficace, en un temps raisonnable et un effort de calcul faible.

Pour cela, nous exploitons la précision variable pour toujours estimer la fonction objectif avec une précision au juste nécessaire : suffisamment grande pour affirmer qu'un point est meilleur qu'un autre, et assez basse pour que ces calculs soient les plus rapides possibles.

ABSTRACT

This document aims to introduce the reader to some research in a sub-topic in optimisation, a field of applied mathematics which tries to maximise or minimise a quantity named *objective function*. This quantity depends on some *variables* that can be chosen by the optimiser.

Usually, one may exploits its knowledge of the structure of the function to make the optimisation as fast as possible. For example, knowing that a given variable unconditionally diminishes the objective while augmenting, one can decide to maximise that variable. One may also think about the derivatives of the function, as knowing these helps to easily find some promising directions to search on.

In *blackbox optimisation*, the topic this work will discuss, such knowledge in intractable. When the objective is a *blackbox*, its behaviour cannot be recovered from observations and calculus. As this kind of function is naturally hard to minimise, finding efficient ways to perform it is the main goal of this field.

In addition to these difficulties, one may face an harder blackbox problem named *noisy blackbox problem*. In this context, the blabkbox to optimise is *noisy*, which means any value it returns is possibly inaccurate. The value given to the optimiser is the real value one should expect, plus a random error. We will denote by *magnitude of the noise* the standard deviation of these errors. It will be assumed that the magnitude can be chosen by the optimiser, with the drawback that the lower the desired magnitude is, the higher the computation time is.

Then, this document studies *noisy blackboxes with adaptive precision* : functions with unpredictable behaviour (*blackboxes*) which return inaccurate values (because the functions are *noisy*) but with a controllable amount of noise (as the *precision* is *adaptive*).

The main goal of this research is to find an efficient way to minimise such a function. By *minimising*, we mean finding a set of variables which have the lowest expected returned value : assuming that the errors are void in average, we target the lowest real, non-noisy, value returned by the function.

To reach that goal, the *adaptive precision* tradeoff have been useful. We tried to perform any calculus at the lowest precision acceptable, in order to make the computation as fast as possible, but accurate enough to ensure the optimisation determines with few risks where the interesting solutions are.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

Ce premier chapitre a comme vocation de présenter tous les prérequis nécessaires à la bonne compréhension de ce projet. Nous proposons une brève introduction aux concepts couramment utilisés en optimisation pour analyser l'*optimalité* d'une solution. Ce terme sera rigoureusement défini, via une condition utile sur le plan théorique. Le cadre général de l'optimisation de boîte noire sera présenté. Nous y ajouterons divers éléments spécifiques au projet, notamment les notations que nous utiliserons. Les rappels de probabilités se contenteront de présenter la loi normale, ainsi que les théorèmes qui nous seront utiles lors de l'analyse de convergence de nos algorithmes. Le concept de *substitut à précision variable* est également introduit comme cadre englobant nos fonctions bruitées à précision variable. Une illustration concrète d'un tel substitut sera donnée via les *simulations de Monte-Carlo*.

1.1 Définitions et concepts fondamentaux

Les résultats présentés ici sont issus de connaissances générales en mathématiques, et peuvent apparaître dans de nombreux domaines. Nous les rappelons pour introduire des notions telles que *solution optimale*, dont la définition dépend de ces concepts.

1.1.1 Notions d'analyse mathématique par différentiabilité

L'analyse par différentiabilité permet d'établir des garanties que pour une fonction f donnée, un point x qu'on pense optimal «ne peut être amélioré», au sens où on ne peut trouver de points y voisins de x tels que $f(y) < f(x)$. Un tel point x est alors un *optimum local* de f, et x est une *solution optimale* au problème de minimisation de f.

Continuité de Lipschitz

Une fonction $f : \mathbb{R}^n \to \mathbb{R}^m$ est *lipschitzienne* sur $\mathcal{X} \subset \mathbb{R}^n$, et notée $f \in \mathcal{C}^{0^+}(\mathcal{X}, \mathbb{R}^m)$, si $\exists K > 0 \mid \forall x, y \in \mathcal{X}^2,\ \|f(x) - f(y)\| \leq K \|x - y\|$.

Par extension, on pourra parler de fonction *localement lipschitzienne* au voisinage d'un point x quelconque, lorsque la fonction est lipschitzienne sur un voisinage de x. Ceci est vérifié si $\exists K(x) > 0$ et $\exists r > 0 \mid \forall u, v \mid \|u - x\| < r$ et $\|v - x\| < r,\ \|f(u) - f(v)\| \leq K(x) \|u - v\|$.

On peut noter qu'une fonction lipschitzienne est nécessairement continue, d'où la notation employée $\mathcal{C}^{0^+}$.

Dérivée, gradient

Une fonction continue f définie de $\mathbb{R}$ dans $\mathbb{R}$ est dite *dérivable en* x si $\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$ existe et est finie. Ce taux limite est alors nommé *taux d'accroissement en* x et noté $f'(x)$; il représente le taux de variation de la fonction autour du point x.

Cette notion se généralise aux fonctions de plusieurs variables. Soit f définie de $\mathbb{R}^n$ dans $\mathbb{R}$. On a $\forall i \in [\![1,n]\!]$ la i^{eme} *dérivée partielle en* x : $\frac{\partial f}{\partial x_i}(x) = \lim_{h\to 0}\frac{f(x+he_i)-f(x)}{h}$, où les e_i forment la base canonique. Cette définition ne tient évidemment que si cette limite existe et est finie.

Si une fonction $f : \mathbb{R}^n \to \mathbb{R}$ est telle que toutes les opérations de dérivation partielle sont répétables k fois, et que toutes les dérivées obtenues sont continues, alors f est dite k *fois continûment dérivable* ; ce qu'on note $f \in \mathcal{C}^k(\mathbb{R}^n, \mathbb{R})$.

Pour une fonction $f \in \mathcal{C}^1(\mathbb{R}^n, \mathbb{R})$, il est utile de définir le vecteur *gradient* de f en x : $\nabla f(x) = \left[\frac{\partial f}{\partial x_i}(x) \mid i \in [\![1,n]\!]\right]^\top$, représentant la direction dans $\mathbb{R}^n$ qui, partant de x, donne la plus forte augmentation de f.

Enfin, on définit la *dérivée directionnelle* en x dans la direction unitaire d, par $f'(x;d) = \lim_{h\searrow 0}\frac{f(x+hd)-f(x)}{h}$ lorsque cette limite existe. On a l'implication très utile : si x est un *minimum local* (ou *optimum local*) de $f \in \mathcal{C}^1(\mathbb{R}^n, \mathbb{R})$, alors $\forall d \in \mathbb{R}^n,\ f'(x;d) \geq 0$.

Dérivée généralisée

Toutes les définitions présentées ci-avant peuvent ne pas s'appliquer dans le cadre de ce projet, où les fonctions ne seront supposées, au mieux, que lipschitziennes $\left(\mathcal{C}^{0^+}\right)$. Néanmoins, il est dommage de s'en priver totalement car elles fournissent des informations très utiles (le gradient, par exemple, indique les directions dans lesquelles il est intéressant de chercher pour améliorer une solution). Il existe heureusement des définitions plus larges, et applicables à des fonctions n'étant que $\mathcal{C}^{0^+}$, permettant de généraliser ces concepts. Notamment, la *dérivée généralisée* de Clarke (Clarke (1983)).

Définition 1.1.1 (Dérivée généralisée)**.** *Soit* $x \in \mathbb{R}^n$, $f \in \mathcal{C}^{0^+}(\mathbb{R}^n, \mathbb{R})$ *au voisinage de* x, *et* $d \in \mathbb{R}^n$. *La dérivée généralisée, nommée aussi dérivée de Clarke, de* f *en* x *dans la direction unitaire* d *est la quantité*

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

lorsque cette limite existe.

Pour des fonctions $\mathcal{C}^1$, la dérivée généralisée coïncide avec la dérivée directionnelle.

De manière analogue au cas $\mathcal{C}^1$, on a le résultat d'optimalité suivant sur les fonctions $\mathcal{C}^{0^+}$ au voisinage de x :

Si x est un *minimum local* (ou *optimum local*) de f, alors $\forall d \in \mathbb{R}^n,\ f^\circ(x;d) \geq 0$.

Suite raffinante

Beaucoup d'algorithmes présentés par la suite évalueront des points autour d'une solution courante donnée. Si on nomme $(x_k)_k$ la suite des optimums courants, $(d_k)_k$ la suite des directions de recherche et $(t_k)_k$ la suite des pas, tous les points évalués à l'itération k sont de la forme $x_k + t_k d_k$. On utilise alors le concept de *suite raffinante* proposé par Audet et Dennis, Jr. (2006), lié à ces définitions :

Définition 1.1.2 (Suite raffinante)**.** *Considérant une suite $(x_k)_k$ de points, $(d_k)_k$ de directions unitaires et $(t_k)_k$ de réels positifs, la suite $(x_k)_k$ est dite raffinante si on a :*

$$\begin{cases} \exists x \mid x_k \underset{k\to\infty}{\longrightarrow} x \\ \exists d \mid d_k \underset{k\to\infty}{\longrightarrow} d \\ t_k \underset{k\to\infty}{\longrightarrow} 0 \end{cases}$$

ou, autrement dit, si la suite $(x_k)_k$ des points converge vers un point limite x, et qu'en même temps les directions de recherche unitaires $(d_k)_k$ convergent vers une direction d et les pas de recherche $(t_k)_k$ tendent vers 0. On dira qu'alors la suite est raffinante vers $(x;d)$.

On a alors (Audet et Hare (2017)) un critère pour vérifier la positivité de $f^\circ(x;d)$:

Théorème 1.1.3 (Critère d'optimalité d'un point dans une direction)**.** *Soient x et d deux points de $\mathbb{R}^n$ (où d sera unitaire, et vu comme une direction). Soit $(x_k)_k$, $(d_k)_k$ et $(t_k)_k$ une suite raffinante vers $(x;d)$. On a alors :*

$$\lim_{k\to\infty} \frac{f(x_k + t_k d_k) - f(x_k)}{t_k} \leq f^\circ(x;d)$$

et ainsi, si la limite exprimée en fonction de la suite raffinante est positive, la dérivée $f^\circ(x;d)$ l'est également.

1.1.2 Optimisation de boîte noire

L'optimisation de fonctions dites «boîte noire» a pour but de considérer un cadre très général et peu restrictif. Les hypothèses faites sur la fonction objectif et les contraintes sont les plus faibles possible. De fait, dans ce domaine très peu de structures mathématiques sont exploitables (les dérivées ou la convexité ne le sont pas, par exemple). On cherchera donc des méthodes n'exploitant que les valeurs des fonctions aux points déjà évalués.

Exemple d'application

L'archétype d'une fonction de type «boîte noire» est une lourde simulation informatique, renvoyant la performance d'un jeu de variables de dimensionnement. Par exemple, les variables x peuvent définir le profil d'une aile d'avion, et $f(x)$ renverra une estimation numérique de la portance de cette aile. Une telle fonction est non-dérivable puisqu'une légère modification d'une longueur peut créer des turbulences dans le flux d'air, et ainsi briser la portance. De fait, en tant qu'optimiseur nous n'avons pas accès à son expression analytique (car sa valeur est calculée par simulation informatique). Les seules informations accessibles sont les valeurs renvoyées aux points déjà considérés.

Formalisation mathématique

Soit $n \in \mathbb{N}^*$ un entier correspondant au nombre de variables du problème. L'espace des variables est un ensemble $\mathcal{X} \subset \mathbb{R}^n$ sur lequel la fonction objectif f est définie. On note en général $f : x \in \mathbb{R}^n \to \overline{\mathbb{R}}$, avec $\overline{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$ et $f(x) = +\infty$ lorsque $x \notin \mathcal{X}$.

On peut ajouter au problème diverses contraintes. Notamment des bornes sur les valeurs de x : soient $l \in \overline{\mathbb{R}}^n$ et $u \in \overline{\mathbb{R}}^n$ deux vecteurs de bornes définissant la boîte $[l, u]$ par $l \leq x \leq u$.[1] De plus, on peut imposer m contraintes modélisées par une fonction $c : \mathbb{R}^n \to \overline{\mathbb{R}}^m$ et la contrainte $c(x) \leq 0$.

Un problème d'optimisation de boîte noire se formule de la façon suivante :

$$\left[\begin{array}{rcll} n, m & \in & \mathbb{N}^* \times \mathbb{N} & \text{Déterminer } \min\limits_{x \in \mathbb{R}^n} \quad f(x) \\ f & \in & Fct(\mathbb{R}^n, \overline{\mathbb{R}}) & \text{sous contrainte} \quad x \in \mathcal{D} \\ c & \in & Fct(\mathbb{R}^n, \overline{\mathbb{R}}^m) & \\ (l, u) & \in & \overline{\mathbb{R}}^n \times \overline{\mathbb{R}}^n,\ l \leq u & \text{avec } \mathcal{D} = \{x \in \mathcal{X} \mid l \leq x \leq u \text{ et } c(x) \leq 0\} \\ \mathcal{X} & = & \{x \mid f(x) < +\infty\} & \text{l'ensemble des solutions dites } \textit{réalisables}. \end{array}\right]$$

1. L'inégalité $a \leq b$ entre deux vecteurs signifie ici l'inégalité sur chaque composante : $\forall i \in [\![1, n]\!], a_i \leq b_i$

1.1.3 Notions de probabilités

Ici, nous ne présenterons que quelques résultats de la théorie des probabilités utiles pour saisir le problème et la solution apportée. L'univers duquel on tire les réalisations sera noté Ω. Une discussion approfondie des fondamentaux théoriques est trouvable dans Durrett (2010).

Loi normale

Il s'agit d'une loi de probabilité très commune, et intervenant de façon cruciale dans la théorie. Commençons par rappeler son expression générale :

$$\mathcal{N}\left(\mu,\sigma^2\right):\begin{cases}\text{Moyenne}: & \mu\\ \text{Variance}: & \sigma^2\\ \text{Densité}: & \forall x\in\mathbb{R},\ \phi_{\mu,\sigma}(x)=\dfrac{1}{\sigma\sqrt{2\pi}}\exp\left\{-\dfrac{1}{2}\left(\dfrac{x-\mu}{\sigma}\right)^2\right\}\\ \text{Distribution}: & \forall z\in\mathbb{R},\ \Phi_{\mu,\sigma}(z)=\mathbb{P}(\mathcal{N}\left(\mu,\sigma^2\right)\leq z)=\displaystyle\int_{-\infty}^{z}\phi_{\mu,\sigma}(x)dx\end{cases}$$

dont on peut spécifier la forme dite *centrée réduite* :

$$\mathcal{N}\left(0,1\right):\begin{cases}\text{Moyenne}: & 0\\ \text{Variance}: & 1\\ \text{Densité}: & \forall x\in\mathbb{R},\ \phi(x)=\phi_{0,1}(x)=\dfrac{1}{\sqrt{2\pi}}\exp\left\{-\dfrac{x^2}{2}\right\}\\ \text{Distribution}: & \forall z\in\mathbb{R},\ \Phi(z)=\Phi_{0,1}(z)=\mathbb{P}(\mathcal{N}\left(0,1\right)\leq z)=\displaystyle\int_{-\infty}^{z}\phi(x)dx.\end{cases}$$

On introduira également la fonction Ψ suivante, qui détermine la probabilité qu'une observation y d'une loi $\mathcal{N}\left(\mu_y,\sigma_y^2\right)$ soit inférieure à une autre observation x d'une loi $\mathcal{N}\left(\mu_x,\sigma_x^2\right)$, sachant que les lois sont indépendantes et que l'écart entre leurs moyennes est $\delta=\mu_y-\mu_x$:

$$\begin{aligned}\Psi(\delta,\sigma_x,\sigma_y) &= \mathbb{P}\left(Y\leq X\mid X\sim\mathcal{N}\left(0,\sigma_x^2\right),\ Y\sim\mathcal{N}\left(\delta,\sigma_y^2\right)\right)\\ &= \mathbb{P}\left(Y-X\leq 0\mid X\sim\mathcal{N}\left(0,\sigma_x^2\right),Y\sim\mathcal{N}\left(\delta,\sigma_y^2\right)\right)\\ &= \mathbb{P}\left(Z\leq 0\mid Z\sim\mathcal{N}\left(\delta-0,\sigma_x^2+\sigma_y^2\right)\right)\\ &= \mathbb{P}\left(\frac{Z-\delta}{\sqrt{\sigma_x^2+\sigma_y^2}}\leq\frac{0-\delta}{\sqrt{\sigma_x^2+\sigma_y^2}}\mid Z\sim\mathcal{N}\left(\delta,\sigma_x^2+\sigma_y^2\right)\right)\\ &= \mathbb{P}\left(U\leq\frac{-\delta}{\sqrt{\sigma_x^2+\sigma_y^2}}\mid U\sim\mathcal{N}\left(0,1\right)\right)\\ &= \Phi\left(\frac{-\delta}{\sqrt{\sigma_x^2+\sigma_y^2}}\right).\end{aligned}$$

Lois des grands nombres

Ces deux lois contrôlent l'incertitude associée à un grand nombre d'observations d'un phénomène aléatoire. Elles stipulent, de façon plus ou moins forte, que la moyenne empirique des observations tendra toujours vers la moyenne théorique de la loi observée. Les deux lois s'expriment comme suit :

Théorème 1.1.4 (Loi faible des grands nombres)**.** *Soit $(X_n)_{n\in\mathbb{N}^*}$ une suite de variables aléatoires indépendantes telles que $\exists(\mu,\sigma) \mid \forall n \in \mathbb{N}^*,\ \mathbb{E}(X_n) = \mu$ et $\mathbb{V}(X_n) = \sigma^2$. Soit $Y_n = \frac{1}{n}\sum_{i=1}^{n} X_i$ la moyenne empirique des n premières observations de la suite des X_i. On a :*

$$\forall \epsilon > 0,\ \lim_{n\to\infty} \mathbb{P}(|Y_n - \mu| \geq \epsilon) = 0.$$

Théorème 1.1.5 (Loi forte des grands nombres)**.** *Soit $(X_n)_{n\in\mathbb{N}^*}$ une suite de variables aléatoires, indépendantes et intégrables, suivant toutes une même loi de probabilité d'espérance finie E. Soit $M_n = \frac{1}{n}\sum_{i=1}^{n} X_i$ la moyenne empirique des n premières observations de la suite des X_i. On a :*

$$\mathbb{P}\left(\lim_{n\to\infty} M_n = E\right) = 1.$$

Théorème central limite

Ce très grand résultat de probabilités permet de ramener un ensemble d'observations indépendantes à une distribution bien connue (une normale). Une utilisation classique est de considérer un grand nombre d'observations d'une même loi inconnue : leur moyenne suit une loi normale de moyenne et variance égales à celles de la loi. Le théorème s'énonce comme suit :

Théorème 1.1.6 (Théorème central limite (TCL))**.** *Soit L une loi de probabilité de moyenne μ et d'écart-type $\sigma \neq 0$. Soient $n \in \mathbb{N}$, et $X_1, \ldots, X_n$ n variables aléatoires indépendantes et de même loi L. Soient $M_n = \frac{1}{n}\sum_{i=1}^{n} X_i$ et $Z_n = \frac{M_n - \mu}{\sigma/\sqrt{n}}$. On a alors :*

$$\forall z \in \mathbb{R},\ \lim_{n\to\infty} \mathbb{P}(Z_n \leq z) = \Phi(z).$$

Ce théorème montre que la moyenne empirique de n observations d'une même loi L suit (lorsque n devient grand) une loi normale centrée en la moyenne μ de L et d'écart-type $\sigma/\sqrt{n}$, où σ est l'écart-type de L. La loi normale est donc une hypothèse très naturelle lorsqu'on manipule une loi inconnue mais dont on dispose de beaucoup d'observations.

Lemme de Borel-Cantelli

Ce lemme formalise la probabilité, connaissant une suite d'événements potentiels, d'en observer une infinité. Il stipule que :

Théorème 1.1.7 (Lemme de Borel-Cantelli)**.** *Soit* $(A_i)_{i\in\mathbb{N}}$ *une suite d'événements de probabilités* $\mathbb{P}(A_i)$*. On a une condition pour déterminer si une infinité d'événements se réalisent :*

$$(\exists L \subset \mathbb{N}, Card(L) = \infty \mid \forall l \in L,\ A_l \text{ se réalise}) \iff \left(\sum_{i=0}^{\infty} \mathbb{P}(A_i) = +\infty\right).$$

p-valeur statistique d'une hypothèse

De façon générique, la *p-valeur* statistique d'une hypothèse désigne la cohérence entre une hypothèse et les observations d'un système.

Supposons par exemple que l'on dispose d'observations $l_1, \ldots, l_n$ d'une loi inconnue L, et que toutes sont négatives : $\forall i \in [\![1, n]\!],\ l_i < 0$. On peut s'interroger sur la moyenne de L, qu'on désignera par μ. Puisqu'elle est inconnue, on peut faire plusieurs hypothèses sur sa valeur, notamment $H_+ : \mu \geq 0$ opposée à l'hypothèse contraire $H_- : \mu < 0$. On souhaite alors déterminer la plus plausible, au vu des observations $l_1, \ldots, l_n$. Supposons de plus, pour simplifier, qu'on sache que $L \sim \mathcal{N}(\mu, \sigma^2)$ avec σ connue. La seule incertitude vient de sa moyenne. Sachant qu'on a toujours toutes les observations $l_1, \ldots, l_n$ négatives, il vient que l'hypothèse H_- est plus crédible que H_+ : si μ est positif (ce que propose H_+), la probabilité qu'on observe $l_1 < 0, \ldots, l_n < 0$ est très faible. En d'autres termes, si H_+ est vraie, il est fortement improbable qu'on observe ce qu'on a pourtant observé.

C'est cette probabilité qu'on appelle *p-valeur*. Il s'agit de la probabilité à priori que sous une hypothèse, une observation égale celle réellement obtenue : $\mathbb{P}(x \mid H)$, où x est un ensemble d'observations effectuées et H une hypothèse sur la loi qu'elles suivent. La *p-valeur* est la cohérence statistique entre une hypothèse et des observations.

Un cas d'application très utile est la comparaison de deux réels μ_1 et μ_2 inconnus, pour lesquels on a des estimations $\hat{\mu_1}$ et $\hat{\mu_2}$ obtenus comme dans l'annexe A. On cherche à savoir lequel des μ_i est le plus petit. On fait donc l'hypothèse «$\mu_1 < \mu_2$» et en calcule sa *p-valeur* : $\mathbb{P}(\hat{\mu_1} < \hat{\mu_2} \mid \mu_1 < \mu_2 \text{ et } \hat{\mu_1}, \hat{\mu_2} \text{ issus d'observations connues})$. Si cette p-valeur est proche de 0.5, on ne peut rien conclure de nos observations : les deux hypothèses sont aussi crédibles l'une que l'autre. Par contre, si elle est proche de 0 ou de 1, on peut sans grand risque accepter l'une de nos hypothèses : proche de 1 signifie que l'hypothèse «$\mu_1 < \mu_2$» est très plausible, tandis que c'est «$\mu_1 \geq \mu_2$» qui est très plausible si la p-valeur est proche de 0.

1.2 Éléments de la problématique

Dans le cadre de ce projet, il faut avoir conscience d'autres éléments complexifiant la problématique. Le cadre générique de boîte noire présenté ci-avant ne suffit pas à englober l'intégralité des notions utiles par la suite.

Cette section a pour but de présenter les notions de *substitut* à la fonction objectif, ainsi que les fonctions à *précision variable*, dont les *simulations de Monte-Carlo* en sont un exemple.

1.2.1 Simulation de Monte-Carlo

Une simulation de Monte-Carlo a pour objectif d'estimer une quantité $\mathcal{A}$ définie comme l'intégrale sur un espace, par la moyenne des évaluations en un nombre fini de points de cet espace. Le principe de cette simulation est d'exploiter la «presque-égalité» suivante :

$\forall a \in Fct(\mathbb{R}^n, \mathbb{R})$ qu'on souhaite intégrer sur un ensemble borné $A \subset \mathbb{R}^n$, de volume défini par $Vol(A) = \int_{\vec{x} \in A} d\vec{x}$, et $\forall N \in \mathbb{N}^*,\ \vec{x_1}, \ldots, \vec{x_N}$ un échantillon de N observations indépendantes d'un vecteur aléatoire $\vec{X}$ de loi uniforme sur A,

$$\mathcal{A} = \int_{\vec{x} \in A} a(\vec{x}) d\vec{x} \approx \widetilde{\mathcal{A}} = Vol(A) \times \frac{1}{N} \sum_{i=1}^{N} a(\vec{x_i}).$$

La valeur estimée $\widetilde{\mathcal{A}}$ est donc aléatoire, mais de moyenne centrée sur $\mathcal{A}$. On peut également calculer la variance de cet estimé :

$$\begin{aligned}
\mathbb{V}\left(\widetilde{\mathcal{A}}\right) &= \frac{Vol(A)^2}{N} \mathbb{V}(a(\vec{X})) \\
&\underset{\overline{a}=\frac{\mathcal{A}}{Vol(A)}}{=} \frac{Vol(A)^2}{N} \mathbb{E}\left(\left(a(\vec{X}) - \overline{a}\right)^2\right) \\
&= \frac{1}{N} \underbrace{Vol(A)^2 \int_{\vec{x} \in A} \left(a(\vec{x}) - \overline{a}\right)^2 \frac{1}{Vol(A)} d\vec{x}}_{\text{constante, notée cste}} \\
&= \frac{cste}{N}.
\end{aligned}$$

Ainsi, sous réserve que l'intégrale ci-dessus existe, la variance est finie et diminue lorsque le nombre N de tirages utilisés dans la simulation augmente.

Une conséquence intéressante de ce résultat est le lien explicite qu'il donne entre le nombre de tirages utilisés et la variance obtenue. Via cet intermédiaire, à variance maximale imposée, on peut déterminer la valeur minimale de N à utiliser pour respecter cette demande.

Cependant, augmenter le nombre de tirages dans la simulation augmente le temps de calcul requis. Il n'est donc pas viable de pousser trop loin la précision demandée. Les courbes suivantes illustrent le problème. On propose ici d'estimer la valeur de π via la méthode de Monte-Carlo, ce qui revient à intégrer la fonction $a : \begin{cases} [-1,1]^2 & \to & \{0,1\} \\ (x,y) & \to & 1 \text{ si } x^2+y^2 \leq 1, \text{ 0 sinon} \end{cases}$ sur $[-1,1]^2$.

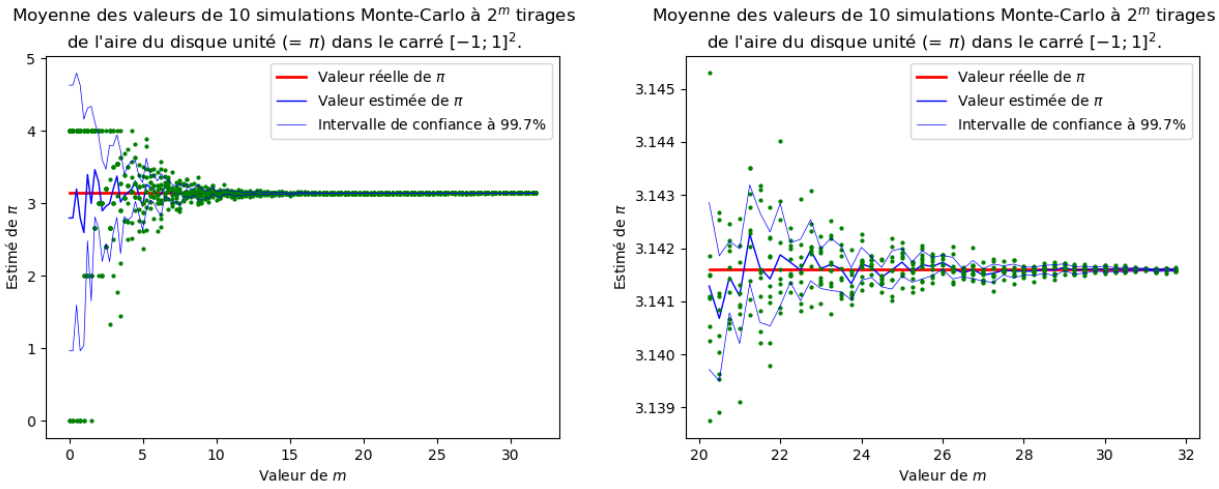

Figure 1.1 Effet du nombre de tirages de Monte-Carlo sur la précision.

Sur cette première figure (1.1), on peut observer que même avec un nombre de tirages élevé, l'imprécision reste grande. Il faut monter à 2^{28} tirages pour que chacune des 10 simulations effectuées renvoient un estimé erroné à moins de 10^{-4}. Ainsi, même s'il y a diminution de l'incertitude entourant les estimés lorsque le nombre de tirages augmente, il paraît clair que ce nombre doit être grand pour avoir des résultats significatifs.

Cela amène au problème sus-cité, observable sur la figure 1.2. Le temps de calcul requis pour effectuer la simulation croît très fortement avec le nombre de tirages demandé. Il est difficile d'envisager l'optimisation d'une fonction effectuant des simulations précises, car 2^{32} tirages par simulation seraient trop long à calculer. Le diagramme de phase proposé donne le lien empirique entre écart-type désiré et temps de calcul requis. On y observe par exemple qu'une évaluation ayant un écart-type imposé à 10^{-4} requiert environ 200 secondes sur un Intel Core i7-6700 cadencé à 3.40 GHz.

On peut remarquer un détail intéressant : en vertu du théorème central limite (1.1.6), l'estimé $\widetilde{\mathcal{A}}$ suit une loi normale dépendant du nombre de tirages, qu'on peut définir par :

$$\widetilde{\mathcal{A}} \sim \mathcal{N}\left(\mathcal{A}, \frac{cste}{N}\right). \tag{1.1}$$

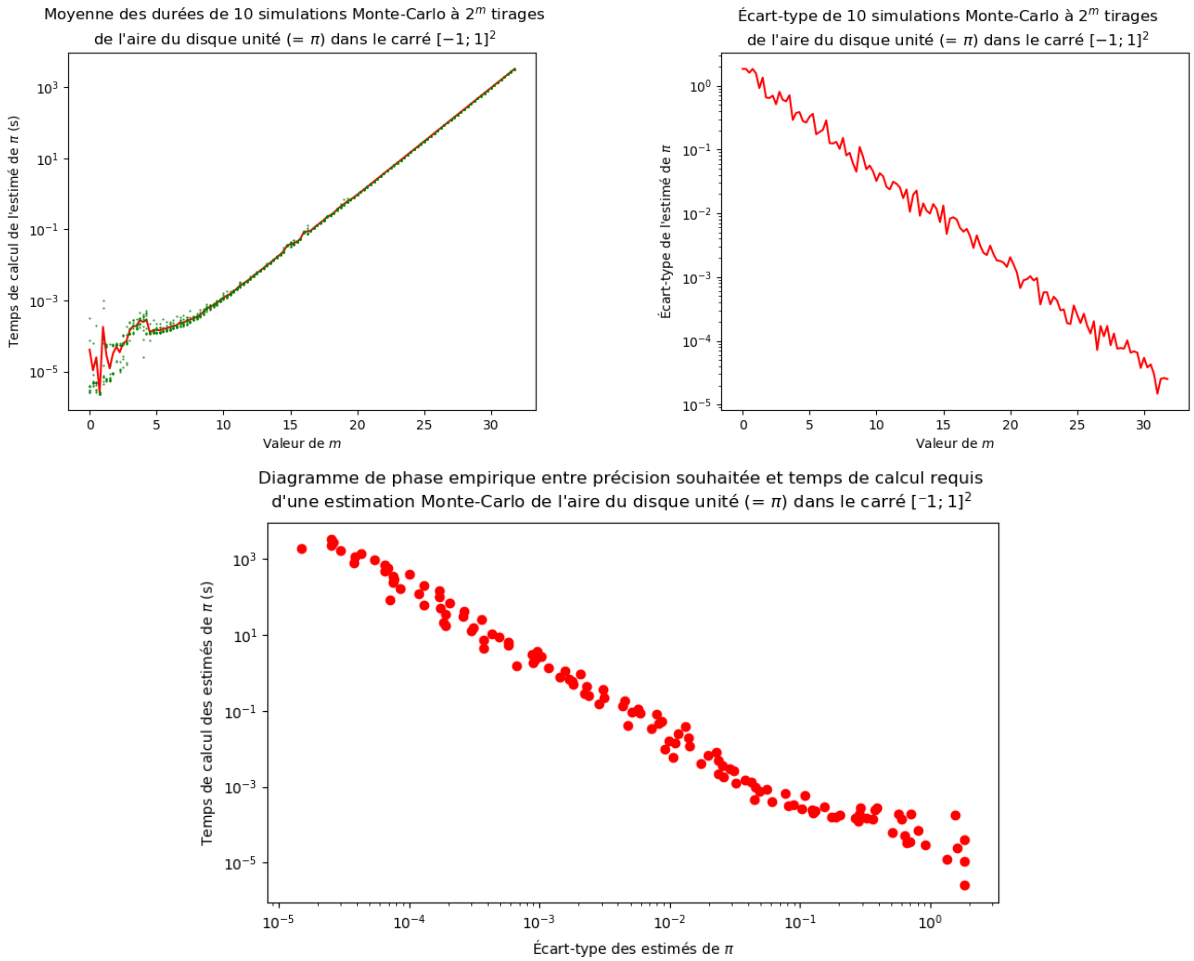

Figure 1.2 Relations entre nombre de tirages, temps de calcul et écart-type.

1.2.2 Fonctions substituts

Comme nous l'avons déjà évoqué, les problèmes de boîte noire sont fondamentalement difficiles sur plusieurs aspects. Notamment, l'important temps de calcul requis pour évaluer la fonction en un point, ou encore l'absence de quelconque information additionnelle (comme la dérivée) pour accélérer la recherche d'un optimum. Pour combler ces faiblesses, différents types de *substituts* (ou *surrogates* dans la littérature anglophone, concept formalisé par Booker *et al.* (1999)) peuvent être définis. Il s'agit de fonctions rapides à calculer et «approximant» la véritable fonction à optimiser. L'intérêt des substituts est de guider l'optimisation en exploitant l'information imparfaite mais rapidement accessible qu'ils fournissent. Par exemple :

1. Dans une simulation aérodynamique, les frottements fluides peuvent être négligés.
2. Les équations de dynamique des fluides de Navier-Stokes peuvent être remplacées par les approximations d'Euler.

3. Si la boîte noire estime une valeur via un algorithme de convergence, on peut augmenter le seuil du résidu en-dessous duquel il considérera avoir convergé.
4. Il est possible de construire un modèle de la fonction en exploitant les valeurs connues aux points où on l'a déjà évaluée.

On peut constater que les substituts peuvent être de nature différentes. Ici, les trois premiers sont *statiques* et le dernier *dynamique*, en le sens que les premiers sont des modifications fixées du problème (d'un modèle physique sous-jacent, d'un intermédiaire calculatoire, etc.) alors que le dernier anticipe le comportement de la fonction via toute l'information obtenue.

Pour ce projet, la notion de *substitut à précision réglable* sera vitale. On considérera ici ne pas avoir accès à la vraie fonction, mais uniquement à des approximations aussi précises qu'on le souhaite. De tels cas ne sont pas rares. Citons par exemple :

1. Comme évoqué ci-avant, un algorithme effectuant une convergence vers un point fixe. Chaque itération de cet algorithme réduit la distance entre son estimé du point fixe et la valeur réelle. Après une infinité d'itérations les deux valeurs coïncident, mais dans un cas réel on ne peut en exécuter qu'un nombre fini. En pratique, on se contente de fixer un seuil d'erreur maximale en-dessous duquel on considérera avoir convergé. Ainsi, tout seuil non-nul amène à une estimation de la vraie valeur, qui peut ne pas l'égaler mais dont la précision augmente lorsque le seuil diminue.
2. Une boîte noire renvoyant une valeur calculée par une simulation de Monte-Carlo. La vraie valeur de la fonction serait obtenue avec une infinité de tirages dans la simulation, ce qui est impensable en pratique. Cependant, on peut obtenir une estimation de cette valeur en utilisant un nombre fini de tirages, qu'on peut choisir aussi grand que désiré.

Cela nous amène à une situation très particulière : on ne peut pas pleinement se fier aux valeurs renvoyées par la fonction. Il faut constamment se souvenir qu'on ne manipule que des estimations, mais obtenues à une précision connue, variable et réglable. De plus, une grande précision (seuil de convergence au point fixe très bas, beaucoup de tirages Monte-Carlo, etc.) s'obtient toujours au prix d'un temps de calcul pouvant grandir démesurément. Ce travail vise à trouver un compromis efficace entre temps de calcul raisonnable et qualité du minimum estimé. Notamment, une stratégie proposant de considérer le problème comme déterministe en effectuant toutes les évaluations à une grande précision est une mauvaise idée, car n'a aucune robustesse et considérera comme optimal le premier estimé sous-évaluant la fonction.

1.2.3 Fonctions bruitées par bruit aléatoire

Dans le cadre de ce projet, les fonctions à optimiser peuvent ne pas être calculées exactement. Chaque évaluation est bruitée par une erreur aléatoire d'amplitude contrôlable. De fait,

la boîte noire n'est pas une véritable fonction mathématique : la calculer deux fois en un même point x donnera deux valeurs différentes. Ainsi, exécuter la boîte noire en un point x renvoie une valeur qu'on ne peut pas nommer $f(x)$: le faire impliquerait $x_1 = x = x_2$ mais $f(x) = f(x_1) \neq f(x_2) = f(x)$. De plus, à cause des erreurs aléatoires (non corrélées), les estimés en x et $x + dx$ n'ont aucune raison d'être proches. On ne peut donc pas invoquer d'argument de continuité lorsqu'on considère les résultats renvoyés par la boîte noire.

Par la suite, on supposera que les erreurs sont de loi $\mathcal{N}\left(0, \sigma^2\right)$: centrées d'écart-type σ. La figure 1.3 représente une telle situation, où deux évaluations au même point peuvent donner deux valeurs différentes. Pour une entrée donnée, la sortie est aléatoire autour d'une valeur moyenne, qu'on considérera comme la «vraie» valeur mais qu'on n'atteint pas à cause d'un bruit aléatoire venant à chaque évaluation. On peut observer, en comparant les courbes bleue et verte, que deux évaluations aux mêmes points donnent des valeurs différentes, et que les erreurs effectuées sont de même écart-type σ.

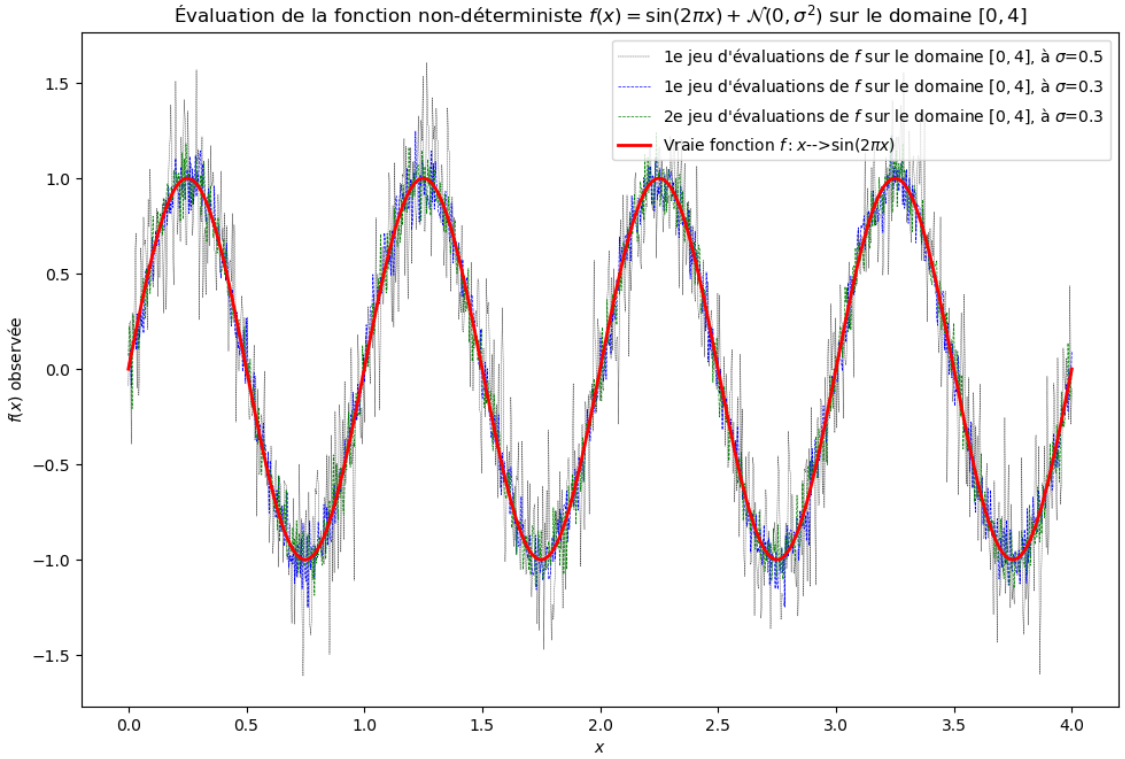

Figure 1.3 Diverses observations successives d'une même fonction non-déterministe.

On remarquera la possibilité de régler σ pour obtenir des garanties d'approximations plus fines : les courbes bleue et verte sont moins dispersées que la noire. On peut également observer la discontinuité d'un ensemble d'observations, alors même que la vraie fonction est continue.

Ainsi, le problème peut être formulé comme suit :

Notations liées à la fonction objectif

$$f(x) \quad : \quad \begin{cases} f : x \in \mathbb{R}^n \to f(x) \in \overline{\mathbb{R}}, \\ f(x) \text{ la valeur de la vraie fonction en } x \text{ (inaccessible en pratique).} \end{cases}$$

$$f_\sigma(x) \quad : \quad \begin{cases} \forall \sigma \in \mathbb{R}^{+*} : \ f_\sigma : x \in \mathbb{R}^n \to f(x) + \eta, \text{ avec } \eta \sim \mathcal{N}\left(0, \sigma^2\right) \text{ un bruit aléatoire,} \\ f_\sigma(x) \text{ une variable aléatoire de loi } \mathcal{N}\left(f(x), \sigma^2\right). \end{cases}$$

$$f_{\sigma(\omega)}(x) \quad : \quad \begin{cases} f_\sigma(x) : \omega \in \Omega \to (f(x) + \eta)(\omega) \in \mathbb{R}, \\ f_{\sigma(\omega)}(x) \text{ une observation } \omega \text{ de la variable aléatoire } f_\sigma(x). \end{cases}$$

Ces notations sont liées à la fonction objectif. La fonction que l'on souhaite minimiser est f. Il s'agit d'une fonction déterministe, mathématiquement définie selon les conventions de l'optimisation de boîte noire. Elle est cependant supposée inaccessible en pratique, et inconnue.

La seule information qu'on puisse avoir à son sujet est une approximation à précision variable. Par exemple, $f(x)$ peut être une quantité $\mathcal{A}(x)$ qu'on ne peut qu'estimer par une simulation de Monte-Carlo, effectuée avec un nombre de tirages déterminé. En vertu de l'équation (1.1), on peut connaître l'écart-type σ de l'estimé renvoyé par cette simulation. On désigne donc par $f_\sigma(x)$ la «fonction» lançant une telle simulation. Cet objet est une variable aléatoire.

Lorsqu'on a réellement effectué une simulation (et donc obtenu un estimé), on désigne la valeur obtenue par $f_{\sigma(\omega)}(x)$. Cette notation est proposée comme une analogie de la théorie des probabilités, dans laquelle on note X une variable aléatoire, et $X(\omega)$ une réalisation de X qu'on a observée. Il s'agit d'un réel.

Notation liée à la précision variable

$$\rho(r) \quad : \quad \begin{cases} \rho : r \in \mathbb{R} \to \rho(r) = \sigma \in \mathbb{R}^{+^*} \text{ bornée, positive et strictement décroissante sur } \mathbb{R}, \\ \rho \text{ traduit une précision } r \in \mathbb{R} \text{ arbitraire en un écart-type } \sigma \text{ des erreurs.} \end{cases}$$

Cette fonction effectue le lien entre un indice arbitraire de précision et un écart-type réel imposé à la boîte noire. Dans nos méthodes, on manipulera un indice $r \in \mathbb{Z}$ de précision arbitraire. Lorsque r augmente, on impose une précision plus grande à la boîte noire (au sens où on force l'écart-type de l'estimé qu'elle renvoie à être plus faible). Le lien explicite entre l'indice arbitraire r et l'écart-type réel imposé σ est donc la fonction $\rho : \rho(r) = \sigma$.

