Algorithmes évolutionnaires assistés par des méthodes d'apprentissage en grandes dimensions

Kamal Abboud

Thèse présentée pour obtenir le grade de

DOCTEUR DE L'ÉCOLE POLYTECHNIQUE

spécialité : Mathématiques Appliquées

par

Kamal ABBOUD

Algorithmes évolutionnaires assistés par des méthodes d'apprentissage en grandes dimensions

Soutenue le 19 octobre 2004 devant le jury composé de

MM.	Jean-Marc ALLIOT	Rapporteur
	Cyril FONLUPT	Examinateur
	Mohamed MASMOUDI	Rapporteur
	Jean-Claude NÉDÉLEC	Examinateur
	Marc SCHOENAUER	Directeur de thèse
	Michèle SEBAG	Examinateur

Algorithmes évolutionnaires assistés par des méthodes d'apprentissage en grandes dimensions

Kamal ABBOUD

6 octobre 2004

Table des matières

Introduction

Les algorithmes évolutionnaires sont des algorithmes d'optimisation stochastique d'ordre zéro qui s'appliquent à des espaces de recherche non standard, pouvant par exemple combiner des variables continues et discrètes. Ces algorithmes sont utilisés pour résoudre des problèmes difficiles où ils permettent souvent de proposer de meilleures solutions que les algorithmes classiques d'optimisation lorsque ces derniers s'appliquent. Leur principal point faible est la nécessité d'évaluer la fonction objectif un grand nombre de fois. Ceci les rend inapplicables en pratique pour des problèmes où l'évaluation de cette fonction est coûteuse en temps de calcul (résultat d'un lourd calcul d'éléments finis par exemple).

Le but de la thèse est de proposer une méthode permettant de diminuer le nombre de calculs de la fonction objectif dans le cas de fonctions coûteuses en temps de calcul et ainsi de diminuer le coût calcul. La méthode consiste à introduire un nouvel opérateur de mutation, nommé SDM (pour *Surrogate Deterministic Mutation*, basé sur une méthode d'approximation (nous étudierons en particulier comme méthodes d'approximation les Machines à Support Vecteur – SVM – et la méthode du Krigeage). L'opérateur SDM utilise des individus déjà évalués et se trouvant localement autour de l'individu devant être muté, pour définir une fonction modèle qui est facile à optimiser. Pour obtenir un bon modèle, non seulement le nombre de points d'exemples est important, mais aussi leur disposition autour de l'individu à muter. L'individu trouvé après optimisation du modèle est évalué avec la vraie fonction objectif et réinjecté dans la population. Nous avons démontré sur différentes fonctions-tests une nette amélioration de la convergence locale par rapport à un algorithme évolutionnaire classique, en l'occurrence les stratégies d'évolution (ES).

La thèse est formée de six chapitres :

Le premier chapitre consiste en un survol de l'état de l'art des différentes

techniques qui seront utilisées ensuite dans le reste de la thèse. En premier lieu, nous présentons les algorithmes d'optimisation déterministes d'ordre supérieur à un (optimisation différentiable).

Ensuite, nous introduisons les algorithmes stochastiques allant des méthodes de Monte Carlo, passant par le recuit simulé, les algorithmes évolutionnaires (dont les stratégies d'évolutions (ES)) et pour finir la méthode de l'Adaptation de la Matrice de Covariance (CMA). Puis nous passons en revue les principales méthodes d'approximation disponibles : des représentations sous forme de maillage éléments finis aux méthodes spectrales (dont l'approximation de Legendre) et les méthodes à noyau (dont la méthode de Machines à Support Vecteur et la méthode de Krigeage).

Finalement, nous faisons un état de l'art détaillé des méthodes hybrides précédemment proposées combinant une approche déterministe et/ou une approche stochastique et/ou une méthode d'approximation. Le chapitre se termine sur la présentation de l'ensemble des fonctions-tests qui seront ensuite utilisées pour valider notre approche.

Le deuxième chapitre discute la possibilité de modéliser la fonction objectif à partir de points pré-évalués et d'utiliser les optimaux des modèles approchés comme des candidats-optima de la fonction objectif. Nous utilisons comme méthodes d'approximations les SVM et le Krigeage. Pour déterminer les optima du modèle nous utilisons comme algorithme déterministe la méthode L-BFGS-B. Ces méthodes d'approximation déterminent d'une manière unique un modèle à partir des points évalués et des paramètres représentant la régularité et l'erreur souhaitée sur les points (SVM). Le choix de ces paramètres offre la possibilité de contrôler les modèles de façon à trouver un modèle adapté à nos besoins. Nous testons différents critères pour le choix de ces paramètres. Notons que le but de l'approximation n'est pas ici d'interpoler le mieux possible la fonction objectif, mais de trouver un individu meilleur que les points d'apprentissage en un temps raisonnable. Les différents tests de ce chapitre seront faits sur la fonction sphère.