Notations liées à la mémoire algorithmique des observations passées

Tous les algorithmes proposés par la suite auront un fonctionnement dit *itératif*, au sens où ils répéteront une suite donnée d'instructions (formant une itération) tant qu'un certain critère d'arrêt ne sera pas atteint. On désignera par k l'indice d'itération et définira alors :

$$\mathcal{V}^k(x) = \Big\{(f_{\sigma(\omega)}(x), \sigma) \mid f_{\sigma(\omega)}(x) \text{ a été observée lors des } k \text{ premières itérations}\Big\},$$
$\mathcal{V}^k(x)$ est formé de l'ensemble des couples connus à l'issue de l'itération k au point x, de la forme (valeur observée de $f(x)$, écart-type de l'observation).

$$\mathcal{V}^k = \underset{x\in\mathbb{R}^n}{\cup}\Big\{\Big(x, \mathcal{V}^k(x)\Big)\Big\} = \underset{x|\mathcal{V}^k(x)\neq\emptyset}{\cup}\Big\{\Big(x, \mathcal{V}^k(x)\Big)\Big\} \text{ la cache à l'issue de l'itération } k.$$

Ces deux ensembles contiennent l'historique de l'intégralité des interactions avec la boîte noire bruitée au cours des k premières itérations d'une optimisation.

Le premier, $\mathcal{V}^k(x)$, contient l'ensemble des estimés obtenus de $f(x)$ lors des k premières itérations (donc des objets de la forme $f_{\sigma(\omega)}(x)$) ainsi que, pour chacun, l'écart-type σ qui avait été imposé à la boîte noire lorsqu'on en a lancé l'exécution pour observer $f_\sigma(x)$. On retrouve donc dans $\mathcal{V}^k(x)$ l'ensemble des couples $\Big(f_{\sigma(\omega)}(x), \sigma\Big)$ ayant été obtenus.

Si on désigne par $\omega_1, \ldots, \omega_m$ les m estimations de f en x, chacune ayant été faite à un écart-type imposé à $\sigma_1, \ldots, \sigma_m$, alors on aura $\mathcal{V}^k(x) = \Big\{\Big(f_{\sigma_1(\omega_1)}(x), \sigma_1\Big), \ \ldots, \ \Big(f_{\sigma_m(\omega_m)}(x), \sigma_m\Big)\Big\}$.

De là, l'ensemble $\mathcal{V}^k$ contient l'ensemble de tous les appels à la boîte noire bruitée, en tous les points x ayant été utilisés pour au moins un appel au cours de l'optimisation.

Par abus de notation, on s'autorisera également à voir $\mathcal{V}(x)$ comme une fonction qui renvoie toutes les évaluations de $f(x)$ contenues dans la cache $\mathcal{V}$: $\mathcal{V}(x) = \begin{cases} \emptyset & \text{si} \quad (x, .) \notin \mathcal{V} \\ \mathcal{E} & \text{si} \quad (x, \mathcal{E}) \in \mathcal{V} \end{cases}$.

Notations liées aux estimés des valeurs de la fonction

$$f^k(x) \ : \ \begin{cases} f^k : x \in \mathbb{R}^n \to \dfrac{\sum_{(\lambda,\sigma)\in\mathcal{V}^k(x)} \lambda/\sigma^2}{\sum_{(\lambda,\sigma)\in\mathcal{V}^k(x)} 1/\sigma^2} \in \mathbb{R}, \\ f^k(x) \text{ le meilleur estimé [2] de } f(x) \text{ possible après } k \text{ itérations.} \end{cases}$$

$$\sigma^k(x) \ : \ \begin{cases} \sigma^k : x \in \mathbb{R}^n \to \sqrt{\dfrac{1}{\sum_{(\lambda,\sigma)\in\mathcal{V}^k(x)} 1/\sigma^2}} \in \mathbb{R}^{+^*}, \\ \sigma^k(x) \text{ l'écart-type de l'estimé } f^k(x). \end{cases}$$

2. Il s'agit de l'expression analytique du maximum de vraisemblance. Voir l'annexe A.

Ces deux notations représentent les «meilleures» valeurs qu'un algorithme puisse proposer pour estimer $f(x)$ après k itérations. $f^k(x)$ est calculé par maximum de vraisemblance (cette méthode est détaillée dans l'annexe A) à partir de l'ensemble des observations en x de la boîte noire bruitée. Dit autrement, il est construit à partir de l'ensemble des éléments contenus dans $\mathcal{V}^k(x)$. On note également $\sigma^k(x)$ l'écart-type associé à l'estimé $f^k(x)$.

Pour les points auxquels on n'a effectué aucune observation, il est impossible de proposer une valeur puisque l'ensemble $\mathcal{V}^k(x)$ est vide. Dans ce cas, on prendra par convention $f^k(x) = +\infty$ et $\sigma^k(x) = +\infty$ pour signifier qu'on n'a aucune connaissance du comportement de la fonction f au point x. En pratique, on ne stockera que ces deux valeurs dans la cache puisque l'annexe A montre qu'il n'est pas nécessaire de sauvegarder les observations ayant amené à ces estimés.

Formalisation du problème d'optimisation

Puisque f est inaccessible, chercher à la minimiser est impossible. Il n'est pas pertinent de s'attarder sur le problème déterministe habituel :

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

puisque même s'il s'agit de notre objectif réel, il nous est impossible de calculer exactement les valeurs renvoyées par f. Ceci n'est donc que ce vers quoi on souhaite tendre théoriquement, mais la façon de l'atteindre doit dépendre d'un objectif différent. Le problème ne doit faire intervenir que les estimés f^k. Cependant, résoudre

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

ne serait pas pertinent non plus. Un solveur pourrait volontairement garder grand le bruit aléatoire, puisqu'alors les erreurs resteraient grandes, et pourraient artificiellement générer de très basses valeurs de f^k. Le problème ci-dessus n'impose aucun contrôle sur la qualité des approximations : un mauvais estimé (sujet à une grande variance) dont la valeur numérique est très basse sera préféré à un bon estimé (variance plus faible) à valeur numérique plus grande.

On cherchera plutôt une optimisation de f^k s'assurant de l'exactitude de sa valeur lorsque $k \to \infty$. Cela se traduit par un problème de recherche du point x minimisant la «fonction estimée limite» obtenue lorsque la variance des estimés tend vers 0. On cherchera un point x_* pour lequel l'estimé $f^k(x_*)$ est le plus bas parmi tous, mais sous la garantie que ce n'est pas à

cause d'une trop grande imprécision. On veut que la solution x_* optimale au problème soit, de façon certaine et pas à cause de bruits aléatoires incontrôlés, un point minimisant f^k. Il faut que $\forall y \neq x_*, \mathbb{P}\left(f^k(y) < f^k(x_*)\right)$ soit négligeable [3]. On propose donc le problème suivant :

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \lim_{k\to\infty|\sigma^k(x)\searrow 0} f^k(x) \\ \text{s.c.} \quad & x \in \mathcal{D} \end{aligned}$$

Les optima de ce problème sont ceux du problème initial, car lorsque $\sigma^k(x) \to 0$, on a $f^k(x) \to f(x)$ de façon certaine (d'après l'annexe A) : $\forall x,\ \mathbb{P}\left(\lim_{k\to\infty|\sigma^k(x)\searrow 0} f^k(x) = f(x)\right) = 1$. Résoudre ce problème portant sur les estimés (accessibles) est donc équivalent à résoudre le problème initial sur la vraie fonction f (qui, elle, ne peut être calculée).

1.3 Objectifs de recherche et plan du mémoire

L'objectif premier de ce projet est de concevoir une méthode d'optimisation de fonctions «boîte noire» bruitées dont on peut contrôler l'amplitude du bruit à chaque évaluation. Comme présenté ci-avant, on supposera manipuler des fonctions non-dérivables, où chaque évaluation est parasitée par une erreur aléatoire. On supposera également avoir la possibilité de régler l'écart-type de ces erreurs, qu'on peut donc diminuer autant qu'on le souhaite en contrepartie d'un temps de calcul plus élevé.

Diverses méthodes existent déjà pour assurer une convergence vers un minimum local malgré le bruit inhérent à chaque appel à la boite noire, aussi notre proposition devra être compétitive face à ces algorithmes déjà existants dans la littérature. De plus, on souhaitera qu'elle soit exacte et accompagnée d'une analyse de convergence. Il serait également intéressant que notre algorithme soit économe sur les ressources mobilisées : nous essaierons de limiter le temps de calcul (ou l'intensité de calcul) requis pour assurer une convergence.

Dans la suite de ce document, nous présenterons en premier lieu une revue de littérature (au chapitre 2) contenant diverses techniques pour répondre au problème que nous venons de proposer. Ensuite, le chapitre 3 détaillera les algorithmes que nous proposons, avant de les tester numériquement au chapitre 4. Enfin, nous discuterons au chapitre 5 de quelques techniques pour faire entrer un problème quelconque dans notre cadre théorique, et des façons de le résoudre en pratique.

3. À l'exception des éventuels autres points de l'ensemble $\arg\min_{x\in\mathcal{D}} f^k$.

CHAPITRE 2 REVUE DE LITTÉRATURE

Cette revue de littérature présente de façon sommaire plusieurs techniques couramment utilisées en optimisation de boîte noire déterministe, et pouvant être adaptées au contexte bruité. Chaque algorithme sera présenté dans le contexte déterministe, puis sera suivi d'une discussion sur la façon de l'adapter à des fonctions stochastiques.

Nous introduirons d'abord une heuristique connue pour son efficacité pratique malgré un manque de garanties théorique : l'algorithme de Nelder-Mead, très utilisé sur des problèmes réels. Ensuite, nous présenterons un algorithme exact au fonctionnement reposant sur les surfaces de réponse : l'algorithme *EGO*. Enfin, nous discuterons des méthodes exactes de recherche directe. L'objectif de cette section sera de parvenir à présenter l'algorithme *MADS*.

Notons que les problèmes contraints ne sont pas considérés dans cette revue de littérature. Parmi les algorithmes présentés ici, seul *MADS* peut être adapté pour les traiter.

2.1 Heuristique de Nelder-Mead

Cette heuristique est un algorithme célèbre car souvent très efficace en pratique sur les problèmes non contraints. Bien que la version originale (Nelder et Mead (1965)) ne dispose d'aucune garantie de convergence vers un optimum, elle traite avec succès de nombreux problèmes. Les créateurs de cet algorithme le surnommaient *Méthode du simplexe mobile*, ce qui résume son fonctionnement. Le principe repose sur un simplexe, un «hyper-triangle» qui est à $\mathbb{R}^n$ ce qu'un triangle est à $\mathbb{R}^2$: un simplexe dans $\mathbb{R}^n$ est un ensemble de $n+1$ points dont l'enveloppe convexe est d'intérieur non vide. Ce simplexe va se transformer dans l'espace $\mathcal{X}$ des variables via des expansions ou contractions dans différentes directions.

2.1.1 Algorithme déterministe

À chaque itération, l'algorithme évalue chacun des points du simplexe $\mathbb{Y} = [y_0, \ldots, y_n]$, et les note y_0 à y_n par ordre croissant de leurs images par f. Ainsi, y_0 est le meilleur point trouvé par la méthode, et y_n le plus mauvais. L'algorithme va chercher à se débarrasser de y_n en considérant que s'il s'agit du plus mauvais point, cela signifie que la fonction décroît lorsqu'on s'approche des autres.

Une itération de l'algorithme consiste alors à marquer le centroïde des n premiers points (x_c) et à projeter y_n vers les directions jugées prometteuses, en prenant sa réflexion par rapport à x_c. On obtient alors le point réfléchi x_r, en lequel on évalue f pour obtenir la valeur f_r. Selon

la qualité de f_r, on peut conserver le point x_r ou proposer d'autres candidats pouvant être intéressants. Dans tous les cas, l'idée est de proposer un remplaçant à y_n, un nouveau point qui a une meilleure image par f. Cette propriété sera notée comme suit :

$$x \prec y \iff f(x) < f(y).$$

La figure 2.1 illustre l'ensemble des stratégies testées par l'algorithme.

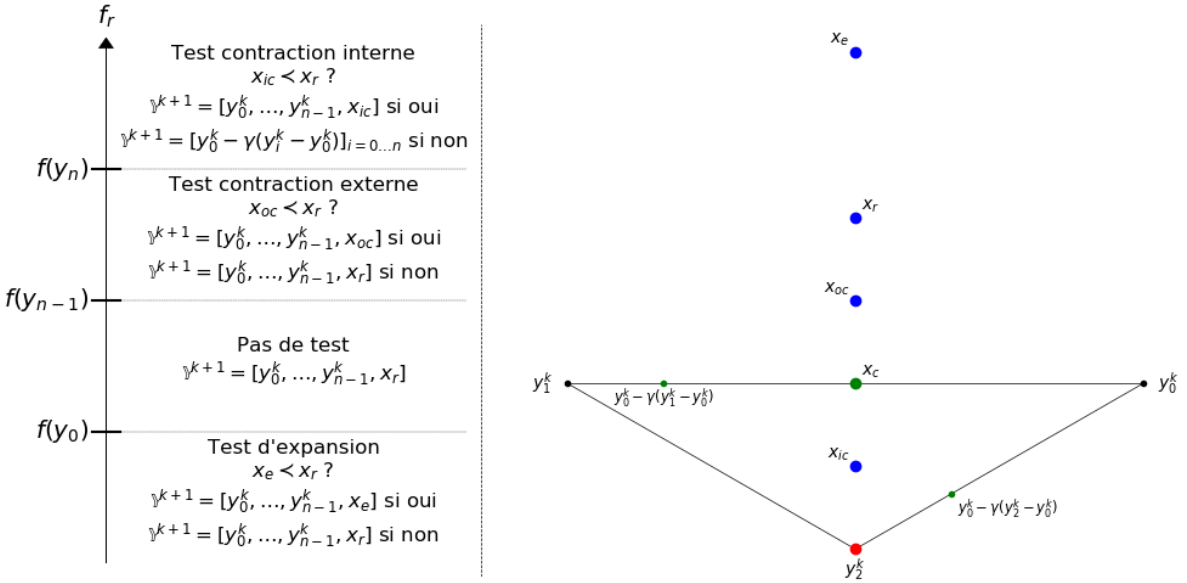

Figure 2.1 Illustration des points testés par Nelder-Mead sur un simplexe dans $\mathbb{R}^2$.

On peut y observer le simplexe $\left\{y_0^k, y_1^k, y_2^k\right\}$, avec $y_0^k \prec y_1^k$ et $y_1^k \prec y_2^k$. Le centroïde x_c est donc le point milieu du segment $[y_0^k, y_1^k]$. De là, on peut générer x_r, la réflexion de y_2^k par rapport à x_c. Ensuite, selon la qualité de $f_r = f(x_r)$, d'autres points peuvent être générés et évalués pour construire le simplexe de l'itération suivante : x_e une extension externe, x_{oc} une contraction externe, ou x_{ic} une contraction interne.

Le déroulé de l'algorithme est donc :

- Détection du point y_n et du centroïde x_c, pour créer x_r et évaluer f_r,
- Comparer f_r à $f(y_0)$, $f(y_{n-1})$ et $f(y_n)$ et en déduire le test adéquat en fonction de cette comparaison,
- Mettre à jour le simplexe et passer à l'itération suivante.

Ainsi, l'algorithme s'écrit comme suit :

Algorithme 1 Algorithme de Nelder et Mead (*NM*)

fonction NELDER_MEAD(f ; $y_0^0, \ldots, y_n^0$; $\delta_e = 2, \delta_{oc} = 0.5, \delta_{ic} = 0.5, \gamma = 0.67$)
  $k \leftarrow 0$
  tant que non(*Critère d'arrêt atteint*) **faire**
    $\mathbb{Y}^k \leftarrow [y_0^k, \ldots, y_n^k]$ rangés par images par f croissantes
    $f_b^k \leftarrow f(y_0^k)$
    $x_c^k, x_r^k, f_r^k \leftarrow Reflexion(f; \mathbb{Y}^k)$
    si $f_r^k < f_b^k$ **alors** $\mathbb{Y}^{k+1} \leftarrow Expansion(f, \delta_e; x_c^k, \mathbb{Y}^k, f_r^k, x_r^k)$
    si $f_b^k \leq f_r^k < f(y_{n-1}^k)$ **alors** $\mathbb{Y}^{k+1} \leftarrow [y_0^k, \ldots, y_{n-1}^k, x_r^k]$
    si $f(y_{n-1}^k) \leq f_r^k < f(y_n^k)$ **alors** $\mathbb{Y}^{k+1} \leftarrow ContractExt(f, \delta_{oc}; x_c^k, \mathbb{Y}^k, f_r^k, x_r^k)$
    si $f(y_n^k) \leq f_r^k$ **alors** $\mathbb{Y}^{k+1} \leftarrow ContractInt(f, \delta_{ic}, \gamma; x_c^k, \mathbb{Y}^k, f_r^k)$
    $k \leftarrow k + 1$
  renvoyer $\arg\min_{y \in \mathbb{Y}^k} f(y)$

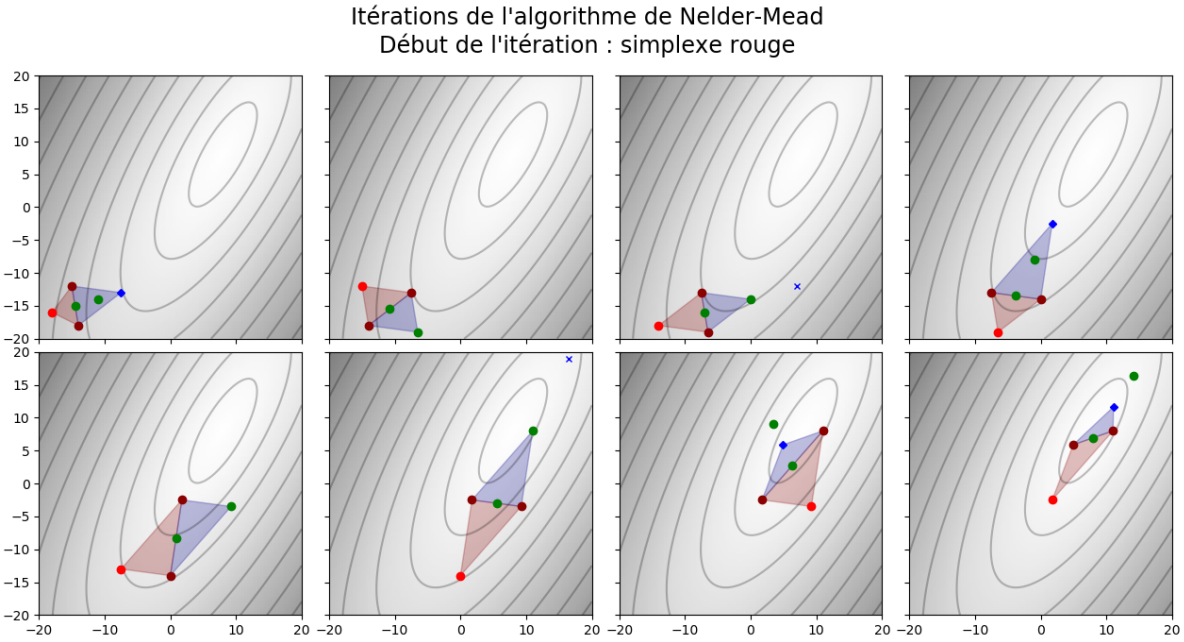

Figure 2.2 Évolution d'un simplexe de Nelder-Mead sur un problème dans $\mathbb{R}^2$.

Le détail des fonctions internes suit la figure 2.1, et commence par l'évaluation de f_r :

1: **fonction** REFLEXION(f ; $\mathbb{Y}$)
2: $x_c \leftarrow \frac{1}{n}\sum_{i=0}^{n-1} y_i$
3: $x_r \leftarrow x_c + (x_c - y_n)$
4: $f_r \leftarrow f(x_r)$
5: **renvoyer** x_c, x_r, f_r

Si x_r est meilleur que y_0 (c'est-à-dire $f_r < f(y_0)$), la réflexion a apporté un gain important, donc l'algorithme va se projeter encore plus loin pour explorer cette direction. On propose un facteur d'expansion δ_e puis évalue f en x_e, l'extension de x_r dans la direction $x_r - y_n$. On teste ensuite la qualité de x_e et le retient s'il bat x_r.

1: **fonction** EXPANSION(f, δ_e ; $x_c, \mathbb{Y}, f_r, x_r$)
2: $x_e \leftarrow x_c + \delta_e(x_c - y_n)$
3: $f_e \leftarrow f(x_e)$
4: **si** $f_e < f_r$ **alors renvoyer** $[y_0, \ldots, y_{n-1}, x_e]$ **sinon renvoyer** $[y_0, \ldots, y_{n-1}, x_r]$

Sinon, plusieurs cas sont possibles. Si x_r est moins bon que y_0 mais meilleur que le 2^e pire point du simplexe (y_{n-1}), l'algorithme crée un nouveau simplexe en n'éliminant que y_n.

Si toutes ces conditions ne sont pas vérifiées (on est donc dans le cas où $f(y_{n-1}) \leq f_r$), on sait que la réflexion de y_n a probablement créé x_r trop loin des zones prometteuses. L'algorithme va donc rétracter x_r vers x_c. Là encore, une disjonction de cas est utile.

Si $f_r < f(y_n)$, il est tout de même pertinent de rester plus proche de x_r que de y_n. On tente donc une contraction externe d'un facteur δ_{oc} et retient x_{oc} s'il bat x_r.

1: **fonction** CONTRACTEXT(f, δ_{oc} ; $x_c, \mathbb{Y}, f_r, x_r$)
2: $x_{oc} \leftarrow x_c + \delta_{oc}(x_c - y_n)$
3: $f_{oc} \leftarrow f(x_{oc})$
4: **si** $f_{oc} < f_r$ **alors renvoyer** $[y_0, \ldots, y_{n-1}, x_{oc}]$ **sinon renvoyer** $[y_0, \ldots, y_{n-1}, x_r]$

Sinon, la réflexion a été un échec. On évalue donc si f décroit lorsqu'on s'approche de x_c depuis y_n (via une étape de contraction interne d'un facteur $-\delta_{ic}$). Si cette dernière tentative échoue, alors tout le simplexe va se contracter vers son meilleur point y_0 d'un facteur γ : chaque point y_k va voir sa distance à y_0 être multipliée par γ.

1: **fonction** CONTRACTINT(f, δ_{ic}, γ ; $x_c, \mathbb{Y}, f_r$)
2: $x_{ic} \leftarrow x_c - \delta_{ic}(x_c - y_n)$
3: $f_{ic} \leftarrow f(x_{ic})$
4: **si** $f_{ic} < f_r$ **alors renvoyer** $[y_0, \ldots, y_{n-1}, x_{ic}]$ **sinon renvoyer** $[y_0 - \gamma(y_0 - y_i),\ i \leq n]$

L'algorithme de Nelder-Mead est une heuristique simple à comprendre et à implémenter, et souvent efficace en pratique. Mais il n'a aucune garantie de convergence vers un optimum. Un contre-exemple célèbre est donné par McKinnon (1998), où la fonction mettant NM en échec est pourtant strictement convexe et très lisse (elle est $\mathcal{C}^2(\mathbb{R}^2, \mathbb{R})$). Il existe cependant des modifications dont on peut prouver l'exactitude, comme celle proposée par Tseng (1999) à optimalité garantie sur des fonctions $\mathcal{C}^1$ bornées inférieurement, ou celle de Audet et Tribes (2017) héritant des propriétés de l'algorithme $MADS$ présenté ci-après.

2.1.2 Variante non déterministe

L'algorithme présenté précédemment, déterministe, n'est certainement pas approprié pour le contexte de notre projet. Cependant, il est possible de l'adapter pour lui intégrer la composante stochastique des fonctions que l'on souhaite minimiser. Une variante possible est celle proposée par Chang (2012). Ici, la précision de chaque évaluation de la fonction n'est pas réglable. On considère seulement que la «vraie» valeur de f est la moyenne de la loi observée à chaque évaluation (ou, dit autrement, que le bruit stochastique est de moyenne nulle). Le principe repose sur la loi des grands nombres. L'algorithme propose, à chaque itération k et pour chaque point x, d'approximer $f(x)$ par la moyenne de N^k observations. Dans nos notations, en notant η la loi des erreurs, le calcul des estimés est donc : $\forall x,\ f^k(x) = \sum_{i=1}^{N^k} f_{\eta(\omega_i)}(x)/N^k$. Pour assurer la convergence, l'algorithme va faire tendre $N^k \underset{k\to+\infty}{\longrightarrow} +\infty$ pour chaque point du simplexe. Ainsi, la loi des grands nombres assure que les valeurs limites des estimés égalent celles de la vraie fonction.

L'algorithme 2 encode la mécanique proposée par Chang.

On peut noter que cet algorithme pourrait être adapté pour notre contexte de précision variable, puisque le nombre N d'évaluations de $f(x)$ choisi pour limiter l'amplitude de l'aléatoire pourrait être remplacé par une unique observation à une précision donnée. Au lieu d'un nombre N imposé d'observations (ce qui, indirectement, donne une borne supérieure de la variance de l'erreur faite en moyennant chaque observation), on pourrait imposer un écart-type maximal et lancer une nouvelle observation assez précise pour assurer que l'estimé obtenu est en-dessous de ce seuil.

Algorithme 2 Algorithme de Nelder et Mead (*NM*) stochastique de Chang

fonction NELDER_MEAD_CHANG(f ; $y_0^0, \ldots, y_n^0$; $\delta_e, \delta_{oc}, \delta_{ic}, \gamma$)
  $k \leftarrow 0$
  $\forall x \in \mathcal{X},\ \mathcal{V}(x) \leftarrow \emptyset$
  tant que non(*Critère d'arrêt atteint*) **faire**
    $N^k \leftarrow \left\lfloor \sqrt{k+1} \right\rfloor$
    pour $i \in [\![0, n]\!]$ **faire** $f^k(y_i^k) \leftarrow Evaluation(f, y_i^k, N^k)$
    $\mathbb{Y}^k \leftarrow [y_0^k, \ldots, y_n^k]$ rangés par images par f^k croissantes
    $f_b^k \leftarrow f^k(y_0^k)$
    $x_c^k, x_r^k, f_r^k \leftarrow Reflexion(f; \mathbb{Y}^k; N^k)$
    si $f_r^k < f_b^k$ **alors** $\mathbb{Y}^{k+1} \leftarrow Expansion(f, \delta_e; x_c^k, \mathbb{Y}^k, f_r^k, x_r^k; N^k)$
    si $f_b^k \leq f_r^k < f^k(y_{n-1}^k)$ **alors** $\mathbb{Y}^{k+1} \leftarrow [y_0^k, \ldots, y_{n-1}^k, x_r^k]$
    si $f^k(y_{n-1}^k) \leq f_r^k < f^k(y_n^k)$ **alors** $\mathbb{Y}^{k+1} \leftarrow ContractExt(f, \delta_{oc}; x_c^k, \mathbb{Y}^k, f_r^k, x_r^k; N^k)$
    si $f^k(y_n^k) \leq f_r^k$ **alors** $\mathbb{Y}^{k+1} \leftarrow ContractInt(f, \delta_{ic}, \gamma; x_c^k, \mathbb{Y}^k, f_r^k; N^k)$
    $k \leftarrow k + 1$
  renvoyer $\underset{y \in \mathbb{Y}^k}{\arg\min} f(y)$

La différence algorithmique avec l'original déterministe repose dans l'affectation des valeurs de la fonction. Au lieu d'une évaluation déterministe suivie d'une affectation (par exemple, $f_r \leftarrow f(x_r)$), on fera ici appel à la fonction *Evaluation* suivante :

fonction EVALUATION(f, x, N)
  tant que $Card(\mathcal{V}(x)) < N$ **faire**
    $\mathcal{V}(x) \leftarrow \mathcal{V}(x) \cup \left\{ f_{\eta(\omega)}(x) \right\}$ ▷ Ajout d'une nouvelle observation ω de $f_\eta(x)$
  renvoyer $\dfrac{1}{N} \displaystyle\sum_{i=1}^{Card(\mathcal{V}(x))} f_{\eta(\omega_i)}(x)$

Ainsi, dans la variante de Chang, toute affectation directe doit être remplacée par un appel à *Evaluation* avec le paramètre N^k. Par exemple, la fonction *Reflexion* deviendrait ici :

fonction REFLEXION($f, x; N$)
  $x_c \leftarrow \dfrac{1}{n} \displaystyle\sum_{i=0}^{n-1} y_i$
  $x_r \leftarrow x_c + (x_c - y_n)$
  $f_r \leftarrow Evaluation(f, x_r, N)$
  renvoyer x_c, x_r, f_r

2.2 Surface de réponse

Les méthodes présentées ici exploitent les évaluations de la fonction objectif pour estimer son comportement global. Le principe sous-jacent est d'enrichir la *surface de réponse* au fil de l'optimisation, pour qu'à terme elle représente efficacement le comportement réel de la fonction. Ainsi, un point minimisant la surface de réponse minimisera aussi la fonction. On désigne par *surface de réponse* une fonction g définie de $\mathbb{R}^n$ dans $\mathbb{R}$, qui coïncide avec l'objectif réel f aux points où on l'a déjà évalué. La façon de construire g est laissée libre à la méthode, mais dans tous les cas elle sera actualisée au fil de l'optimisation. Les méthodes présentées ici, ainsi que d'autres suivant le même principe, sont notamment présentées dans la thèse de Sasena (2002).

2.2.1 Krigeage

La technique de *krigeage* (dit *kriging* en littérature anglophone) tire son nom du géostatisticien D.G. Krige, qui l'a développée en 1951 pour estimer l'emplacement de gisements d'or sur un site d'excavation. Originellement très appliquée, elle a vite et naturellement été étendue à beaucoup d'autres contextes. Il s'agit maintenant d'une stratégie très commune en optimisation. On pourra se référer à la revue de littérature de Kleijnen (2009).

L'idée sous-jacente repose sur une hypothèse implicite, stipulant que les fonctions «réelles» qu'on optimise sont souvent assez régulières. On ne va pas jusqu'à imposer la dérivabilité, mais on supposera que lorsqu'on considère deux points assez proches l'un de l'autre, les valeurs de leurs objectifs seront proches également. Dit autrement, cette technique exploite une hypothèse implicite de continuité de Lipschitz locale.

Ceci étant supposé, on modélise alors la fonction comme une exécution donnée d'un processus stochastique à déterminer. Connaissant les valeurs prises en certains points (ceux auxquels f a déjà été estimée), on peut alors prédire le comportement du dit processus en tout point de l'espace. La technique de krigeage va proposer un processus stochastique particulier, et mettre en lumière la façon d'en déduire son comportement le plus probable dans l'ensemble de l'espace (notamment, les points où il est probable qu'il prenne une valeur basse). En calculant ensuite f en certains points, on peut mettre à jour le comportement du processus stochastique ; et à terme le faire coïncider avec f sur l'ensemble de l'espace des variables.

La méthode suppose avoir un ensemble $\mathcal{D} = \left\{(x^i, f(x^i) \mid i \in [\![1, p]\!]\right\}$ de points en lesquels on connaît la valeur de f. Elle propose alors un processus stochastique $\mathcal{S}$ tel que $\forall x,\ \mathcal{S}(x) = \mu + \epsilon(x)$, où $\epsilon(x) \sim \mathcal{N}\left(0, \sigma^2\right)$ et cherche à reconstruire une exécution depuis le jeu d'observations partiel $\forall i \in [\![1, p]\!],\ \mathcal{S}(x^i) = f(x^i)$. De plus, l'hypothèse de continuité de Lipschitz mentionnée

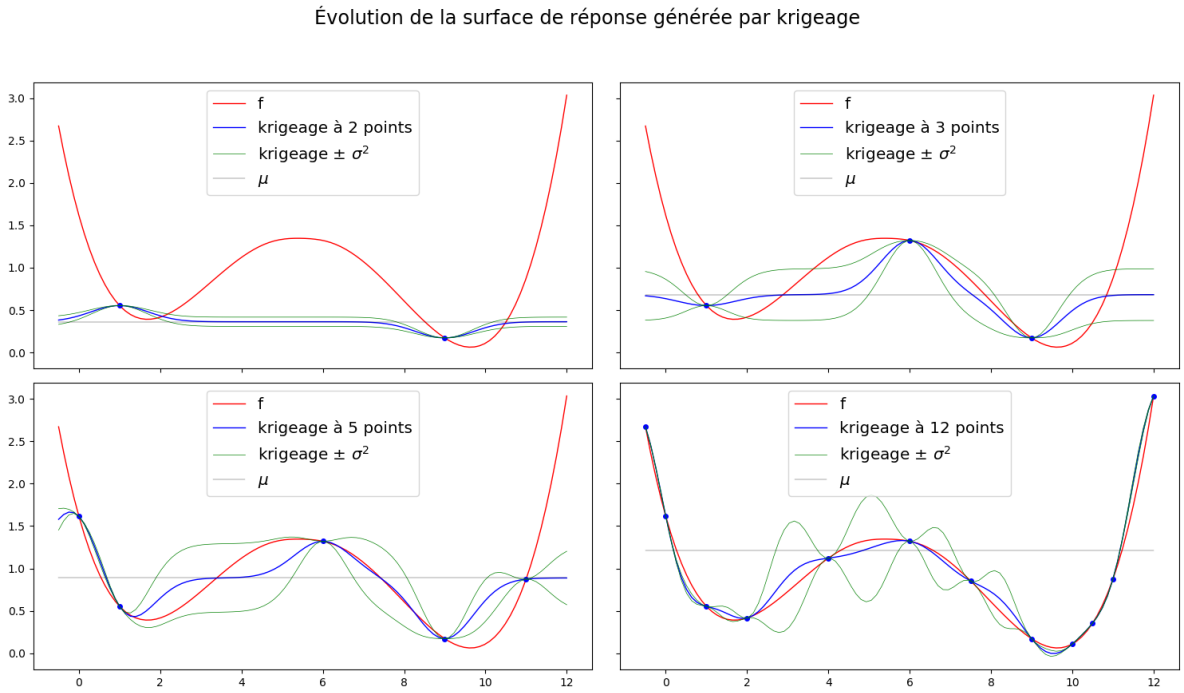

Figure 2.3 Illustration de surfaces de réponse dans $\mathbb{R}$ par krigeage.

ci-avant se traduit ici par une forte corrélation entre les ϵ pour des points très proches. La corrélation est exprimée comme suit : $\mathcal{C}orr(\epsilon(x), \epsilon(y)) = \exp\left\{-\sum_{i=1}^{n} 10^{\theta_i}|x_i - y_i|^{p_i}\right\}$.

Pour manipuler le processus, il faut donc estimer μ, σ^2, les θ_i et les p_i. Une fois ceci fait (par maximum de vraisemblance, en général), on a accès en tout point à la valeur la plus vraisemblable prise par le processus, ainsi que la variance associée. En d'autres mots, on a en tout point la valeur la plus vraisemblable de la fonction f, et une distribution de probabilité autour de cette valeur. La figure 2.3 résume l'intérêt de la méthode.

Il est également possible de renforcer la méthode, par exemple en supposant que la moyenne μ n'est plus constante mais fonction de x : $\mu(x)$ en tout point $x \in \mathbb{R}^n$. De telles modifications améliorent grandement la puissance de la méthode, notamment car elles saisissent mieux l'amplitude de l'inconnu loin des points d'évaluation. On peut se référer au premier cadre de la figure 2.3 pour observer que dans la méthode initiale, lorsqu'on se trouve loin de toute observation (en $x = 6$ par exemple), le processus se stabilise à sa moyenne constante, et considère qu'il est très peu probable que la vraie fonction s'en éloigne. Cet important défaut est corrigé par la moyenne mobile. En effet, si la moyenne dépend de x, il est impossible de prédire efficacement sa valeur dans ces zones, ainsi la méthode considérera avoir une source d'incertitude supplémentaire venant de cette imprécision.

2.2.2 Algorithme *EGO*

Cet algorithme, proposé initialement par Jones *et al.* (1998) pour les cas non contraints, exploite le principe du krigeage pour effectuer une *Efficient Global Optimisation* (d'où son nom), à savoir une optimisation qui va à chaque itération évaluer le point le plus prometteur que la méthode parvient à localiser, sans se restreindre à un point voisin de l'optimum courant.

Pour cela, à chaque itération, l'approximation de f obtenue par krigeage est observée. En chaque point x, elle donne $\widetilde{f}(x)$ la valeur estimée de $f(x)$ ainsi que la variance $\widetilde{\sigma}(x)$ (l'incertitude) de cette estimation, et x_* minimisant $\widetilde{f}$. Cela permet de calculer une quantité nommée *expected improvement*, ou *amélioration moyenne*, en chaque point. Cette quantité se calcule comme suit : $EI(x) = \mathbb{E}\left[\max(f(x_*) - \widetilde{f}(x), 0)\right]$.[1] Le point la maximisant est le plus intéressant, puisqu'il s'agit du point en lequel le gain espéré est le plus grand.

Cette méthode offre un compromis naturel entre exploration dans les zones incertaines et intensification dans les zones prometteuses. Dans les zones incertaines $EI(x)$ est grand, car l'estimé est très imprécis (donc potentiellement beaucoup plus grand que la vraie valeur de f). Proche de l'optimum courant, l'estimé a une variance très faible mais une excellente valeur. Ainsi, la méthode va soit choisir un point nouveau car dans une zone inconnue, soit un voisin de l'optimum actuel car les modèles affirment qu'il améliore la solution courante.

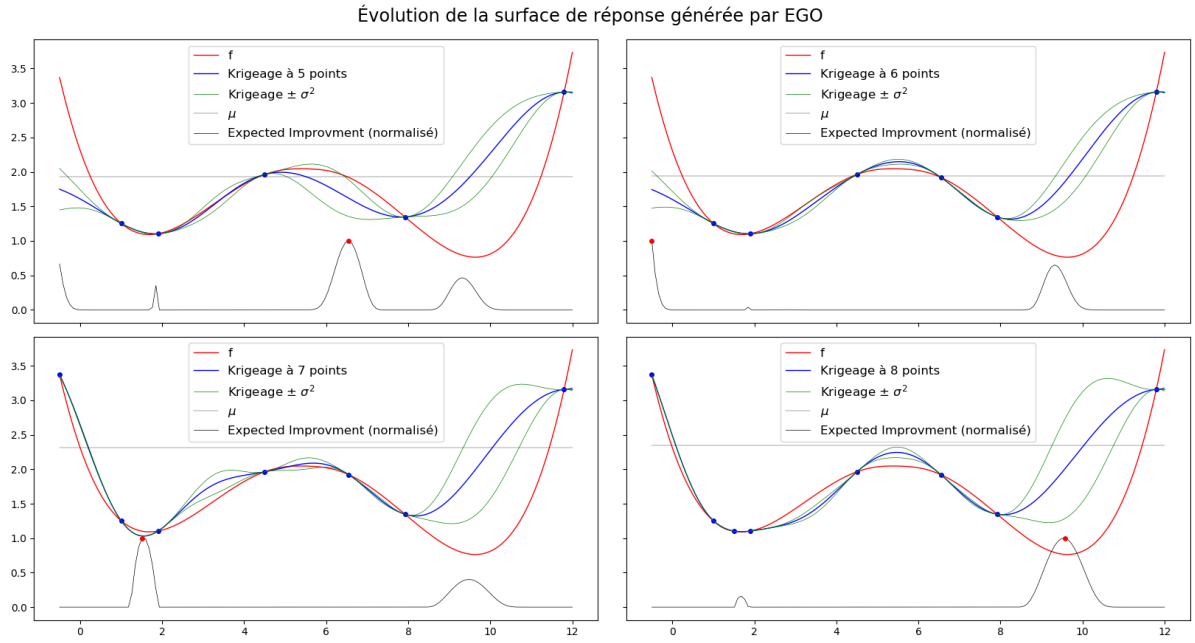

Figure 2.4 Application d'*EGO* à une surface de réponse dans $\mathbb{R}$.

1. Son expression analytique est $EI(x) = \left(f(x_*) - \widetilde{f}(x)\right)\Phi\left(\frac{f(x_*) - \widetilde{f}(x)}{\widetilde{\sigma}(x)}\right) + \widetilde{\sigma}(x)\phi\left(\frac{f(x_*) - \widetilde{f}(x)}{\widetilde{\sigma}(x)}\right)$.

2.2.3 Intégration d'un bruit stochastique réglable

Il existe des modifications de l'algorithme *EGO* pouvant traiter les fonctions bruitées par des erreurs stochastiques à précision réglable. Par exemple, la variante présentée ici, issue d'un document de travail de Picheny *et al.* (2012).

Tout d'abord, il faut adapter le krigeage pour qu'il tienne compte de l'incertitude entourant chaque évaluation. Il est possible de modifier les formules pour obtenir un résultat satisfaisant, comme celui présenté en figure 2.5. On peut observer, sur la figure, l'amplitude de l'écart-type imposé aux évaluations de f, et les estimés obtenus. De là, le krigeage trace la courbe (bleue) du maximum de plausibilité de f, et celle de la variance de l'estimation (en vert).

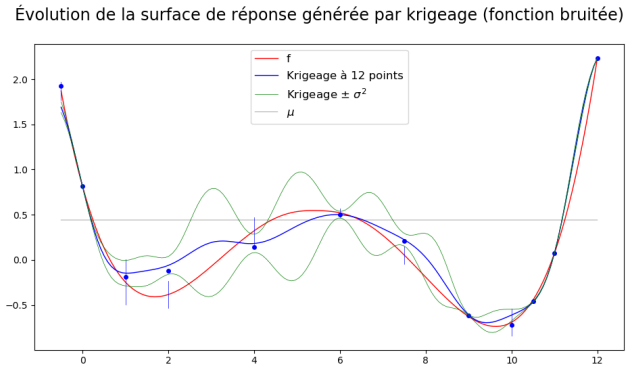

Figure 2.5 Krigeage d'une fonction bruitée.

Il est alors possible d'utiliser cette nouvelle mécanique de krigeage pour modifier l'algorithme *EGO*. Réutiliser son critère de sélection serait risqué, puisque le bruit peut maintenant fausser la valeur de $f(x_*)$ et le point x_* courant. Pour choisir un nouveau point en lequel évaluer f, *EGO* cherchait à maximiser l'*expected improvement*. Ici, il faut modifier cette quantité pour qu'elle tienne compte de deux choses : la valeur de la surface de réponse aux points déjà évalués est bruitée, et celle au point qu'on s'apprête à choisir est également stochastique.

On désignera toujours par $\widetilde{f}^k$ la fonction du maximum de plausibilité de f à l'issue de l'ensemble des observations faites avant l'itération k. De même, $\widetilde{\sigma}^k$ sera la variance ce cet estimé. On a cette fois des estimations $y_1, \ldots, y_k$ de f en chacun des points $x_1, \ldots, x_k$ faites à des écarts-types $\sigma_1, \ldots, \sigma_k$. Ainsi, $\widetilde{f}^k$ et $\widetilde{\sigma}^k$ sont connues conditionnellement aux observations passées $\widetilde{A}_k = \{(x_1, y_1, \sigma_1), \ldots, (x_k, y_k, \sigma_k)\}$. La modification proposée par l'article de Picheny *et al.* (2012) consiste à calculer le β-quantile en chaque point. En notant $\widetilde{A}_k$ l'ensemble des observations passées, on définit le β-quantile en x comme suit, $\forall \beta \in [0,1]$ et $\forall x \in \mathcal{X}$:

$$q_k(x) = \inf\left\{u \in \mathbb{R} \mid \mathbb{P}\left(\widetilde{f}^k(x) \leq u \mid \widetilde{A}_k\right) \geq \beta\right\} = \widetilde{f}^k(x) + \Phi^{-1}(\beta)\widetilde{\sigma}^k(x).$$

Cette quantité représente la plus petite valeur y telle qu'il y ait une probabilité supérieure ou égale à β que la fonction en x estimée par le krigeage soit inférieure à y. On peut donc trouver x_*^k le point le plus intéressant trouvé à l'issue de l'itération k : $x_*^k \in \underset{x \in \mathcal{X}}{\arg\min}\ q_k(x)$.

Enfin, puisqu'on peut à chaque évaluation choisir la précision à laquelle faire l'estimation, on désignera par $Q^{k+1}(x,\sigma)$ le quantile de krigeage (aléatoire) au point x, conditionnellement à $\widetilde{A}_k$ et avec une évaluation de $f(x)$ faite à un écart-type de σ. La source d'aléatoire vient du fait que $y = f(x) + \epsilon,\ \epsilon \sim \mathcal{N}\left(0, \sigma^2\right)$ n'a pas encore été observé.