Le troisième chapitre a pour but de définir un algorithme stochastique itératif simple, nommé AlgoAlea, basé sur une loi définie à travers des paramètres constants. Après une brève étude de la convergence de cet algorithme, nous l'hybridons respectivement avec une méthode déterministe (BFGS) pour définir un premier algorithme nommé AlgoDet, puis avec une méthode d'approximation (SVM ou Krigeage) et une méthode déterministe (BFGS), définissant l'algorithme AlgoApp.

Dans le cadre itératif et dans le souci d'optimiser la recherche des paramètres de régularité et les autres paramètres définissant le modèle, nous

proposons une approche dynamique dans la recherche des paramètres d'une itération à l'autre. Nous validons notre approche sur les fonctions-tests décrites au chapitre 2.

Pour finir, nous définissons une variante de l'algorithme hybride, variante nommée AlgoAppBis. Au lieu d'utiliser un algorithme déterministe pour optimiser le modèle, nous utilisons un algorithme évolutionnaire, les stratégies d'évolution (ES). Nous comparons enfin les différents algorithmes sur les fonctions-tests, et nous montrons que les algorithmes itératifs AlgoApp et AlgoAppbis ont un comportement quasi-déterministe, et nous pointons des difficultés d'un algorithme déterministe comme BFGS sur certaines fonctions-tests.

Le quatrième chapitre consiste à pallier les problèmes rencontrés dans le chapitre précédant en proposant une hybridation de l'algorithme AlgoApp avec un algorithme évolutionnaire global, ici les Stratégies d'évolution isotropes, l'algorithme AlgoApp se chargeant de la recherche locale, et ES de la recherche globale. Pour cela nous introduisons l'opérateur de mutation SDM (*Surrogate Determnistic Mutation*). Chaque mutation correspond de fait à une itération élémentaire de l'algorithme AlgoApp. L'interaction entre SDM et ES s'effectue à travers les paramètres des mutations gaussiennes. Nous définissons ainsi un paramètre de contrôle qui agit directement sur l'équilibre exploration-exploitation.

Nous comparons notre approche avec une autre approche, inspirée de MAES introduite par Thomas Bäck (elle-même représentante de l'approche dynamique définie par Andy Keane), et nous la validons sur les fonctions-tests.

Dans le sixième chapitre, nous appliquons SDM-ES à un problème pratique provenant de la finance quantitative. Il s'agit d'un modèle de marché incomplet qui traite la problématique du *smile* tout en proposant une couverture dynamique optimale. La calibration d'un tel modèle conduit à la résolution d'un problème de minimisation mal posé. Nous montrons que les méthodes hybrides de type SDM-ES peuvent apporter un plus par rapport à des méthodes déterministes comme L-BFGS-B qui ne donnent pas toujours la bonne solution.

Enfin, le sixième chapitre conclut la thèse par un bref rappel des principaux résultats obtenus. Prenant comme exemple le cas de la fonction Rosenbrock, réputée difficile, nous donnons quelques perspectives sur les adaptations possibles des hybridations proposées aux cas mal conditionnés. Nous montrons qu'un couplage de SDM avec la méthode plus récente de l'Adapta-

tion de la Matrice de Covariance (CMA) semble une voie très prometteuse. Nous montrons également que le noyau sigmoïde, bien que beaucoup plus délicat à maîtriser que le noyau Gaussien le plus utilisé (et en particulier utilisé dans la thèse), est très intéressant et mérite une étude systématique.

Chapitre 1

État de l'art

1.1 Le cadre de la thèse

Le problème classique d'optimisation consiste à rechercher le minimum global d'une fonction f définie sur un sous-ensemble X de $\mathbb{R}^n$ à valeurs dans $\mathbb{R}$:

$$\min_{x \in X} f(x) \tag{1.1}$$

Les algorithmes numériques de minimisation ont deux objectifs contradictoires dans la pratique : l'un est de trouver l'optimum global et l'autre est de garantir une précision minimale de la convergence, selon des critères forcément locaux. Les méthodes déterministes sont plutôt efficaces localement alors que les méthodes stochastiques ont pour objectif de l'être globalement.

Un premier objectif de la thèse est de proposer et d'étudier une nouvelle façon d'hybrider ces deux types d'algorithmes en profitant des avantages de chacun. Nous travaillerons plus particulièrement l'hybridation de méthodes déterministes (*e.g.* BFGS, voir section 1.3) avec les méthodes stochastiques évolutionnaires (*e.g.* ES, voir section 1.4).

Mais l'objectif de cette hybridisation est de diminuer le nombre d'évaluations de la fonction objectif, afin de minimiser le temps de calcul nécessaire à l'optimisation dans le cas de fonctions chères. Le moyen pour cela sera l'approximation locale de la fonction objectif.

Enfin, nous nous sommes concentrés surtout sur des méthodes pouvant s'appliquer en grandes dimensions.

Ce chapitre est consacré à "planter le décor" de l'ensemble de la thèse. En particulier, il ne contient pas de contribution nouvelle. Nous commençons

par un panorama des algorithmes d'optimisation déterministes puis stochastiques. Cette description ne prétend pas à l'exhaustivité, se concentrant sur les méthodes que nous utilisons par la suite.

Nous passons ensuite en revue différentes méthodes d'approximation de fonctions pour aboutir aux choix qui seront retenus finalement dans la thèse.

L'hybridation des méthodes stochastiques et déterministes n'est pas une idée neuve : nous faisons par la suite une revue des différents travaux dans ce domaine, qui, elle, se veut plus complète puisqu'elle concerne directement nos travaux. Enfin, nous terminons ce chapitre par la définition des fonctions-tests qui sont utilisées dans la suite de la thèse dans les expérimentations numériques.

1.2 Définitions et notations

1.2.1 Le problème d'optimisation

Étant donnée une fonction f à valeur dans $\mathbb{R}$ définie sur $X \subset \mathbb{R}^n$, le problème d'optimisation s'énonce de la manière suivante :

Trouver $x_* \in X$ tel que $[(\forall x \in X)(f(x_*) \leq f(x))]$.

La fonction f est appelée indifféremment fonction objectif, critère, fonction-coût, performance ou *fitness* dans les diverses communautés travaillant sur l'optimisation. Dans cette thèse, nous parlons de fonction objectif dans le cadre déterministe, et de fonction fitness dans le cadre évolutionnaire.

Plus formellement, le problème se met sous la forme :

$$(P_X) \left\{ \operatorname*{Arg\,min}_{x \in X} \{f(x)\} \right. \tag{1.2}$$

Remarques : lorsqu'il y a plusieurs critères, c'est-à-dire que l'espace d'arrivée est de dimension supérieure à 1, on parle alors d'optimisation multi-critères. Mais nous n'évoquons pas ici ce type de problèmes, pas plus que les problèmes d'optimisation sous contraintes.

On définit alors les notions de minimum global et de minimum local :
- Une solution de (P_X) est appelée *minimum global* de f.
- $x \in X$ est un *minimum local* si
  $(\exists \varepsilon > 0)\ \ (\forall y \in B(x, \varepsilon))\ \ (f(x) \leq f(y))$

Nous disposons sur $\mathbb{R}^n$ du produit scalaire canonique $\langle .,. \rangle$ et on notera $\|.\| = \langle .,. \rangle^{1/2}$ la norme associée. $\mathbb{R}^n$ est muni de la topologie euclidienne usuelle.

1.2.2 Conditions d'optimalité

Lorsque la fonction f est différentiable, on note $\nabla f(x)$ et $\nabla^2 f(x)$ le vecteur gradient et la matrice hessienne de f en x.

Les solutions du problème (P_X) vérifient alors des équations qu'on appelle les conditions d'optimalité. Ces équations peuvent être des conditions nécessaires ou suffisantes. Elles sont à la base des algorithmes numériques déterministes de résolution du problème (P_X). Elles servent aussi à définir des tests d'arrêt pour les algorithmes.

Dans le cas sans contraintes, $X = \mathbb{R}^n$, rappelons que les conditions d'optimalité sont :

Conditions nécessaires : si x_* est un minimum local, alors

(1.3) CN du 1^{er} ordre : $\nabla f(x) = 0$

(1.4) CN du 2^{nd} ordre : $\nabla^2 f(x)$ est semi-définie positive.

Condition suffisante : inversement, si x_* est tel que $\nabla f(x) = 0$ et $\nabla^2 f(x)$ est définie positive, alors x_* est un minimum local(*i.e.* $f(x) \geq f(x_*)$ pour x proche de x_*).