Ainsi, le nouveau critère est la maximisation de l'*expected quantile improvement* sous une précision proposée σ pour la prochaine évaluation : $x_*^{k+1} \in \underset{x\in\mathcal{X}}{\arg\max}\ EQI^k(x,\sigma)$, avec :

$$\forall x \in \mathcal{X},\ \forall \sigma > 0,\ EQI^k(x,\sigma) = \mathbb{E}\left[\max\left(q_k(x_*^k) - Q_{k+1}(x,\sigma), 0\right) \mid \widetilde{A}_k\right]$$

dont on peut donner une expression analytique :

$$\begin{aligned} EQI^k(x,\sigma) &= I^k(x)\Phi\left(\frac{I^k(x)}{s^k(x)}\right) + s^k(x)\phi\left(\frac{I^k(x)}{s^k(x)}\right) \\ \text{avec } I^k(x) &= \widetilde{f}^k(x_*^k) - \left(\widetilde{f}^k(x) + \Phi^{-1}(\beta)\sqrt{\frac{\sigma^2 \times \widetilde{\sigma}^k(x)^2}{\sigma^2 + \widetilde{\sigma}^k(x)^2}}\right) \\ \text{et } s^k(x) &= \sqrt{\frac{\widetilde{\sigma}^k(x)^2}{\sigma^2 + \widetilde{\sigma}^k(x)^2}}. \end{aligned}$$

Enfin, une technique avancée de choix de la valeur de σ à utiliser à chaque itération est discutée dans l'article pour limiter au maximum les efforts de calcul. β est lui déterminé pour l'ensemble de l'optimisation, et sa valeur influe sur le comportement général de la méthode : on aura un comportement plus exploratoire s'il est proche de 0, tandis que β proche de 0.5 incitera la méthode à intensifier autour de l'optimum estimé.

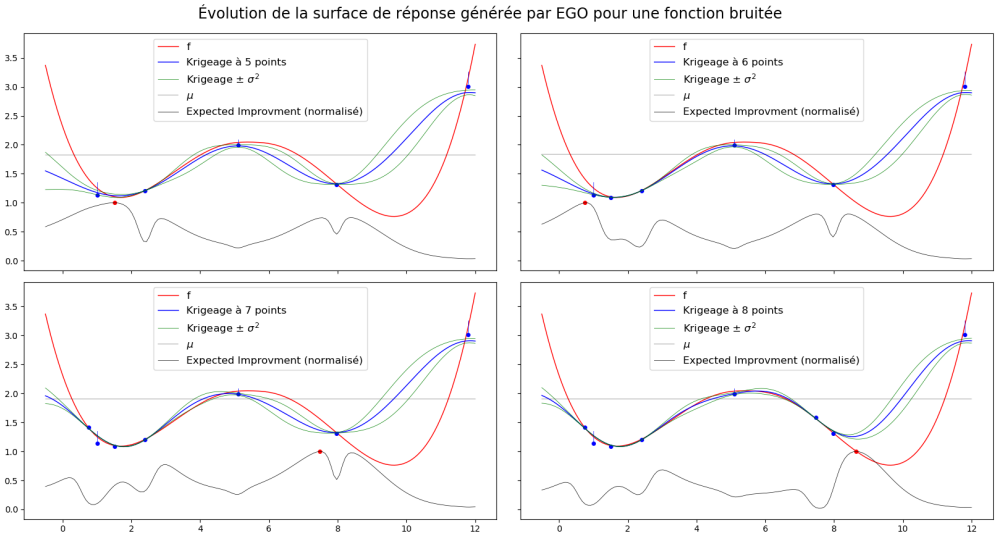

Figure 2.6 Application d'*EGO* à une surface de réponse bruitée dans $\mathbb{R}$.

2.3 Méthodes de recherche directe

L'optimisation de boîte noire par recherche directe est une famille de méthodes exploitant le principe d'amélioration par voisinage. Le concept repose sur la notion de *solution courante*, le point le plus intéressant connu à un instant donné. Autour de celui-ci, une itération de l'algorithme va évaluer un certain nombre de points *candidats*. Ces candidats appartiennent à un voisinage de la solution courante. Si l'un des candidats bat la solution courante, alors à l'itération suivante l'algorithme l'utilisera comme nouvelle solution courante. Enfin, à la fin de chaque itération, l'algorithme adapte la structure de voisinage qui sera utilisée par la suite (par exemple, en la «raffinant» : générer de plus en plus de points, de plus en plus proches de l'optimum estimé, pour à terme avoir des garanties sur son optimalité).

2.3.1 Recherche par coordonnées

Un exemple très simple de recherche directe est la *recherche par coordonnées*, dite *CS* (pour *Coordinate Search*), proposé dès les années 1950 par Fermi et Metropolis (1952).

Algorithme 3 Recherche par coordonnées (*CS*)

fonction RECHERCHE_PAR_COORDONNÉES(f ; x_*^0)
  $k \leftarrow 0$
  $\delta^0 \leftarrow 1$
  tant que non(*Critère d'arrêt atteint*) **faire**
    $\mathcal{P}^k \leftarrow \left\{x_*^k \pm \delta^k e \mid e \text{ vecteur de la base canonique de } \mathbb{R}^n\right\}$
    $x_n^k \in \arg\min\left\{f(x) \mid x \in \mathcal{P}^k\right\}$
    si $f(x_n^k) < f(x_*^k)$ **alors**
      $x_*^{k+1} \leftarrow x_n^k$
      $\delta^{k+1} \leftarrow \delta^k$
    sinon
      $x_*^{k+1} \leftarrow x_*^k$
      $\delta^{k+1} \leftarrow \delta^k/2$
    $k \leftarrow k+1$
  renvoyer x_*^k

Le voisinage utilisé dans *CS* autour d'un point est l'ensemble des points ne différant que d'une valeur précise, sur une seule coordonnée. On définit un pas δ et la structure de recherche (souvent nommée sonde, ou *Poll*) $\mathcal{P}_\delta(x) = \{x \pm \delta e \mid e \text{ vecteur de la base canonique de } \mathbb{R}^n\}$. À chaque itération k, *CS* partira du meilleur point courant x_*^k et du pas de sonde courant δ^k, et examinera des candidats appartenant à $\mathcal{P}^k = \mathcal{P}_{\delta^k}\left(x_*^k\right)$. Il convergera naturellement autour d'un optimum en divisant δ^k par 2 à chaque fois que $\mathcal{P}^k$ échoue à proposer un candidat

améliorant la solution courante.

Il est aisé de démontrer qu'un tel algorithme converge vers un point x_* pour lequel toutes les dérivées directionnelles dans les directions canoniques sont ou nulles ou non définies. Torczon (1997) l'a prouvé sur les fonctions $\mathcal{C}^2$ et Audet et Dennis, Jr. (2003) pour le cas $\mathcal{C}^0$:

Théorème 2.3.1 (Convergence de *CS*)**.** *Soit $f : \mathbb{R}^n \to \mathbb{R}$ ayant des ensembles de niveau bornés. Soit x_* un point d'accumulation généré par une exécution de l'algorithme de recherche par coordonnées, sans critère d'arrêt. On a le résultat suivant :*

$$\forall d \in \{\pm e \mid e \in \textit{base canonique}\}, \ (f^\circ(x_*; d) \textit{ n'existe pas}), \textit{ ou } (f^\circ(x_*; d) \geq 0).$$

2.3.2 Recherche par motif généralisé

Cependant, les performances de *CS* ne sont pas satisfaisantes. Des améliorations existent, comme la recherche par motif généralisé (dite *GPS*, pour *Generalized Pattern Search*).

Algorithme 4 Recherche par motif généralisé (*GPS*)

fonction RECHERCHE_PAR_MOTIF_GÉNÉRALISÉ(f ; x_*^0, $\mathbf{D}$)
  $\delta^0 \leftarrow 1$
  tant que non(*Critère d'arrêt atteint*) **faire**
    $\mathbf{D}^k \leftarrow$ base positive de $\mathbb{R}^n \mid \forall d \in \mathbf{D}^k, \ d \in \mathbf{D}$
    $\mathcal{P}^k \leftarrow \left\{x_*^k \pm \delta^k d \mid d \in \mathbf{D}^k\right\}$
    $x_n^k \in \arg\min \left\{f(x) \mid x \in \mathcal{P}^k\right\}$
    si $f(x_n^k) < f(x_*^k)$ **alors**
      $x_*^{k+1} \leftarrow x_n^k$
      $\delta^{k+1} \leftarrow \delta^k/2$
    sinon
      $x_*^{k+1} \leftarrow x_*^k$
      $\delta^{k+1} \leftarrow 2\delta^k$
    $k \leftarrow k + 1$
  renvoyer x_*^k

L'algorithme *GPS* prend un paramètre supplémentaire par rapport à *CS* : la matrice $\mathbf{D}$. Il faut la voir comme un ensemble de directions potentielles que *GPS* utilisera pour créer les points candidats dans la sonde $\mathcal{P}^k$.

Rigoureusement, on définit $\mathbf{D} = \mathbf{GZ}$, où $\mathbf{G}$ est une base de $\mathbb{R}^n$ ($\mathbf{G} \in GL_n(\mathbb{R})$) et $\mathbf{Z}$ une matrice $n \times p$ (avec $p \geq n + 1$) d'entiers relatifs ($\mathbf{Z} \in M_{n,p}(\mathbb{Z})$) telle que les colonnes de $\mathbf{Z}$ forment une famille génératrice positive de $\mathbb{R}^n$. Via cette matrice, *GPS* a donc p directions à sa disposition. À chaque itération k, il en choisira un certain nombre (formant tout de même une base positive de l'espace) pour créer les points candidats : $\mathbf{D}^k$ une base positive formée

de colonnes de $\mathbf{D}$, et $\mathcal{P}^k = \left\{x_*^k + \delta^k d \mid d \in \mathbf{D}^k\right\}$.

De là, deux scénarios sont possibles. Soit $\mathcal{P}^k$ contient un point améliorant la solution courante x_*^k, soit tous les candidats échouent. Dans le premier cas, cela signifie que *GPS* vient de trouver une direction selon laquelle f semble diminuer. Il va donc augmenter le pas δ^k, pour aller plus rapidement explorer plus loin dans cette direction prometteuse. Dans le second cas, comme le faisait *CS*, *GPS* réduira δ^k pour entamer sa convergence vers x_*^k. On a finalement le résultat suivant (Torczon (1997) pour le cas $\mathcal{C}^2$, Audet et Dennis, Jr. (2003) pour le cas $\mathcal{C}^0$) :

Théorème 2.3.2 (Convergence de *GPS*)**.** *Soit $f : \mathbb{R}^n \to \mathbb{R}$ ayant des ensembles de niveau bornés. Soit x_* un point d'accumulation généré par une exécution de l'algorithme de recherche par motif généralisé, sans critère d'arrêt. On a le résultat suivant :*

$\forall d \in \mathbf{D} \mid d$ *apparaît dans une infinité de* $\mathbf{D}^k$, $(f^\circ(x_*; d)$ *n'existe pas), ou* $(f^\circ(x_*; d) \geq 0)$.

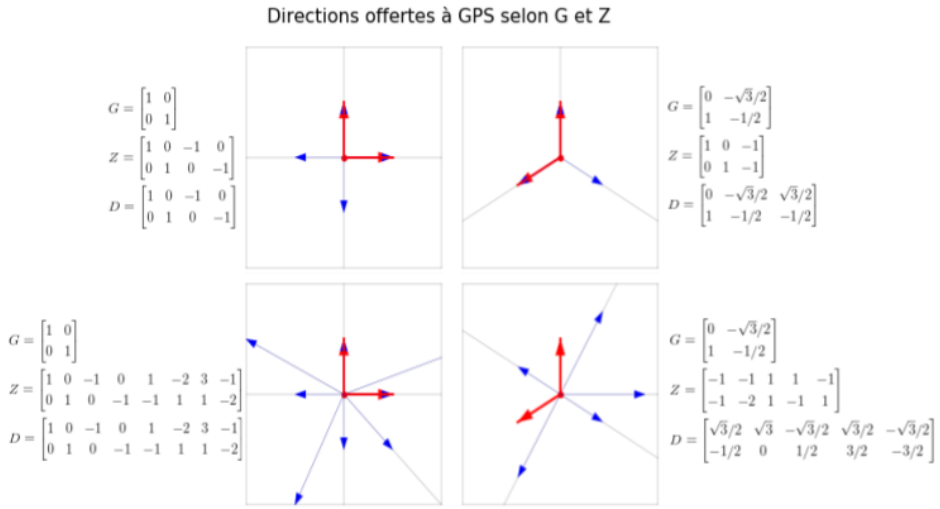

Figure 2.7 Construction de l'ensemble de directions utilisables par *GPS*.

2.3.3 Recherche directe par treillis adaptatif

Même si *GPS* est une grande amélioration de *CS* (au sens où il permet au pas de sonde δ d'augmenter pour aller explorer de nouvelles zones plus rapidement), sa performance reste peu satisfaisante. Il souffre en effet toujours de l'un des problèmes fondamentaux de *CS* : ses directions de recherche sont restreintes. Ainsi, un optimum renvoyé par *GPS* n'a de garanties d'optimalité que dans l'ensemble des directions contenues dans $\mathbf{D}$. On peut d'ailleurs remarquer, sur la figure 2.7, que selon le choix de $\mathbf{D}$, *GPS* peut comme *CS* être restreint aux directions canoniques.

Ce défaut peut être corrigé par une autre grande amélioration algorithmique : *MADS* (pour *Mesh Adaptative Direct Search*, par Audet et Dennis, Jr. (2006)). Il est sans doute plus performant que les deux précédents, et est un grand renforcement de *CS* et *GPS* sur plusieurs aspects. En premier lieu, comme dans *GPS* la taille du voisinage utilisé ne suit pas la décroissance monotone proposée par *CS*. Ici, les points candidats sont choisis sur un treillis de paramètre δ_m^k pouvant augmenter en cas de découverte d'un candidat intéressant. Cela permet d'aller rapidement explorer des zones prometteuses de l'espace. Dans l'algorithme *MADS*, on pourra observer l'évolution de δ_m^k qui traduit cette idée. La figure 2.8 en démontre l'intérêt, et met en lumière la puissance de *MADS* en exploration.

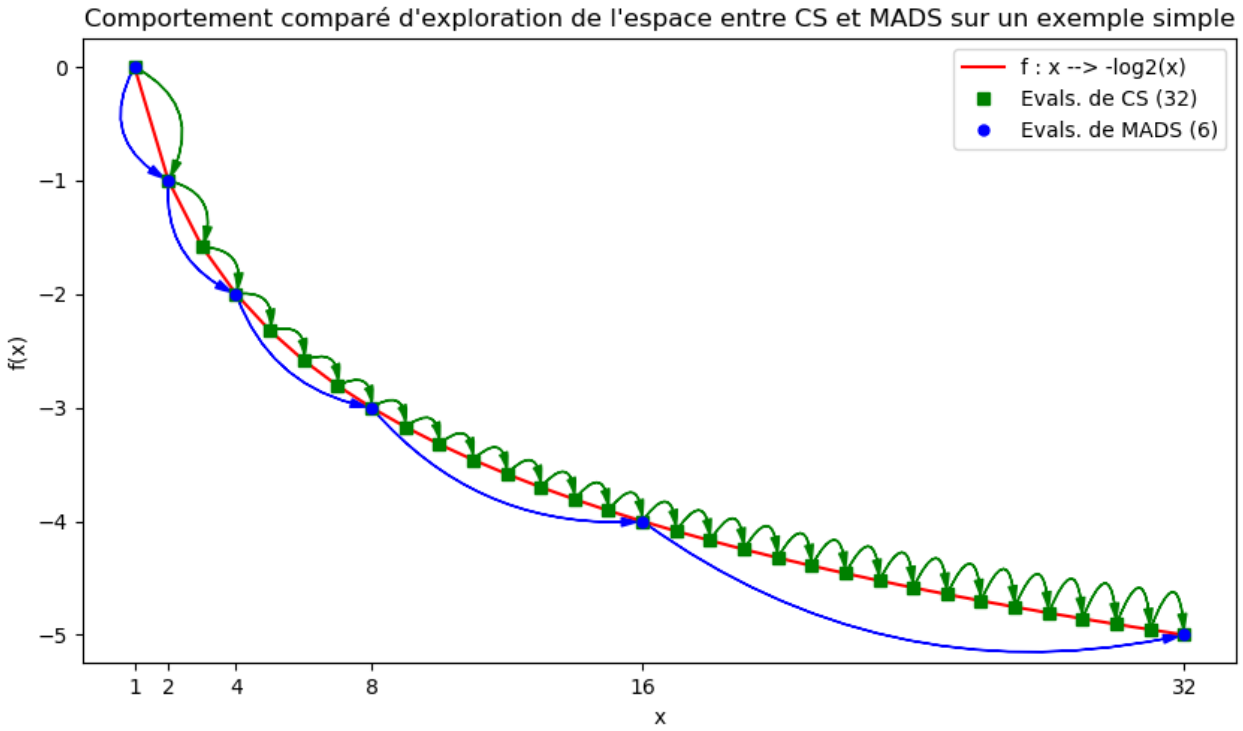

Figure 2.8 Intérêt de l'adaptation de la taille du treillis en exploration.

La fonction à minimiser, de $\mathbb{R}^+$ dans $\mathbb{R}$, est $f : x \to -\log_2(x)$ partant de $x_*^0 = 1$. L'algorithme *CS* va, à chaque itération, évaluer $x_*^k + 1$ et constater que f est plus basse en ce point. Il va donc avancer, à pas constant valant 1, vers $+\infty$. On peut par exemple observer qu'il lui faut 32 évaluations de la boîte noire pour atteindre le point $x = 32$. Dans le même temps, *MADS* se comportera comme *GPS* : il va lui aussi partir de $x_*^0 = 1$ et $\delta_m^0 = 1$. Cependant, à chaque itération, il va constater que $x_*^k + \delta_m^k$ améliore f. Il va donc prendre $x_*^{k+1} = x_*^k + \delta_m^k$, et actualiser $\delta_m^{k+1} = 2\delta_m^k$. Ainsi, la suite des points optimums proposés croît beaucoup plus vite vers les zones intéressantes où f est basse. Il ne faut à *MADS* que 6 appels à la boîte noire pour atteindre le point $x = 32$.

Cet exemple permet de comprendre comment *GPS* et *MADS* améliorent *CS* lors des phases d'exploration de l'espace : lorsqu'ils détectent une direction intéressante, ils augmentent le pas (la distance à la solution courante) auquel ils évaluent les nouveaux points. Ainsi, ils peuvent très vite s'éloigner de leur solution courante pour explorer les zones prometteuses détectées.

Ensuite, remarquons que *CS* et *GPS* n'examinaient qu'un ensemble très réduit de directions, et n'en considéraient jamais d'autres. Cela génère un manque de fiabilité pour *CS*, même sur des fonctions très simples comme la norme infinie [2]. En reprenant *CS*, on peut observer que seules les directions canoniques sont étudiées. Ainsi, un point «optimal» pour *CS* est un point autour duquel f ne décroît pas lorsqu'on change une seule composante. De même, *GPS* souffre d'une faiblesse analogue, car il n'élargit le champ des directions qu'a un ensemble toujours restreint et choisi à priori. Pour corriger ce défaut, *MADS* propose un comportement plus performant, synthétisé par la figure 2.9.

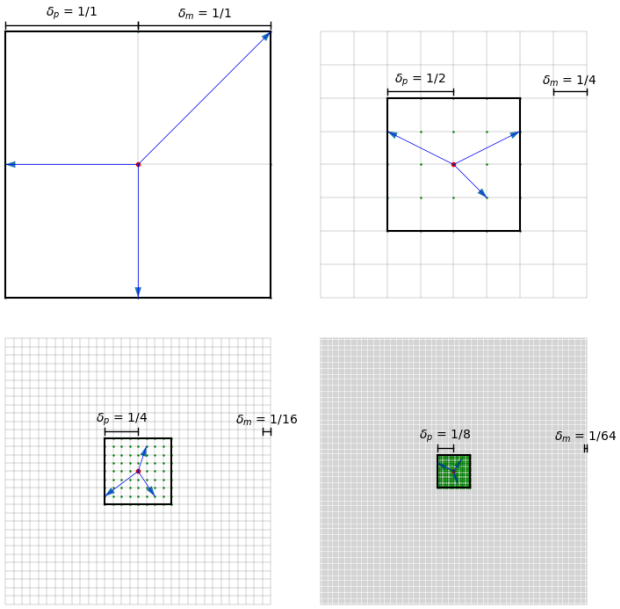

Figure 2.9 Directions possibles dans $\mathcal{P}$ selon δ_p.

MADS construit automatiquement divers ensembles de directions, dont l'union normalisée est dense sur la sphère unité. Lorsque l'algorithme stagne en une solution, il pourra donc chercher dans toutes les directions possibles. On peut observer, dans l'algorithme ou la figure 2.9, que l'ensemble $\mathbf{D}^k$ n'est pas restreint à un ensemble fini de directions prédéterminées. L'atout de *MADS* sur *GPS* est cette capacité à explorer un grand nombre de directions. L'algorithme propose également une *recherche globale* permettant de proposer des points quelconques à évaluer. En ignorant cette recherche globale, *MADS* exécute les instructions suivantes :

2. Exemple 3.3 de Audet et Hare (2017) : $\min \|x\|_\infty$ depuis $x^0 = [1, 1]^\top$. *CS* ne quittera jamais x^0.

Algorithme 5 Recherche directe par treillis adaptatif (*MADS* sans recherche globale)

fonction RECHERCHE_DIRECTE_PAR_TREILLIS_ADAPTATIF(f ; x_*^0)
  $\delta_p^0 \leftarrow 1$
  $k \leftarrow 0$
  tant que non(*Critère d'arrêt atteint*) **faire**
    $\delta_m^k \leftarrow \min\left(\delta_p^k, \left(\delta_p^k\right)^2\right)$
    $\mathcal{M}^k \leftarrow \left\{x_*^k + \delta_m^k m \mid m \in \mathbb{Z}^n\right\}$
    $\mathbf{D}^k \leftarrow$ base positive de $\mathbb{R}^n \mid \forall d \in \mathbf{D}^k,\ x_*^k + d \in \mathcal{M}^k$ et $\|d\|_\infty \leq \delta_p^k$
    $\mathcal{P}^k \leftarrow \left\{x_*^k + d \mid d \in \mathbf{D}^k\right\}$
    $x_n^k \in \arg\min\left\{f(x) \mid x \in \mathcal{P}^k\right\}$
    si $f(x_n^k) < f(x_*^k)$ **alors**
      $\delta_p^{k+1} \leftarrow 2\delta_p^k$
    sinon
      $\delta_p^{k+1} \leftarrow \delta_p^k/2$
    $x_*^{k+1} \in \arg\min\left(f\left(x_*^k\right), f\left(x_n^k\right)\right)$
    $k \leftarrow k+1$
  renvoyer x_*^k

Lorsque *MADS* évalue un ensemble de points autour de sa solution courante x_*^k, il commence par créer un «cadre de sonde» délimité par le paramètre δ_p^k. Dans ce cadre, il génère ensuite un treillis de pas δ_m^k beaucoup plus petit que δ_p^k (son carré). De là, *MADS* n'est pas restreint dans le choix des points de ce treillis qu'il évaluera. La seule contrainte est que l'ensemble des points retenus forme une base positive (pour assurer que tout l'espace est observé). On peut voir sur la figure 2.9 que les directions choisies changent d'une itération à l'autre. Ce comportement, combiné à un choix non-naïf, garantit que l'algorithme explorera finalement un ensemble de directions dense dans la sphère unité. En d'autres termes, un point d'accumulation renvoyé par *MADS* a une garantie d'optimalité bien plus forte que s'il venait de *GPS* : autour de ce point la fonction croît dans «toutes» les directions, et pas uniquement dans un ensemble prédéfini et restreint. Cette densité permet même de généraliser à toutes les directions possibles, donnant ainsi un résultat bien plus fort que pour *CS* et *GPS*.

Théorème 2.3.3 (Convergence de *MADS*)**.** *Soit $f : \mathbb{R}^n \to \mathbb{R}$ ayant des ensembles de niveau bornés. Soit x_* un point d'accumulation généré par une exécution de l'algorithme MADS, sans critère d'arrêt. Alors on a (Audet et Dennis, Jr. (2006)) :*

$$\forall d \mid \|d\|_2 = 1,\ \ f^\circ(x_*; d) \geq 0.$$

2.3.4 Prise en compte du bruit

Les méthodes de recherche directe ont naturellement été adaptées aux contextes bruités. Nous présenterons ici deux propositions : *Robust-MADS*, qui moyenne les observations voisines d'un point, et l'algorithme de Polak et Wetter qui exploite une précision variable pour l'augmenter dès que cela devient nécessaire.

Algorithme Robust-MADS

L'article (Audet *et al.* (2018)) propose une modification de *MADS* capable de traiter des fonctions bruitées (par un bruit stochastique ou non) via un processus de *smoothing*, qu'on peut traduire par *lissage*. L'idée est d'estimer $f(x)$ à chaque itération k par une valeur $f^k(x)$, une moyenne pondérée des observations en des points voisins de x. Plus un point y est proche de x, plus la pondération de son observation $f_\omega(y)$ est importante. Chaque itération va d'abord actualiser les estimés en tous les points déjà évalués (l'ensemble $\mathcal{V}^k$), puis générer une recherche locale $\mathcal{P}^k$ autour du point x_*^k ayant l'estimé le plus bas. Le point $x_n^k \in \underset{x\in\mathcal{P}^k}{\arg\min} f^k(x)$, meilleur point trouvé à l'itération k, est alors comparé à x_*^k pour décider d'une modification de δ_p^k. Lorsque l'algorithme va converger, la valeur de l'estimé $f^k(x_*^k)$ va tendre vers la vraie valeur $f(x_*^k)$ car la moyenne des bruits faussant les estimés au voisinage de x_*^k va s'atténuer.

Dans cet algorithme, on se contente, pour chaque point x auquel on souhaiterait calculer f, d'une unique observation. Pour alléger les notations, on dénotera ici cette unique observation par $f_\omega(x)$. On a $\forall x\ \exists!\ (k_0,\omega) \in \overline{\mathbb{N}} \times \Omega \mid \forall k < k_0,\ \mathcal{V}^k(x) = \emptyset$ et $\forall k \geq k_0,\ \mathcal{V}^k(x) = \{f_\omega(x)\}$.

Pour construire les estimés, il faut se munir d'une fonction φ de *kernel*, ou *noyau*, qu'on prendra ici gaussien. Une telle fonction détermine la pondération qu'on souhaite affecter à un estimé en un point voisin, connaissant sa distance au point actuel :

$$\forall x,u \in (\mathbb{R}^n)^2,\ \varphi(x,u;s) = \frac{\exp\left\{-\|x-u\|_2^2/2s^2\right\}}{\sqrt{2\pi}s}.$$

Ici, s est choisi dépendant de l'itération k, au sens où il diminue lorsque le treillis se raffine : $s^k = \beta\delta_p^k$, où β est une constante, et $\varphi^k(x,y) = \varphi(x,y;s^k)$ le noyau utilisant $s = s^k$.

Ainsi, on construit les estimés f^k de la façon suivante :

$$\forall k\ \forall x \in \mathbb{R}^n,\ f^k(x) = \frac{1}{\sum\limits_{y|\mathcal{V}^k(y)\neq\emptyset} \varphi^k(x,y)} \sum_{y|\mathcal{V}^k(y)\neq\emptyset} \varphi^k(x,y) f_\omega(y).$$

Cet algorithme est accompagné d'une analyse de convergence, plus faible que celle de *MADS* :

Théorème 2.3.4 (Convergence de Robust-*MADS*)**.** *Toute exécution de Robust-MADS sur une fonction f à observations bruitées créera une suite d'optimums estimés $(x_*^k)_k$ et de pas de treillis $(\delta_p^k)_k$ dont on peut extraire une sous-suite raffinante :*

$$\mathbb{P}\left(\exists x_*,\ \exists L \subset \mathbb{N},\ Card(L) = \infty\ \mid (\delta_p^k)_{k\in L} \to 0 \text{ et } (x_*^k)_{k\in L} \to x_*\right) = 1.$$

Algorithme 6 Recherche directe par treillis adaptatif robuste (Robust-*MADS*)

fonction RECHERCHE_DIRECTE_TREILLIS_ADAPTATIF_ROBUSTE(f ; x_*^0; $\beta = 1$)
  $k \leftarrow 0$
  $\delta_p^0 \leftarrow 1$
  $\mathcal{V}^{-1} \leftarrow Evaluation\left(\left\{x_*^0\right\}, \emptyset\right)$
  tant que non(*Critère d'arrêt atteint*) **faire**
    $\delta_m^k \leftarrow \min\left(\delta_p^k, \left(\delta_p^k\right)^2\right)$
    $s^k \leftarrow \beta\delta_p^k$
    $\mathcal{M}^k \leftarrow \left\{x_*^k + \delta_m^k m \mid m \in \mathbb{Z}^n\right\}$
    $\mathbf{D}^k \leftarrow$ base positive de $\mathbb{R}^n \mid \forall d \in \mathbf{D}^k,\ x_*^k + d \in \mathcal{M}^k$ et $\|d\|_\infty \leq \delta_p^k$
    $\mathcal{P}^k \leftarrow \left\{x_*^k + d \mid d \in \mathbf{D}^k\right\}$
    $\mathcal{V}^k \leftarrow Evaluation\left(\mathcal{P}^k, \mathcal{V}^{k-1}\right)$
    pour $x \in \mathcal{P}^k$ **faire** $f^k(x) \leftarrow Estimation(x, \mathcal{V}^k, s^k)$
    $x_n^k \in \arg\min\left\{f^k(x) \mid x \in \mathcal{P}^k\right\}$
    si $x_n^k = x_*^k$ **alors**
      $\delta_p^{k+1} \leftarrow 2\delta_p^k$
    sinon si $\mathcal{V}^{k-1}(x_n^k) \neq \emptyset$ **alors**
      $\delta_p^{k+1} = \delta_p^k$
    sinon
      $\delta_p^{k+1} \leftarrow \delta_p^k/2$
    $x_*^{k+1} \in \arg\min_{x\in\mathcal{V}^k} f^k(x)$
    $k \leftarrow k + 1$
  renvoyer x_*^k

fonction EVALUATION($\mathcal{P}, \mathcal{V}$)
  renvoyer $\mathcal{V} \cup \bigcup_{x\in\mathcal{P}\mid\mathcal{V}(x)=\emptyset} \{(x, f_\omega(x)\}$

fonction ESTIMATION($x, \mathcal{V}, s$)
  renvoyer $\dfrac{1}{\sum_{(y,.)\in\mathcal{V}^k} \varphi(x, y; s)} \sum_{(y,f_\omega(y))\in\mathcal{V}^k} \varphi(x, y; s) f_\omega(y)$

Bruit à amplitude réglable : algorithme de Polak et Wetter

Une stratégie commune reliant les méthodes directes et la précision variable est d'augmenter la précision (donc diminuer l'écart-type des erreurs) lorsque le pas de sonde diminue. L'idée sous-jacente est que lorsque l'algorithme entame sa convergence vers un optimum, la réduction de variance plus forte que celle du pas de sonde garantit que l'amplitude des erreurs est assez faible pour ne pas fausser la relation d'ordre entre les valeurs des estimés de la fonction aux points d'évaluation. On peut notamment citer, comme exemple de cette stratégie, l'article de Polak et Wetter (2006).

Cet article utilise une version «pseudo-CS» des méthodes de recherche directe, entre *CS* et *GPS*. La recherche ne se restreint pas aux coordonnées mais à des directions prises dans un ensemble prédéterminé (comme dans *GPS*). Cependant, l'augmentation du pas de sonde n'est pas envisagée. Le comportement est donc celui de *CS* avec plus de directions possibles.

Là-dessus, le traitement de l'incertitude vient s'ajouter. Il est à noter que l'algorithme de Polak et Wetter (2006) traite des erreurs pouvant être non-déterministes, mais à amplitude maximale bornée. En notant η le bruit à précision fixée, $\exists K_\eta \mid \forall \omega \in \Omega,\ |f_{\eta(\omega)}(x) - f(x)| < K_\eta$: toute réalisation du bruit donne une erreur qui ne peut être arbitrairement grande[3]. Le cadre originel présente cependant les erreurs comme étant déterministes (des approximations de modèle, par exemple) mais inconnues. Puisqu'on a supposé que l'amplitude des erreurs ne peut diverger (et que leur borne supérieure $K_\eta \to 0$ lorsque l'amplitude du bruit $\eta \to 0$), il est possible d'assurer une convergence non parasitée par les erreurs. Le cadre proposé par l'article est le suivant :

$$
\begin{array}{rcll}
f & \in & \mathcal{C}^{0^+}(\mathbb{R}^n, \mathbb{R}) & : \text{la fonction à minimiser (inaccessible)} \\
\epsilon & \in & \left(\mathbb{R}^+\right)^q & : \text{les tolérances du modèle d'approximation (réglables)} \\
f^* & \in & Fct((\mathbb{R}^q)^+ \times \mathbb{R}, \mathbb{R}) & : \text{la fonction d'approximation de } f \text{ aux seuils } \epsilon \\
\varphi & \in & Fct((\mathbb{R}^q)^+ \to \mathbb{R}^+) & : \text{la fonction de borne maximale de l'erreur aux tolérances } \epsilon \\
\rho & \in & Fct(\mathbb{R}^+ \to \left(\mathbb{R}^+\right)^q) & : \text{la fonction de lien entre le pas de sonde et un vecteur } \epsilon
\end{array}
$$

L'idée dans l'article est alors, lorsque δ_p (le pas de sonde) diminue, d'affecter à ϵ la valeur $\rho(\delta_p)$. Ainsi, si ρ décroît suffisamment rapidement, les erreurs décroîtront plus vite que la taille du treillis. Pour assurer que les erreurs soient négligeables, diverses stratégies sont possibles. Les auteurs proposent deux variantes dans l'article. La première est l'idée fondamentale, la variation limite la diminution de δ_p pour qu'il reste supérieur à l'amplitude des erreurs.

La première variante est :

3. Ce qui implique notamment que le bruit normal n'est, rigoureusement, pas traité par cet algorithme.

Algorithme 7 Algorithme de Polak et Wetter

fonction POLAK_WETTER(f^* ; $x_*^0, \mathbf{D}; \varphi, \rho, \zeta \geq 0$)
  $k \leftarrow 0$
  $\left(x_*^0, \delta^0, \epsilon^0\right) \leftarrow \left(x_*^0, 1, \rho(1)\right)$
  tant que non(*Critère d'arrêt atteint*) **faire**
    $\mathbf{D}^k \leftarrow$ base positive de $\mathbb{R}^n \mid \forall d \in \mathbf{D}^k,\ d \in \mathbf{D}$
    $\mathcal{P}^k \leftarrow \left\{x_*^k \pm \delta^k d \mid d \in \mathbf{D}^k\right\}$
    $x_n^k \in \arg\min \left\{f^*(\epsilon^k, x) \mid x \in \mathcal{P}^k\right\}$
    si $f^*(\epsilon^k, x_n^k) - f^*(\epsilon^k, x_*^k) < -\zeta\varphi(\epsilon^k)$ **alors**
      $\left(x_*^{k+1}, \delta^{k+1}, \epsilon^{k+1}\right) \leftarrow \left(x_n^k, \delta^k, \epsilon^k\right)$
    sinon
      $\left(x_*^{k+1}, \delta^{k+1}, \epsilon^{k+1}\right) \leftarrow \left(x_*^k, \delta^k/2, \rho(1/\delta^{k+1})\right)$
    $k \leftarrow k + 1$
  renvoyer x_*^k

Il est suivi d'une variante qui utilise le nombre N d'échecs comme paramètre à passer à ρ :

Algorithme 8 Algorithme de Polak et Wetter - Variante

fonction POLAK_WETTER_VARIANTE(f^* ; $x_*^0, \mathbf{D}; \varphi, \rho, \zeta \geq 0, \alpha \in]0, 1[$)
  $k \leftarrow 0$
  $\left(N, x_*^0, \delta^0, \epsilon^0\right) \leftarrow \left(0, x_*^0, 1, \rho(1)\right)$
  tant que non(*Critère d'arrêt atteint*) **faire**
    $\mathbf{D}^k \leftarrow$ base positive de $\mathbb{R}^n \mid \forall d \in \mathbf{D}^k,\ d \in \mathbf{D}$
    $\mathcal{P}^k \leftarrow \left\{x_*^k \pm \delta^k d \mid d \in \mathbf{D}^k\right\}$
    $x_n^k \in \arg\min \left\{f^*(\epsilon^k, x) \mid x \in \mathcal{P}^k\right\}$
    si $f^*(\epsilon^k, x_n^k) - f^*(\epsilon^k, x_*^k) < -\zeta\varphi(\epsilon^k)$ **alors**
      $\left(N, x_*^{k+1}, \delta^{k+1}, \epsilon^{k+1}\right) \leftarrow \left(N, x_n^k, \delta^k, \epsilon^k\right)$
    sinon
      $\left(N, x_*^{k+1}, \delta^{k+1}, \epsilon^{k+1}\right) \leftarrow \left(N + 1, x_*^k, \delta^k/2, \rho\left(N\right)\right)$
      tant que $\varphi\left(\epsilon^{k+1}\right)^\alpha < \left(\delta^{k+1}\right)^2$ **faire**
        $\delta^{k+1} \leftarrow \delta^{k+1}/2$
    $k \leftarrow k + 1$
  renvoyer x_*^k

L'article donne enfin une analyse de convergence analogue à *GPS* (Polak et Wetter (2006)) :

Théorème 2.3.5 (Convergence des algorithmes de Polak et Wetter)**.** *Soit $f : \mathbb{R}^n \to \mathbb{R}$ ayant des ensembles de niveau bornés. Soit x_* un point d'accumulation généré par une exécution de l'un des algorithmes de Polak et Wetter, sans critère d'arrêt. Alors on a :*

$$\forall d \in \mathbf{D} \mid d \text{ apparaît dans une infinité de } \mathbf{D}^k,\ f^\circ(x_*; d) \geq 0.$$

2.4 Autres algorithmes récents

Les quelques stratégies présentées ici ne sont bien évidemment qu'en nombre restreint, et ce chapitre est loin d'être exhaustif.

Par exemple, une stratégie courante en optimisation repose sur l'idée des *régions de confiance*, ou *trust region* en littérature anglophone. Il est notamment possible de se référer à Conn *et al.* (2000). Cette approche a été utilisée dans des contextes bruités (en section 10.6), ainsi qu'à des boîtes noires. L'optimisation de boîte noire bruitée a également été envisagée, avec notamment deux études récentes de Chen *et al.* (2018a) et Shashaani *et al.* (2016). Le second est particulièrement intéressant pour notre contexte puisque la méthode est spécifiquement conçue pour traiter des fonctions approximées par des estimations Monte-Carlo. Elle définit un modèle stochastique auquel elle applique les mécaniques de régions de confiance, tout en augmentant le nombre de tirages Monte-Carlo effectués au fil des itérations pour générer à terme une convergence presque sûre vers un optimum local.

Parmi les approches classiques de l'optimisation, les «line search», ou «recherches linéaires», sont souvent vues comme alternative aux régions de confiance. Le contexte bruité est étudié via ces approches, par exemple par Paquette et Scheinberg (2018).

L'idée du lissage présentée dans Robust-*MADS* (6) est également proposée par d'autres auteurs. Par exemple, Chen *et al.* (2018b) l'adapte au contexte spécifique d'optimisation de fonction bruitée car estimées via des simulations de Monte-Carlo. Les auteurs proposent ici d'augmenter le nombre de tirages Monte-Carlo par évaluation à chaque échec, et dans le même temps de réduire les effets du lissage. Cela amène alors à une optimisation estimant les valeurs de f par une annulation moyenne du bruit lors des premières itérations, puis accordant plus de crédit à chaque valeur l'incertitude issue des estimés Monte-Carlo diminue au fil des échecs.

Enfin, rappelons que les algorithmes présentés ici sont dans leur variante fondamentale. Tous ont connu de grandes améliorations. *MADS*, par exemple, est maintenant apte à traiter les contraintes. Ses variantes pratiques implémentent également la recherche opportuniste, le redimensionnement du problème en temps réel, la création de modèles, ou encore les variables granulaires facilitant la lecture des solutions générées.

Pour conclure ce chapitre, constatons que la littérature en optimisation de boîte noire, en général déterministe, a un alter ego apte à traiter les cas bruités. Beaucoup des méthodes existantes ont des variantes adaptées aux fonctions parasitées par du bruit, pouvant être déterministe comme stochastique. On pourra également noter que la réponse commune à ces problèmes tient dans une augmentation de la précision (une diminution de l'amplitude du bruit) à chaque fois que cela devient nécessaire.

CHAPITRE 3 ÉTUDE DE DEUX STRATÉGIES DE CONTRÔLE DE LA PRÉCISION

La revue de littérature du chapitre 2 a illustré la diversité des stratégies pour traiter des problèmes de boîte noire bruitée. Nous en retiendrons qu'il semble se dégager deux grandes directions possibles.

La première, qu'on nommera *intensification monotone*, calcule à précision constante tant que cela suffit à discriminer les solutions, puis l'augmente dès que nécessaire. Dans son principe, toute avancée vers un nouvel optimum courant est alors certaine, et on réduit constamment l'amplitude du bruit tant qu'on ne peut affirmer avoir trouvé de nouvelle solution. Cette stratégie facilite l'analyse de convergence, puisqu'à chaque itération k, le point x_*^k qu'on pense être le meilleur n'est dominé par aucun des autres déjà visités. L'enjeu de cette stratégie est alors de bien déterminer le pas d'augmentation de précision, pour éviter au maximum de la faire diverger trop vite (ce qui impacterait grandement le temps de calcul).

L'autre approche, à l'opposé, donnera un comportement moins rigide à la précision. On nommera cette stratégie *précision dynamique*. Ici, la précision pourra varier à chaque itération, et pas uniquement vers une augmentation. L'idée repose sur le fait qu'à une itération donnée et une précision donnée, on connaît la qualité statistique des estimés de l'objectif en chaque point étudié. On peut alors déterminer quel point est probablement le meilleur (celui d'estimé le plus bas), et quelle est cette probabilité (en fonction de l'écart-type statistique associé à chacun des estimés, selon le principe de la p-valeur détaillé en 1.1.3). Si cette probabilité est trop faible, on ne peut rien affirmer car nos estimés sont trop imprécis ; on va donc augmenter la précision. À l'opposé, si cette probabilité est très proche de 1, nos calculs étaient très précis, et peut-être aurait-on pu avoir des résultats aussi satisfaisants avec une précision plus faible ; on va donc la diminuer légèrement. Sous ce principe, l'optimisation est conduite à une précision offrant une incertitude jamais négligeable, mais toujours assez faible pour que le risque d'erreur soit contenu. On peut donc espérer avoir une convergence statistique vers l'optimum, en ayant toujours investi aussi peu d'efforts de calcul que nécessaire. L'enjeu de cette stratégie est de repérer ce «seuil» de précision nécessaire pour que nos affirmations soient juste assez précises pour trouver rapidement un optimum, et de créer une mécanique amenant efficacement la précision proche de ce seuil.

Cet chapitre présente deux algorithmes dérivés de *MADS*, aux fonctionnements suivant chacune de ces stratégies. La section 3.3 les munit d'une analyse de convergence. On trouvera aussi des adaptations de l'algorithme *NM* entrant dans ces deux paradigmes.

3.1 Intensification monotone : Augmentation en réponse à un échec

L'intensification monotone est la première grande approche envisageable pour manipuler la précision variable. L'idée est de toujours rester avec autant d'imprécision que possible pour éviter des calculs inutiles, et d'augmenter la précision dès que nécessaire afin de garantir des résultats jamais perturbés par un trop grand bruit aléatoire. Un exemple connu en optimisation de boîte noire est la méthode de Polak et Wetter (2006).

La méthode présentée dans l'article de Polak et Wetter (2006) et détaillée en 2.3.4 pilote l'optimisation de fonctions calculables uniquement de façon déterministe approchée. Plus précisément, elle traite des fonctions dont les approximations sont de précision réglable, comme par exemple la convergence numérique d'un système d'équations linéaires, à erreur maximale de résidu imposée par une borne supérieure η. Il s'ensuit que les approximations sont déterministes mais d'erreur exacte inconnue.

Cette méthode propose d'augmenter la précision durant l'optimisation, de telle sorte que l'erreur qui peut affecter le modèle reste toujours plus faible que la moitié de l'écart entre les valeurs de deux estimés. Ainsi, le point de meilleur estimé est, très probablement voir certainement, le point minimisant la vraie fonction.

Si l'optimisation se poursuit indéfiniment, le paramètre de précision tend vers l'infini (donc l'erreur faite par les approximés tend vers 0), ce qui assure que le point optimum obtenu satisfait des conditions d'optimalité locales. En effet, toutes les évaluations dans son voisinage sont alors moins bonnes (puisque sinon, il ne serait pas le point optimal retenu), et cette affirmation n'est plus parasitée par les erreurs de modèle car elles sont devenues négligeables.

Dans notre cas, les erreurs impactant les estimés ne sont pas déterministes. Au lieu d'une borne supérieure sur leur valeur, nous pouvons ici régler l'écart-type qui leur est associé. De fait, la «borne supérieure de l'erreur» η peut être remplacée par un seuil μ de confiance en-dessous duquel l'erreur restera probablement. On parlera donc d'écart-type σ et d'un seuil de confiance μ prédéterminés là où Polak et Wetter mentionnaient un seuil η. Ainsi, les mécaniques fondamentales de leur méthode restent valides malgré le changement de contexte.

Avec cette approche, il existe maintenant un risque, faible mais non nul, que les erreurs dépassent le seuil μ. Ceci est problématique car l'algorithme considère que ce seuil est une borne supérieure sur l'amplitude des erreurs. Ainsi, il peut occasionnellement se présenter un cas faussant l'interprétation des résultats faite par l'algorithme : si un estimé sous-évalue grandement la vraie fonction en un point donné x, l'algorithme va considérer x comme optimal, puisque selon son interprétation l'estimé ne peut être grandement erroné. Une erreur, improbable mais pas impossible, bloque donc l'algorithme sur un point qui n'est pas optimal.

Pour corriger ce problème, nous introduisons une fonction de raffinement des estimés : celui au point considéré comme optimal est amélioré via une nouvelle observation, pour statistiquement le ramener sous le seuil μ selon le principe développé dans l'annexe A. Cet ajout offre une garantie que les erreurs ne peuvent retenir l'algorithme sur un faux optimal. Cet algorithme est à convergence garantie vers un minimum local de la fonction non-bruitée : la preuve de convergence présentée en 3.3 peut s'y appliquer.

On se propose également d'étendre l'algorithme aux mécaniques de *MADS* au lieu de celles de *GPS*, ce qui se fait sans difficulté et améliore grandement la qualité de la recherche.

3.1.1 Application à l'algorithme *MADS*

L'algorithme "d'inspiration Polak et Wetter" que nous utiliserons se présente comme suit :

Algorithme 9 MADS à précision adaptative - Méthode à variation monotone

fonction MADS_Précision_Monotone(f ; x_*^0 ; ρ, $\alpha = 0.4$)
  $k \leftarrow 0$
  $r^0 \leftarrow 0$
  $\sigma^0 \leftarrow \rho\left(r^0\right)$
  $\mathcal{V}^0 \leftarrow \left\{\left(x_*^0, NouvelleEstimation(x_*^0, \sigma^0)\right)\right\}$
  $\delta_p^0 \leftarrow 1$
  tant que non(*Critère d'arrêt atteint*) **faire**
    $\sigma^k \leftarrow \rho\left(r^k\right)$
    $\delta_m^k \leftarrow \min\left(\delta_p^k, \left(\delta_p^k\right)^2\right)$
    $x_n^k, Statut, \mathcal{V}^{k+1} \leftarrow RechercheLocale\left(x_*^k; \delta_m^k, \sigma^k, \mathcal{V}^k\right)$
    si $Statut == Succes$ **alors**
      $r^{k+1} \leftarrow r^k$
      $\delta_p^{k+1} \leftarrow 2\delta_p^k$
    sinon si $Statut == EchecDansX$ **alors**
      $r^{k+1} \leftarrow r^k + 1$
      $\sigma^{k+1} \leftarrow \rho(r^{k+1})$
      tant que $\mu\left(\sigma^{k+1}\right)^\alpha < \left(\delta_p^k\right)^2$ **faire**
        $\delta_p^k \leftarrow \delta_p^k/2$
      $\delta_p^{k+1} \leftarrow \delta_p^k$
    sinon
      $\delta_p^{k+1} \leftarrow \delta_p^k/2$
    $k \leftarrow k + 1$
    $x_*^k \in \underset{y \in \mathcal{V}^k}{\arg\min}(f^k(y))$
  renvoyer x_*^k

fonction RECHERCHELOCALE(x_*^k ; δ_m^k, δ_p^k, σ^k, $\mathcal{V}^k$; $\zeta = 10^{-1}$)
  $\mathcal{M}^k \leftarrow \left\{\delta_m^k m \mid m \in \mathbb{Z}^n\right\}$
  $\mathcal{D}^k \leftarrow$ une base positive de $\mathbb{R}^n \mid \forall d \in \mathcal{D}^k,\ d \in \mathcal{M}^k$ et $\|d\|_\infty \leq \delta_p^k$
  $\mathcal{P}^k \leftarrow \left\{x_*^k + d \mid d \in \mathcal{D}^k \cup \{\vec{0}\}\right\}$ ▷ L'ajout de $\vec{0}$ force à réestimer $f(x_*^k)$
  pour $x \in \mathcal{P}^k$ **faire**
    si $\mathcal{V}^k(x) = \emptyset$ **alors**
      $\mathcal{V}^k(x) \leftarrow NouvelleEstimation(x, \sigma^k)$
    sinon
      $\sigma_{calc} \leftarrow \dfrac{1}{1/\sigma^k - 1/\sigma^k(x)}$
      $\mathcal{V}^k(x) \leftarrow AmeliorationEstime(x, \sigma_{calc}, \mathcal{V}^k(x))$
  $x_n^k \in \arg\min\left\{f^k(x) \mid x \in \mathcal{P}^k\right\}$
  si $f^k(x_n^k) - f^k(x_*^k) < -\zeta\mu(\sigma)$ **alors**
    $Statut \leftarrow Succes$
  sinon
    $Statut \leftarrow EchecDansX$ **si** $x_n^k \in \mathcal{X}$ **sinon** $Echec$
  $\mathcal{V}^{k+1} \leftarrow \bigcup\limits_{x \in \mathbb{R}^n} \left\{(x, \mathcal{V}^k(x))\right\}$
  renvoyer x_n^k, $Statut$, $\mathcal{V}^{k+1}$

fonction NOUVELLEESTIMATION(x, σ_{calc})
  $f^k(x) \leftarrow f_{\sigma_{calc}(\omega)}(x)$ une première observation ω ▷ Ne rien faire si $\sigma_{calc} \leq 0$
  $\sigma^k(x) \leftarrow \sigma_{calc}$
  $\mathcal{V}^k(x) \leftarrow [f^k(x), \sigma^k(x)]$
  renvoyer $\mathcal{V}^k(x)$

fonction AMELIORATIONESTIME(x, σ_{calc}, $\mathcal{V}^k(x)$)
  $f_{actuel}, \sigma_{actuel} \leftarrow$ Valeurs $f^k(x)$ et $\sigma^k(x)$ présentes dans $\mathcal{V}^k(x)$
  $f_{nouveau} \leftarrow f_{\sigma_{calc}(\omega)}(x)$ la nouvelle observation ω ▷ Ne rien faire si $\sigma_{calc} \leq 0$
  $f^k(x) \leftarrow \dfrac{f_{actuel}/\sigma_{actuel}^2 + f_{nouveau}/\sigma_{calc}^2}{1/\sigma_{actuel}^2 + 1/\sigma_{calc}^2}$
  $\sigma^k(x) \leftarrow \dfrac{1}{1/\sigma_{actuel}^2 + 1/\sigma_{calc}^2}$
  $\mathcal{V}^k(x) \leftarrow [f^k(x), \sigma^k(x)]$
  renvoyer $\mathcal{V}^k(x)$

3.1.2 Application à une heuristique : Adaptation de Nelder-Mead - Chang

L'algorithme de Nelder-Mead modifié par Chang (2012), détaillé en 2.1.2, traite des fonctions bruitées mais à précision pas nécessairement réglable. Pour les optimiser, il propose de multiplier les observations en les points du simplexe pour que la loi des grands nombres ramène les estimés à la vraie valeur de l'objectif. Dans notre projet, on dispose d'un levier supplémentaire avec la précision variable. Il est donc naturel de chercher à adapter l'algorithme pour proposer une variante exploitant cette hypothèse.

L'algorithme de Chang impose, à l'itération k, un nombre N^k tel que chaque point rencontré à l'itération k soit analysé avec un estimé calculé via N^k observations. Ce nombre $N^k = \left\lfloor \sqrt{k} \right\rfloor$ diverge lorsque k augmente, ce qui impose aux itérations de manipuler des estimés de plus en plus précis. Lorsque l'algorithme rencontre un point x, il lance $N^k - Card(\mathcal{V}^k(x))$ évaluations de la boîte noire pour finalement obtenir $Card(\mathcal{V}^k(x)) = N^k$ observations de $f(x)$, et construit ensuite l'estimé à partir de ces N^k observations.

Dans notre cas, il suffit de remplacer N^k par un écart-type maximal σ^k. Là où l'algorithme de Chang lançait $N^k - Card(\mathcal{V}^k(x))$ évaluations, nous pouvons demander une unique observation, à une précision calculée pour amener l'estimé global à un écart-type σ^k.

L'algorithme modifié que nous proposons a la même structure globale que l'original :

Algorithme 10 Algorithme de Nelder et Mead (NM) stochastique - «Type Chang»

fonction NELDER_MEAD_CHANG(f ; $y_0^0, \dots, y_n^0$; $\delta_e, \delta_{oc}, \delta_{ic}, \gamma$; ρ)
  $k \leftarrow 0$
  pour $x \in \mathcal{X}$ **faire** $\mathcal{V}(x) \leftarrow \emptyset$
  tant que non(*Critère d'arrêt atteint*) **faire**
    $\sigma^k \leftarrow \rho(k)$
    pour $i \in [\![0, n]\!]$ **faire** $f^k(y_i^k) \leftarrow Evaluation(f, y_i^k, \sigma^k)$
    $\mathbb{Y}^k \leftarrow [y_0^k, \dots, y_n^k]$ rangés par images par f^k croissantes
    $f_b^k \leftarrow f^k(y_0^k)$
    $x_c^k, x_r^k, f_r^k \leftarrow Reflexion(f; \mathbb{Y}^k; \sigma^k)$
    si $f_r^k < f_b^k$ **alors** $\mathbb{Y}^{k+1} \leftarrow Expansion(f, \delta_e; x_c^k, \mathbb{Y}^k, f_r^k, x_r^k; \sigma^k)$
    si $f_b^k \leq f_r^k < f^k(y_{n-1}^k)$ **alors** $\mathbb{Y}^{k+1} \leftarrow [y_0^k, \dots, y_{n-1}^k, x_r^k]$
    si $f^k(y_{n-1}^k) \leq f_r^k < f^k(y_n^k)$ **alors** $\mathbb{Y}^{k+1} \leftarrow ContractExt(f, \delta_{oc}; x_c^k, \mathbb{Y}^k, f_r^k, x_r^k; \sigma^k)$
    si $f^k(y_n^k) \leq f_r^k$ **alors** $\mathbb{Y}^{k+1} \leftarrow ContractInt(f, \delta_{ic}, \gamma; x_c^k, \mathbb{Y}^k, f_r^k; \sigma^k)$
    $k \leftarrow k + 1$
  renvoyer $\arg\min_{y \in \mathbb{Y}^k} f(y)$

La seule modification vient de la fonction *Evaluation* affectant des valeurs aux estimés f^k :

1: **fonction** EVALUATION(f, x, σ_{calc})
2: **si** $\mathcal{V}^k(x) = \emptyset$ **alors** $\mathcal{V}^k(x) \leftarrow NouvelleEstimation(f, x, \sigma_{calc})$ **sinon** $\mathcal{V}^k(x) \leftarrow AmeliorationEstime(f, x, \sigma_{calc})$
3: **renvoyer** $f^k(x)$ contenue dans $\mathcal{V}^k(x)$

1: **fonction** NOUVELLEESTIMATION(f, x, σ_{calc})
2: **renvoyer** $[f_{\sigma_{calc}(\omega)}(x), \sigma_{calc}]$ une première observation ω

1: **fonction** AMELIORATIONESTIME(f, x, σ_{calc})
2: $f_{actuel}, \sigma_{actuel} \leftarrow$ valeurs $f^k(x)$ et $\sigma^k(x)$ présentes dans $\mathcal{V}^k(x)$
3: $f_{nouveau} \leftarrow f_{\sigma_{calc}(\omega)}(x)$ la nouvelle observation ω
4: $\sigma^k(x) \leftarrow \left(1/\sigma^2_{actuel} + 1/\sigma^2_{calc}\right)^{-1}$
5: $f^k(x) \leftarrow \left(f_{actuel}/\sigma^2_{actuel} + f_{nouveau}/\sigma^2_{calc}\right)/\sigma^k(x)$
6: **renvoyer** $[f^k(x), \sigma^k(x)]$

Les autres fonctions, elles, restent inchangées depuis la variante de Chang présentée dans 2 :

1: **fonction** REFLEXION(f ; $\mathbb{Y}$; σ)
2: $x_c \leftarrow \dfrac{1}{n}\sum_{i=0}^{n-1} y_i$
3: $x_r \leftarrow x_c + (x_c - y_n)$
4: $f_r \leftarrow Evaluation(f, x_r, \sigma)$
5: **renvoyer** x_c, x_r, f_r

1: **fonction** EXPANSION(f, δ_e ; $x_c, \mathbb{Y}, f_r, x_r$; σ)
2: $x_e \leftarrow x_c + \delta_e(x_c - y_n)$
3: $f_e \leftarrow Evaluation(f, x_e, \sigma)$
4: **si** $f_e < f_r$ **alors renvoyer** $[y_0, \ldots, y_{n-1}, x_e]$ **sinon renvoyer** $[y_0, \ldots, y_{n-1}, x_r]$

1: **fonction** CONTRACTEXT(f, δ_{oc} ; $x_c, \mathbb{Y}, f_r, x_r$; σ)
2: $x_{oc} \leftarrow x_c + \delta_{oc}(x_c - y_n)$
3: $f_{oc} \leftarrow Evaluation(f, x_{oc}, \sigma)$
4: **si** $f_{oc} < f_r$ **alors renvoyer** $[y_0, \ldots, y_{n-1}, x_{oc}]$ **sinon renvoyer** $[y_0, \ldots, y_{n-1}, x_r]$

1: **fonction** CONTRACTINT(f, δ_{ic}, γ ; $x_c, \mathbb{Y}, f_r$; σ)
2: $x_{ic} \leftarrow x_c - \delta_{ic}(x_c - y_n)$
3: $f_{ic} \leftarrow Evaluation(f, x_{ic}, \sigma)$
4: **si** $f_{ic} < f_r$ **alors renvoyer** $[y_0, \ldots, y_{n-1}, x_{ic}]$ **sinon renvoyer** $[y_0 - \gamma(y_0 - y_i),\ i \leq n]$

3.2 Précision dynamique : compromis intensification - temps de calcul

L'approche présentée dans la section précédente exploite un principe de renforcement de la précision lorsque l'algorithme commence à converger vers un minimum local. De fait, la probabilité de faire un mauvais choix (privilégier un point qu'on pense optimal alors qu'il ne l'est pas) devient négligeable lorsque le treillis autour du point optimal diminue en taille. De façon grossière, on peut retenir que l'intensification permanente mise sur une approche robuste, en le sens qu'elle limite au maximum les effets indésirables liés à l'aléatoire et ne se fie qu'à des résultats considérés comme très fiables. Pour cela, aucun compromis n'est fait sur la précision, et la méthode ne considère pas la possibilité de la rabaisser pour économiser des ressources. Il s'agit de la principale différence avec l'approche présentée dans cette section.

L'idée ici est de faire tous les calculs d'estimés à une précision «moyenne». Cela signifie que lors d'une itération, on ne cherche pas forcément à avoir une probabilité d'amélioration proche de 1, mais simplement au-delà d'un certain seuil $\beta > 1/2$. Si un point *améliorant* est trouvé (au sens où il présente une probabilité supérieure à β de battre l'optimum courant), il devient le nouvel optimum. De plus, on va ré-estimer plus souvent les valeurs de la fonction aux points déjà visités, tant que la probabilité de conclure sur leur qualité est trop faible.

L'idée sous-jacente repose sur l'observation statistique : Si un point est, selon la vraie fonction, meilleur qu'un autre, cette propriété se manifestera tôt ou tard au fil des réévaluations car les approximés tendront vers la véritable valeur de la fonction. En réduisant petit à petit les amplitudes des incertitudes, on peut conclure sur la qualité d'un point en investissant un minimum de ressources.

De plus, on peut remarquer qu'il n'est pas toujours nécessaire d'estimer f de façon très précise. Si deux estimés sont très éloignés et avec un écart-type pas trop grand, il est inutile de raffiner leur précision pour savoir en quel point la vraie fonction est la plus basse. L'approche monotone ne tient pas compte de ceci. Si elle a au préalable augmenté grandement l'indice de précision, elle va investir beaucoup plus de ressources que nécessaire pour analyser les points suivants. Cette méthode va corriger cette faiblesse en adaptant la précision.

À la fin d'une itération, il suffit de calculer la probabilité d'amélioration entre l'optimum et le nouveau candidat. Si elle est proche de 1/2, on ne peut rien conclure et il faut donc augmenter la précision. Si, au contraire, elle est proche de 0 ou de 1, on peut sans risque conclure sur la qualité des nouveaux points, et donc diminuer la précision pour les itérations suivantes : on vient d'observer qu'on investit plus de ressources de calcul que nécessaire. On peut illustrer ce principe simplement.

On considère, sur les trois figures suivantes, deux points x_1 et x_2 dont les estimés diffèrent

d'une unité. La différence entre les figures vient de la «précision» à laquelle l'estimé en x_2 est calculé : L'écart-type associé à l'incertitude autour de la vraie valeur de $f(x_2)$ change l'interprétation qu'on peut donner aux figures. Dans chaque cas, on calculera la probabilité que f soit plus faible en x_2 qu'en x_1, connaissant les valeurs des estimés et leurs écarts-types. Il s'agit, rigoureusement, de la p-valeur statistique de l'hypothèse «$f(x_2) < f(x_1)$».

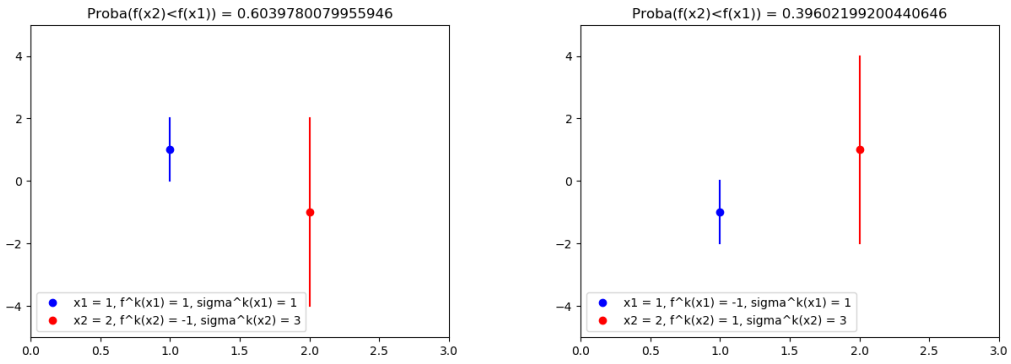

Figure 3.1 Comparaison de deux points : estimés distants mais trop peu précis.

Dans le premier cas (figure 3.1), l'imprécision est grande et ne peut presque rien conclure. Même si l'estimé $f^k(x_2)$ est assez éloigné de $f^k(x_1)$, l'hypothèse «$f(x_2) < f(x_1)$» est presque aussi viable que sa complémentaire «$f(x_2) \geq f(x_1)$». On ne peut donc pas conclure sur le point le plus optimal entre x_1 et x_2, à cause d'une trop faible précision. Observant cela, on sait donc qu'il faudra l'augmenter pour espérer avoir des résultats exploitables.

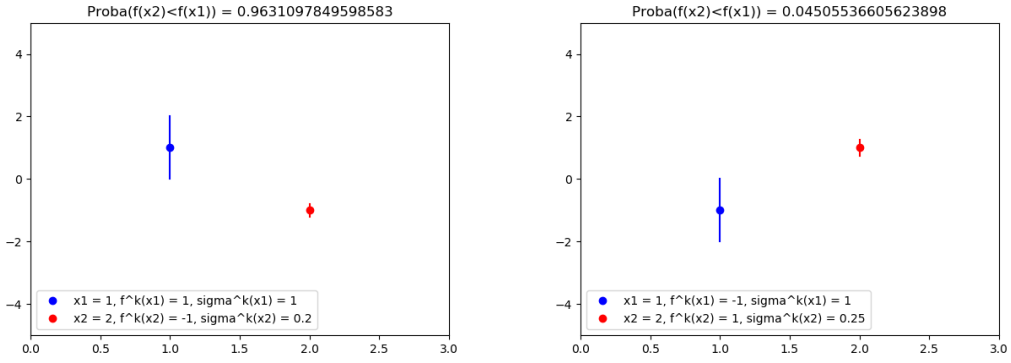

Figure 3.2 Comparaison de deux points : estimés distants et trop précis.

La deuxième figure (3.2) illustre le problème inverse. Les estimés $f^k(x_1)$ et $f^k(x_2)$ sont distants, et l'imprécision autour de leur valeur est très faible. Ainsi, on peut sans grand risque déterminer le point minimisant la vraie fonction. Ces conditions sont excellentes pour l'optimisation,

puisqu'il n'y a alors presque pas de risque de prendre une décision erronée. Cependant, parvenir à une telle situation peut coûter très cher en terme de ressources temporelles et calculatoires. Le temps de calcul et les ressources à mobiliser croissent très fortement lorsqu'on cherche à diminuer l'écart-type des estimés. Ainsi, aussi avantageuse soit la situation sur cette figure, elle n'est pas envisageable dans un cas réel puisque les efforts pour y parvenir sont trop importants.

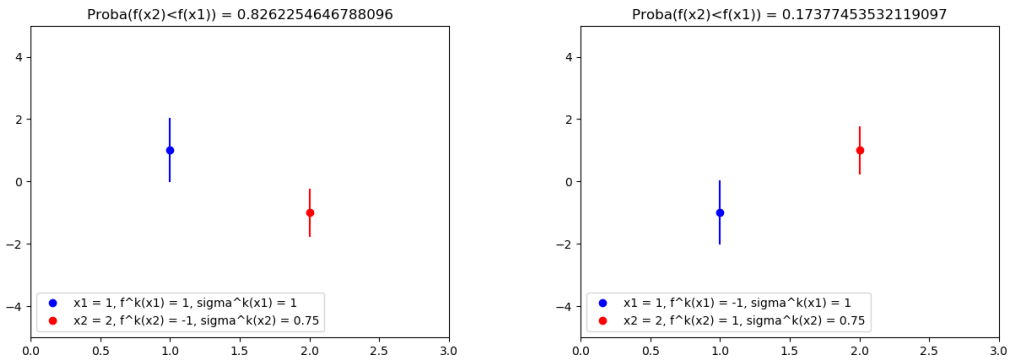

Figure 3.3 Comparaison de deux points : estimés distants et précis au nécessaire.

Un compromis intéressant est proposé en figure 3.3. Ici, les estimés ont été calculés avec une précision «moyenne», au sens où l'écart-type de l'incertitude reste non négligeable mais pas démesuré. Dans ces conditions, la p-valeur de l'hypothèse «$f(x_2) < f(x_1)$» est supérieure à 80%, mais l'écart-type assez grand a requis beaucoup moins d'efforts de calcul que celui du cas précédent pour être atteint. Ainsi, dans cette situation, on peut prendre des décisions avec un risque toujours limité, mais cette fois sans dépenser un budget de calcul trop grand.

Cette remarque illustre la principale différence entre la méthode de Polak et Wetter et celle que nous nous apprêtons à présenter. L'algorithme suivant cherche à toujours dépenser le moins de budget de calcul possible, en investissant juste assez pour avoir des résultats viables : il cherche à toujours se placer dans le cas illustré par la figure 3.3. Pour cela, à la fin de chaque itération, il calculera la p-valeur de «$f(x_{candidat}) < f(x_{opti\ actuel})$», et adaptera la précision de calcul à l'itération suivante pour qu'elle reste toujours relativement loin de 50%[1]. On obtient alors un compromis naturel entre certitude des décisions et consommation de ressources.

3.2.1 Application à l'algorithme *MADS*

Une variante de *MADS* encodant ces idées peut être envisagée :

1. Une hypothèse est alors dite «viable» si sa p-valeur p vérifie $p \geq 0.5 + \beta$, avec β fixé. On cherche donc à ce que la précision soit la plus faible possible, tout en rendant l'une des hypothèses viables.

Algorithme 11 MADS à précision adaptative - Méthode dynamique

fonction MADS_PRECISION_DYNAMIQUE(f ; x_*^0 ; ρ, $\beta_l = 0.25$, $\beta_u = 0.75$)
  $k \leftarrow 0$
  $r^0 \leftarrow 0$
  $\mathcal{V}^{-1} \leftarrow \left\{\left(x_*^0, NouvelleEstimation(x_*^0, \sigma^0)\right)\right\}$
  $\delta_p^0 \leftarrow 1$
  tant que non(*Critère d'arrêt atteint*) **faire**
    $\sigma^k \leftarrow \rho(r^k)$
    $\delta_m^k \leftarrow \min\left(\delta_p^k, \left(\delta_p^k\right)^2\right)$
    $x_n^k, Statut, \mathcal{V}^k \leftarrow RechercheLocale(x_*^k, \delta_m^k, \delta_p^k, \sigma^k, \mathcal{V}^{k-1})$
    $p^k \leftarrow ProbaGain(f^k(x_*^k),\ \sigma^k(x_*^k),\ f^k(x_n^k),\ \sigma^k(x_n^k))$
    si $Statut == Succes$ **alors**
      si $p^k > \beta_u$ **alors** $\delta_p^{k+1} \leftarrow 2\delta_p^k$
      sinon $\delta_p^{k+1} \leftarrow \delta_p^k$
    sinon
      si $p^k < \beta_l$ **alors** $\delta_p^{k+1} \leftarrow \delta_p^k/2$
      sinon $\delta_p^{k+1} \leftarrow \delta_p^k$
    $r^{k+1} \leftarrow r^k + MaJPrecision(p^k)$
    $\mathcal{V}^k \leftarrow ReestimeCache(\mathcal{V}^k, \sigma = \rho(r^k - 5))$
    $x_*^k \in \arg\min_{y \in \mathcal{V}^k}(f^k(y))$
    $k \leftarrow k + 1$
  renvoyer x_*^k

fonction PROBAGAIN(f_{est}^1, σ^1, f_{est}^2, σ^2)
  renvoyer $\Psi(f_{est}^2 - f_{est}^1, \sigma^1, \sigma^2)$

Où Ψ est définie comme en 1.1.3 : $\Psi(\delta, \sigma_x, \sigma_y) = \mathbb{P}\left(Y \leq X \mid X \sim N(0, \sigma_x^2),\ Y \sim N(\delta, \sigma_y^2)\right)$.

fonction MAJPRECISION(p_{amel})
  $p \leftarrow |p_{amel} - 0.5|$
  renvoyer
$$\begin{cases} -3 & \text{si} \quad 0.50 \geq p > 0.49 \\ -2 & \text{si} \quad 0.49 \geq p > 0.45 \\ -1 & \text{si} \quad 0.45 \geq p > 0.40 \\ 0 & \text{si} \quad 0.40 \geq p > 0.25 \\ 1 & \text{si} \quad 0.25 \geq p > 0.15 \\ 2 & \text{si} \quad 0.15 \geq p > 0.10 \\ 3 & \text{si} \quad 0.10 \geq p \geq 0 \end{cases}$$

fonction RECHECHELOCALE(x_*^k, δ_m^k, δ_p^k, σ^k, $\mathcal{V}^k$)
  $\mathcal{M}^k \leftarrow \left\{\delta_m^k m \mid m \in \mathbb{Z}^n\right\}$
  $\mathcal{D}^k \leftarrow$ une base positive de $\mathbb{R}^n \mid \forall d \in \mathcal{D}^k,\ d \in \mathcal{M}^k$ et $\|d\|_\infty \leq \delta_p^k$
  $\mathcal{P}^k \leftarrow \left\{x_*^k + d \mid d \in \mathcal{D}^k\right\}$
  pour $x \in \mathcal{P}^k$ **faire**
    si $\mathcal{V}^k(x) = \emptyset$ **alors**
      $\mathcal{V}^k(x) \leftarrow NouvelleEstimation(x, \sigma^k)$
    sinon
      $\sigma_{calc} \leftarrow \dfrac{1}{1/\sigma^k - 1/\sigma^k(x)}$
      $\mathcal{V}^k(x) \leftarrow AmeliorationEstime(x, \sigma_{calc}, \mathcal{V}^k(x))$
  $x_n^k \in \arg\min\left\{f^k(x) \mid x \in \mathcal{P}^k\right\}$
  si $f^k(x_n^k) < f^k(x_*^k)$ **alors**
    $Statut \leftarrow Succes$
  sinon
    $Statut \leftarrow Echec$
  $\mathcal{V}^{k+1} \leftarrow \bigcup_{x \in \mathbb{R}^n}\left\{(x, \mathcal{V}^k(x))\right\}$
  renvoyer x_n^k, *Statut*, $\mathcal{V}^{k+1}$

fonction NOUVELLEESTIMATION(x, σ_{calc})
  $f^k(x) \leftarrow f_{\sigma_{calc}(\omega)}(x)$ une première observation ω ▷ Ne rien faire si $\sigma_{calc} \leq 0$
  $\sigma^k(x) \leftarrow \sigma_{calc}$
  $\mathcal{V}^k(x) \leftarrow [f^k(x), \sigma^k(x)]$
  renvoyer $\mathcal{V}^k(x)$

fonction AMELIORATIONESTIME(x, $\sigma_{calc}, \mathcal{V}^k(x)$)
  $f_{actuel}, \sigma_{actuel} \leftarrow$ Valeurs $f^k(x)$ et $\sigma^k(x)$ présentes dans $\mathcal{V}^k(x)$
  $f_{nouveau} \leftarrow f_{\sigma_{calc}(\omega)}(x)$ la nouvelle observation ω ▷ Ne rien faire si $\sigma_{calc} \leq 0$
  $f^k(x) \leftarrow \dfrac{f_{actuel}/\sigma_{actuel}^2 + f_{nouveau}/\sigma_{calc}^2}{1/\sigma_{actuel}^2 + 1/\sigma_{calc}^2}$
  $\sigma^k(x) \leftarrow \dfrac{1}{1/\sigma_{actuel}^2 + 1/\sigma_{calc}^2}$
  $\mathcal{V}^k(x) \leftarrow [f^k(x), \sigma^k(x)]$
  renvoyer $\mathcal{V}^k(x)$

fonction REESTIMECACHE($\mathcal{V}^k, \sigma_{calc}$)
  pour $x \in \mathbb{R}^n \mid \mathcal{V}^k(x) \neq \emptyset$ **faire** $\mathcal{V}^k(x) \leftarrow AmeliorationEstime(x, \sigma_{calc}, \mathcal{V}^k(x))$
    ▷ En pratique, ssi $x \mid ProbaGain(f^k(x_*^k), \sigma^k(x_*^k), f^k(x), \sigma^k(x)) \geq 0.1$
  $\mathcal{V}^{k+1} \leftarrow \bigcup_{x \in \mathbb{R}^n}\left\{(x, \mathcal{V}^k(x))\right\}$
  renvoyer $\mathcal{V}^k$

3.2.2 Application à une heuristique : Nelder-Mead à précision dynamique

La section précédente montre comment l'algorithme *MADS* à précision dynamique (11) modifiait le comportement de la précision par rapport à l'adaptation de la méthode de Polak et Wetter (9). En suivant un principe analogue, on peut proposer une variante à l'heuristique de *NM* que nous proposions en 10. Au lieu de toujours augmenter la précision à chaque itération (via l'instruction $\sigma^k \leftarrow \rho(k)$), on peut lui donner des variations suivant celles du diamètre du simplexe. Ainsi, en notant $Diam(\mathcal{E}) = \sup_{(x,y)\in\mathcal{E}^2} \|y - x\|$ le diamètre d'un ensemble $\mathcal{E}$, on se propose de considérer le quotient $\dfrac{Diam([y_0^0, \ldots, y_n^0])}{Diam([y_0^k, \ldots, y_n^k])}$, qui détermine le ratio entre le diamètre du simplexe au début de l'optimisation et son diamètre à l'itération k. Plus le diamètre courant décroît, plus ce ratio augmente. On se propose alors de le prendre comme indicateur du niveau de précision à l'itération $k+1$: plus $Diam([y_0^k, \ldots, y_n^k])$ est petit, plus les points sont proches, et plus on souhaitera une précision élevée. Ainsi, la seule modification est l'affectation de l'écart-type désiré σ^k, qui se fait maintenant via :

$$\sigma^k \leftarrow \rho\left(\frac{Diam([y_0^0, \ldots, y_n^0])}{Diam([y_0^k, \ldots, y_n^k])}\right).$$

Algorithme 12 Algorithme de Nelder et Mead (*NM*) stochastique - Précision dynamique

fonction NELDER_MEAD_PRECISION_DYNAMIQUE(f ; $y_0^0, \ldots, y_n^0$; $\delta_e, \delta_{oc}, \delta_{ic}, \gamma; \rho$)
  $k \leftarrow 0$
  $D_0 \leftarrow Diam([y_0^0, \ldots, y_n^0])$
  pour $x \in \mathcal{X}$ **faire** $\mathcal{V}(x) \leftarrow \emptyset$
  tant que non(*Critère d'arrêt atteint*) **faire**
    $\sigma^k \leftarrow \rho\left(D_0/Diam([y_0^k, \ldots, y_n^k])\right)$
    pour $i \in [\![0, n]\!]$ **faire** $f^k(y_i^k) \leftarrow Evaluation(f, y_i^k, \sigma^k)$
    $\mathbb{Y}^k \leftarrow [y_0^k, \ldots, y_n^k]$ rangés par images par f^k croissantes
    $f_b^k \leftarrow f^k(y_0^k)$
    $x_c^k, x_r^k, f_r^k \leftarrow Reflexion(f; \mathbb{Y}^k; \sigma^k)$
    si $f_r^k < f_b^k$ **alors** $\mathbb{Y}^{k+1} \leftarrow Expansion(f, \delta_e; x_c^k, \mathbb{Y}^k, f_r^k, x_r^k; \sigma^k)$
    si $f_b^k \leq f_r^k < f^k(y_{n-1}^k)$ **alors** $\mathbb{Y}^{k+1} \leftarrow [y_0^k, \ldots, y_{n-1}^k, x_r^k]$
    si $f^k(y_{n-1}^k) \leq f_r^k < f^k(y_n^k)$ **alors** $\mathbb{Y}^{k+1} \leftarrow ContractExt(f, \delta_{oc}; x_c^k, \mathbb{Y}^k, f_r^k, x_r^k; \sigma^k)$
    si $f^k(y_n^k) \leq f_r^k$ **alors** $\mathbb{Y}^{k+1} \leftarrow ContractInt(f, \delta_{ic}, \gamma; x_c^k, \mathbb{Y}^k, f_r^k; \sigma^k)$
    $k \leftarrow k + 1$
  renvoyer $\arg\min_{y\in\mathbb{Y}^k} f(y)$

3.3 Analyse de convergence

Cette section a pour but d'étudier la convergence théorique des deux variantes de *MADS* proposées dans cette section (9 et 11). Nous démontrerons ici que s'ils tournent sans critère d'arrêt, ils généreront un point d'accumulation x_*^∞ qui satisfera les conditions d'optimalité locale sur la fonction non-bruitée. L'argumentaire suivra l'analyse de convergence de *MADS* dans le cas déterministe proposée dans Audet et Hare (2017).

En premier lieu, rappelons quelques propriétés inhérentes à nos algorithmes :

- $\forall k,\ x_*^k$ est le point de meilleur estimé au début de l'itération $k : \forall x,\ f^k(x) \geq f^k(x_*^k)$,
- $\forall k,\ \sigma^{k+1}(x_*^k) < \sigma^k(x_*^k)$: l'estimé de la valeur de x_*^k est amélioré à chaque itération,
- $\forall k,\ \forall x \in \mathcal{P}^k,\ \sigma^{k+1}(x) < \sigma^k(x)$: De même pour tous les points candidats,
- $\forall k,\ \forall x,\ \sigma_{max} \geq \sigma^k(x) \geq 0$: L'écart-type du bruit est borné (par $\sigma_{max} = \lim_{r \to -\infty} \rho(r)$).

Pour simplifier la lecture des preuves à venir, on prendra $\sigma_{max} = 1$. Le cas général est totalement analogue et les calculs sont inchangés.

Il nous faut également établir quelques lemmes contrôlant l'imprécision. Ensuite, comme dans Audet et Hare (2017), on va montrer que les points considérés par *MADS* sont tous dans un ensemble borné. De là, la taille du treillis sera bornée supérieurement, et on pourra en déduire que sa limite inférieure sera 0. De là, une sous-suite raffinante des optimums estimés permettra de prouver que le point limite est un minimum Clarke-optimal de la fonction.

L'un des éléments essentiels pour traiter l'incertitude sera le résultat suivant :

Lemme 3.3.1 (Consistance de l'estimation du minimum d'un nombre fini de points)**.** *Soit E un ensemble de cardinal fini. Soit $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ et $\forall x \in E, f^i(x)$ l'estimateur de f construit par maximum de vraisemblance après i évaluations bruitées de la fonction en x. Soit $x_* \in \arg\min_{x \in E}(f(x))$. On a le résultat suivant :*

$$\mathbb{P}\left(\exists I \in \mathbb{N} : \forall i \geq I,\ x_* \in \arg\min_{x \in E}\left\{f^i(x)\right\}\right) = 1. \tag{3.1}$$

Autrement dit, après suffisamment d'évaluations des éléments de E, l'estimateur de f est minimisé en un point minimisant également la vraie fonction.

Démonstration. Notons $x_1, \ldots, x_e, x_*$ les éléments de E (qui est donc de cardinal fini $e+1$). On supposera pour alléger la preuve que x_* est l'unique point minimisant f dans E, c'est à dire qu'on a $\arg\min_{x \in E}(f(x)) = \{x_*\}$. On dénote par μ_i les écarts entre la valeur de f en x_i et

x_*, qui sont tous strictement positifs par définition (et unicité) de x_* :

$$\forall i \in [\![1, e]\!],\ \mu_i = f(x_i) - f(x_*) > 0.$$

On a de plus, par la loi forte des grands nombres :

$$\forall x \in \{x_1, \dots, x_e, x_*\},\ \mathbb{P}\left(f^k(x) \underset{k\to\infty}{\longrightarrow} f(x)\right) = 1,$$

de laquelle on peut déduire :

$$\begin{cases} \forall i \in [\![1, e]\!] & \mathbb{P}\left(\exists K_i \mid \forall k \geq K_i,\ f^k(x_i) > f(x_i) - \mu_i/2\right) = 1 \\ \forall \eta > 0 & \mathbb{P}\left(\exists K_{\eta_*} \mid \forall k \geq K_{\eta_*},\ f^k(x_*) < f(x_*) + \eta/2\right) = 1 \end{cases}. \tag{$*$}$$

Considérons $\eta = \min\{\mu_1, \dots, \mu_e\}$. Alors $K = \max\{K_1, \dots, K_e, K_{\eta_*}\}$ existe avec $\mathbb{P} = 1$, en tant que max d'un nombre fini de constantes existantes avec $\mathbb{P} = 1$. On a donc, via ($*$) :

$$\forall i \in [\![1, e]\!],\ \forall k \geq K,\ f^k(x_i) > f(x_i) - \mu_i/2 \geq {}^{2} f(x_*) + \eta/2 > f^k(x_*),$$

qui est bien équivalent au résultat voulu :

$$\forall k \geq K,\ x_* \in \arg\min_{x\in E}(f^k(x)).$$

□

On remarquera que ce lemme permet de déduire un résultat plus fort :

Corollaire 3.3.2 (Consistance de la relation d'ordre entre les estimés d'un nombre fini de points)**.** *Après suffisamment d'évaluations de f dans E, l'estimateur et la vraie fonction définissent la même relation d'ordre :* $\exists K \mid \forall k \geq K,\ \forall x, y,\ f(x) < f(y) \iff f^k(x) < f^k(y)$.

Démonstration. Notant $E = \{x_1, \dots, x_{e+1}\}$ avec les x_i rangés par ordre croissant de la vraie fonction f, il suffit d'appliquer récursivement le raisonnement de la démonstration 3.3.1 à E, puis $E \setminus \{x_1\}$, puis $E \setminus \{x_1, x_2\}$, …, jusqu'à $E \setminus \{x_1, \dots, x_{e-1}\}$. On obtient des constantes qu'on notera $K_0, \dots, K_e$. Ainsi la constante $K = \sum_{i=0}^{e} K_i$ convient pour notre résultat. □

Il est maintenant possible de démontrer la convergence de nos versions de *MADS* non-déterministes vers un optimum local de f. En premier lieu, nous nous assurerons que les algorithmes convergent. On a de manière analogue au cas déterministe le résultat suivant :

2. $f(x_i) - \mu_i/2 \geq f(x_*) + \eta/2 \iff \mu_i \geq \eta/2 + \mu_i/2 \iff \mu_i \geq \eta = \min\{\mu_k\}$, ce qui est toujours vrai.

Lemme 3.3.3 (Conservation de l'appartenance à un treillis)**.** *Soit $x \in \mathcal{X}$. Soit $\delta_m^k > 0$ et $\mathcal{M}_k(x) = \left\{x^k + \delta_m^k m \mid m \in \mathbb{Z}^n\right\}$. Alors une itération depuis x_*^k donne $x_*^{k+1} \in \mathcal{M}_k(x_*^k)$.*

Démonstration. Le résultat est évident en retenant qu'une itération de *MADS* n'évalue que des points de la sonde $\mathcal{P}^k$, incluse dans $\mathcal{M}_k(x_*^k)$, et que le point x_*^{k+1} retenu sera soit x_*^k (qui y appartient), soit l'un des points de $\mathcal{P}^k$ (qui y appartiennent), soit un point évalué dans l'une des itérations précédentes (qui, récursivement, y appartient également). □

Les lemmes suivants assurent que l'algorithme ne pourra pas passer un temps infini dans des zones de l'espace trop inintéressantes. Plus précisément, ils garantissent que le pas de sonde reste borné, et qu'il existe une boule de laquelle l'algorithme ne sortira pas. Le rayon de cette boule est aléatoire, et peut changer d'une exécution à l'autre, mais sera toujours fini.

Lemme 3.3.4 (Ensemble de niveau de f jamais dépassé par les optimums)**.** *Soit une fonction $f : \mathbb{R}^n \to \mathbb{R}$ dont les ensembles de niveau sont bornés, et $x_*^0 \in \mathcal{X}$. Toute exécution de MADS sur des évaluations bruitées de f générera une suite d'optimums estimés notée $(x_*^k)_{k\in\mathbb{N}}$. Il existe une boule centrée en 0 de laquelle ces optimums ne sortiront pas :*

$$\mathbb{P}\left(\forall \left(x_*^k\right)_{k\in\mathbb{N}},\ \exists M > 0 : \forall k \in \mathbb{N},\ x_*^k \in B_M(0)\right) = 1.$$

Démonstration. Puisque $\forall y \in \mathbb{R}$, l'ensemble de niveau y de f est borné, on peut trouver une boule de rayon fini contenant l'ensemble entier. Notamment, on définit :

$$\forall\ \alpha > 0, L_\alpha = \left\{x \in \mathbb{R}^n \mid f(x) \leq f(x_*^0) + \alpha\right\}, \tag{L_α}$$

$$\forall\ \alpha > 0, d_\alpha > 0 \mid L_\alpha \subset B_{d_\alpha}(0), \tag{d_α}$$

$$\forall k \in \mathbb{N},\ N^k = \min\{n \in \mathbb{N} \mid x_*^k \in L_n\}. \tag{N_k}$$

La démonstration revient à s'assurer que $N = \sup_{k\in\mathbb{N}} N^k$ est fini, puisqu'alors d_N conviendra.

Pour ce faire, considérons d'abord la probabilité suivante : $\mathbb{P}(x_*^{k+1} \notin L_n \mid x_*^k \in L_n)$, la probabilité que l'itération k trouve un point hors de L_n considéré comme optimal (car d'estimé le plus faible), sachant que $x_*^k \in L_n$. L'idée est de majorer cette probabilité par un terme dépendant de n, dont la somme sur n converge. La raison sera plus claire par la suite.

On peut noter que si tous les points évalués à l'itération k sont dans L_n, cette probabilité est nulle. On prendra donc la convention suivante dans le calcul à venir : si l'ensemble $\mathcal{P}^k \cap L_n^c = \emptyset$, l'expression calculée vaut 1. En d'autres termes :

$$\prod_{x\in\emptyset} A(x) = 1 \text{ et } \mathbb{P}(\forall x \in \emptyset,\ B(x)) = 1 \text{ avec } A \text{ une fonction et } B(x) \subset \Omega \text{ un évènement.}$$

De plus, on considérera se placer à une itération $k \geq \kappa$, où κ est tel que $f^{\kappa}(x_*^0) \leq f(x_*^0) + 1/2$. Cette constante κ existe et est finie, en vertu de la loi des grands nombres.