1.3 Méthodes déterministes à directions de descente

Nous supposons dans cette section que la fonction f est différentiable, et nous rappelons les grandes lignes d'un ensemble d'algorithmes basés sur la notion de direction de descente. Pour plus de détails voir le livre de F. Bonnans *et al.* [98].

On dit que d est une **direction de descente** de f en $x \in \mathbb{R}^n$ si

$$\langle \nabla f(x), d \rangle < 0$$

Géométriquement cela veux dire que d fait avec l'opposé du gradient $-\nabla f(x)$ un angle θ strictement plus petit que $\frac{\pi}{2}$.

Si d est une direction de descente, en utilisant une formule de Taylor, on voit que

$$f(x+\alpha d) < f(x), \quad \text{pour} \quad \alpha > 0 \text{ suffisamment petit}$$

En se déplaçant (peu) suivant d, on va donc faire décroître f. Les méthodes à directions de descente utilisent cette idée pour minimiser une fonction. L'algorithme général définit une suite $\{x_k\}_{k>=1}$ de la façon suivante :

$$\begin{cases} x_0 \in X \\ x_{k+1} = x_k + \alpha_k d_k, \forall k \geq 0 \end{cases}$$

où $\alpha_k > 0$ est appelé le pas et d_k est une direction de descente en x_k.

Le pas doit être déterminé de telle sorte que la valeur de la fonction décroisse le plus possible. On utilise en général un algorithme de recherche linéaire, qui consiste à rechercher un pas "optimal". L'algorithme s'écrit alors plus précisément :

Choix d'un itéré initial $x_0 \in \mathbb{R}^n$, et d'un paramètre d'arrêt ε.
Pour $k \geq 0$, faire

1. test d'arrêt : $\|g_k\| < \varepsilon$, où g_k est le gradient de f en x_k ;
2. choix d'une direction de descente d_k ;
3. recherche linéaire : déterminer un pas $\alpha_k > 0$ minimisant $f(x_k + \alpha d_k)$;
4. $x_{k+1} = x_k + \alpha_k . d_k$.

La condition d'arrêt porte donc sur la norme du gradient, utilisant la condition d'optimalité 1.3. Les deux ingrédients d'une méthode à direction de descente sont donc d'une part le choix de la direction, d'autre part le choix de la méthode de recherche linéaire. Nous allons maintenant en donner des exemples.

1.3.1 Choix de la direction de descente

1. **Algorithme du gradient**(ou de la plus forte pente) :
   L'inverse du gradient est évidemment une direction de descente si x_k n'est pas un point stationnaire.

   $$d_k = -g_k$$

   Loin de la solution, cette direction est une bonne direction de descente. Dès que l'on s'approche de la solution optimale les termes du second ordre d'un développement de Taylor de f autour de x_* jouent un plus grand rôle, et l'algorithme devient moins performant.

2. **Algorithme du gradient conjugué** :
   Pour un simple problème fortement elliptique ($f(x) = x^T Ax$, A étant une matrice symétrique définie positive ayant des valeurs propres d'ordres de grandeur différents), il est clair que le gradient n'est pas la meilleure direction de descente, *i.e.* ne va pas beaucoup rapprocher la recherche de l'optimum. On conjugue alors les directions de descentes (on dit que u et v sont conjugués si $u^T Av = 0$). L'algorithme s'écrit alors

$$d_k = \begin{cases} -g_1 & \text{si } k = 1 \\ -g_k + \beta_k d_{k-1} & \text{si } k \geq 2. \end{cases}$$

   La valeur β_k est calculée de telle sorte que les directions d_k soient conjuguées 2 à 2, et s'écrit :

$$\beta_k = \frac{\|g_k\|^2}{\|g_{k-1}\|^2}$$

   A noter que la recherche linéaire peut alors se faire exactement, et on trouve :

$$\alpha_k = -\frac{g_k^T d_k}{d_k^T A d_k}.$$

   L'algorithme a été étendu aux fonctions non quadratiques. Par exemple, la méthode de Fletcher-Reeves utilise $\beta_k^{FR} = \frac{\|g_k\|^2}{\|g_{k-1}\|^2}$, et celle de Polak-Ribière, $\beta_k^{PR} = \frac{g_k^T y_{k-1}}{\|g_{k-1}\|^2}$. Ces deux méthodes sont identiques à la méthode de gradient conjugué dans le cas d'une fonction quadratique et avec recherche linéaire exacte.