Ainsi, on peut maintenant manipuler la probabilité $\mathbb{P}(x_*^{k+1} \notin L_n \mid x_*^k \in L_n)$:

$$\begin{aligned}
\mathbb{P}\left(x_*^{k+1} \notin L_n \mid x_*^k \in L_n\right) &= \mathbb{P}\left(\exists x \in \mathcal{P}^k : x \notin L_n \text{ et } f^k(x) < f^k(x_*^k)\right) \\
&= 1 - \mathbb{P}\left(\forall x \in \mathcal{P}^k : x \in L_n \text{ ou } f^k(x) \geq f^k(x_*^k)\right) \\
&= 1 - \mathbb{P}\left(\forall x \in \mathcal{P}^k \cap L_n^c : f^k(x) \geq f^k(x_*^k)\right) \\
&\underset{f^k(.) \text{ indept}}{=} 1 - \prod_{x \in \mathcal{P}^k \cap L_n^c} \mathbb{P}\left(f^k(x) \geq f^k(x_*^k)\right) \\
&\underset{\eta \sim \mathcal{N}\left(0, \sigma^k(x)^2\right)}{=} 1 - \prod_{x \in \mathcal{P}^k \cap L_n^c} \mathbb{P}\left(f(x) + \eta \geq f^k(x_*^k)\right) \\
&\underset{f^k(x_*^k) \leq f^k(x_*^0)}{\leq} 1 - \prod_{x \in \mathcal{P}^k \cap L_n^c} \mathbb{P}\left(\eta \geq f^k(x_*^0) - f(x)\right) \\
&= 1 - \prod_{x \in \mathcal{P}^k \cap L_n^c} \left[1 - \Phi_{0,\sigma^k(x)}(f^k(x_*^0) - f(x))\right] \\
&\underset{\sigma^k(x) \leq \sigma_{max} \leq 1}{\leq} 1 - \prod_{x \in \mathcal{P}^k \cap L_n^c} \left[1 - \Phi(f^k(x_*^0) - f(x))\right] \\
&\underset{k \geq \kappa}{\leq} 1 - \prod_{x \in \mathcal{P}^k \cap L_n^c} \left[1 - \Phi(f(x_*^0) + 1/2 - f(x))\right] \\
&\underset{f(x) \geq f(x_*^0) + n}{\leq} 1 - \prod_{x \in \mathcal{P}^k \cap L_n^c} \left[1 - \Phi(-n + 1/2)\right] \\
&= 1 - \left[1 - \Phi(-n + 1/2)\right]^{Card(\mathcal{P}^k \cap L_n^c)} \\
&\leq 1 - \left[1 - \Phi(-n + 1)\right]^{Card(\mathcal{P}^k \cap L_n^c)}.
\end{aligned}$$

Ainsi, on peut déduire de cette expression que $\forall k \in \mathbb{N}$:

$$\begin{aligned}
\mathbb{P}\left(N^{k+1} > N^k\right) &\leq 1 - \left[1 - \Phi(-N^k + 1)\right]^{Card(\mathcal{P}^k \cap L_n^c)} \\
&\underset{Card(\mathcal{P}^k) = p = 2n}{\leq} 1 - \left[1 - \Phi(-N^k + 1)\right]^{p} \\
&= a_{N^k}.
\end{aligned}$$

Considérons alors le pire des cas : (N^k) est monotone croissante avec k. On a dans ce cas :

$$\begin{aligned}
\mathbb{P}\left(N^{k+1} > N^k\right) &\leq a_{N^k}, \\
\mathbb{P}\left(N^{k+1} = N^k\right) &\geq 1 - a_{N^k}.
\end{aligned}$$

Notons alors K l'ensemble des itérations k telles que $N^{k+1} > N^k$. De là, on va conclure via le lemme de Borel-Cantelli. Puisqu'il donne l'implication suivante :

$$\sum_{k \in K} a_{N^k} < +\infty \Rightarrow \mathbb{P}\left(\limsup_{k \in \mathbb{N}} N^k = +\infty\right) = 0,$$

il suffit de prouver que la somme des $(a_k)_{k\in\mathbb{N}}$ converge. Or, en majorant le terme général a_k :

$$\begin{aligned}
\forall k \in \mathbb{N}^*,\ a_k &= 1 - \mathbb{P}\left(\eta \geq -k+1 \mid \eta \sim N(0,1)\right)^p \\
&= 1 - \left(\int_{u=-k+1}^{\infty} \frac{\exp\{-u^2/2\}}{\sqrt{2\pi}} du\right)^p \\
&= 1 - \int_{\vec{u}\in[-k+1;+\infty[^p} \frac{\exp\{-\left\|\vec{u}\right\|_2^2/2\}}{(2\pi)^{p/2}} d\vec{u} \\
&\leq \int_{\vec{u}\notin B_{k-1}(0)} \frac{\exp\{-\left\|\vec{u}\right\|_2^2/2\}}{(2\pi)^{p/2}} d\vec{u} \\
&= \int_{r=k-1}^{\infty} \frac{\exp\{-r^2/2\}}{(2\pi)^{p/2}} \times Vol(S_{p-1}(r)) dr \\
&\underset{p=2n,\text{ pair}}{=} \int_{r=k-1}^{\infty} \frac{\exp\{-r^2/2\}}{(2\pi)^{p/2}} \frac{\pi^{p/2} r^p}{(p/2)!} dr \\
&\leq \frac{1}{2^{p/2}\,(p/2)!} \int_{r=k/2}^{\infty} \exp\left\{-r^2/2\right\} r^p dr \\
&\underset{u=r-k/2}{=} \frac{\exp\left\{-k^2/8\right\}}{2^{p/2}\,(p/2)!} \int_{u=0}^{\infty} \exp\left\{-u^2/2\right\} \exp\{-uk/2\} \left(u+\frac{k}{2}\right)^p du \\
&\underset{\exp\{-uk/2\}\leq 1}{\leq} \frac{\exp\left\{-k^2/8\right\} (k/2)^p}{2^{p/2}\,(p/2)!} \int_{u=0}^{\infty} \exp\left\{-u^2/2\right\} \left(\frac{u}{k/2}+1\right)^p du \\
&\underset{u/k/2\leq u}{\leq} \exp\left\{\frac{-k^2}{8}\right\} k^p \times \frac{1}{2^{3p/2}\,(p/2)!} \int_{u=0}^{\infty} \exp\left\{-u^2/2\right\} (u+1)^p\, du \\
&= \exp\left\{\frac{-k^2}{8}\right\} k^p \times (cste < +\infty),
\end{aligned}$$

Il apparaît clairement que a_k est sommable, et donc que la somme sur $k \in \mathbb{N}$ des a_k converge. On peut donc affirmer que $\sum_{k\in K} a_{N^k}$ converge et conclure via le lemme de Borel-Cantelli. □

Ce résultat peut notamment servir à démontrer que le pas du treillis reste borné lui aussi.

Lemme 3.3.5 (Existence d'une borne supérieure à la taille de treillis δ_m^k)**.** *Soit une fonction $f : \mathbb{R}^n \to \mathbb{R}$ dont les ensembles de niveau sont bornés, et $x_*^0 \in \Omega$. Une exécution de MADS sur des évaluations bruitées de f, partant de x_*^0, ne peut créer de treillis de pas tendant vers $+\infty$. Autrement dit, en notant δ_m^k la suite des pas de treillis créés,*

$$\mathbb{P}\left(\exists\ \delta_m^{sup} : \forall k \in \mathbb{N},\ \delta_m^k \leq \delta_m^{sup}\right) = 1.$$

Démonstration. Soit $M \mid \forall k \in \mathbb{N},\ x_*^k \in B_M(0)$. Cette constante existe d'après le lemme 3.3.4.

On a donc $\forall k \in \mathbb{N},\ \delta_m^k > 2M \Rightarrow \mathcal{M}_k(x^k) \cap B_M(0) = \left\{x_*^k\right\}$. Ainsi, à l'itération k on aura $\mathcal{P}^k \cap B_M(0) = \emptyset$, et donc tous les points évalués ne peuvent devenir des optimums estimés (par définition, si l'un d'eux le devenait on aurait un x_*^k hors de $B_M(0)$). Cela signifie que l'itération est nécessairement un échec. Donc à cette itération, $\delta_m^{k+1} \leq \delta_m^k$ puisque l'inverse n'est possible que si l'itération est un succès. Ainsi, la suite des δ_m^k ne peut devenir arbitrairement grande. □

On peut alors démontrer le théorème suivant :

Théorème 3.3.6. *Soit $f : \mathbb{R}^n \to \mathbb{R}$ dont tous les ensembles de niveau sont bornés. Alors, $\forall x_*^0 \in \mathcal{X}$, toute exécution de MADS vérifiera :*

$$\liminf_{k\to\infty} \delta_m^k = 0. \tag{3.2}$$

Démonstration. Supposons que $\exists \delta_m^{inf} > 0 \mid \forall k \in \mathbb{N},\ \delta_m^k \geq \delta_m^{inf}$. Rappelons de plus que par construction, la suite des δ_m^k contient exclusivement des multiplications ou divisions par 2 de δ_m^0, ce qui implique que

$$\forall k \in \mathbb{N},\ \exists s^k \in \mathbb{Z} \mid \delta_m^k = 2^{s^k}\delta_m^0.$$

Comme de plus cette suite est bornée supérieurement (via le lemme 3.3.5), on a alors

$$\exists\ s^{inf}, s^{sup} \in \mathbb{Z}^2 \mid \forall k \in \mathbb{N},\ \delta_m^{inf} = 2^{s^{inf}}\delta_m^0 \leq \delta_m^k = 2^{s^k}\delta_m^0 \leq 2^{s^{sup}}\delta_m^0 = \delta_m^{sup}.$$

On peut de plus déduire, via le lemme 3.3.3, que $\forall k \in \mathbb{N},\ x_*^k \in \mathcal{M}_{\delta_m^{inf}}(x_*^0)$. Considérons alors l'ensemble $B_{M+\delta_m^{sup}}(0)$, ensemble qui contient tous les points évalués par *MADS* lors de son exécution. L'ensemble $\mathcal{M}_{\delta_m^{inf}}(x_*^0) \cap B_{M+\delta_m^{sup}}(0)$ est de cardinal fini, et *MADS* n'évaluera pas de point hors de cet ensemble. D'après le lemme 3.3.1, il existe donc un nombre κ d'itérations au-delà duquel les estimés de f en tous les points suivent nécessairement la même relation d'ordre que la vraie fonction. Ainsi, pour toute itération d'indice $k \geq \kappa$, les images par f^k de tous les points examinés durant l'itération k (les points de $\mathcal{P}^k \cup \{x_*^k\}$) suivent la même relation d'ordre que les estimés f. Ainsi, l'itération k sera nécessairement un échec, ce qui implique que $\forall k \geq \kappa,\ \delta_m^{k+1} = \delta_m^k/2$, contredisant ainsi l'hypothèse initiale. □

Pour finaliser la preuve, on aura besoin de la notion de *sous-suite raffinante* :

Définition 3.3.7. *On nommera sous-suite raffinante de $(x_*^k)_{k\in\mathbb{N}}$ une suite extraite $(x_*^k)_{k\in K}$ qui est raffinante (au sens donné en 1.1.2). C'est-à-dire une sous-suite convergente (vers une limite notée x_*) et telle que $\lim_{k\in K}\delta_m^k = 0$. On désignera de plus par direction raffinante d toute*

direction pour laquelle $\exists L \subset K \mid \forall k \in L,\ \exists d^k \in \mathcal{D}^k$ *et que cette suite vérifie* $\lim_{k\in L} \frac{d^k}{\|d^k\|_2} = \frac{d}{\|d\|_2}$.

La dernière étape de la preuve de convergence tient enfin dans le résultat suivant :

Théorème 3.3.8. *Soit f une fonction déterministe dont les ensembles de niveau sont bornés, bruitée par une erreur aléatoire normale centrée, à écart-type réglable. Soit (x_*^k) la suite des minimums estimés par MADS sur f. On peut en extraire une sous-suite raffinante vers x_* et une certaine direction d, qu'on notera toujours (x_*^k), (δ_m^k) et (d^k). On a le résultat suivant :*

$$\mathbb{P}\left(f^\circ\left(x_*; d\right) \geq 0\right) = 1.$$

En d'autres termes, le point optimum trouvé par MADS après un temps infini est, de façon presque sûre, un minimum local Clarke-optimal de la fonction déterministe non-bruitée.

Démonstration. Rappelons d'abord que :

$$\begin{aligned}
f^\circ(x_*; d) &= \limsup_{x\to x_*,\ t\searrow 0} \frac{f(x+td)-f(x)}{t} \\
&= \limsup_{x\to x_*,\ t\searrow 0,\ v\to d} \frac{f(x+tv)-f(x)}{t} \text{ (Audet et Dennis, Jr. (2006), propriété 3.9)} \\
&\geq \limsup_{k\to+\infty} \frac{f(x_*^k+\delta_m^k d^k)-f(x_*^k)}{\delta_m^k \|d^k\|_2} \\
&= \limsup_{k\to+\infty} \frac{f^k(x_*^k+\delta_m^k d^k)-\eta^k(x_*^k+\delta_m^k d^k)-f^k(x_*^k)+\eta^k(x_*^k)}{\delta_m^k \|d^k\|_2},
\end{aligned}$$

où, ici, $\eta^k(.)$ est la valeur de l'erreur entre l'estimé $f^k(.)$ et $f(.)$ à l'itération k.

Par la suite, on notera également

$$\forall k,\ \mu^k = f^k(x_*^k+\delta_m^k d^k) - f^k(x_*^k) \geq 0.$$

On a $\mu^k \geq 0\ \forall k$ car les indices k considérés dans la sous-suite raffinante sont des indices d'échec d'itération.

Ainsi, on a :

$$\begin{aligned}
\mathbb{P}\left[f^\circ(x_*; d) \geq 0\right] &\geq \mathbb{P}\left[\limsup_{k\to+\infty} \frac{\mu^k-\eta^k(x_*^k+\delta_m^k d^k)+\eta^k(x_*^k)}{\delta_m^k \|d^k\|_2} \geq 0\right] \\
&= \mathbb{P}\left[\lim_{k\to+\infty} \sup_{n\geq k} \mu^k-\eta^k(x_*^k+\delta_m^k d^k)+\eta^k(x_*^k) \geq 0\right].
\end{aligned}$$

Or, pour une suite $(A_{n,k})$ de réels donnée,

$$\left(\lim_{k\to+\infty}\sup_{n\geq k} A_{n,k}\right) \geq 0 \iff \forall(k,\epsilon)\in\mathbb{N}\times\mathbb{R}^{+^*},\ \exists n\geq k \mid A_{n,k}\geq -\epsilon$$
$$\text{d'où}\quad \left(\lim_{k\to+\infty}\sup_{n\geq k} A_{n,k}\right) < 0 \iff \exists(k,\epsilon)\in\mathbb{N}\times\mathbb{R}^{+*} \mid \forall n\geq k,\ A_{n,k}<-\epsilon.$$

En appliquant ceci à $\mathbb{P}\left[f^\circ(x_*;d)\geq 0\right]$, on obtient alors :

$$\begin{aligned}
&= \mathbb{P}\left[\forall(k,\epsilon)\in\mathbb{N}\times\mathbb{R}^{+*},\ \exists n\geq k : \mu^n-\eta^n(x_*^n+\delta_m^n d^n)+\eta^n(x_*^n)\geq-\epsilon\right]\\
&= 1-\mathbb{P}\left[\exists(k,\epsilon)\in\mathbb{N}\times\mathbb{R}^{+*} : \forall n\geq k,\ \eta^n(x_*^n)<-\epsilon-\mu^n+\eta^n(x_*^n+\delta_m^n d^n)\right]\\
&\geq 1-\mathbb{P}\left[\exists(k,\epsilon)\in\mathbb{N}\times\mathbb{R}^{+*} : \forall n\geq k,\ \eta^n(x_*^n)<-\epsilon-0+\eta^n(x_*^n+\delta_m^n d^n)\right].
\end{aligned}$$

On sait de plus que les erreurs $\eta^k(.)$ suivent une loi connue : $\mathcal{N}\left(0,\sigma^k(.)^2\right)$, avec $\sigma^k(.)\leq 1$.
Or, $\forall\epsilon\in\mathbb{R}^{+*},\forall(\sigma_1,\sigma_2)\in]0,1[^2$:

$$\mathbb{P}\left[\eta_1<\eta_2-\epsilon \mid \eta_1\sim\mathcal{N}\left(0,\sigma_1^2\right),\ \eta_2\sim\mathcal{N}\left(0,\sigma_2^2\right)\right]=\Psi(\epsilon,\sigma_1,\sigma_2)\leq\Psi(\epsilon,1,1)<1.$$

Ainsi, $\forall\epsilon>0,\ \forall k$:

$$\begin{aligned}
&\mathbb{P}\left[\forall n\geq k,\ \eta^n(x_*^n)<-\epsilon+\eta^n(x_*^n+\delta_m^n d^n)\right]\\
=\ &\prod_{n=k}^{\infty}\mathbb{P}\left[\eta^n(x_*^n)<-\epsilon+\eta^n(x_*^n+\delta_m^n d^n)\right]\\
=\ &\prod_{n=k}^{\infty}\Psi\left(\epsilon,\sigma^n(x_*^n),\sigma^n(x_*^n+\delta_m^n d^n)\right)\\
\leq\ &\prod_{n=k}^{\infty}\Psi\left(\epsilon,1,1\right)\\
=\ &0.
\end{aligned}$$

De fait,

$$\mathbb{P}\left[\exists(k,\epsilon)\in\mathbb{N}\times\mathbb{R}^{+*} \mid \forall n\geq k,\ \eta^n(x_*^n)<-\epsilon+\eta^n(x_*^n+\delta_m^n d^n)\right]=0.$$

Nous permettant donc de conclure, comme souhaité, que

$$\mathbb{P}\left[f^\circ(x_*;d)\geq 0\right]=1$$

□

Enfin, la construction des points d'évaluation à chaque itération (proposée par Audet et Hare (2017)) assure que l'algorithme crée un ensemble de directions raffinantes dense dans la sphère

unité de $\mathbb{R}^n$. En utilisant encore le résultat 3.9 de Audet et Dennis, Jr. (2006), on peut alors affirmer que 3.3.8 est vrai pour toute direction de la sphère unité de $\mathbb{R}^n$. D'où le résultat final :

Théorème 3.3.9 (Garantie de convergence vers un minimum local)**.** *Soit f une fonction déterministe dont les ensembles de niveau sont bornés, bruitée par une erreur aléatoire normale centrée, à écart-type réglable. Soit x_* un optimum estimé, généré via une sous-suite raffinante d'optimums proposés par MADS.*

$$\forall d \in \textit{sphère unité de } \mathbb{R}^n,\ \mathbb{P}\left[f^\circ\left(x_*; d\right) \geq 0\right] = 1.$$

Les deux variantes de *MADS* proposées dans ce chapitre (algorithmes 11 et 9) ont donc des propriétés de convergence analogues, et identiques à celle de l'algorithme déterministe dont elles s'inspirent (5). Elles ont cependant des mécaniques internes différentes, pouvant générer des variations de performances dans les cas pratiques.

Notons toutefois que la méthode à précision monotone requiert que le bruit imposé aux estimations puisse devenir aussi faible que désiré, car la diminution de δ_p fait de telle sorte que δ_p reste $\leq \mu(\sigma^k)^\alpha$, mais tout en restant aussi grand que possible. Ainsi, si σ^k est borné inférieurement par une constante $\sigma_{min} > 0$, le pas de sonde ne tendra pas vers 0. Ceci ne peut être corrigé qu'en implémentant une mécanique analogue à la première version de l'algorithme de Polak et Wetter (7), qui diminuerait δ_p à chaque échec.

Pour ces raisons, il sera alors bon de comparer nos deux algorithmes sur quelques problèmes conçus pour illustrer les difficultés que chacune peut naturellement surpasser, ou au contraire qui auront tendance à poser des problèmes à certaines méthodes.

CHAPITRE 4 COMPORTEMENTS NUMÉRIQUES

Ce chapitre a pour objectif de présenter divers problèmes conçus pour illustrer les différences entre les algorithmes, dans leur fonctionnement réel et leurs spécificités théoriques. Nous proposons ici cinq problèmes défiant les algorithmes sur leur capacité à converger (en section 4.2), leur façon de surpasser les obstacles durant l'optimisation (4.3), et enfin sur une famille de problèmes plus complexes (4.4).

Dans un contexte de précision variable, la façon d'analyser et de comparer des algorithmes diffère des techniques utilisées dans les cas déterministes. Les profils usuels de performance de Moré et Wild (2009), par exemple, perdent de leur pertinence puisqu'un nombre d'évaluations de la boîte noire n'est plus significatif de la qualité d'un algorithme : peu d'évaluations mais à grand coût peuvent être moins efficaces que beaucoup d'évaluations à une grande imprécision. Nous proposerons donc en premier lieu (en section 4.1) des stratégies de comparaison de performances d'algorithmes dans des contextes stochastiques.

4.1 Techniques de comparaison d'algorithmes en contexte bruité

Certains des problèmes proposés dans ce chapitre n'ont pas vocation à représenter des boîtes noires réelles, au sens où ils ne cherchent pas à reproduire les temps de calcul requis par les problèmes réels. De fait, le bruit stochastique des cas pratiques (venant par exemple de simulations de Monte-Carlo) est ici remplacé par un bruit artificiellement ajouté à la vraie valeur de la fonction : au lieu d'une simulation Monte-Carlo, nos problèmes calculeront la vraie valeur analytique $f(x)$ et lui ajouteront un bruit normal artificiel $\mathcal{N}\left(0, \sigma^2\right)$ pour renvoyer la valeur bruitée $f(x) + \epsilon$, avec ϵ une observation de $\mathcal{N}\left(0, \sigma^2\right)$.

Cependant, ce changement n'affecte pas l'information que les algorithmes pourront tirer de l'observation des valeurs bruitées. Par analogie avec les simulations de Monte-Carlo (1.1), on parlera de nombre équivalent de tirages Monte-Carlo lorsqu'on évaluera la fonction à un bruit d'amplitude σ donné : un bruit artificiel $\mathcal{N}\left(0, \sigma^2\right)$ est statistiquement équivalent à une simulation Monte-Carlo à $N = cste/\sigma^2$ tirages. Ce nombre de tirages équivalent nous donnera une statistique simple à sommer, et dont la valeur nous renseignera sur l'effort de calcul réel qu'aurait coûté l'évaluation si on avait utilisé des véritables simulations de Monte-Carlo au lieu d'un bruit normal artificiel.

Grâce à cette statistique (ou au véritable nombre de tirages Monte-Carlo utilisés, si c'est possible), on peut comparer les performances des algorithmes via divers éléments. Notamment,

on se propose d'utiliser les profils suivants :

Définition 4.1.1 (Profil de résultats)**.** *Pour tout problème P, on lance p exécutions de k algorithmes dans les mêmes conditions. On obtient alors $(f_1, \dots, f_p)_1, \dots, (f_1, \dots, f_p)_k$ les p valeurs optimales proposées par chaque algorithme ($f_{i,j}$ est l'optimum suggéré par la i^{eme} exécution du j^{eme} algorithme) et $(s_1, \dots, s_p)_1, \dots, (s_1, \dots, s_p)_k$ le nombre total de tirages Monte-Carlo utilisés pour aboutir à ces solutions.*

Un profil de résultats sur le problème P est un nuage de points $(s_{i,j}, f_{i,j})$, $\forall i \in [\![1, p]\!]$, $\forall j \in [\![1, k]\!]$ représentant les solutions proposées par chaque exécution de chaque algorithme, en fonction de l'effort de calcul requis pour y parvenir.

Définition 4.1.2 (Profil de convergence en tirages Monte-Carlo)**.** *Pour tout problème P, on lance p exécutions de k algorithmes dans les mêmes conditions. On obtient alors, pour chaque exécution de chaque algorithme, la suite (f^k, s^k) des optimums courants proposés durant l'optimisation, et le nombre total de tirages Monte-Carlo consommés pour s'y rendre.*

Un profil de convergence en tirages Monte-Carlo sur un problème P est un ensemble de courbes (une par exécution d'un algorithme) représentant f^k en fonction de s^k.

Définition 4.1.3 (Profil de consommation instantanée)**.** *Pour tout problème P, on lance p exécutions de k algorithmes dans les mêmes conditions. On obtient alors, pour chaque exécution de chaque algorithme, la suite (f^k, r^k) des optimums courants trouvés durant l'optimisation, et la précision imposée aux évaluations à chaque itération.*

Un profil de consommation instantanée sur un problème P est un ensemble de courbes (une par exécution d'un algorithme) représentant $\rho(r^k)$ en fonction de f^k.

Définition 4.1.4 (Profil de précision instantanée)**.** *Pour tout problème P, on lance p exécutions de k algorithmes dans les mêmes conditions. On obtient alors, pour chaque exécution de chaque algorithme, la suite (f^k, σ^k) des optimums courants trouvés durant l'optimisation, et l'écart-type de ces estimés courants.*

Un profil de précision instantanée sur un problème P est un ensemble de courbes (une par exécution d'un algorithme) représentant σ^k en fonction de f^k.

On utilisera également une version modifiée des profils de performance (de Dolan et Moré (2002)), de données (de Moré et Wild (2009)) et de qualité (de Beiranvand *et al.* (2017)). Au lieu d'un nombre d'appels à la boîte noire, l'unité discriminant les algorithmes sera le nombre de tirages Monte-Carlo utilisés. On utilisera la forme de f non bruitée lorsqu'elle est connue. Sinon, on se contentera des valeurs des estimés f^k. L'annexe B détaille ces adaptations.

4.2 Comportements comparés en convergence : boîte noire «Norm2»

Cette section introduit un problème très simple à résoudre, dont l'analyse nous permettra de saisir les différences fondamentales dans la façon dont les algorithmes pilotent leurs convergences autour d'une solution. Ici, cette solution sera même optimale.

Présentation du problème

Ce problème ne présente aucune difficulté théorique, et peut être utilisé pour comparer la vitesse de convergence de différents algorithmes vers l'unique optimum que la fonction objectif convexe présente. La boîte noire prend ici deux variables x et y, et calcule la norme euclidienne du vecteur (x, y). Dans le cas non-bruité, on cherche à résoudre :

$$\min_{(x,y)\in\mathbb{R}^2} \quad \|X\|_2$$
$$\text{assujetti à} \quad \begin{cases} (x_0, y_0) &= \left(\pi^2, e^2\right) \\ X &= (x, y) \end{cases}$$

dont l'unique solution optimale est $(x_*, y_*) = (0, 0)$.

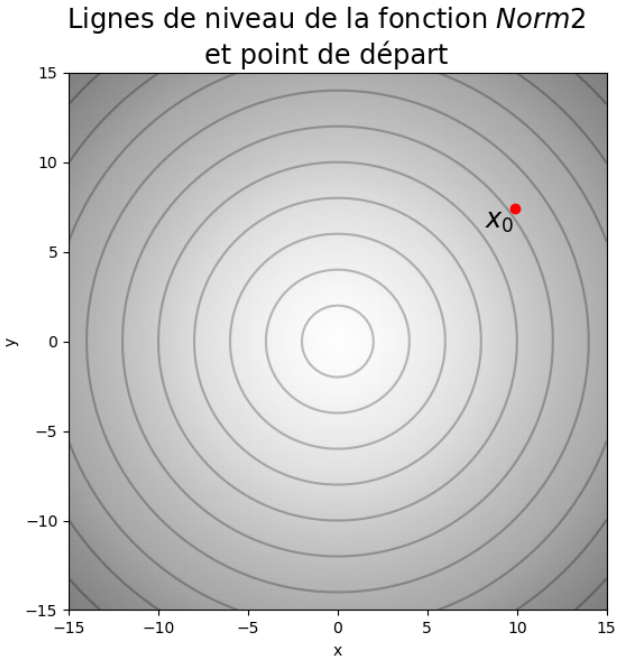

Figure 4.1 Fonction «Norm2», et point X_0.

Nous en proposerons également une version bruitée :

$$\min_{(x,y)\in\mathbb{R}^2} \quad \lim_{k\to\infty|\sigma^k(X)\searrow 0} f^k(X)$$
$$\text{assujetti à} \quad \begin{cases} (x_0, y_0) &= \left(\pi^2, e^2\right) \\ X &= (x, y) \\ f^k \text{ et } \sigma^k \text{ définis en 1.2.3 via} \\ f_\sigma(.) = \|.\|_2 + \mathcal{N}\left(0, \sigma^2\right) \end{cases}$$

où l'amplitude σ du bruit sera réglable, équivalente en tirages de Monte-Carlo à $1/\sqrt{\sigma}$.

L'objectif de ce problème, de par sa nature très simple, n'est pas d'étudier la *capacité* des algorithmes à converger. Il est acquis que tous vont localiser l'optimum et réussir à s'en approcher. L'intérêt du problème réside dans l'observation du *coût* de cette convergence : les algorithmes parviennent-ils à s'approcher de l'origine sans augmenter démesurément leur consommation de tirages Monte-Carlo ? Quel est l'effet de chacune des deux stratégies, et laquelle doit-on préférer si on dispose d'un budget limité ? Ou, au contraire, si on privilégie l'exactitude de la solution sans se soucier du coût ?

Résultats numériques

À l'issue de 10 exécutions de chacune des deux stratégies (*MADS* à précision monotone (9) et dynamique (11)), les résultats montrent comme prévu que toutes ont convergé :

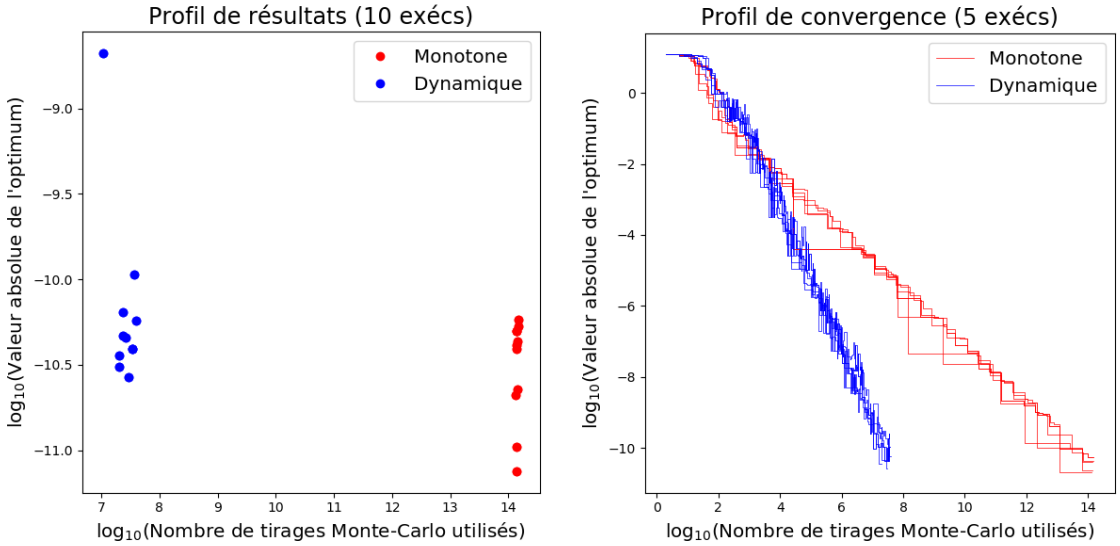

Figure 4.2 «Norm2» - Profils de (a) Résultats. (b) Convergence Monte-Carlo.

On remarque de plus que la stratégie monotone a de meilleures garanties de résultats, mais à un coût bien plus élevé, que la stratégie dynamique. Cela est confirmé par les autres profils :

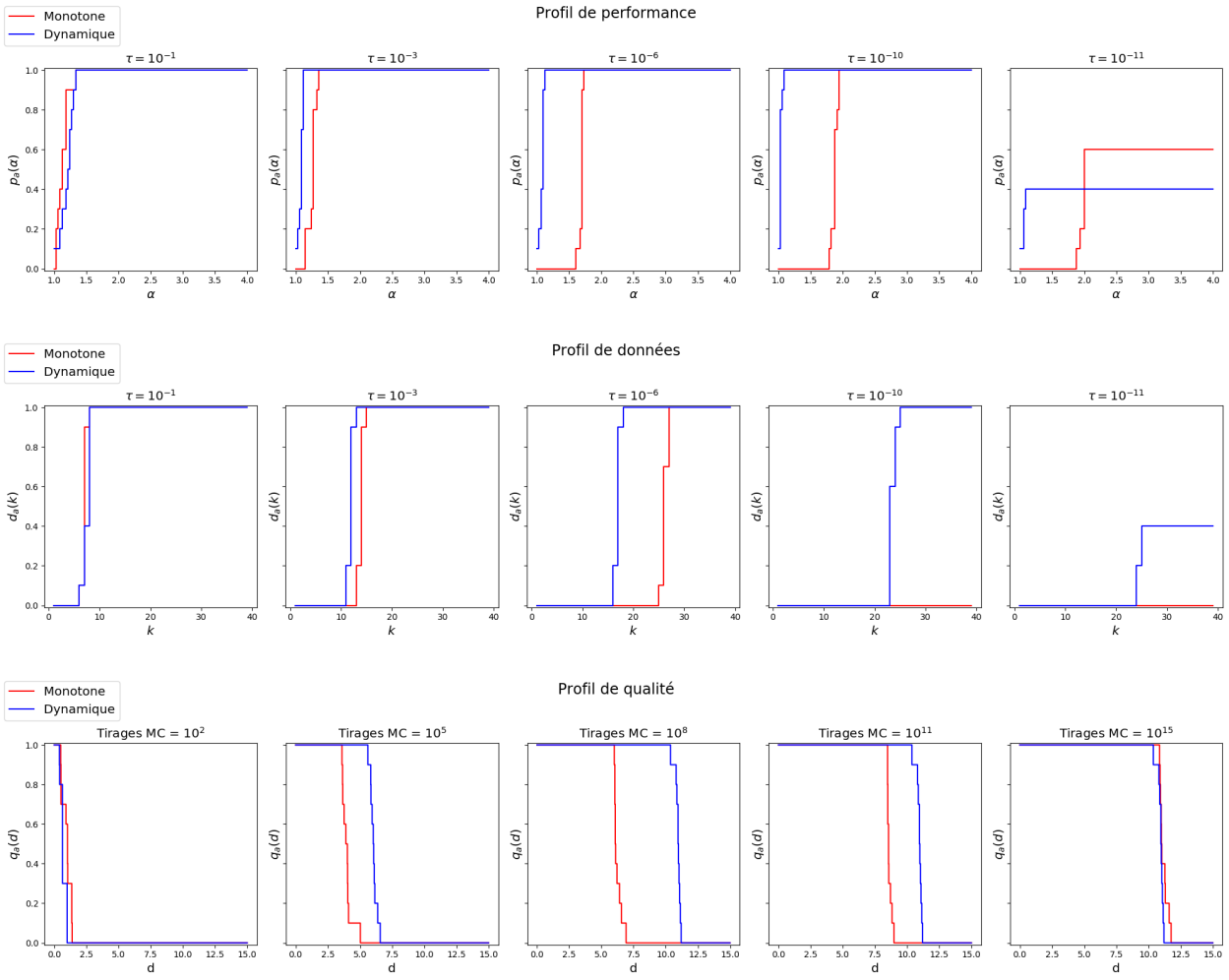

Figure 4.3 «Norm2» - Profils de (a) Performances. (b) Données. (c) Qualité.

Le profil de performance prouve que les deux algorithmes convergent, même pour des tolérances τ faibles. Cependant, plus τ diminue, plus la stratégie monotone requiert de tirages. Cela prouve sa surconsommation par rapport à la stratégie dynamique. Cependant, la monotone est plus performante lorsque la tolérance devient extrêmement faible. Le profil de données est, pour ceci, encore plus parlant. La courbe de la stratégie monotone s'effondre lorsque τ diminue, là où celle de la stratégie dynamique n'atteint plus les 100 % de réussite qu'à partir d'une tolérance très faible. Enfin, le profil de qualité montre qu'il faut accorder beaucoup de tirages à la stratégie monotone pour qu'elle égale la dynamique, mais qu'elle est alors capable d'aller un peu plus proche de l'optimum sous un budget conséquent.

4.3 Comportements comparés en exploration : obstacles durant l'optimisation

Ici, l'objectif sera d'étudier empiriquement comment les algorithmes évoluent durant leur phase de recherche d'un optimum, en phase d'exploration. Notamment, lorsqu'ils rencontrent une difficulté à surmonter. Réussissent-ils à s'en échapper sans intensifier inutilement autour de la solution courante ? Comment leur gestion de l'aléatoire peut compliquer cette extraction ? Quelles sont les conséquences de ce contretemps sur la suite de l'optimisation ?

4.3.1 Plateaux (sans contraintes) : Boîte noire «Quasi-Stairway»

Ce premier problème est trivial en contexte déterministe, mais illustrera l'importance d'une bonne gestion de l'incertitude en contexte bruité.

Présentation du problème

La fonction f à minimiser est définie comme suit, pour un paramètre $\lambda \in]1,2[$ donné :

$$\begin{array}{rcccl} \theta & : & x \in [1,+\infty) & \rightarrow & \theta(x) = \max\{m \in \mathbb{N} \mid 2^m \leq x\} = \lfloor \log_2(x) \rfloor \\ f_\lambda & : & x \in [1,+\infty) & \rightarrow & \left\{ \begin{array}{cl} -1 & \text{si } x = 1 \\ f(2^{\theta(x)}) - \log\left(\dfrac{x}{2^{\theta(x)}}\right) & \text{si } x \in (2^{\theta(x)}, \lambda 2^{\theta(x)}] \\ f(2^{\theta(x)+1/2}) - \left(x - \lambda 2^{\theta(x)}\right) & \text{si } x \in (\lambda 2^{\theta(x)}, 2^{\theta(x)+1}] \end{array} \right. \end{array}$$

dont on peut donner une forme analytique directe, $\forall N \in \mathbb{N}$:

$$f_\lambda : x \in [2^N, 2^{N+1}) \rightarrow \left\{ \begin{array}{lll} -1 - N\log(\lambda) - (2-\lambda)\,2^N\left(1 - 2^{-N}\right) & \text{si} & x = 2^N \\ f\left(2^N\right) - \log\left(\dfrac{x}{2^N}\right) & \text{si} & \dfrac{x}{2^N} \in (1, \lambda] \\ f\left(2^N\right) - \log(\lambda) - 2^N\left(\dfrac{x}{2^N} - \lambda\right) & \text{si} & \dfrac{x}{2^N} \in (\lambda, 2) \end{array} \right. .$$

L'intérêt de cette fonction est sa décroissance de -1 vers $-\infty$ sur l'espace des variables $[1,+\infty)$. La fonction non bruitée est donc très simple à minimiser : il suffit de maximiser la variable x. On pourra ainsi comparer la capacité des algorithmes à aller explorer l'infini, depuis le point $x_0 = 1$ et malgré l'incertitude entourant chaque évaluation.

Le nom «Quasi-Stairway» paraîtra sûrement plus clair au vu du graphe de la fonction, puisqu'on peut observer une alternance entre deux comportements : des zones très plates, où la fonction est quasi-constante, suivies de pentes abruptes dans lesquelles elle décroît plus fortement. On notera que λ influe sur la largeur des plateaux (figure 4.4) :

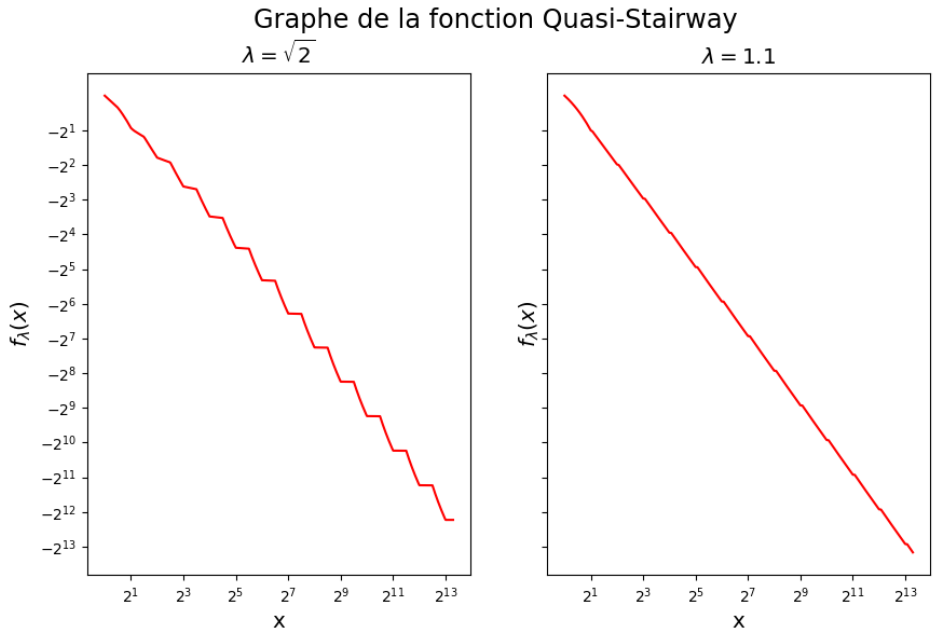

Figure 4.4 Fonctions «Quasi-Stairway» pour deux valeurs de λ.

Les deux problèmes (bruités) que l'on soumet aux algorithmes sont alors :

$$\min_{(x)\in[1;\infty)} \lim_{k\to\infty | \sigma^k(x)\searrow 0} f_\lambda^k(x)$$

$$\text{assujetti à} \begin{cases} x_0 = 1 \\ f_\lambda^k \text{ et } \sigma^k \text{ tels qus définis en 1.2.3} \\ 10^{12} \text{ tirages Monte-Carlo au maximum} \\ \text{Observer } f_\sigma(.) \text{ coûte } \dfrac{1}{\sqrt{\sigma}} \text{ tirages : l'estimé via } N \text{ tirages suit } \mathcal{N}\left(f(.), \dfrac{1}{N}\right) \\ \forall k,\ \delta_p^k \leq 1 \text{ : le cadre de sonde reste petit} \\ \lambda = \sqrt{2} \text{ et } \lambda = 1.1 \text{ (deux problèmes distincts)} \end{cases}$$

Ces fonctions vont permettre d'évaluer l'intérêt d'un pilotage dynamique et non monotone de la précision. Lorsque l'algorithme arrive sur l'un des plateaux, il est nécessaire d'utiliser une grande précision pour compenser le fait que la fonction est presque constante (puisque l'amplitude σ du bruit entre $f^k(x)$ et $f^k(y)$ doit vérifier $|f(y) - f(x)| \gg \sigma$ pour que la relation d'ordre entre les estimés respecte certainement celle de la vraie fonction). Ainsi, la traversée d'un plateau va coûter très cher aux algorithmes, qui vont augmenter grandement leurs précisions pour s'en échapper. Cependant, hors des plateaux la fonction varie plus fortement, avec une pente unitaire grâce à laquelle un bruit aléatoire plus important peut être toléré. Un algorithme capable de saisir cette information pourra relâcher sa précision et s'épargner une surconsommation inutile de tirages Monte-Carlo, et ainsi aller plus loin

pour un budget maximal de tirages fixé (à 10^{12} ici). En contrepartie, la stratégie dynamique exploitant cette remarque impose de nombreuses réévaluations des précédents points dont la valeur par f est proche de celle de l'optimum courant. Ainsi, sur les plateaux, cela va générer une grande consommation de tirages car tous les points ont des images très proches. Il y a donc une comparaison intéressante à faire entre les forces et faiblesses de chaque méthode sur un problème comme «Quasi-Stairway», qui mélange en une seule fonction les comportements mettant en lumière ou en difficulté chacun des deux algorithmes.

Enfin, il est bon de rappeler que l'une des forces de *GPS* (4) et *MADS* (5) est d'augmenter la taille de leur treillis pour survoler de tels plateaux. Avec cet ajout, même les algorithmes stochastiques pourraient éviter de stagner dans les zones où la fonction est quasi-constante. Cependant, puisque l'objectif de ce problème est d'illustrer les difficultés que de tels plateaux amènent, on forcera ici la taille δ_p des treillis à rester petite ($\delta_p \leq 1$).

Résultats numériques

Pour le problème difficile (à $\lambda = \sqrt{2}$), les résultats sont en faveur de la stratégie monotone :

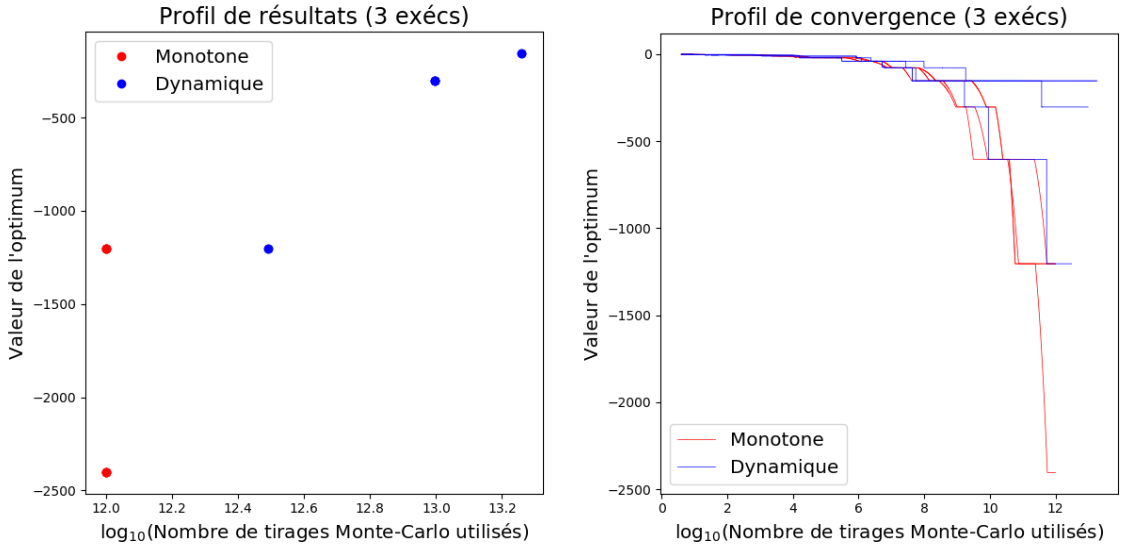

Figure 4.5 «$\sqrt{2}$-Quasi-Stairway» - Profils de (a) Résultats. (b) Convergence Monte-Carlo.

On y observe en effet que la stratégie monotone domine sur les deux aspects : qualité de la solution proposée et nombre de tirages consommés. Cela peut se comprendre en observant que la stratégie dynamique a besoin d'un grand nombre de réévaluations de ses anciennes solutions pour avancer. Il y a donc une surconsommation de tirages lors de la traversée des plateaux,

car la méthode dynamique y paie le prix de la relativement grande incertitude qu'elle laisse derrière elle lors de son optimisation. Les autres profils sont ici moins intéressants :

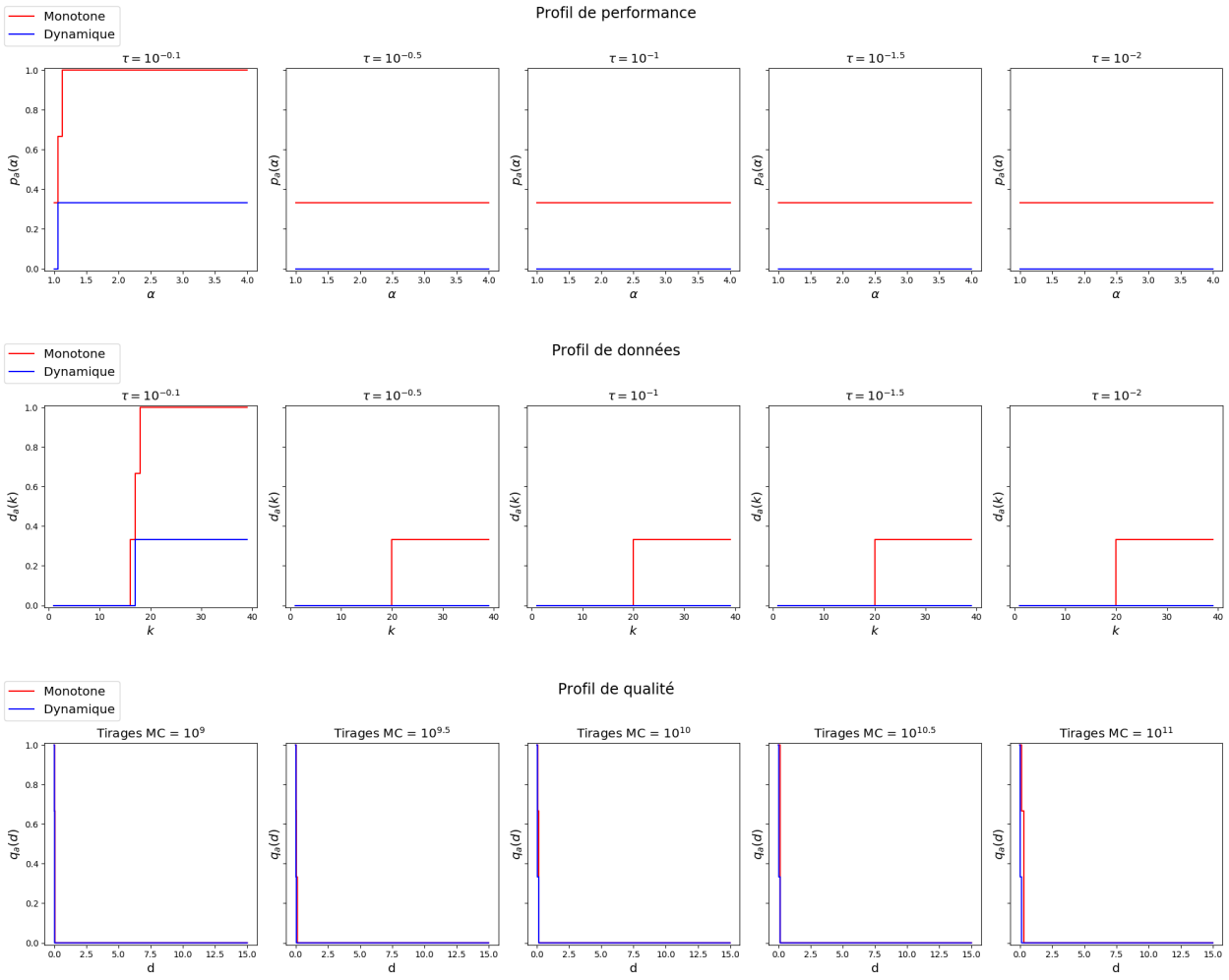

Figure 4.6 «$\sqrt{2}$-Quasi-Stairway» - Profils de (a) Performances. (b) Données. (c) Qualité.

On y observe, sous différents aspects, le fait que la stratégie dynamique échoue à aller aussi loin dans l'optimisation que la monotone. Il faut en effet choisir de très grandes tolérances pour observer des courbes non nulles.

Pour illustrer les raisons de la grande domination de la stratégie monotone sur ce problème, il faut donc chercher d'autres profils. Ici, ceux de *précision instantanée* et de *consommation instantanée* nous apporterons plus d'informations. En effet, il semble que l'explication tienne dans le fait que la méthode dynamique échoue à obtenir une précision assez élevée pendant un nombre d'itérations assez grand pour réussir à avancer sur les plateaux. Pour étayer cette hypothèse, on peut se référer aux deux profils de la figure 4.7 :

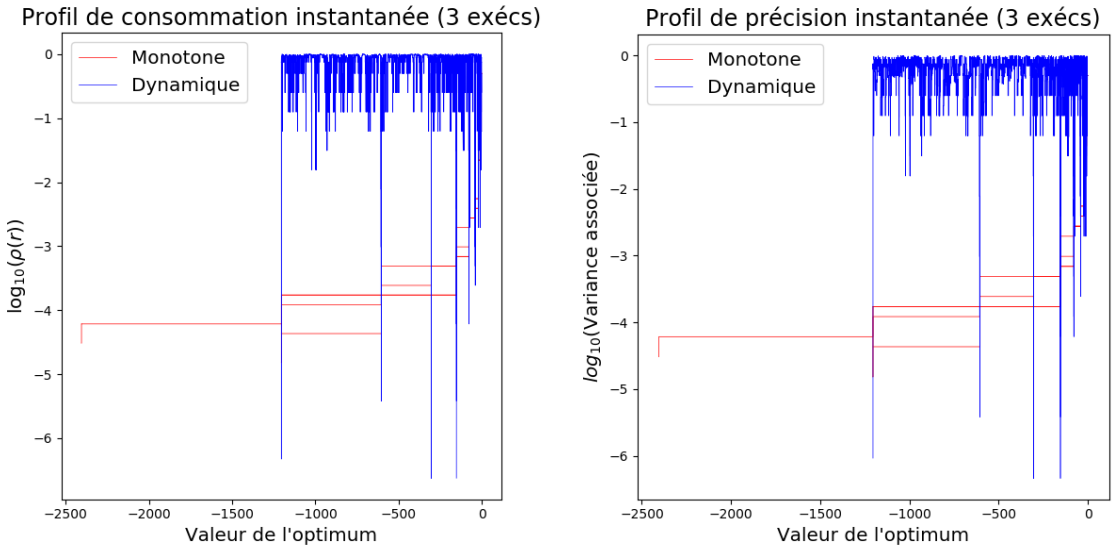

Figure 4.7 «$\sqrt{2}$-Quasi-Stairway» - Profils de (a) Consommation instantanée. (b) Précision.

Si la méthode de précision dynamique avait bien fonctionné, la consommation instantanée (l'écart-type imposé à toutes les évaluations) aurait été faible dans les zones où la fonction est raide, et aurait augmenté sur les plateaux. Dans le même temps, la précision instantanée (la précision de l'optimum courant) aurait suivie cette tendance, comme dans le cas de «Norm2». Or, ici les courbes de la précision dynamique oscillent fortement à une fréquence très élevée. Pour le comprendre, le profil de précision est utile. On y observe que l'écart-type de l'optimum courant revient régulièrement à une très grande valeur. Cela est dû aux points intermédiaires visités lors de l'arrivée sur un plateau : l'optimum courant est le point de meilleur estimé, qui stagne parmi les points trop imprécis que l'algorithme a survolé durant sa phase à haute précision. Durant les itérations où il amène toute sa cache à une précision suffisante, les points encore bruités faussent son interprétation en s'imposant comme des optimaux apparents (car leurs valeurs fortement bruitées les placent bien en-deçà de leurs voisins précis). La méthode de précision dynamique se perd donc dans son incertitude latente affectant beaucoup de points qu'il est très difficile de départager. Dans le même temps, la stratégie de précision monotone a su traverser ces plateaux en imposant rapidement une grande précision. Ce gain est ici assez important pour compenser la surconsommation inutile dans les zones moins restrictives.

L'analyse précédente amène alors naturellement à un corollaire : Réduire la taille des plateaux devrait diminuer le phénomène affectant la stratégie dynamique, puisque le nombre de points difficile à départager diminuerait d'autant. C'est en effet ce qu'on observe sur la deuxième fonction «Quasi-Stairway». Les résultats de son optimisation sont en figures 4.8, 4.9 et 4.10.

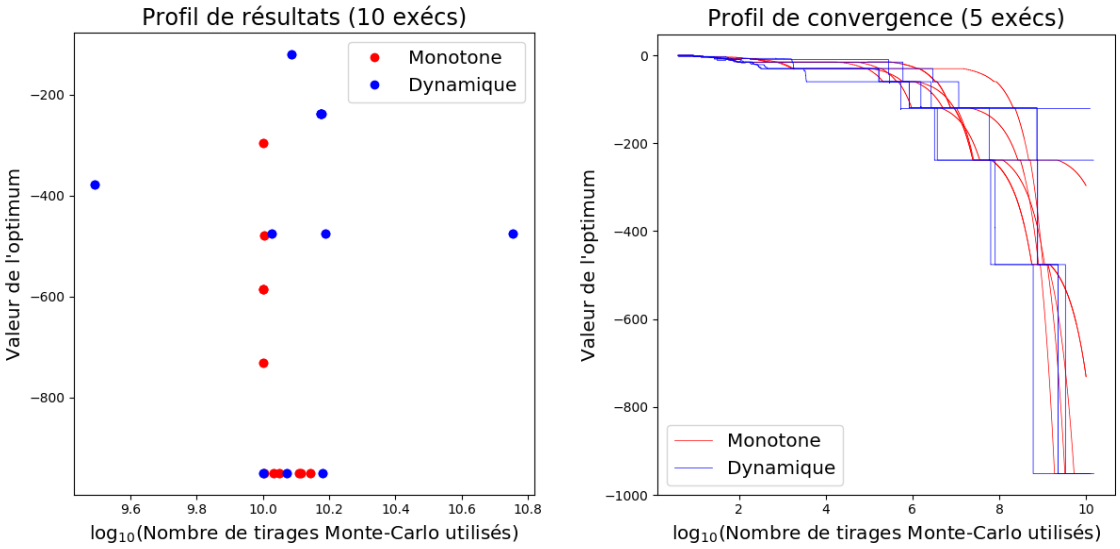

Figure 4.8 «1.1-Quasi-Stairway» - Profils de (a) Résultats. (b) Convergence Monte-Carlo.

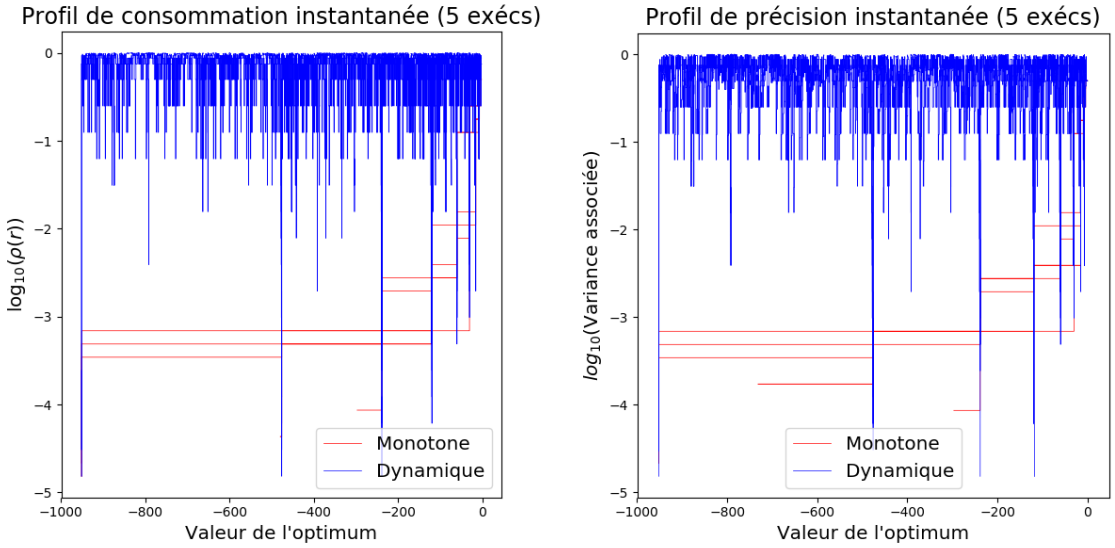

Figure 4.9 «1.1-Quasi-Stairway» - Profils de (a) Consommation instantanée. (b) Précision.

La surconsommation dûe à l'incertitude latente de la précision dynamique est toujours présente. On peut observer que les résultats sont moins disparates, mais pas encore équivalents. Les fonctions «Quasi-Stairway» montrent donc que la stratégie monotone est efficace sur des fonctions présentant de très grandes zones sur lesquelles la fonction est quasi-constante.

Enfin, les autres profils ne sont, là encore, pas particulièrement instructifs. Il faut prendre des tolérances bien trop grandes pour être pertinentes pour observer des courbes non nulles, ce

qui traduit la difficulté du problème et la robustesse encore faillible des méthodes.

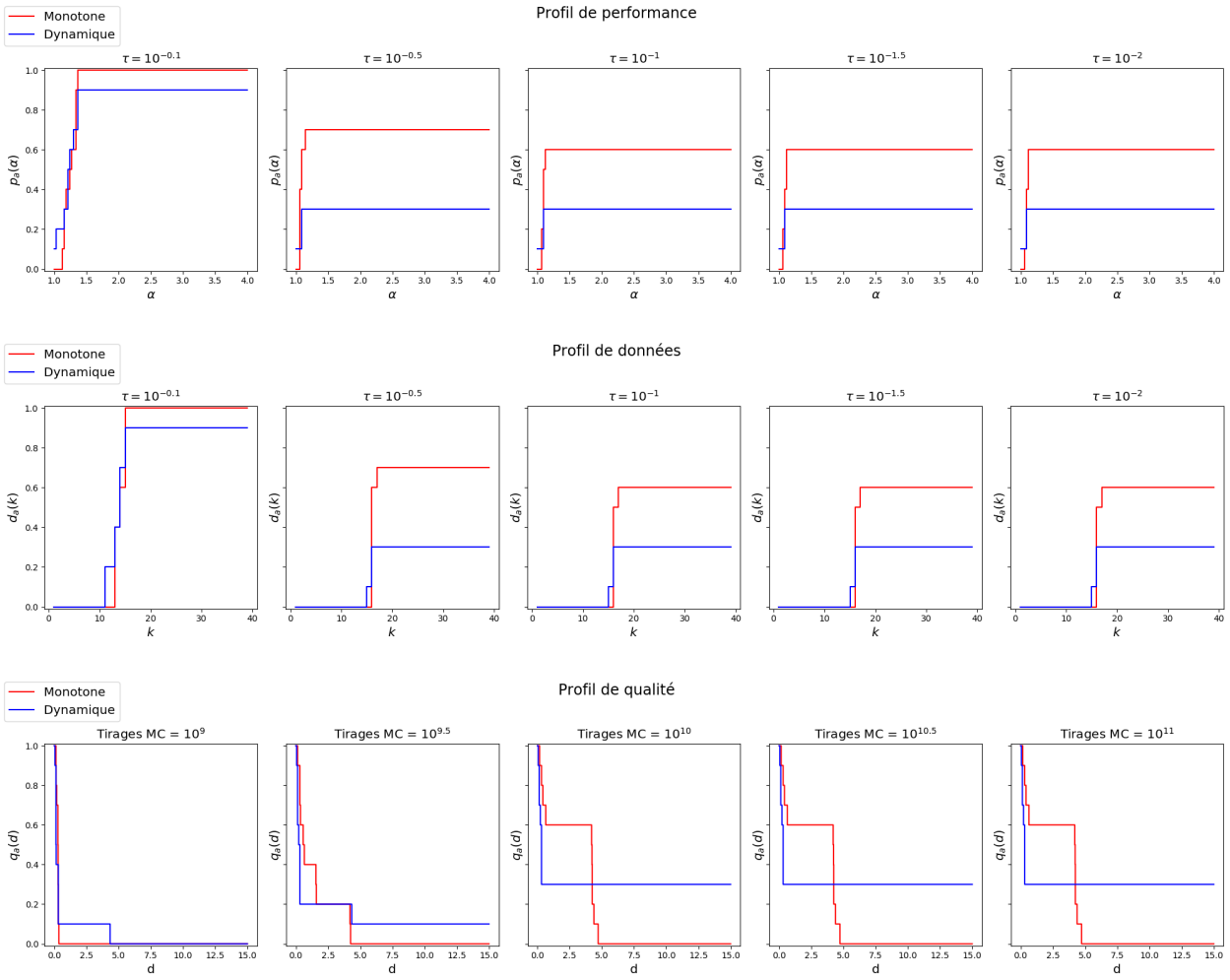

Figure 4.10 «1.1-Quasi-Stairway» - Profils de (a) Performances. (b) Données. (c) Qualité.

4.3.2 Obstacle (avec contraintes) : Boîte noire «Moustache»

Le problème «Snake» (d'Audet et Dennis, Jr. (2009), en section 4.1) est souvent utilisé dans les tests numériques en optimisation de boîte noire déterministe. Il permet d'étudier la gestion des contraintes et les effets d'une diminution de la taille du treillis. Nous allons le modifier pour qu'il illustre une difficulté propre aux problèmes bruités : un besoin temporaire de réduire la taille du treillis amène souvent à augmenter la précision (puisque les points évalués sont très proches, les vraies valeurs de l'objectif le sont également). Ainsi, sur un problème dans lequel le treillis doit souvent devenir petit, la stratégie monotone subira de nombreuses augmentations de précision, là où la méthode dynamique parviendra à limiter cet effet.

Présentation du problème

Comme dans les cas précédents, la fonction objectif sera artificiellement bruitée par un bruit normal ajouté par-dessus sa valeur déterministe. Le problème requiert une fonction intermédiaire (qui forcera l'optimisation à modifier régulièrement l'ordonnée de ses solutions) :

$$\begin{array}{rcl} (l, L) & = & (0.05, 0.1) \\ g : x \in [0, 20] & \rightarrow & g(x) = \left[\sin(x)\left(|\cos(x)| + 0.1\right) \pm \left(L - \dfrac{l - L}{1 + |x - 11|}\right)\right] \end{array}$$

permettant de définir le problème «*Moustache*» déterministe :

$$\max_{(x,y)\in\mathbb{R}^2} \quad x$$
$$\text{s.c.} \quad \begin{cases} (x, y) & \in & [0, 20] \times [0, 4] \\ (x_0, y_0) & = & (0, 2) \\ y - 2 & \in & g(x) \end{cases}$$

qu'on va bruiter en modifiant la fonction objectif. Ce ne sera plus x mais $x + \mathcal{N}\left(0, \sigma^2\right)$. Remarquons que cela ne signifie pas que la valeur de la variable x est aléatoire : le bruit n'intervient pas dans la valeur de x en tant que variable, mais uniquement lorsqu'on l'utilise dans la fonction objectif. C'est $f(x, y) = x$ qui sera bruitée, pas la variable x elle-même.

Le problème bruité est donc (une fois passé sous format «minimisation» via $f(x, y) = -x$) :

$$\min_{(x,y)\in\mathbb{R}^2} \quad \lim_{k\to\infty | \sigma^k(x,y) \searrow 0} f^k(x, y)$$
$$\text{s.c.} \quad \begin{cases} (x, y) & \in & [0, 20] \times [0, 4] \\ (x_0, y_0) & = & (0, 2) \\ y - 2 & \in & g(x) \\ f^k \text{ et } \sigma^k \text{ tels que définis en 1.2.3} \\ \text{Observer } f_\sigma(.) \text{ coûte } \sqrt{\sigma}^{-1} \text{ tirages} \end{cases}$$

La contrainte sur la valeur de y sera ici traitée comme une contrainte forte, et non relaxable. Nous considérerons que l'espace des variables $\mathcal{X}$ est restreint aux points la respectant :

$$\mathcal{X} = \left\{ (x, y) \in \mathbb{R}^2 \mid \begin{array}{l} 0 \leq x \leq 20 \\ 0 \leq y \leq 4 \\ |y - 2 - \sin(x)\left(|\cos(x)| + 0.1\right)| \leq \left(L - \dfrac{l - L}{1 + |x - 11|}\right) \end{array} \right\}.$$

Ainsi, $f(x, y) = +\infty$ si $(x, y) \notin \mathcal{X}$. Le problème est donc considéré comme non contraint, mais avec un espace des variables très difficile à explorer. La figure 4.11 illustre l'espace $\mathcal{X}$:

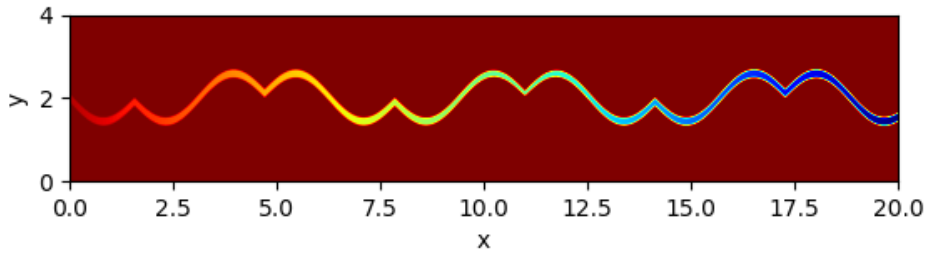

Figure 4.11 Fonction «Moustache».

Résultats numériques

Les résultats à venir ont été générés avec un unique critère d'arrêt, concernant la taille du treillis δ_p^k. L'optimisation d'un algorithme s'arrête dès qu'une itération k vérifie $\delta_p^k < 10^{-5}$, et aucun autre critère ne peut stopper l'optimisation (notamment, il n'y a pas de limite supérieure au budget de tirages Monte-Carlo). Les résultats bruts sont alors :

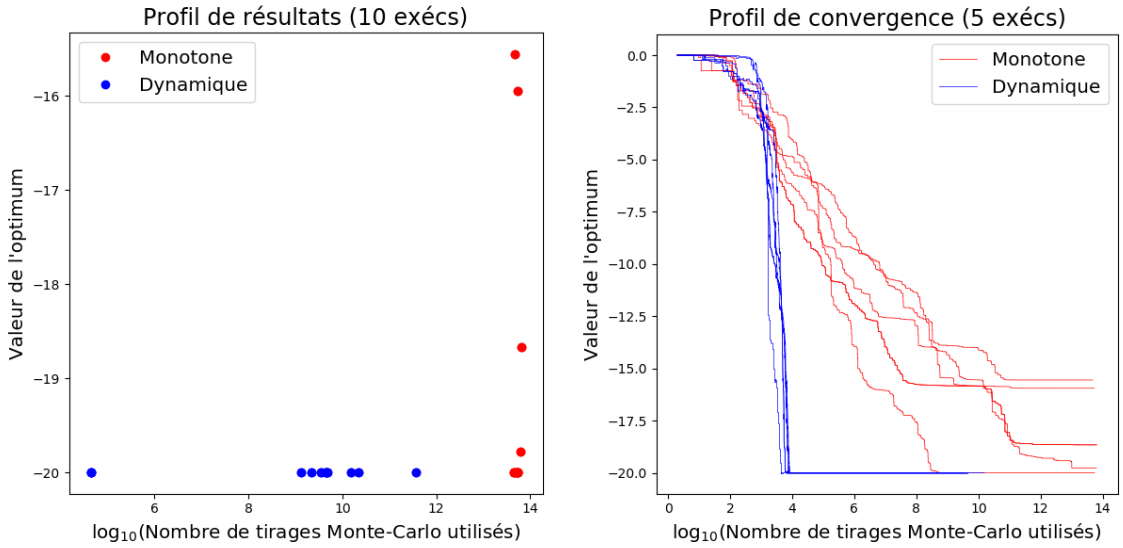

Figure 4.12 «Moustache» - Profils de (a) Résultats. (b) Convergence Monte-Carlo.

On y observe que la stratégie dynamique réussit à résoudre le problème, et que les optimums sont en réalité atteints dès 10^4 tirages (et non pas 10^{10} comme le profil de résultats le laisse penser). On constate aussi que la méthode monotone a parfois échoué, probablement à cause

de nombreux échecs dans les zones difficiles à $x \approx 8; 11; 14$ l'ayant incitée à fortement réduire son treillis, combiné à une évaluation particulièrement surestimée d'un point.

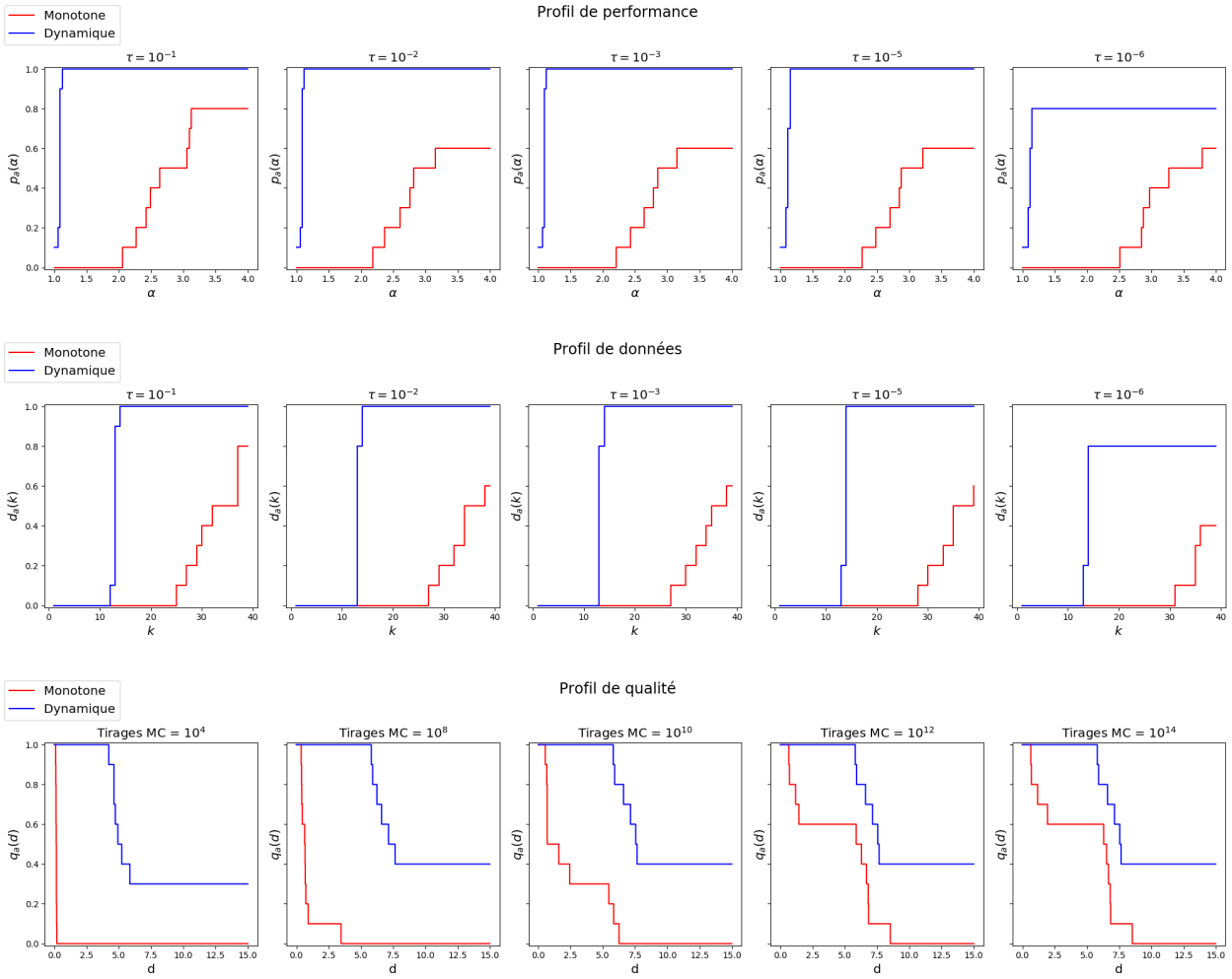

Figure 4.13 «Moustache» - Profils de (a) Performances. (b) Données. (c) Qualité.

La grande robustesse de la stratégie dynamique se retrouve clairement sur les profils précédents (figure 4.13).

Sur les profils de performance, on peut observer que la tolérance imposée n'a aucune influence sur la courbe de la stratégie monotone. Elle ne commence à s'effondrer qu'à partir de $\tau = 10^{-6}$, ce qui est une conséquence du fait que l'optimisation s'arrêtait à une taille du cadre de sonde δ_p de 10^{-5}. On peut remarquer un comportement analogue pour la courbe de la méthode monotone, ce qui laisse penser que soit cette stratégie réussit, soit elle échoue toujours dans des zones très semblables.

De même, sur les profils de données on peut observer que la stratégie dynamique a une consommation de tirages Monte-Carlo variant très peu d'une exécution à l'autre. Comme on

le voyait sur le profil de convergence 4.12, toutes les exécutions ont convergé après 10^4 points. Cela explique le comportement «binaire» de la courbe dynamique, qui passe de 0 à 1 presque directement, et ce pour toute tolérance τ. La stratégie monotone, elle, semble augmenter bien plus linéairement avec la consommation de tirages. C'est une conséquence du fait que la précision reste constante dès lors qu'elle est suffisante ; là où la dynamique l'avait grandement relaxée dès que possible pour avoir une consommation instantanée très faible.

Enfin, les profils de qualité prouvent la convergence des algorithmes jusqu'à une tolérance de $1 - 10^{-8}$ environ. C'est une conséquence de deux phénomènes : le fait que pour approcher l'optimum encore plus près, il aurait fallu autoriser le cadre de sonde à diminuer plus fortement ; et le fait que lorsque deux points évalués sont extrêmement proches, l'écart entre leurs images par la vraie fonction objectif est aussi extrêmement faible (trop pour que les algorithmes aient réussi à imposer une précision suffisante pour l'observer).

Enfin, on peut examiner les profils de consommation et précision. Ceux-ci sont très représentatifs des différences fondamentales entre les deux stratégies :

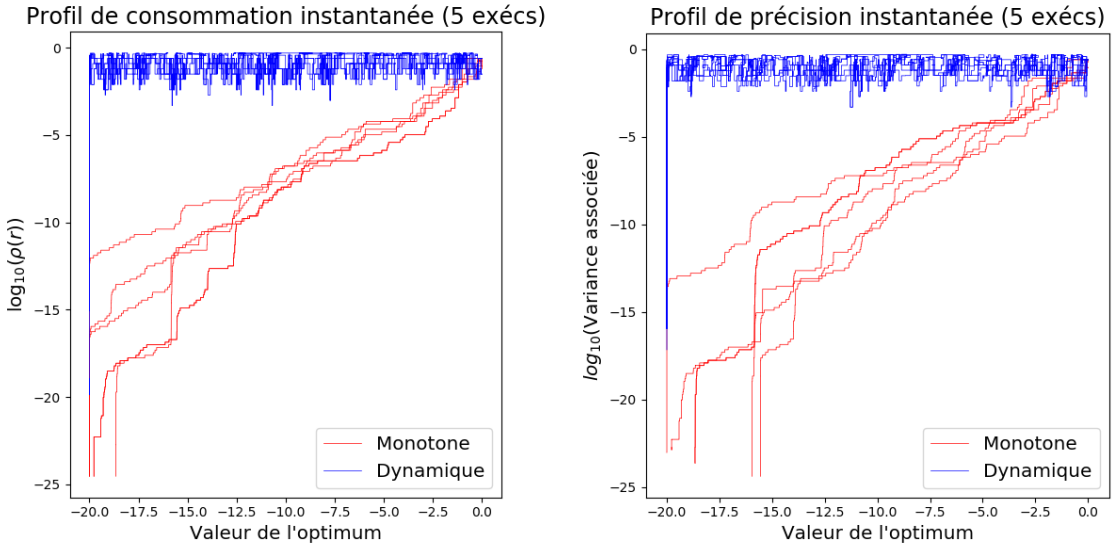

Figure 4.14 «Moustache» - Profils de (a) Consommation instantanée. (b) Précision.

On y observe en effet que la stratégie dynamique conserve le plus possible une précision faible, et qu'elle avance dans une incertitude relative (même l'optimum courant est toujours relativement incertain) ; alors qu'à l'opposé la méthode monotone augmente sa précision dès qu'elle rencontre une difficulté, pour n'exploiter que des informations très fiables.

4.4 Étude d'un problème plus complexe : boîte noire «Polygons»

Tous les tests précédents impliquent des problèmes purement artificiels, conçus pour comparer les performances d'algorithmes qui voient comme des boîtes noires des fonctions pourtant parfaitement analytiques. Dans tous ces problèmes, l'optimum global était connu et identifiable. De plus, les fonctions étaient relativement lisses et simple à étudier. Dans la famille de problèmes qui suit, ces remarques ne seront plus avérées. Les problèmes «Polygons» sont complexes, avec une fonction objectif difficilement dérivable et loin d'être lisse. De plus, toute évaluation va prendre un temps de calcul important, et sera bruitée car calculée via une vraie simulation de Monte-Carlo. Ces problèmes sont donc de véritables boîtes noires, en un sens très proche de la définition donnée en introduction (1.1.2).

Nous parlons de *famille* de problèmes car nous pouvons en définir un par valeur de $n \in [\![3, +\infty[\![$. À n fixé, on désignera par $\mathcal{Q}_n$ l'ensemble des polygones à n sommets dont le diamètre[1] est au plus unitaire. L'objectif du problème est de trouver le polygone de $\mathcal{Q}_n$ d'aire maximale. Le problème $Polygons_n$ est donc : résoudre $\max_{Q \in \mathcal{Q}_n} \mathcal{A}(Q)$.

Pour modéliser ce problème, on considérera deux variables par sommet du polygone : ses coordonnées polaires. Le problème se ramène donc à la recherche du jeu de coordonnées maximisant le volume qu'elles génèrent, sous la contrainte de diamètre. Pour n sommets $v_0, \ldots, v_{n-1}$, on note $r_0, \ldots, r_{n-1}$ leurs rayons et $\theta_0, \ldots, \theta_{n-1}$ leurs arguments. Il y a donc $2n$ variables pour représenter entièrement le polygone. Dans un bon repère, on peut se contenter de $2n-3$ variables (r_1 à r_{n-1} plus θ_1 à θ_{n-2}) via la modélisation présentée en figure 4.15.

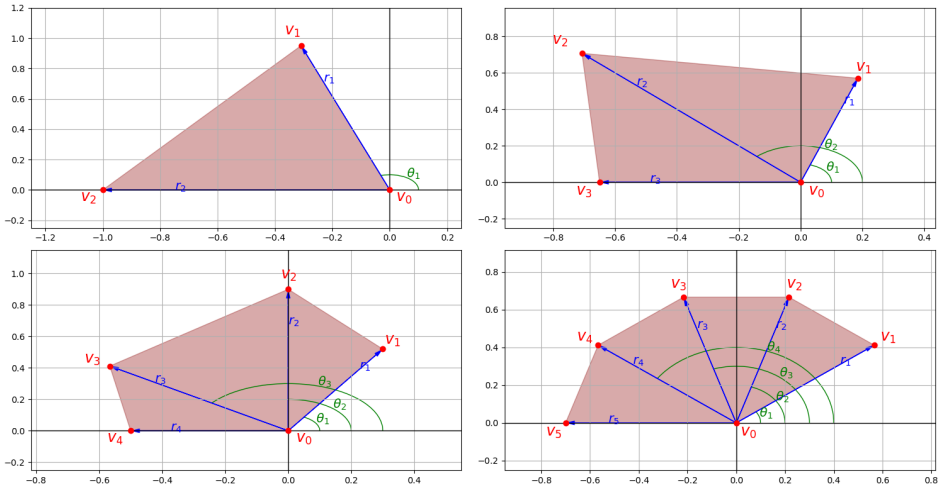

Figure 4.15 Modélisation d'un polygone à n sommets par $2n-3$ variables.

1. Le *diamètre* d'un polygone est la plus grande des distances séparant deux de ses sommets.

Afin de calculer efficacement l'aire, on vérifiera l'appartenance d'un point m à Q via un système de contrainte. Un point est dans un polygone si, pour chaque face, il se trouve dans le demi-espace contenant les autres points. La figure 4.16 illustre ce phénomène. Ainsi, l'appartenance à un polygone se traduit par le respect de cette contrainte, pour chaque couple de deux sommets successifs.

De plus, cette contrainte est exprimable analytiquement via un produit vectoriel bien choisi :

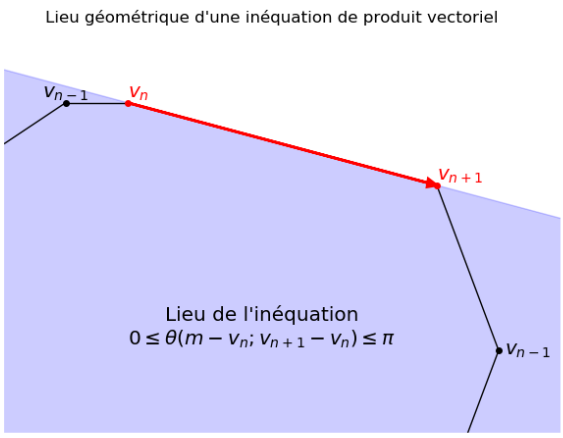

Figure 4.16 Lieu d'une contrainte.

$$\begin{aligned} \theta(m - v_i; v_{i+1} - v_i) \in [0, \pi] &\iff \{(\vec{m} - \vec{v_{i+1}}) \times (\vec{v_{i+1}} - \vec{v_i})\} \cdot \vec{e_3} \geq 0 \\ &\iff \left(\begin{bmatrix} x - x_{i+1} \\ y - y_{i+1} \\ 0 \end{bmatrix} \times \begin{bmatrix} x_{i+1} - x_i \\ y_{i+1} - y_i \\ 0 \end{bmatrix} \right) \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \geq 0 \\ &\iff (x - x_{i+1})(y_{i+1} - y_i) \geq (y - y_{i+1})(x_{i+1} - x_i)\,. \end{aligned} \tag{4.1}$$

Ainsi, on peut poser l'ensemble C des points vérifiant 4.1 sur chaque face (on prendra $v_n = v_0$) :

$$\forall\ Q = (v_0, \ldots, v_{n-1})\,,\ C = \Big\{(x, y) \in \mathbb{R}^2 \mid \forall i \in [\![0, n-1]\!],\ (x, y) \text{ vérifie } 4.1\Big\}\,.$$

Lorsque Q est convexe, C coïncide pleinement avec l'ensemble des points le constituant. On peut donc l'exploiter pour estimer l'appartenance d'un point à Q par un test analytique.

Toutefois, il est intéressant de noter que le comportement de C devient erratique lorsque Q n'est plus convexe. La figure 4.17 illustre l'effet du jeu d'équations 4.1 sur quelques polygones particuliers.

Cependant, on se convaincra aisément qu'un polygone de $\mathcal{Q}_n$ non convexe ne peut être

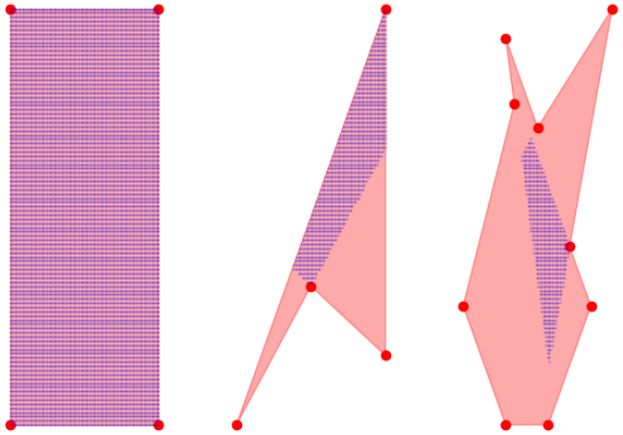

Figure 4.17 Ensembles C de quelques polygones.

optimal, puisque son enveloppe convexe respecte également la contrainte de diamètre, et a une aire strictement supérieure. On a donc une garantie que le comportement erratique de la figure 4.17 ne perturbera pas l'algorithme. Par conséquent, on peut utiliser C pour estimer l'aire de Q de façon non biaisée lorsqu'on approche une solution qui fait sens. Le calcul de son aire (qui revient donc à celle de C) peut se faire de différentes façons. Il est évidemment possible de la calculer exactement, mais afin de simuler au mieux le comportement réel de vraies boîtes noires, on utilisera plutôt une approximation de Monte-Carlo. Pour cela, on définira en premier lieu la plus petite boîte contenant Q :

$$\mathcal{B}(Q) = \left[\min_{(x,y)\in Q} x, \max_{(x,y)\in Q} x\right] \times \left[\min_{(x,y)\in Q} y, \max_{(x,y)\in Q} y\right].$$

On a alors, pour Q convexe et de manière analogue à ce qui est présenté en 1.1 :

$$\begin{aligned} \mathcal{A}(Q) = \mathcal{A}(C) &= \int_{\vec{x}\in\mathcal{B}(Q)} \mathbb{1}_{\vec{x}\in C} d\vec{x} \\ &\approx \widetilde{\mathcal{A}}_N = Vol(\mathcal{B}(Q)) \times \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{\vec{x_i}\in C} \end{aligned}$$

où les $\vec{x_i}, \forall i \in [\![1; N]\!]$, sont des observations indépendantes de la loi uniforme sur $\mathcal{B}(Q)$; et où l'appartenance à C est calculée en vérifiant le respect des contraintes 4.1.

Enfin, pour éviter la génération inutile de polygones trop erratiques, on imposera que la suite des arguments des sommets soit croissante : $\forall i \in [\![1, n-3]\!],\ 0 \leq \theta_i \leq \theta_{i+1} \leq \pi$.