3. **Algorithme de Newton** :
   L'algorithme de Newton est une méthode pour résoudre un système d'équations non linéaires très général. Dans le cadre de l'optimisation, elle est utilisée sur la condition d'optimalité

$$\nabla f(x) = 0$$

   Il s'agit bien d'un système de n équations non-linéaires à n inconnues. L'équation de Newton s'obtient en linéarisant en x_k le système d'équations d'optimalité ci-dessus, et qui s'écrit :

$$\nabla^2 f(x_k) d_k = -\nabla f(x_k) \tag{1.5}$$

   L'algorithme est alors :
   Choix d'un itéré initial $x_0 \in \mathbb{R}^n$.
   Pour $k \geq 0$

(a) Test d'arrêt : $\|g_k\| < \varepsilon$
(b) Calcul d_k comme solution de l'équation 1.5
(c) $x_{k+1} = x_k + d_k$;

Une autre approche, aboutissant au même algorithme, consiste à chercher l' itéré suivant en minimisant l'approximation quadratique de f autour de l'itéré courant (voir section 1.3.3 pour plus de détails sur la minimisation avec approximation quadratique).
L'algorithme décrit ci-dessus est l'algorithme de Newton original. Pour rendre la méthode plus globale (la fonction quadratique est définie localement), il est courant d'ajouter un pas α, et un algorithme de recherche linéaire pour déterminer un pas optimal.

4. **Algorithmes de quasi-Newton**
La méthode de Newton nécessite le calcul de la hessienne, puis son inversion pour définir une direction descente. De plus, on ne dispose pas toujours des dérivées secondes. L'idée des méthodes de quasi-Newton (dont la plus utilisée est la méthode BFGS) est de remplacer la matrice exacte $\nabla^2 f^{-1}$ par une approximation mise à jour itérativement. La direction de descente s'écrit alors

$$d_k = -M_k^{-1} g_k$$

où la matrice M_k doit converger vers la matrice hessienne lorsqu'on s'approche de l'optimum.
Pour tout k, la matrice M_k doit vérifier plusieurs conditions comme par exemple la symétrie :

$$M_k = M_k^T.$$

D'autre part, en posant

$$s_k = x_{k+1} - x_k \text{ et } \ y_k = g_{k+1} - g_k,$$

on impose à la matrice M_{k+1} de vérifier la relation

$$y_k = M_{k+1} s_k.$$

qui porte le nom d'équation de quasi-Newton.
Plusieurs méthodes ont été proposées pour la mise à jour de M_k :
– La formule Symétrique de Rang 1(SR1) :

$$M_{k+1} = M_k + \frac{(y_k - M_k s_k)(y_k - M_k s_k)^T}{(y_k - M_k s_k)^T s_k}.$$

Cette formule est utilisée lorsqu'il n'est pas possible ou qu'il n'est pas nécessaire que M_{k+1} soit définie positive.

– La formule de BFGS (Broyden, Fletcher, Godlfarb, Shanno) [98] :

$$M_{k+1} = M_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{M_k s_k s_k^T M_k}{s_k^T M_k s_k},$$

Dans les deux cas, le pas α_k est déterminé avec la recherche linéaire de *Wolfe*, exposée dans la section suivante.

1.3.2 La recherche linéaire

La recherche linéaire consiste à trouver une valeur "optimale" de α_k qui est le pas le long de la direction de descente fixée. La valeur optimale de α_k serait celle qui minimise la fonction f dans la direction d_k. À défaut de trouver un α_k optimal, on doit satisfaire les deux conditions suivantes : La première est de faire décroître f. La deuxième est d'empêcher le pas $\alpha_k > 0$ d'être trop petit sinon cela pourrait provoquer une fausse convergence, c'est-à-dire une convergence des itérés vers un point non-stationnaire.

La recherche linéaire contribue à la convergence des méthodes à directions de descente grâce à la condition de Zoutendijik :

Il existe une constante $C > 0$ telle que pour tout indice $k \geq 1$ on ait :

$$f(x_{k+1}) \leq f(x_k) - C\|g_k\|^2 \cos^2(\theta_k) \quad \text{avec} \quad \cos\theta_k = \frac{\langle -g_k, d_k \rangle}{\|g_k\|\|d_k\|}. \tag{1.6}$$

Plusieurs méthodes de recherche linéaire ont été proposées, que nous allons maintenant passer en revue.