On peut alors formuler le problème déterministe «Polygons», à un nombre n de sommets fixé :

$$(Polygons_n) : \begin{cases} \min\limits_{\vec{r}\in\mathbb{R}^{n-1}, \vec{\theta}\in\mathbb{R}^{n-2}} & -\mathcal{A}\left(Q(\vec{r}, \pi\vec{\theta})\right) \\ \text{s.c.} & \begin{cases} \vec{r} \in [0,1]^{n-1} \\ \vec{\theta} \in [0,1]^{n-2} \\ \theta_i \leq \theta_{i+1},\ \forall i \in [\![1, n-3]\!] \\ 1 \geq Diam\left(Q(\vec{r}, \pi\vec{\theta})\right) \end{cases} \end{cases}$$

et sa variante non déterministe :

$$(PolygonsNoisy_n) : \begin{cases} \min\limits_{\vec{r}\in\mathbb{R}^{n-1}, \vec{\theta}\in\mathbb{R}^{n-2}} & \lim\limits_{N\to+\infty} -\widetilde{\mathcal{A}}_N\left(Q(\vec{r}, \pi\vec{\theta})\right) \\ \text{s.c.} & \begin{cases} \vec{r} \in [0,1]^{n-1} \\ \vec{\theta} \in [0,1]^{n-2} \\ \theta_i \leq \theta_{i+1},\ \forall i \in [\![1, n-3]\!] \\ 1 \geq Diam\left(Q(\vec{r}, \pi\vec{\theta})\right) \end{cases} \end{cases}$$

La solution optimale à ces problèmes n'est pas toujours triviale. Il est prouvé que lorsque le nombre de sommets est impair, le polygone optimal est le régulier (Reinhardt (1922)). L'analyse de la performance des algorithmes est donc très simple lorsque n est impair, car la simple représentation visuelle de la solution qu'ils proposent permet immédiatement d'en déterminer la qualité. Cependant, lorsque le nombre de sommets manipulés devient pair, l'optimal est très différent de la solution régulière. Par exemple, lorsqu'on cherche le meilleur hexagone (pour $n = 6$, donc), Graham (1975) prouve que l'optimum au problème est unique, et environ 4% plus grand que le régulier. Cette unique solution est représentée en figure 4.18.

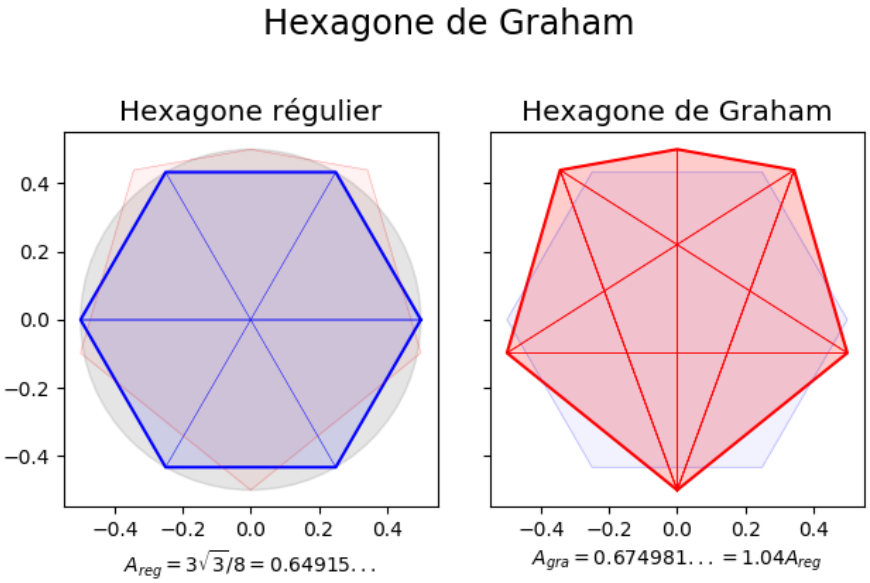

Figure 4.18 Supériorité de l'aire de l'hexagone de Graham sur celle du régulier.

Les optimums de certains problèmes pairs sont connus, analytiquement pour $n = 6$ et numériquement $n \in \{8, 10, 12\}$. Ils sont présentés dans Graham (1975), Audet *et al.* (2002) et Henrion et Messine (2011). La question reste ouverte pour les tailles paires supérieures.

La question est pertinente, surtout pour n pair, notamment car il y a alors de nombreux optimums locaux. Cette famille de problèmes est alors doublement intéressante, de par sa difficulté naturelle à cause des nombreux optimums locaux, et dans sa variante bruitée par le fait que la fonction objectif est de très faible amplitude, ce qui imposera aux algorithmes d'effectuer toutes leurs évaluations à une grande et coûteuse précision.

Résultats numériques

Cette famille de problèmes a mis les algorithmes en difficulté. Pour des raisons pratiques, nous avons dû imposer un critère d'arrêt restrictif (arrêt dès que $\delta_p \leq 10^{-5}$). Cela est nécessaire

pour stopper le problème en un nombre contenu de tirages Monte-Carlo.

La stratégie monotone échoue à achever son optimisation. En effet, la précision atteint après un certain nombre d'itérations une valeur trop élevée pour rester acceptable en pratique. En se souvenant des courbes présentées en 1.2, on comprend qu'une très grande précision amène à un temps de calcul par évaluation très élevé. De fait, les optimisations pilotées par la stratégie monotone ont dû être avortées car elles tournaient toujours après plusieurs semaines de calcul, alors que la stratégie dynamique ne requiert que quelques heures pour s'achever. Les résultats liés à la méthode monotone dans les courbes suivantes sont les meilleures obtenus malgré l'arrêt anticipé des optimisations.

On pourrait penser à chercher un moyen d'empêcher la stratégie monotone d'atteindre des niveaux de précision trop importants. Une solution pourrait être de borner inférieurement l'écart-type imposé à la boîte noire (en modifiant la fonction ρ les générant, comme expliqué dans le chapitre 5.3), mais cela a échoué également. La raison est cette fois l'incompatibilité entre l'incertitude résultante de ce choix et le besoin de résultats parfaitement fiables requis par la méthode monotone : elle se refusait à avancer dans l'optimisation car les résultats n'étaient plus assez fiables, mais l'augmentation de précision qu'elle imposait en réponse ne pouvait plus corriger ce manque. Au cumulé, cette approche était donc inefficace, et les résultats qui en sont issus ne seront pas discutés ici.

À titre de comparaison, les deux stratégies seront comparées à une exécution du solveur NOMAD (Abramson *et al.* (2015)). Ce solveur manipule des fonctions déterministes ; aussi nous lui passerons la boîte noire bruitée, mais à une précision fixée et très élevée. Le solveur la considérera comme déterministe (ce qu'elle n'est pas, puisqu'on a un très faible bruit la parasitant). Il sera intéressant de comparer nos solutions à cette stratégie «naïve» consistant à négliger l'aléatoire en imposant une précision constante et très élevée.

Analysons d'abord le cas très simple du triangle, pour lequel a fonction objectif est convexe. La stratégie monotone n'a, comme annoncé, jamais achevé sa première optimisation (d'où l'unique résultat présenté). La méthode dynamique a réussi à s'achever à chaque fois, mais l'une des optimisation est restée bloquée sur une solution loin d'être optimale (ceci est probablement dû à une trop faible précision lors de certaines évaluations, combiné à une stratégie de réévaluation des points passés trop peu agressive). Enfin, sur ce problème assez simple, l'optimisation «faussement déterministe» a convergé, mais au prix de plus de tirages que nos deux stratégies. Les résultats des optimisations (au critère d'arrêt $\delta_p \leq 10^{-5}$) sont en figure 4.19.

On peut également se concentrer sur les profils en figure 4.20. Notons un point important : l'optimum du problème a ici été considéré comme la meilleure solution trouvée par l'un des algorithmes. Nous n'avons pas choisi l'optimum réel ($\sqrt{3}/4$, atteint pour le triangle unitaire

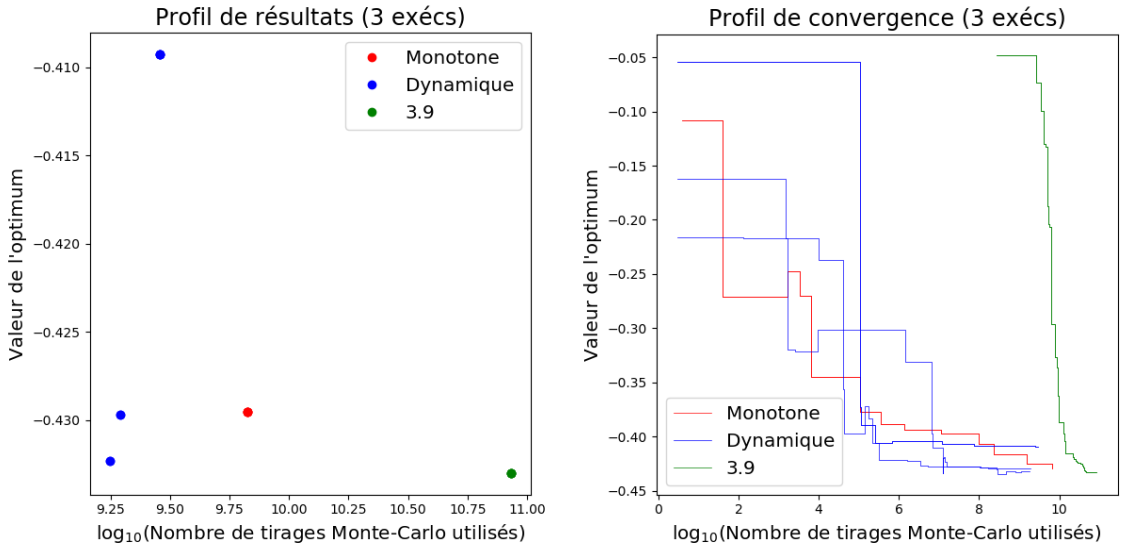

Figure 4.19 «Polygons$_3$» - Profils de (a) Résultats. (b) Convergence Monte-Carlo.

équilatéral) car aucun algorithme ne l'a approché d'assez près pour que les profils soient pertinents. À défaut, nous préférons donc comparer les algorithmes entre eux, en retenant qu'aucun n'a réellement trouvé l'optimum (car notre critère d'arrêt ne leur permettait pas).

Le profil de performance peut paraître surprenant. Lorsque la tolérance est grande ($\tau = 10^{-0.1}$), la courbe associée à *NOMAD 3.9.1* est nulle, et ne croît que lorsque la tolérance diminue. Cela prouve deux choses : *NOMAD 3.9.1* a trouvé la meilleure solution (puisque sa courbe, issue d'une unique optimisation, monte à 1) mais à un grand coût. En effet, pour que la courbe décolle de 0 sur le premier profil, il aurait fallu monter α à près de 15 : la stratégie «pseudo-déterministe» requiert 15 fois plus de tirages Monte-Carlo que les monotones et dynamiques pour approcher l'optimum. Par contre, sur ce problème très simple, elle parvient à aller un peu plus loin que les autres.

Le profil de données traduit aussi cette idée, en montrant notamment que la stratégie dynamique reste, une fois sur trois, aussi performante que la quasi-déterministe, pour une tolérance de 10^{-2}. On y observe aussi que la stratégie monotone avait localisé l'optimum à $10^{-1.5}$ près avant de faire grandir fortement ses coûts de calcul. C'est donc, comme dans «Norm2», la phase d'intensification qui a coûté cher à la méthode monotone.

Enfin, on voit sur le profil de qualité que la méthode quasi-déterministe est la meilleure en terme de qualité, mais à un coût de fonctionnement élevé.

Cependant, les résultats changent lorsque le nombre de sommets du polygone augmente. En passant sur un nombre pair (on prendra 6), les algorithmes ne se comportent plus de la même

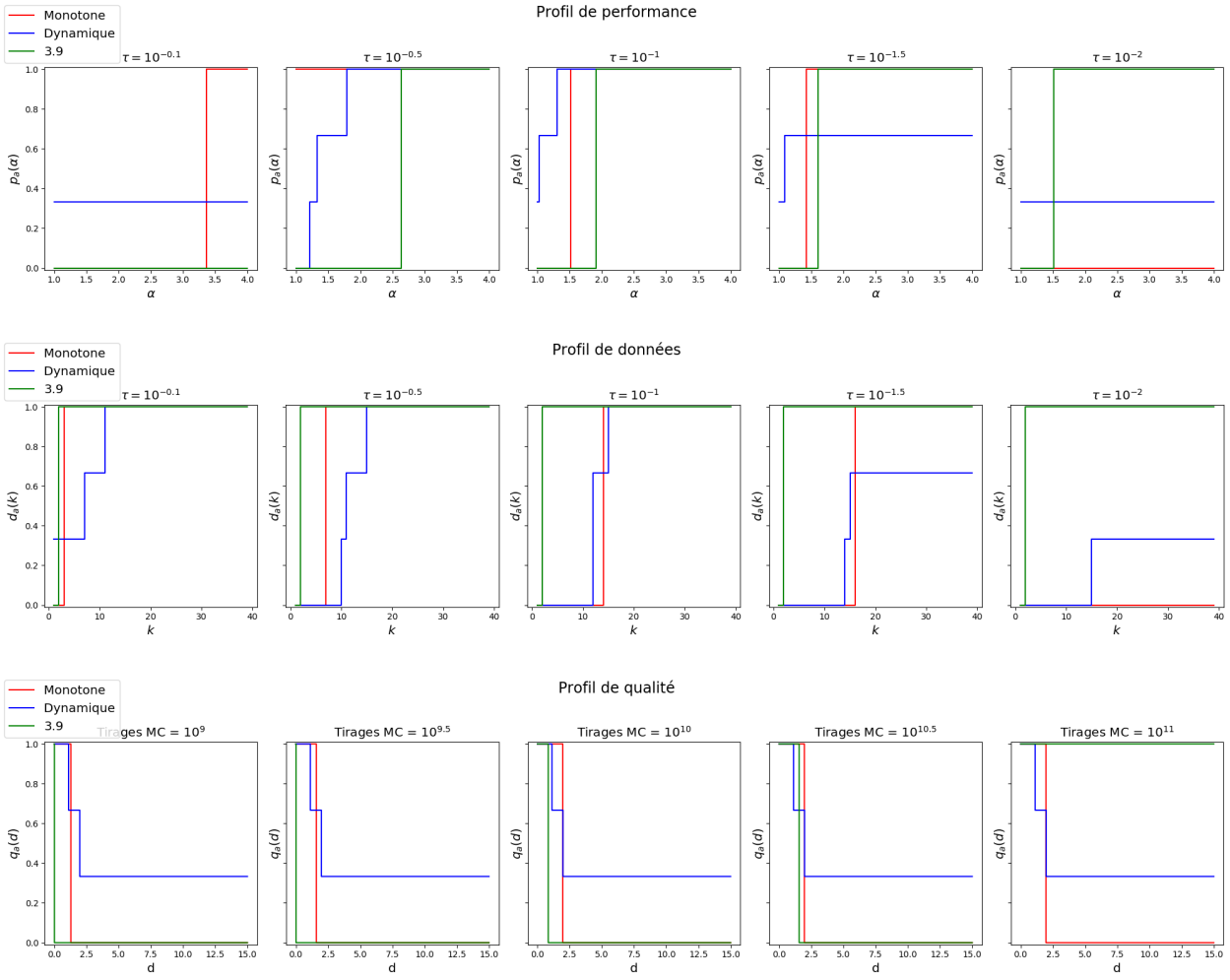

Figure 4.20 «Polygons$_3$» - Profils de (a) Performances. (b) Données. (c) Qualité.

manière. Là encore, on prendra comme optimum au problème la meilleure solution trouvée par l'un des algorithmes, car aucun n'a approché le polygone de Graham d'assez près [2]. Cela peut s'expliquer via la condition d'arrêt des optimisations, qui était la taille du cadre de sonde $\delta_p \leq 10^{-5}$.

On remarquera qu'ici, les trois exécutions de la stratégie monotone ont tourné en un temps raisonnable. Cependant, les résultats ne sont pas de qualité. Cela est probablement dû aux nombreux optimums locaux que ce problème présente : la stratégie monotone a probablement plongé dans l'un d'eux et, puisque sa précision est grande, n'a pas pu en sortir. La méthode quasi-déterministe est probablement tombée dans un piège analogue.

À l'opposé, la stratégie dynamique (qui reste toujours à une incertitude relativement élevée)

2. La meilleure solution trouvée est ≈ 0.667370, alors que le polygone de Graham donne ≈ 0.674981.

a survolé ces optimums sans s'y attarder, et n'a intensifié qu'autour du plus intéressant après les réévaluations faites dans la cache.

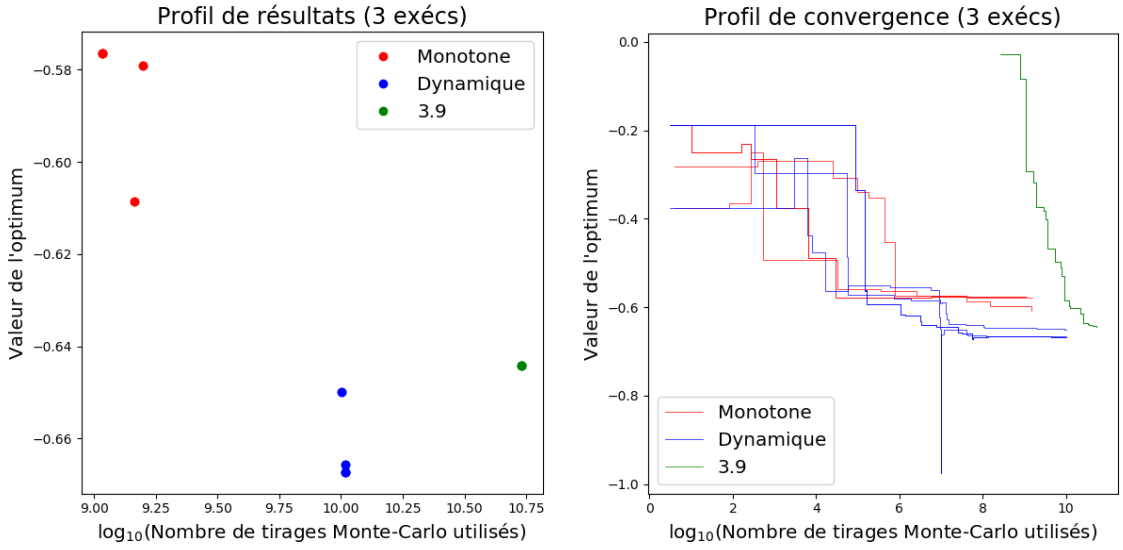

Figure 4.21 «Polygons$_6$» - Profils de (a) Résultats. (b) Convergence Monte-Carlo.

Cependant, ces résultats sont à relativiser, au vu de la meilleure solution proposée :

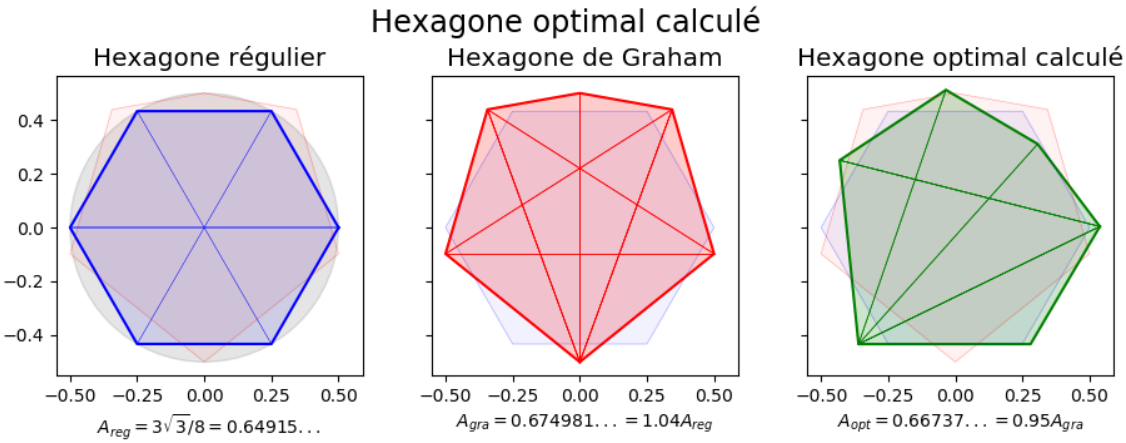

Figure 4.22 Échec de l'ensemble des optimisations à approcher l'optimum de Graham.

On peut constater que cette solution (issue de la stratégie dynamique), la meilleure parmi toutes celles proposées durant nos optimisations, est encore assez lointaine de l'optimum réel. Cependant, on peut l'expliquer. Les diagonales tracées sur les figures sont celles de longueur (proche d'être) unitaires. La solution proposée est donc relativement contrainte, et le seul

moyen de s'en échapper pour se rapprocher de l'hexagone de Graham serait de temporairement réduire le polygone pour desserrer les contraintes. Mais, en ce faisant, la qualité de la solution diminuerait, ce que les méthodes n'autorisent pas. Le relatif échec des optimisations vient donc de la difficulté naturelle du problème, plus que de l'impact de l'aléatoire.

Conclusions issues des tests numériques

Les divers tests présentés ici permettent finalement de cerner le comportement pratique des différentes méthodes que nous proposons. Plusieurs points intéressants peuvent être notés.

D'abord, le problème «Polygons$_6$» prouve que la stratégie «naïve», qui consiste à effectuer l'optimisation via un algorithme conçu pour les cas déterministes, en fixant la précision à une valeur constante et élevée, n'est pas efficace. Elle n'est ni performante en terme de résultats (sauf peut-être sur des cas très simples), ni en terme de consommation de ressources (dans tous les cas, elle a consommé significativement plus de tirages Monte-Carlo que les autres méthodes). Cela laisse penser qu'il y a un besoin réel d'utiliser des méthodes spécifiquement adaptées lorsqu'on manipule une boîte noire bruitée. Le bruit impose de la prudence lors de l'optimisation, que la stratégie naïve n'a pas. De plus, lorsque la précision est variable, des méthodes conçues pour exploiter cette information permettent d'économiser beaucoup de ressources calculatoires pour parvenir à des résultats aussi bons, et souvent meilleurs.

Ensuite, on constate que les deux stratégies (monotone et dynamique) n'ont pas les mêmes forces et faiblesses. Au vu de «Quasi-Stairway», il semble que la stratégie monotone gère mieux les fonctions à très faible amplitude, car la dynamique échoue à discriminer rapidement les solutions qu'elle examine, et réévalue trop souvent des points «potentiellement intéressants». Par contre, lorsque les algorithmes entament leur convergence vers une solution optimale («Norm2»), la méthode monotone a tendance à surconsommer. Elle parvient à aller plus loin que la stratégie dynamique, mais à un coût prohibitif. Enfin, sur des fonctions peu plates mais dont l'optimisation impose de petites tailles de treillis («Moustache»), la stratégie monotone surconsomme grandement, et risque d'échouer ; alors que la dynamique gère bien mieux les difficultés.

Pour conclure, la stratégie quasi-déterministe ne semble pas viable, et la monotone globalement dominée par la dynamique. Il reste cependant des cas précis où la méthode dynamique est en difficulté, mais les problèmes mettant ce phénomène en évidence sont très particuliers.

CHAPITRE 5 TECHNIQUES D'IMPLÉMENTATION PRATIQUE

Ce dernier chapitre a vocation de discuter de la viabilité pratique de nos hypothèses. Nous proposerons quelques lignes directrices pour faire entrer un problème bruité dans le cadre théorique développé précédemment. Le solveur développé pour implémenter les algorithmes et effectuer les comparaisons numériques est disponible sur mon compte GitHub [1]. Il s'agit d'un code en Python 3.6 nommé NOMAD Lite Python. Il s'inspire du logiciel NOMAD (Abramson *et al.* (2015)) développé au GERAD par l'équipe d'optimisation de boîte noire, qui offre une implémentation efficace de *MADS* déterministe enrichie de nombreux ajouts performants.

Pour entrer dans notre cadre théorique, une boîte noire doit vérifier certaines propriétés :

- $\forall x \in \mathbb{R}^n,\ \mathbb{P}(f(x) = +\infty) = 0$ ou 1 : le calcul échoue ou réussit presque sûrement.
- Le bruit entourant les estimations de $f(x)$ doit être stochastique, et usuellement normal.
- $\forall x \in \mathcal{X},\ \forall \sigma > 0$, la boîte noire peut générer et observer $f_\sigma(x) \sim \mathcal{N}\left(f(x), \sigma^2\right)$.

5.1 Discussion des hypothèses

Identification des sources de bruit

Dans les problèmes réels, les sources courantes du bruit numérique sont :

- Des quantités imprécises (par exemple expérimentales, ou générées aléatoirement),
- Des erreurs d'approximations et d'arithmétique flottante,
- La résolution de systèmes numériques, convergence vers un point fixe, etc., considérée réussie à un seuil de tolérance donné,
- L'estimation d'un phénomène (d'une valeur moyenne par exemple) par un tirage fini de scénarios (comme proposé par les simulations de Monte-Carlo (1.1) par exemple).

Les trois premières sources amènent à un bruit contenu, au sens où on maîtrise bien son amplitude et sa valeur maximale. Un tel bruit est traitable par des stratégies comme celle de Polak et Wetter (2006), car il s'agit exactement du type de problème bruité que leur article proposait de résoudre. Ils ne rentrent pas tout à fait dans notre cadre de travail car il est possible que le bruit ne soit pas aléatoire : les erreurs peuvent être déterministes. Si elles ne le sont pas, alors notre stratégie peut convenir en précisant à l'algorithme la distribution (uniforme) des erreurs. Cependant, notre cadre théorique traite bien mieux la dernière source d'erreurs : le bruit aléatoire venant d'une observation partielle d'un phénomène.

1. Me contacter avec votre courriel ou identifiant GitHub pour l'accès au code : pybouchet@gmail.com

Approximation normale

Il est peu commun de rencontrer de vrais problèmes dans lesquels le bruit suit une loi normale. Les problèmes-test du chapitre 4 n'illustrent pas tous fidèlement la réalité, puisque seul «Polygons» n'ajoute pas un bruit normal artificiel par dessus le calcul exact de la vraie valeur.

Toutefois, cela ne rend pas l'hypothèse normale erronée. Elle reste même très courante et naturelle lorsque le bruit vient d'une estimation d'une valeur par un nombre fini de tirages. Rappelons le théorème central limite (1.1.6) pour s'en convaincre : la moyenne empirique d'un nombre fini N d'observations d'une même loi suit approximativement une loi normale centrée en la véritable moyenne de la loi observée, et d'écart-type de l'ordre de $1/\sqrt{N}$. On sait également que plus grand est N, meilleure est l'approximation normale. On peut de plus montrer, sous certaines hypothèses[2], que la convergence vers la loi normale est uniforme et de l'ordre de $1/\sqrt{N}$ (démontré par Berry (1941)).

De fait, un nombre relativement restreint de variables aléatoires indépendantes suffit pour garantir une distribution quasi-normale à la moyenne empirique de leurs observations. Dans notre cas où les répétitions sont des tirages Monte-Carlo pour estimer une valeur, on aura souvent un grand nombre d'observations. La loi normale fait donc sens pour manipuler des simulations de Monte-Carlo.

Il faut toutefois connaître l'amplitude du problème pour déterminer un nombre de tirages minimal requis pour faire fonctionner cette hypothèse. Quelle est la taille de l'espace dans lequel on effectue les tirages Monte-Carlo ? Comment varie le phénomène observé dans cet espace ? Ou, à défaut, quelle est l'amplitude de ses variations ? Est-il continu, voir lipschitzien ? Ces questions, ainsi que d'autres non listées ici, influent sur la qualité de l'estimation de la moyenne par un faible nombre de tirages. La figure 5.1 illustre quelques situations.

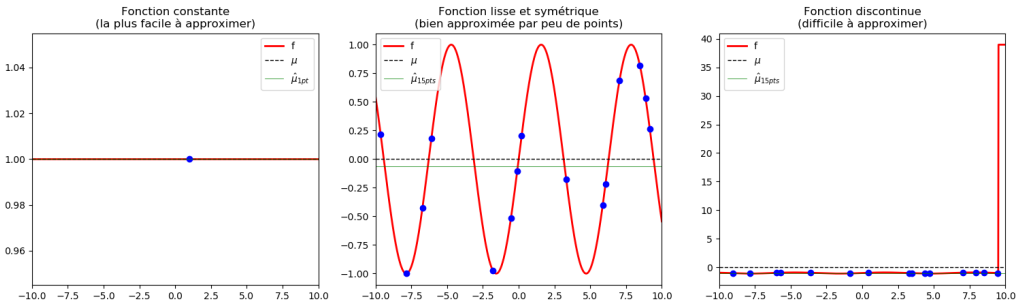

Figure 5.1 Approximations de Monte-Carlo de quantités définies sur $\mathbb{R}$.

On peut observer certains comportements intéressants sur la figure 5.1, généralisables sur

2. L'existence des moments d'ordre 0 à 3 de la loi observée.

des espaces plus larges ou de plus grande dimension. Lorsque la fonction est constante (ou, par extension, quasi-constante), un très faible nombre de tirages suffit à bien l'approximer. À l'opposé, si la fonction est définie par plusieurs expressions différentes dans l'espace des points de Monte-Carlo, il peut être difficile d'obtenir une bonne estimation. Dans le cas de la troisième courbe de la figure 5.1, la raison est le faible volume de l'espace dans lequel la fonction a une grande valeur. De façon générale, un fonction ayant ce comportement est difficile à approximer car il faut un grand nombre de tirages Monte-Carlo pour s'assurer de recouvrir toutes les zones de l'espace de façon suffisante.

Ainsi, lorsqu'on traite un problème qui, pour chaque x, estime une valeur par simulation Monte-Carlo, il est nécessaire d'analyser la boîte noire à priori pour déterminer le nombre minimal de tirages requis pour valider (empiriquement) l'hypothèse de loi normale.

Un tel pré-traitement permet également d'estimer la valeur de la constante déterminant la vitesse de convergence. Comme défini en 1.1, la moyenne empirique suit approximativement une loi $\mathcal{N}(\mu, cste/N)$, où μ est la moyenne réelle qu'on approxime et $cste$ est une constante dépendant de la quantité dont on prend la moyenne. L'estimation de cette constante peut être proposée à l'issue des tests de validation de l'hypothèse normale.

Dépendance de la loi au point d'évaluation

Comme nous venons de le rappeler, les problèmes réels prennent souvent un vecteur de variables en entrée, et l'utilisent pour construire un certain objet d'étude. Cela peut être par exemple un profil d'aile d'avion, calculé depuis un vecteur de variables correspondant à des dimensions ; un chiffre d'affaire généré par une stratégie d'achat dont les dates-clé sont les composantes du vecteur de variables ; un polygone dont l'aire dépend des variables définissant les coordonnées de ses sommets ; etc. Une fois cet objet construit, on peut en estimer la quantité voulue par simulation de Monte-Carlo (respectivement, la portance moyenne de l'aile ; les conséquences de quelques scénarios sur le chiffre d'affaire généré par la stratégie d'achat ; l'aire du polygone) pour ensuite déterminer quel jeu de variables crée l'objet le plus intéressant.

Ces trois exemples mettent en lumière un phénomène que nos hypothèses passent sous silence : notre fonction objectif est la quantité moyenne qu'on obtient en choisissant un jeu de variable, et cette quantité est estimée par la méthode de Monte-Carlo. Dans notre modèle, nous demandons à la fonction de renvoyer un estimé dont elle doit garantir la précision. En d'autres termes, notre modèle considère que la boîte noire est capable de générer la loi $f_\sigma(x)$ et d'en renvoyer une observation. Cependant, dans les cas réels, il est difficile voire impossible de générer cette loi. En effet, la quantité qu'on estime dépend de la valeur des variables x, ce qui

implique qu'on ne la connaît pas exactement puisque la fonction est une boîte noire. Pour «Polygons», par exemple, une simulation de Monte-Carlo sur un polygone de volume nul renverra 0, et ce de façon exacte. Pour un polygone quelconque, la loi dépend de son aire. On ne peut donc, par définition, pas connaître la loi puisque l'aire est exactement la quantité que l'on cherche à estimer. De façon générale, le manque d'information sur l'effet des variables empêche la boîte noire de connaître exactement la loi que suit l'estimé de Monte-Carlo en un point x quelconque. La dépendance au point x est plus profonde qu'en la seule valeur moyenne : la loi de la moyenne empirique de N tirages de Monte-Carlo dépend de l'espace dans lequel les tirages sont lancés, et cet espace dépend de x.

L'hypothèse implicite de notre cadre d'étude : «$\forall x,\ \forall \sigma$, la boîte noire est capable de générer un estimé de $f(x)$ suivant $\mathcal{N}\left(f(x), \sigma^2\right)$» paraît donc difficile à assurer dans la réalité. Pour contourner ce problème, il faut là encore analyser le problème à priori.

Dans le cas de «Polygons», par exemple, on peut facilement prouver que la variance de la simulation à N tirages sur un polygone Q donné a pour variance $\dfrac{Vol(Q)}{Vol(\mathcal{B}(Q))}\left(1 - \dfrac{Vol(Q)}{Vol(\mathcal{B}(Q))}\right)/N$, qu'on peut majorer par $\dfrac{1}{4}/N$. Ainsi, une solution très simple pour coller à l'hypothèse est de recevoir la valeur σ imposée, d'en déduire le nombre de tirages de Monte-Carlo requis $N = \left\lfloor \dfrac{1}{4\sigma} \right\rfloor$ et de renvoyer la moyenne empirique de la simulation à N tirages. En effet, cette moyenne suit une loi plus précise que celle demandée initialement, donc l'algorithme va surestimer l'amplitude du bruit numérique l'entourant. Ceci amènera à quelques calculs supplémentaires inutiles, mais ne peut fausser les résultats renvoyés par l'optimisation.

Dans le cas général, une idée analogue peut être adoptée en cherchant une borne supérieure des variances des estimés à N fixé et lorsque x varie. En d'autres termes, on peut chercher $C = \sup_{x \in \mathcal{X}} \{C(x)\}$, où $\forall x,\ C(x)/N$ est la variance de la moyenne empirique d'une estimation via N tirages de Monte-Carlo dans l'espace défini au point x (la susnommée constante *cste*). Ainsi, lorsque l'algorithme imposera un écart-type σ au point x, la boîte noire pourra lancer une estimation de $f(x)$ à $N = \left\lfloor \dfrac{C}{\sigma} \right\rfloor$ tirages et la renvoyer à l'optimiseur, qui sous-estimera sa qualité en la traitant comme une estimation d'écart-type σ.

5.2 Stratégie de préconditionnement du problème

Toutes ces analyses pré-optimisation peuvent être effectuées dans un même test, utile pour comprendre le problème que l'on cherche à résoudre. La stratégie présentée ici est heuristique, mais vise à déterminer toutes les quantités pratiques présentées ci-avant. Elle doit donc être lancée avant l'algorithme d'optimisation, car les valeurs qu'elle estime servent à formater la boîte noire pour la faire entrer dans le cadre théorique requis par l'optimisation.

L'idée est de se munir d'un ou plusieurs points x (on prendra par exemple le point x_0 pressenti pour être le point de départ de l'optimisation, ainsi que quelques-uns de ses voisins plus ou moins proches) et pour chacun de lancer plusieurs estimations Monte-Carlo à un même nombre N de tirages. Toutes les estimations obtenues suivent la même loi, qui doit être une normale $\mathcal{N}\left(f(x), C(x)/N\right)$ si l'hypothèse de loi normale est vérifiée.

Rigoureusement, on fixe donc un nombre N de tirages par estimation et un ensemble $\mathcal{P}$ de points auxquels on évaluera f. On propose ensuite un nombre r de répétitions effectuées à chaque point (au sens où $\forall x \in \mathcal{P}$, on effectuera r simulations de Monte-Carlo à N tirages). On obtient alors, $\forall x \in \mathcal{P},\ \mathcal{V}(x) = \left\{\widetilde{f}_1(x), \ldots, \widetilde{f}_r(x)\right\}$, où $\forall i \in [\![1; r]\!],\ \widetilde{f}_i(x)$ est l'estimation proposée par la $i^{\text{ème}}$ simulation Monte-Carlo à N tirages. La question est alors de déterminer la *p-valeur* de l'hypothèse de loi normale suivie par les $\widetilde{f}_i(x)$. Si elle est vérifiée, on a donc trouvé un nombre N de tirages pour lequel les approximations sont exploitables par le solveur. On peut alors proposer $C(x) \approx N \times \mathbb{V}\left(\left\{\widetilde{f}_i(x)\right\}\right)$.

Pour vérifier l'hypothèse de loi normale, divers tests statistiques existent. On peut par exemple penser au test de Kolmogorov-Smirnov, appliqué à l'hypothèse de loi normale aux paramètres estimés, en suivant l'article de Lilliefors (1967). En effet, ici, toujours en vertu du théorème central limite 1.1.6, les divers tirages devraient suivre une loi normale de paramètres approximatifs $\hat{\mu} = \dfrac{1}{r}\displaystyle\sum_{\widetilde{f} \in \mathcal{V}(x)} \widetilde{f}$ et $\hat{\sigma}^2 = \dfrac{1}{r}\displaystyle\sum_{\widetilde{f} \in \mathcal{V}(x)} (\widetilde{f} - \hat{\mu})^2$. Le test va alors déterminer la viabilité de l'hypothèse «La moyenne empirique de r approximations Monte-Carlo à N tirages de $f(x)$ suit une loi $\mathcal{N}\left(\hat{\mu}, \hat{\sigma}^2\right)$».

Il suffit alors de chercher $N_m(x)$, la plus petite valeur de N satisfaisant le test, en répétant ces r tirages pour diverses valeurs de N. Ainsi, $N_m(x)$ est le nombre minimal de tirages de Monte-Carlo requis pour rendre viable l'hypothèse normale au point x. De là, $N_m = \max\limits_{x \in \mathcal{P}} N_m(x)$ est la plus petite valeur de N respectant le test en chaque point x de $\mathcal{P}$. Elle peut être considérée comme le nombre minimal de tirages Monte-Carlo requis pour rendre viable l'hypothèse normale sur l'ensemble de l'espace des variables.

Une fois N_m trouvé, on peut calculer $\{C(x) \mid x \in \mathcal{P}\}$ et estimer $C = \sup\limits_{x \in \mathcal{X}} C(x) \approx \sup\limits_{x \in \mathcal{P}} C(x)$.

On propose alors la procédure de pré-traitement suivante. Ici, m_{inf} et m_{sup} détermineront les valeurs de N utilisées, via $N \in \{2^m \mid m \in [\![m_{inf}; m_{sup}]\!]\}$. On prendra également un unique point de test x_0 et créera les autres dans la boîte $[l; u]$ (cette approche pourrait être remplacée par des points générés par hypercube latin dans la boîte). La procédure est :

Algorithme 13 Pré-traitement d'un problème

fonction CALCUL_TIRAGES_X(f ; x ; m_{inf}, m_{sup} ; r)
    $Tirages \leftarrow$Tableau vide à $m_{sup} - m_{inf} + 1$ éléments
    pour $m \in [\![m_{inf}; m_{sup}]\!]$ **faire**
        $N \leftarrow 2^m$
        $\mathcal{V}_N \leftarrow \underset{i \in [\![1;r]\!]}{\cup} \left\{\widetilde{f}_i(x)\right\}$, r estimations de $f(x)$, chacune issue de N tirages Monte-Carlo
        $Tirages(m - m_{inf}) \leftarrow \{N; \mathcal{V}_N\}$
    renvoyer $Tirages$

fonction CREATION_VOISIN(x ; p, k ; l, u)
    $n \leftarrow Card(x)$ le nombre de composantes des vecteurs x, l et u
    $y \leftarrow \vec{0}$
    pour $i \in [\![1; n]\!]$ **faire**
        $a \leftarrow (u_i - l_i) \times Uniforme(-1; 1)$, une valeur aléatoire dans $[-(u_i - l_i); (u_i - l_i)]$
        $y_i \leftarrow \min\left(\max\left(l_i, x_i + \frac{p}{k}a\right), u_i\right)$
    renvoyer y

fonction PRE_TRAITEMENT(f ; x_0 ; m_{inf}, m_{sup} ; r ; l, u ; k)
    $X \leftarrow \underset{i \in [\![0,k]\!]}{\cup} \{Creation\_Voisin(x_0; i, k; l, u)\}$
    $Donnees \leftarrow \underset{x \in X}{\cup} \{x; Calcul\_Tirages\_X(f; x; m_{inf}, m_{sup}; r)\}$
    pour $i \in [\![0, Card(X) - 1]\!]$ **faire**
        $(x; Tirages) \leftarrow Donnees(i)$
        $m \leftarrow m_{inf}$
        tant que $(m \leq m_{sup})$ **et non**$(Kolmogorov\_Smirnov(Tirages(m - m_{inf}))$ **faire**
            $m \leftarrow m + 1$
        si $m \leq m_{sup}$ **alors** $N_m(i) \leftarrow 2^m$ **sinon** $N_m(i) \leftarrow +\infty$
    $N_m \leftarrow \max\{N_m(i) \mid i \in [\![0, Card(X) - 1]\!]\}$
    $Tirages \leftarrow \{Tirages(m - m_{inf}) \mid (x, Tirages) \in Donnees(i),\ i \in [\![0, Card(X) - 1]\!]\}$
    $C \leftarrow \max\{N_m \times \mathbb{V}(T) \mid T \in Tirages\}$
    renvoyer (N_m, C)

5.3 Contrôle de l'évolution de la précision

Dans le cadre théorique proposé, la gestion de la précision demandée à la boîte noire exploite trois intermédiaires.

En premier lieu, l'écart-type du bruit est imposé, via la valeur σ générant la variable aléatoire $f_\sigma(x)$ qu'on demande à la boîte noire d'observer. Les discussions nécessaires liées à ce sujet ont été proposées ci-avant en section 5.1.

En plus des écarts-types, une autre mécanique est utilisée : un indice arbitraire de précision, r, et la fonction ρ faisant le lien entre un indice r donné et un écart-type σ à imposer à la boîte noire. On peut aussi penser, dans le cas de la variante à précision dynamique de *MADS* (algorithme 11), à la mécanique d'évolution de la valeur de r au fil des itérations. Tous ces éléments ont un réel impact sur les performances pratiques, et seront donc discutés ici.

Fonction ρ

Cette fonction n'est que très peu décrite dans le cadre théorique précédent. La seule contrainte qui lui est imposée est sa décroissance sur $\mathbb{R}$ depuis une borne supérieure vers une limite positive. Ces contraintes font sens puisque :

- Si ρ n'est pas bornée supérieurement, faire tendre $r \to -\infty$ donnerait une variance infinie à la boîte noire, ce qui ferait échouer l'hypothèse de loi normale pratique. La théorie se ferait également impacter par le fait que les estimés n'auraient plus aucune viabilité lorsque donnés depuis une variance qui augmente indéfiniment, au point que le maximum de vraisemblance pourrait ne plus converger vers la valeur qu'il approxime.
- La positivité est simplement issue du fait qu'un écart-type négatif n'a pas de signification. ρ doit donc être positive pour que $\sigma = \rho(r)$ soit interprétable comme un écart-type imposé à la boîte noire.
- ρ doit décroître pour qu'une augmentation de l'indice r de précision se traduise par une diminution de l'écart-type $\sigma = \rho(r)$ du bruit.

Bien que ce ne soit pas obligatoire, on la choisira continue et relativement lisse. Cela aidera à garantir qu'une légère augmentation de la précision n'impose pas de diminution drastique de l'écart-type imposé. Si cela n'est pas vérifié, les conséquences pratiques peuvent être grandes. Par exemple, sur les méthodes à précision monotone : lorsqu'on détecte que la précision actuelle est insuffisante, la stratégie consiste à augmenter $r^{k+1} = r^k + 1$, pour que (dans l'idée) l'amplitude du bruit diminue légèrement. Si la fonction est très raide au voisinage de r^k, les itérations $k+1$ et suivantes vont être faites avec un écart-type σ^{k+1} bien plus faible que σ^k. Le temps de calcul requis pour achever une itération va donc croître démesurément entre les

itérations k et $k+1$, rendant l'analyse à posteriori de l'optimisation difficile.

Le choix de fonction retenu dans l'implémentation des algorithmes a été déterminé par l'envie d'estimer l'évolution des écarts-types réels imposés à la boîte noire. Il peut être souhaitable, si on connaît le problème réel, de savoir approximativement comment va évoluer la précision. Pour simplifier ce contrôle, on peut par exemple choisir :

$$\rho : \left\{ \begin{array}{ccl} \mathbb{Z} & \rightarrow & [0,1] \\ r & \rightarrow & \left\{ \begin{array}{lll} 2^{-r/2} & \text{sur} & +\mathbb{N} \\ 2-2^{r/2} & \text{sur} & -\mathbb{N}^* \end{array} \right. \end{array} \right. .$$

Avec ce choix, on peut notamment affirmer que la méthode à précision monotone va diviser l'amplitude du bruit par 2 toutes les deux itérations échouées. Avec la méthode de précision dynamique, une relaxation de précision à un indice $r < 0$ va rapidement amener l'amplitude du bruit proche de sa valeur maximale (ici, 1) pour ne pas perdre d'itérations à inutilement sur-consommer lorsqu'on voudrait (et pourrait) effectuer nos calculs sous un bruit important.

Cette fonction pourrait être adaptée en prenant des valeurs différentes de $2^{\pm r/2}$. Par exemple, avec 10^r sur $r \in \mathbb{N}$, le bruit serait atténué de 10 dB à chaque itération échouée. De même, on peut décaler le point d'inflexion à une valeur $r_0 \neq 0$, modifier l'amplitude maximale à σ_{max} et changer la valeur θ dont r est l'exposant :

$$\rho : \left\{ \begin{array}{ccl} \mathbb{Z} & \rightarrow & [0, \sigma_{max}] \\ r & \rightarrow & \left\{ \begin{array}{lll} (\sigma_{max}/2)\, \theta^{-(r-r_0)} & \text{sur} & [r_0, +\infty) \\ (\sigma_{max}/2) \left(2-\theta^{(r-r_0)}\right) & \text{sur} & (-\infty, r_0) \end{array} \right. \end{array} \right. .$$

Sous ces notations, $\theta = \sqrt{2},\ \sigma_{max} = 1,\ r_0 = 0$ redonnent à ρ la forme proposée ci-avant.

Enfin, la théorie n'impose pas que $\lim_{r \to +\infty} \rho(r) = 0$. Ceci est couramment supposé lorsque ρ fait le lien entre précision et nombre de tirages Monte-Carlo, mais la méthode de précision dynamique ne requiert pas de telle hypothèse. De plus, dans les cas pratiques, on pourrait souhaiter que le solveur n'impose pas de bruit trop faible, car cela générerait des calculs par évaluation trop coûteux. On va donc modifier ρ pour qu'elle reste au-dessus d'un seuil σ_{min}.

Ainsi, ρ peut être personnalisée via les paramètres suivants, ou laissée avec ceux par défaut :

θ	$\in$	$(1, +\infty)$	l'atténuation du bruit lorsque r augmente ($= 1.1;\ \sqrt{2};\ 2;\ 10^{1/10},\ \ldots$)
r_0	$\in$	$\mathbb{Z}$	l'indice du point d'inflexion (forte croissance de σ pour $r \leq r_0$) ($= 0$)
σ_{max}	$\in$	$\mathbb{R}^{+^*}$	l'écart-type maximal qu'on peut soumettre à la boîte noire ($= 1$)
σ_{min}	$\in$	$[0, \sigma_{max})$	l'écart-type minimal qu'on peut soumettre à la boîte noire ($= 0$)

Ces paramètres conduisent à la forme générique de ρ :

$$\rho : \begin{cases} \mathbb{Z} & \rightarrow & [\sigma_{min}, \sigma_{max}] \\ r & \rightarrow & \begin{cases} \sigma_{min} + \dfrac{\sigma_{max} - \sigma_{min}}{2} \theta^{-(r-r_0)} & \text{sur} \quad [r_0, +\infty) \\ \sigma_{min} + \dfrac{\sigma_{max} - \sigma_{min}}{2} \left(2 - \theta^{(r-r_0)}\right) & \text{sur} \quad (-\infty, r_0) \end{cases} \end{cases} .$$

L'interprétation graphique de ρ (figure 5.2) illustre l'effet des paramètres la composant :

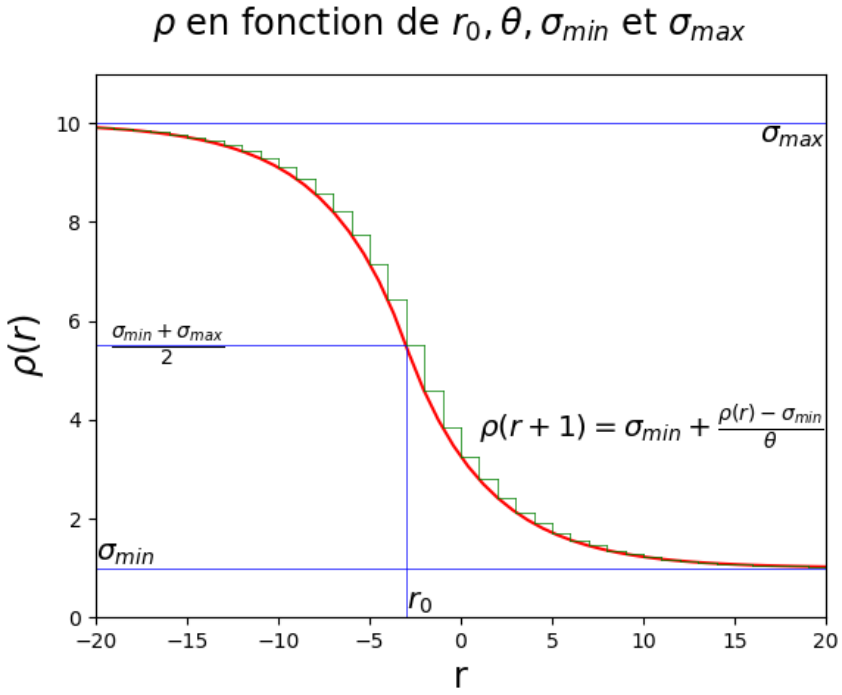

Figure 5.2 Fonction ρ pour $r_0 = -3, \theta = 10^{1/10}, \sigma_{min} = 1, \sigma_{max} = 10$.

Raffinement et relaxation de l'indice de précision

Avec la stratégie de raffinement monotone de la précision (algorithme 9), on peut paramétrer la totalité des effets stochastiques via la seule fonction ρ. En effet, cette stratégie se contente d'augmenter l'indice de précision r de 1 à chaque échec, depuis $r = 0$. Il est donc possible de paramétrer ρ pour connaître exactement l'effet d'une itération échouée sur la précision des suivantes. Seul le paramètre α restera libre. Celui-ci influe sur la valeur à laquelle on force le cadre de sonde δ_p après un échec (on le force à être $\leq \sqrt{\mu(\sigma)^\alpha}$). Il est utile de prendre α grand (proche de 1) sur des problèmes où l'espace est petit et la gestion du bruit coûteuse.

Dans le cas de la stratégie de précision dynamique (11), les choses changent. L'indice de

précision peut varier à la hausse comme à la baisse, mais surtout avec des variations pouvant ne pas être unitaires. Avec *NM*, par exemple, c'est le ratio des diamètres du simplexe (cf l'algorithme 12) qui joue le rôle de r. Ses variations n'ont aucune raison d'être unitaires. De même, avec *MADS*, l'algorithme tel que présenté en (11) permet à l'indice r^{k+1} d'être entre $r^k - 3$ et $r^k + 3$, selon la qualité des conclusions qu'on a pu tirer à l'issue de l'itération k.

Le choix de la façon dont r varie peut avoir un grand impact, et doit être décidé conjointement avec ρ. Par exemple, si ρ atténue le bruit de dix décibels par augmentation de r (dans le cas où $\theta = 10$), une augmentation de r de trois unités ferait grandement augmenter les efforts de calcul requis par l'itération suivante. Dans des cas pratiques, cela peut poser problème. En d'autres termes, il faut porter une grande attention à la façon dont r^k est actualisé au fil des itérations, et à l'impact que ces changements auront sur l'amplitude du bruit.

En actualisant r, l'idée sous-jacente est d'augmenter la précision si les calculs sont trop imprécis, et de la diminuer lorsqu'on constate qu'on a été plus précis que nécessaire. On cherche, à chaque itération, à effectuer les calculs à l'indice r tel que $\sigma = \rho(r)$ soit le plus grand bruit n'offrant pas de résultats trop erronés. Mais pour cela, le solveur ne calcule pas la valeur exacte de cette précision théoriquement attendue. À défaut, on ne fait donc qu'augmenter ou diminuer r pour que $\rho(r)$ suive la même tendance.

Cette décision est calculée comme suit : à l'issue de l'itération k, on connaît x_*^k la meilleure solution courante, et l'estimé $f^k(x_*^k)$ de sa valeur, imprécis à un écart-type $\sigma^k(x_*^k)$. L'itération a généré le point candidat x_n^k, de valeur estimée $f^k(x_n^k)$ à l'écart-type $\sigma^k(x_n^k)$. Pour déterminer si x_n^k est meilleur que x_*^k, on doit décider si $f(x_n^k)$ est plus bas que $f(x_n^*)$, malgré l'incertitude entourant ces deux valeurs. Pour cela, on calcule donc la *p-valeur* de l'hypothèse «$f(x_n^k) < f(x_*^k)$», connaissant leurs estimés.

Si cette *p-valeur* est proche de 0, on peut conclure de façon presque certaine que l'itération a été un échec. On a probablement effectué des calculs plus précis que nécessaire, et aurait pu conclure de façon aussi satisfaisante malgré une incertitude plus élevée. Par conséquent, il semble possible de relaxer la précision aux itérations suivantes. De même, si la *p-valeur* est proche de 1, l'itération est presque certainement une réussite, et les mêmes conclusions tiennent quand à la précision nécessaire aux prochaines itérations. Par contre, si la *p-valeur* est relativement proche de 0.5, il semble difficile de conclure : l'hypothèse «$f(x_n^k) < f(x_*^k)$» est presque aussi viable que sa complémentaire. On doit donc augmenter la précision pour espérer avoir des résultats viables aux prochaines itérations.

Nous traduisons ceci par la variation de r suivante, en notant p la *p-valeur* de « $f(x_n^k) < f(x_*^k)$ » :

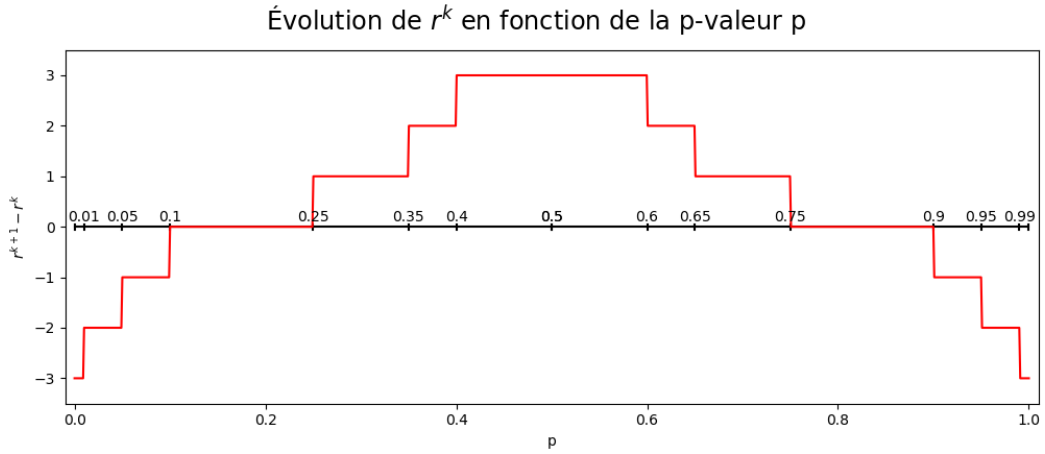

Figure 5.3 Évolution de r^k selon les résultats de l'itération k.

Ce choix est bien évidemment arbitraire, mais reflète l'idée d'augmenter ou diminuer la précision en fonction de la surconsommation observée à l'itération k. Avec ce choix, on considère que la précision peut rester constante lorsque l'une des *p-valeur* est entre 75% et 90%, car cela traduit qu'on peut déterminer lequel des points x_*^k et x_n^* est le meilleur à partir de nos estimés, et que cette déduction est plausible à p%. Si l'une des deux *p-valeurs* tend vers 1 (si elle devient plus grande que 0.9), on considère avoir été plus précis que nécessaire et rabaisse donc la précision. La baisse est d'autant plus marquée que les *p-valeurs* sont proches de 0 et 1. À l'opposé, lorsqu'on ne peut rien conclure (les deux hypothèses ont une viabilité presque égales), on augmente la précision.

Si cette stratégie de variation de r est choisie, il faut adapter ρ en conséquence pour s'assurer que $\rho(r+3)$ n'est jamais trop faible comparé à $\rho(r)$, puisque l'augmentation de précision pourrait autrement générer soudainement des calculs beaucoup trop longs. Le compromis proposé ci-avant tient compte de cette remarque ; par exemple $\theta = 10^{0.1}$ diminue le bruit de $1dB$ par augmentation unitaire de r. Ainsi, le bruit diminue au maximum de $3dB$ par itération, ce qui limite les trop grandes variations soudaines des temps de calcul par itération.

On pourrait aussi envisager, par exemple, une variation continue de r pour ne plus se restreindre à des valeurs discrètes ; ou une évolution beaucoup plus marquée (de $r^k - 10$ à $r^k + 10$ par exemple), et prendre une fonction ρ bien plus lisse (avec un θ proche de 1) pour limiter l'amplitude des variations de l'écart-type σ imposé aux estimations d'une itération à l'autre.

Amélioration des estimés en des points déjà visités

Dans les algorithmes théoriques, la convergence est assurée par une fonction ré-estimant des points déjà visités, à chaque itération. L'objectif de cette fonction est de garantir que la variance de chaque estimé tend vers 0 via la loi des grands nombres, afin de garantir à terme que la relation d'ordre entre les estimés suit la véritable relation d'ordre sur f. Cette fonction est évidemment inutilisable en pratique, puisque la multiplication des points à réévaluer rend les efforts de calcul nécessaires trop importants.

Ainsi, on s'autorisera dans les implémentations à limiter le nombre de points à réévaluer, pour ne se concentrer que sur ceux qui ont encore une chance d'être intéressants. Comme nous l'indiquons en commentaire dans l'algorithme 11, la fonction *ReestimeCache* considère tous les points de la cache $\mathcal{V}^k$, mais ne les réévalue que s'il y a toujours plus de 10% de chance qu'ils battent l'optimum courant. En d'autres termes, les points d'estimés peu précis mais très mauvais ne sont pas réévalués, puisque l'écart entre leur valeur et celle de l'optimum courant est tel qu'il est très peu probable que ce soit uniquement dû au bruit stochastique. En d'autres termes, investir de nouveaux efforts de calcul en ces points n'apportera pas de gain, puisqu'il ne peuvent statistiquement pas améliorer l'optimum courant.

Ce choix, qui limite grandement la consommation de tirages Monte-Carlo dédiés à la réévaluation d'anciens points, détruit cependant la propriété de convergence théorique. Cela dit, cet effet néfaste ne s'est jamais manifesté sur l'ensemble des tests menés avec notre implémentation.

On peut de plus discuter de l'effort de calcul à investir dans chacun des points qu'on décide de réévaluer. Deux solutions sont possibles : les réévaluer à une précision constante, ou dépendante de la précision r^k utilisée à l'itération k où on se trouve. Nous privilégions la seconde stratégie dans notre implémentation, avec une précision demandée au nouvel estimé de $r^k - 5$. On pourrait également prendre, par exemple, $r = -\infty$ pour que l'écart-type du bruit du nouvel estimé soit σ_{max}.

Il peut être bon d'expérimenter divers jeux de paramètres pour spécifier ρ, la variation de précision r^{k+1} en fonction de r^k, et l'écart-type imposé à la réévaluation des points de la cache. On peut observer des gains importants avec de bons paramètres adaptés au problème.

Un bon indicateur de ces gains peut par exemple être le pourcentage de tirages Monte-Carlo consommés à la réévaluation de points déjà dans la cache, qui doit être faible pour optimiser nos ressources.

CHAPITRE 6 CONCLUSION ET RECOMMANDATIONS

À l'issue de ce mémoire, un résumé de l'ensemble des discussions proposées s'impose. Nous tenterons ici de synthétiser les points les plus importants et à retenir en priorité.

6.1 Synthèse des travaux

Partant d'un cadre général de problèmes que nous souhaitions résoudre, nous avons pu constater que les stratégies «naïves» ne sont pas appropriées. Il y a un besoin de proposer des méthodes plus avancées, et exploitant mieux les hypothèses faites sur le problème. Le mot-clé *précision variable* est important ici, et permet diverses approches.

En premier lieu, nous avons suivi le paradigme dominant dans la littérature : la stratégie dite «monotone». Une fois adaptée à notre contexte et à l'algorithme *MADS*, elle devient fonctionnelle. Cependant, elle requiert beaucoup d'efforts calculatoires sur des problèmes lourds, et a une tendance naturelle à surconsommer à l'approche d'un optimum. À l'opposé, notre autre stratégie (dite «dynamique») à une tendance à la consommation de budget de calcul bien plus faible, pour des résultats peu détériorés. Mis à part certains cas précis où son manque de maturité se fait encore ressentir, elle nous semble globalement prometteuse. Enfin, nous avons proposé une implémentation sur *MADS* de ces deux stratégies, nous aidant à offrir des garanties théoriques de convergence vers un optimum, et une efficacité pratique issue de la performance naturelle de cette mécanique.

6.2 Limitations de la solution proposée

Comme les tests numériques l'ont prouvé, la stratégie dynamique peut encore être améliorée. Les principaux défauts sont selon nous la consommation de tirages Monte-Carlo dédiée à la réévaluation de points de la cache. Elle est actuellement très élevée (près de 80 % sur «Snake» et 50 % sur «Norm2»), ce qui semble être un gâchis important de tirages Monte-Carlo. De plus, la stratégie dynamique actuelle se perd encore dans son incertitude latente lorsque beaucoup de points de la cache ont des images par la fonction réelle très proches.

6.3 Améliorations futures

En premier lieu, il faudrait étudier plus en détail les algorithmes *NM*, pour notamment les tester en pratique. Les quelques tests que nous avons effectué n'étaient pas particulièrement

convaincants et n'ont pas été discutés ici. Il y a probablement beaucoup d'améliorations pratiques possibles pour les rendre bien plus efficaces (notamment en discutant plus en détail la fonction passant du diamètre courant du simplexe à une précision imposée aux futurs calculs).

De plus, la méthode principale proposée dans ce document manque encore de maturité. Nous entendons en ce sens qu'il est probablement possible de grandement améliorer ses performances pratiques en spécifiant mieux ses divers paramètres en fonction du problème réel considéré, car les valeurs par défaut peuvent ne pas être totalement appropriés pour tous les problèmes. Il faudrait également munir la stratégie d'un moyen de prédire plus efficacement si un point doit être réévalué ou non.

L'implémentation d'un modèle dynamique de l'objectif, calculé à partir des estimés et tenant compte de l'incertitude, pourrait être une bonne amélioration. Avec un tel modèle, il serait bien plus aisé de détecter en temps réel si un estimé est grandement erroné ; on pourrait ainsi le corriger directement, avant qu'il ne perturbe l'optimisation. Il pourrait également être pertinent de modifier notre contrôle de la précision imposée à chaque itération, car la rendre plus élevée limiterait le besoin de réévaluation lors des itérations suivantes.

En bref, l'exploitation du compromis «grands efforts de calcul instantanés et besoins moindres de réévaluations - calculs moins précis évitant d'inutiles efforts dans les zones inintéressantes» pourrait être mieux exploité. Il serait également intéressant de discuter de nouvelles façons de l'exploiter lorsqu'on dispose de plusieurs processeurs pouvant tourner en parallèle : une stratégie plus agressive en terme de calculs est alors envisageable, tout comme le fait de dédier certains processeurs au travail de réévaluation dans la cache. De manière générale, notre stratégie pourrait probablement être grandement améliorée avec l'ajout du calcul parallèle.

Enfin, la gestion des contraintes (bruitées ou non) pourrait être une avancée algorithmique intéressante. La théorie peut aussi être renforcée par l'ajout d'autres distributions du bruit. Pour tester tous les ajouts, on pourrait également envisager un plus grand nombre de problèmes plus diversifiés.

Ces axes d'amélioration requièrent encore un travail certain, et non trivial. Maintenant que le cadre théorique est posé, l'optimisation pratique de la méthode peut probablement être poussée plus loin qu'à son état actuel. La théorie peut également être enrichie d'ajouts intéressants.

RÉFÉRENCES

M.A. Abramson, C. Audet, G. Couture, J.E. Dennis, Jr., S. Le Digabel et C. Tribes : The NOMAD project. Software available at https://www.gerad.ca/nomad, 2015. URL https://www.gerad.ca/nomad.

C. Audet et J.E. Dennis, Jr. : Analysis of generalized pattern searches. *SIAM Journal on Optimization*, 13(3):889–903, 2003. URL http://dx.doi.org/doi:10.1137/S1052623400378742.

C. Audet et J.E. Dennis, Jr. : Mesh Adaptive Direct Search Algorithms for Constrained Optimization. *SIAM Journal on Optimization*, 17(1):188–217, 2006. URL http://dx.doi.org/doi:10.1137/040603371.

C. Audet et J.E. Dennis, Jr. : A Progressive Barrier for Derivative-Free Nonlinear Programming. *SIAM Journal on Optimization*, 20(1):445–472, 2009. URL http://dx.doi.org/10.1137/070692662.

C. Audet, P. Hansen, F. Messine et J. Xiong : The largest small octagon. *J. Combin. Theory Ser. A*, 98(1):46–59, 2002. ISSN 0097-3165.

C. Audet et W. Hare : *Derivative-Free and Blackbox Optimization*. Springer Series in Operations Research and Financial Engineering. Springer International Publishing, Berlin, 2017. URL https://dx.doi.org/10.1007/978-3-319-68913-5.

C. Audet, A. Ihaddadene, S. Le Digabel et C. Tribes : Robust optimization of noisy blackbox problems using the Mesh Adaptive Direct Search algorithm. *Optimization Letters*, 12(4):675–689, 2018. URL https://doi.org/10.1007/s11590-017-1226-6.

C. Audet et C. Tribes : Mesh-based Nelder-Mead algorithm for inequality constrained optimization. Rapport technique G-2017-90, Les cahiers du GERAD, 2017. URL https://www.gerad.ca/fr/papers/G-2017-90.

Vahid Beiranvand, Warren Hare et Yves Lucet : Best practices for comparing optimization algorithms. *Optimization and Engineering*, 06 2017.

Andrew C. Berry : The accuracy of the Gaussian approximation to the sum of independent variates. *Trans. Amer. Math. Soc.*, 49:122–136, 1941. ISSN 0002-9947. URL https://doi.org/10.2307/1990053.

A.J. Booker, J.E. Dennis, Jr., P.D. Frank, D.B. Serafini, V. Torczon et M.W. Trosset : A Rigorous Framework for Optimization of Expensive Functions by Surrogates. *Structural and Multidisciplinary Optimization*, 17(1):1–13, 1999. URL http://dx.doi.org/10.1007/BF01197708.

K.H. Chang : Stochastic nelder-mead simplex method - a new globally convergent direct search method for simulation optimization. *European Journal of Operational Research*, 220 (3):684–694, 2012. URL http://dx.doi.org/10.1016/j.ejor.2012.02.028.

R. Chen, M. Menickelly et K. Scheinberg : Stochastic optimization using a trust-region method and random models. *Mathematical Programming*, 169(2):447–487, Jun 2018a. ISSN 1436-4646. URL https://doi.org/10.1007/s10107-017-1141-8.

X. Chen, C. Kelley, F. Xu et Z. Zhang : A smoothing direct search method for monte carlo-based bound constrained composite nonsmooth optimization. *SIAM Journal on Scientific Computing*, 40(4):A2174–A2199, 2018b. URL https://doi.org/10.1137/17M1116714.

F.H. Clarke : *Optimization and Nonsmooth Analysis*. John Wiley & Sons, New York, 1983. URL http://www.ec-securehost.com/SIAM/CL05.html. Reissued in 1990 by SIAM Publications, Philadelphia, as Vol. 5 in the series Classics in Applied Mathematics.

A.R. Conn, N.I.M. Gould et Ph.L. Toint : *Trust-Region Methods*. MPS-SIAM Series on Optimization. SIAM, 2000. URL http://www.numerical.rl.ac.uk/trbook/trbook.html.

E.D. Dolan et J.J. Moré : Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002. URL http://dx.doi.org/10.1007/s101070100263.

Rick Durrett : *Probability : Theory and Examples*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 4 édition, 2010.

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

R.L. Graham : The largest small hexagon. *Journal of Combinatorial Theory, Series A*, 18:165–170, 1975.

Didier Henrion et Frederic Messine : Finding largest small polygons with gloptipoly. *Journal of Global Optimization*, 56, 03 2011.

D.R Jones, M. Schonlau et W.J. Welch : Efficient Global Optimization of Expensive Black Box Functions. *Journal of Global Optimization*, 13(4):455–492, 1998. URL http://dx.doi.org/10.1023/A:1008306431147.

Jack P.C. Kleijnen : Kriging metamodeling in simulation : A review. *European Journal of Operational Research*, 192(3):707 – 716, 2009. ISSN 0377-2217. URL http://www.sciencedirect.com/science/article/pii/S0377221707010090.

Hubert W. Lilliefors : On the kolmogorov-smirnov test for normality with mean and variance unknown. *Journal of the American Statistical Association*, 62(318):399–402, 1967. URL https://amstat.tandfonline.com/doi/abs/10.1080/01621459.1967.10482916.

K.I.M. McKinnon : Convergence of the Nelder-Mead simplex method to a nonstationary point. *SIAM Journal on Optimization*, 9(1):148–158, 1998. URL https://dx.doi.org/10.1137/S1052623496303482.

J.J. Moré et S.M. Wild : Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20(1):172–191, 2009. URL http://dx.doi.org/10.1137/080724083.

J. A. Nelder et R. Mead : A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965. URL http://dx.doi.org/10.1093/comjnl/7.4.308.

Courtney Paquette et Katya Scheinberg : A Stochastic Line Search Method with Convergence Rate Analysis. *arXiv e-prints*, page arXiv :1807.07994, Jul 2018.

Victor Picheny, David Ginsbourger, Yann Richet et Grégory Caplin : Quantile-based optimization of Noisy Computer Experiments with Tunable Precision. URL https://hal.archives-ouvertes.fr/hal-00578550. working paper or preprint, mars 2012.

E. Polak et M. Wetter : Precision control for generalized pattern search algorithms with adaptive precision function evaluations. *SIAM Journal on Optimization*, 16(3):650–669, 2006. URL http://dx.doi.org/10.1137/040605527.

K. Reinhardt : Extremale polygone gegebenen durchmessers. *Jahresber. Deutsch. Math. Verein*, 31:251–270, 1922.

Michael James Sasena : *Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations*. Thèse de doctorat, University of Michigan, 2002.

Sara Shashaani, Fatemeh Hashemi et Raghu Pasupathy : ASTRO-DF : A Class of Adaptive Sampling Trust-Region Algorithms for Derivative-Free Stochastic Optimization. *arXiv e-prints*, page arXiv :1610.06506, Oct 2016.

V. Torczon : On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7(1):1–25, 1997. URL http://dx.doi.org/10.1137/S1052623493250780.

P. Tseng : Fortified-Descent Simplicial Search Method : A General Approach. *SIAM Journal on Optimization*, 10(1):269–288, 1999. URL http://dx.doi.org/10.1137/S1052623495282857.

ANNEXE A MAXIMUM DE VRAISEMBLANCE D'UNE SUITE D'OBSERVATIONS DE LOIS NORMALES DE MÊME MOYENNE

Supposons ici que l'on dispose d'un réel μ inconnu, et que l'on souhaite identifier via des observations de lois normales centrées en μ. Plus précisément, on suppose avoir des variables aléatoires indépendantes $X_1, \dots, X_n$ suivant respectivement des lois $\mathcal{N}(\mu, \sigma_1^2), \dots, \mathcal{N}(\mu, \sigma_n^2)$ (où les σ_i sont connus) et $x_1, \dots, x_n$ une observation de chacune de ces variables aléatoires.

Au vu des observations $x_1, \dots, x_n$, on peut estimer la valeur de μ via la technique du *maximum de vraisemblance*. Cette méthode consiste à calculer la plausibilité de l'échantillon observé, sous l'hypothèse que μ a une valeur particulière qu'on lui impose. Si la valeur proposée dans l'hypothèse est trop différente de la vraie, l'échantillon observé aurait été très peu probable à priori. Le maximum de vraisemblance calcule donc une estimation de μ maximisant la probabilité à priori d'obtenir l'échantillon réellement observé.

Pour cela, il cherche la valeur de μ maximisant la densité de probabilité en $(x_1, \dots, x_n)$. L'expression de l'estimé $\hat{\mu}$ de μ par maximum de vraisemblance se calcule donc comme suit :

$$\begin{aligned}
\mu &\approx \hat{\mu} \\
&\in \operatorname*{arg\,max}_{\mu \in \mathbb{R}} \left\{ \mathbb{P}\left(\forall i \in [\![1, n]\!] :\ \mu + \epsilon_i = x_i \mid \epsilon_i \sim \mathcal{N}(0, \sigma_i^2) \text{ et } x_i, \sigma_i \text{ connus} \right) \right\} \\
&= \operatorname*{arg\,max}_{\mu \in \mathbb{R}} \left\{ \prod_{i=1}^{n} \exp\left(\frac{-1}{2} \left(\frac{x_i - \mu}{\sigma_i} \right)^2 \right) \frac{1}{\sqrt{2\pi}\sigma_i} \right\} \\
&= \operatorname*{arg\,max}_{\mu \in \mathbb{R}} \underbrace{\exp\left(\frac{-1}{2} \sum_{i=1}^{n} \left(\frac{x_i - \mu}{\sigma_i} \right)^2 \right)}_{\frac{d}{d\mu} = \frac{-1}{2} \sum_{i=1}^{n} \frac{2\mu - 2x_i}{\sigma_i^2} \exp\{\dots\}} \underbrace{\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_i}}_{\text{cste}} \\
\mu &= \frac{\sum_{i=1}^{n} x_i/\sigma_i^2}{\sum_{i=1}^{n} 1/\sigma_i^2}.
\end{aligned}$$

La dernière ligne vient du fait que l'exponentielle apparaissant dans l'arg max est $\mathcal{C}^2$, et qu'au vu de sa dérivée elle est croissante puis décroissante. L'ensemble arg max considéré est donc restreint à un unique point, et la fonction n'est donc maximale qu'au point annulant ladite dérivée.

On peut s'interroger sur la qualité de cet estimé. On peut montrer qu'il n'est pas biaisé :

$$\mathbb{E}(\hat{\mu}) \underset{\text{linéarité}}{=} \frac{\sum_{i=1}^{n} \mathbb{E}(X_i)/\sigma_i^2}{\sum_{i=1}^{n} 1/\sigma_i^2} \underset{\forall i,\ \mathbb{E}(X_i)=\mu}{=} \frac{\sum_{i=1}^{n} \mu/\sigma_i^2}{\sum_{i=1}^{n} 1/\sigma_i^2} = \mu.$$

De plus, sa variance peut également être calculée :

$$\begin{aligned}
\mathbb{V}(\hat{\mu}) &\underset{\mathbb{V}(aX)=a^2\mathbb{V}(X)}{=} \left(\frac{1}{\sum_{i=1}^{n} 1/\sigma_i^2}\right)^2 \mathbb{V}\left(\sum_{i=1}^{n} X_i/\sigma_i^2\right) \\
&\underset{\text{Somme de VA indép}}{=} \left(\frac{1}{\sum_{i=1}^{n} 1/\sigma_i^2}\right)^2 \sum_{i=1}^{n} \mathbb{V}\left(X_i/\sigma_i^2\right) \\
&\underset{\mathbb{V}(aX)=a^2\mathbb{V}(X)}{=} \left(\frac{1}{\sum_{i=1}^{n} 1/\sigma_i^2}\right)^2 \sum_{i=1}^{n} \mathbb{V}\left(X_i\right)/\sigma_i^4 \\
&\underset{\forall i,\ \mathbb{V}(X_i)=\sigma_i^2}{=} \left(\frac{1}{\sum_{i=1}^{n} 1/\sigma_i^2}\right)^2 \sum_{i=1}^{n} 1/\sigma_i^2 \\
&= \left(\sum_{i=1}^{n} \frac{1}{\sigma_i^2}\right) / \left(\sum_{i=1}^{n} \frac{1}{\sigma_i^2}\right)^2 \\
&= 1/\sum_{i=1}^{n} \frac{1}{\sigma_i^2}.
\end{aligned}$$

Cette expression met en évidence un comportement déjà énoncé par la loi des grands nombres. Lorsque le nombre d'observations tend vers l'infini, la variance de l'estimé tend vers 0, et sa valeur limite sera presque sûrement la vraie valeur de μ.

Enfin, on peut remarquer que cette formule se met très facilement à jour lorsqu'on ajoute une observation x_{n+1} d'écart-type σ_{n+1}, et qu'il n'est pas nécessaire de garder en mémoire l'ensemble des x_i et de leurs σ_i. En notant :

$$\begin{cases} \alpha_1 = x_1/\sigma_1^2 & \text{et } \forall k \geq 1\ \alpha_{k+1} = \alpha_k + x_{k+1}/\sigma_{k+1}^2 \\ \beta_1 = 1/\sigma_1^2 & \text{et } \forall k \geq 1\ \beta_{k+1} = \beta_k + 1/\sigma_{k+1}^2 \end{cases}$$

on a $\forall N$ le maximum de vraisemblance après N observations :

$$\hat{\mu}_N = \frac{\alpha_N}{\beta_N}.$$

Ainsi, seule la mémoire de deux valeurs α_N et β_N est nécessaire, et on peut aisément mettre à jour ces deux valeurs pour qu'elles tiennent compte d'une $(N+1)^{\text{ème}}$ observation.

Cette expression prouve même qu'il est statistiquement équivalent de faire une observation «très» précise ou deux observations «moyennement» précises. Un estimé de μ issu d'une unique observation d'une loi $\mathcal{N}\left(\mu, \sigma^2\right)$ a les même propriétés statistiques qu'un estimé obtenu après deux observations issues de deux lois $\mathcal{N}\left(\mu, \sigma_1^2\right)$ et $\mathcal{N}\left(\mu, \sigma_2^2\right)$ telles que $\sigma^2 = \dfrac{1}{1/\sigma_1^2 + 1/\sigma_2^2}$.

ANNEXE B PROFILS DOLAN-MORÉ ET MORÉ-WILD ADAPTÉS AU CONTEXTE BRUITÉ

Les profils dits «de performance, de données, de qualité» sont issus des articles de Dolan et Moré (2002), Moré et Wild (2009) et Beiranvand *et al.* (2017), qui les introduisent pour proposer des façons efficaces de comparer les performances d'algorithmes sur une banque de problèmes donnée. Ils sont maintenant très utilisés dans toute la littérature pour leur capacité à mettre en lumière certains comportements, mais sont usuellement restreints aux méthodes optimisant des fonctions déterministes. Cet annexe propose une modification des profils pour les rendre aptes à comparer des algorithmes conçus pour traiter des problèmes bruités.

Profils d'algorithmes d'optimisation déterministes

Les profils, tels que présentés originellement, reposent sur un outil de calcul estimant la qualité d'une solution courante à une itération donnée. Pour chaque problème p de la collection P, chaque algorithme a de la collection A est exécuté sur p. Tous partent du même point de départ x_*^0 et dans les mêmes conditions (même nombre maximal d'évaluations, entre autres).

Pour un problème p donné, on notera x_* la meilleure solution obtenue (au sens où elle est la meilleure solution que l'un des algorithmes a réussi à obtenir). Si l'optimum global du problème est connu, on pourra préférer utiliser sa valeur. De fait, $f(x_*)$ est une borne inférieure de l'ensemble des valeur de f examinées par l'ensemble des algorithmes.

On introduit alors, pour un problème p donné et un algorithme a considéré, l'indicateur de qualité de a. On désignera par x_N^a la meilleure solution trouvée par a après N évaluations de la boîte noire. De là, on construit l'objet fondamental exploité par les profils :

$$f_{qual}^N = \frac{f(x_a^N) - f(x_*^0)}{f(x_*) - f(x_*^0)}$$

qui évolue de 0 (lorsque a n'a pas amélioré x_*^0) vers 1 lorsque a améliore sa meilleure solution courante. Si cet indicateur atteint 1, cela signifie que a a obtenu la meilleure solution parmi l'ensemble des algorithmes considérés.

Il est alors possible de déterminer le nombre minimal d'évaluations de f requises par un algorithme pour atteindre une qualité demandée. Pour une tolérance τ donnée, ce nombre est :

$$N_{a,p} \in \arg\min_{N \in \mathbb{N}} \left\{ f_{qual}^N \mid f_{qual}^N \geq 1 - \tau \right\}, \ ou \ +\infty \ si \ \forall N \in \mathbb{N}, \ f_{qual}^N < 1 - \tau.$$

Ces nombres $N_{a,p}$ peuvent être comparés entre eux pour déterminer les performances relatives des algorithmes. Sur un même problème p, pour un seuil τ fixé, on peut définir

$$r_{a,p} = \frac{N_{a,p}}{\min\limits_{b \in A} N_{b,p}}$$

qui donne, en un sens, la performance d'un algorithme a relativement à la meilleure performance observée dans la collection d'algorithmes. Cet indicateur $r_{a,p}$ évolue de 1 (si a est le meilleur algorithme sur p) à $+\infty$ si a n'est pas parvenu à approcher $f(x_*)$ à un seuil τ près. Par exemple, $r_{a,p} = 3$ signifie que a requiert trois fois plus d'évaluations de f que le meilleur algorithme connu pour approcher l'optimum de p à un seuil de tolérance τ.

Ceci étant défini, le *profil de performance* se présente comme suit :

Définition B.0.1 (Profil de performance)**.** *Soit une collection P de problèmes, et A un ensemble d'algorithmes les ayant traités. Pour un seuil $\tau \in (0, 1)$ donné, les relevés des optimisations de chaque algorithme $a \in A$ donneront naissance à une courbe $p_a(\alpha)$ en fonction de α, où $\alpha \in [1, +\infty)$ et*

$$p_a(\alpha) = \frac{1}{Card(P)} Card\{p \in P \mid r_{a,p} \leq \alpha\}.$$

Ainsi, un profil de performance est un ensemble de $Card(A)$ courbes (une par algorithme). Chaque courbe représente le taux de problèmes de P résolus par a à une tolérance τ donnée, en fonction du nombre d'appels à la boîte noire autorisés, relativement au meilleur algorithme de A sur chaque problème.

Cependant, il ne s'agit pas du seul profil pertinent qu'on puisse envisager. Notamment, on peut considérer que le nombre d'appels à la boîte noire requis pour avancer dans l'optimisation n'est pas un indicateur absolu de la qualité des algorithmes. En effet, si le nombre de variables n_p du problème p est grand, il faut un grand d'appels à la fonction pour créer une base positive qui ne laisse aucune zone de l'espace inexplorée.

Pour tenir compte de ce fait, l'article introduit les *profils de données*, qui traduisent la quantité d'information dont chaque algorithme a besoin pour avancer dans son optimisation. Par «quantité d'information», nous entendons ici que $(n_p + 1)$ appels à la boîte noire suffisent à créer une base positive de l'espace des variables, et donc à avancer dans l'optimisation. Le profil de données va représenter la qualité des solutions des algorithmes en fonction du nombre de bases positives que qu'ils auraient pu créer avec leurs nombres d'appels à la boîte noire. Ce profil représentera le taux de problèmes résolus à une tolérance τ et un budget de k bases positives.

Le *profil de données* se définit comme suit :

Définition B.0.2 (Profil de données)**.** *Soit une collection P de problèmes, avec $\forall p \in P,\ n_p$ le nombre de variables du problème, et A un ensemble d'algorithmes les ayant traités. Pour un seuil $\tau \in (0,1)$ donné, les relevés des optimisations de chaque algorithme $a \in A$ donneront naissance à une courbe $d_a(k)$ en fonction de k, où $k \in \mathbb{N}$ et*

$$d_a(k) = \frac{1}{Card(P)} Card\{p \in P \mid N_{a,p} \leq k \times (n_p + 1)\}.$$

Ainsi, un profil de données est un ensemble de $Card(A)$ courbes (une par algorithme). Chaque courbe représente le taux de problèmes de P résolus par a à une tolérance τ donnée, en fonction de la quantité d'information autorisée.

Enfin, une autre approche intéressante consiste à examiner la qualité des solutions proposées par les algorithmes, d'une façon absolue et pas relative à un seuil de tolérance τ fixé. Pour cela, les *profils de qualité* donnent le ratio de problèmes résolus à 10^{-d} près pour un budget d'évaluations de la boîte noire fixé, en fonction de d. On peut les considérer comme une analogie avec un nombre d de décimales exactes imposées :

Définition B.0.3 (Profil de qualité)**.** *Soit une collection P de problèmes, et A des algorithmes les ayant traités. Pour un nombre N d'évaluations autorisées, les relevés des optimisations de chaque algorithme $a \in A$ donneront naissance à une courbe $q_a(d)$ en fonction de d, où $d \in \mathbb{R}^+$ et*

$$q_a(d) = \frac{1}{Card(P)} Card\{p \in P \mid f^N_{qual} \geq 1 - 10^{-d}\}.$$

Ainsi, un profil de qualité est un ensemble de $Card(A)$ courbes (une par algorithme). Chaque courbe représente le taux de problèmes de P résolus par a à une contrainte de qualité d après N appels à la boîte noire, en fonction de d.

Profils d'algorithmes d'optimisation bruitée

Dans le cas non déterministe, et plus encore en précision variable, le nombre d'appels à la boîte noire ne fait plus vraiment de sens et ne peut être utilisé comme indicateur de la performance des algorithmes. En effet, peu d'appels à une grande précision peuvent coûter plus cher (et être moins efficaces) qu'un grand nombre à une précision faible. Il est donc nécessaire d'adapter les profils pour leur faire tenir compte de la véritable quantité pertinente : les efforts de calcul requis. Puisque nous parlons constamment de nombre (équivalent) de tirages Monte-Carlo, c'est cette quantité qui servira ici à discriminer la performance des algorithmes.

Les valeurs de f seront prises exactes si la fonction non bruitée est connue. Si ce n'est pas le cas, on devra se contenter du maximum de vraisemblance de ses valeurs au vu des estimés calculés par les algorithmes. Par exemple, $f(x_*^0)$ sera dans ce cas le maximum de vraisemblance issu de l'ensemble des estimés initiaux. De même, la valeur optimale $f(x_*)$ sera la valeur exacte si elle est connue. Si ce n'est pas le cas, on prendra le meilleur estimé issu de l'ensemble des optimisations des algorithmes. En notant $f^k(x_*)$ cet estimé et $\sigma^k(x_*)$ son écart-type, on pourrait également prendre $f(x_*) \approx f^k(x_*) - 3\sigma^k(x_*)$ une minoration de la valeur réelle de $f(x_*)$ à un seuil de confiance de $99,7$ %.

De là, le calcul de f_{qual}^N se fera de la même manière qu'en contexte déterministe, en considérant les estimés comme exacts. Par contre, on ne parlera plus de nombre N d'appels autorisés à la boîte noire, mais plutôt d'un budget de N tirages de Monte-Carlo autorisés. Ainsi, l'expression est maintenant $f_{qual}^N = \dfrac{f(x_a^N) - f(x_*^0)}{f(x_*) - f(x_*^0)}$, où $N \in \mathbb{N}$ est un budget de tirages Monte-Carlo autorisés, x_a^N la meilleure solution trouvée par l'algorithme a avec ce nombre maximal de tirages, et $f(x_a^N)$ l'estimé de la vraie valeur de la boîte noire en x_a^N qu'il renvoie.

Suivant cette idée, $N_{a,p}$ sera maintenant le nombre minimal de tirages Monte-Carlo à autoriser à l'algorithme a pour qu'il résolve p au seuil de tolérance τ. Pour limiter l'amplitude des valeurs des $N_{a,p}$, on prendra plutôt le logarithme de ce nombre de tirages :

$$N_{a,p} \in \underset{N \in \mathbb{R}^{+*}}{\arg\min} \left\{ f_{qual}^{10^N} \mid f_{qual}^{10^N} \geq 1 - \tau \right\}, \ ou \ +\infty \ si \ \forall N \in \mathbb{N}, \ f_{qual}^N < 1 - \tau.$$

Avec ces nouvelles définitions, les *profil de performance en contexte bruité* et *profil de qualité en contexte bruité* peuvent être créés, de manière analogue au cas déterministe mais avec ces nouvelles définitions des quantités qu'ils manipulent.

Enfin, le profil de données doit être plus profondément adapté. Le groupement de $k \times (n_p + 1)$ évaluations ne signifie plus rien de pertinent. À la place, on notera N_0 le nombre de tirages Monte-Carlo fourni aux algorithmes pour effectuer leur estimation initiale $f^0(x_*^0)$, et l'utilisera en remplacement de n_p dans l'expression de $d_a(k)$:

$$d_a(k) = \frac{1}{Card(P)} Card\{p \in P \mid N_{a,p} \leq k \times \log_{10}(N_0)\}.$$

Le *profil de données en contexte bruité* est alors obtenu de façon analogue à son équivalent déterministe, avec cette nouvelle quantité. Il représente maintenant l'effort de précision requis aux algorithmes pour avancer dans leur optimisation, relativement au nombre minimal de tirages Monte-Carlo requis pour rendre l'estimé en x_*^0 viable.