1. **Recherche linéaires "exacte"**

   Aussi appelée règle de Cauchy, elle consiste à trouver un point stationnaire de f. La recherche linéaire "exacte" est très coûteuse en temps de calcul. Le gain supplémentaire éventuellement apportée à un algorithme par une recherche linéaire exacte ne permet pas, en général, de compenser le temps perdu à déterminer un tel pas.

   Les règles que nous exposons ensuite sont plus adaptées. Il ne s'agit de rechercher un optimum α_k mais d'imposer des conditions moins restrictives, plus facilement vérifiables, dont les solutions ne sont plus des points mais des intervalles de pas. Ce qui permet d'élaborer des algorithmes qui convergent en un nombre limité d'itérations.

2. **Règle d'Armijo et de Goldstein**
   La condition consiste à se contenter d'une décroissance qui n'est qu'un pourcentage de la décroissance qui serait constatée avec une recherche exacte si la fonction f était linéaire. La condition d'Armijo, ou condition de décroissance linéaire, s'écrit :

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \omega_1 \alpha_k \langle g_k, d_k \rangle.$$

3. **Règle de Wolfe**
   Un raffinement consiste à arrêter la recherche du pas α_k dès que les inégalités suivantes, appelées conditions de Wolfe, sont vérifiées :

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \omega_1 \alpha_k \langle g_k, d_k \rangle$$

$$\langle \nabla f(x_k + \alpha_k d_k), d_k \rangle \geq \omega_2 \langle g_k, d_k \rangle$$

   où les constantes ω_1 et ω_2 sont choisies telles que :

$$0 < \omega_1 < \omega_2 < 1$$

1.3.3 Méthode à région de confiance : recherche curviligne

La méthode de la région de confiance est une généralisation des méthodes à direction de descente. Elle globalise l'algorithme en le rendant insensible au choix de l'initialisation. Elle est basée sur un modèle local, au voisinage de l'itéré courant, représentant une perturbation du problème de départ. Il s'agit souvent d'un modèle quadratique, on parle alors de SQP pour Sequential Quadratic Programming. Le modèle est paramétré par une seule variable réelle δ qu'on cherche à optimiser à chaque itération.

Remarquons que la recherche du δ optimal n'est pas une fin en soit ! On limite donc le nombre d'itérations pour trouver un δ «quasi-optimal».

Rappelons que la **recherche linéaire** peut être vue comme un simple cas particulier de la méthode de la région de confiance. En effet, en partant de x, si la direction de descente s'écrit $M^{-1}g$ avec M une matrice symétrique définie positive, alors le minimum du modèle suivant :

$$\hat{f}_\delta(x + h) := f(x) + (g, h) + \frac{1}{2\delta}(Mh, h)$$

est donné par :

$$h_\delta = -\delta M^{-1} g$$

d'où le nouvel itéré x_δ :

$$x_\delta = x + h_\delta$$

La règle de Wolfe peut alors être utilisée pour choisir un δ optimal. Pour tout δ, les x_δ se trouvent sur la droite passant par x de direction $M^{-1}g$.

La **recherche curviligne** consiste à optimiser un modèle $\hat{f}$ dans une boule centrée en x de rayon δ (fixé) :

$$\min_{|h| \leq \delta} \hat{f}(x+h)$$

Le nouvel itéré s'écrit

$$x_\delta = x + h_\delta$$

et on optimise en δ. Contrairement à la recherche linéaire, les x_δ ne sont plus sur une droite mais plutôt sur une courbe gauche. Par exemple, si la fonction $\hat{f}$ est quadratique définie positive, alors son minimum est à une distance finie (le minimum dans la boule de rayon δ est le minimum global si δ est assez grand) et la longueur de la courbe des x_δ est elle-même de longueur finie.

Comme il est difficile de dériver suivant la courbe (qui n'est pas explicite), les algorithmes d'ordre 0 sont les plus adaptés (Wolfe, Goldestein et Price).

1.4 Algorithmes évolutionnaires

1.4.1 Introduction

Les algorithmes évolutionnaires (EA pour Evolutionary Algorithms) font partie de la famille des algorithmes d'optimisation stochastiques, au même titre que les algorithmes de Monte Carlo et l'algorithme du Recuit Simulé. Comme ce dernier, inspiré des mécanismes de la thermodynamique, les algorithmes évolutionnaires sont inspirés d'un paradigme naturel, celui de l'évolution darwinienne : l'émergence d'espèces adaptées à leur milieu est la conséquence de l'interaction entre sélection naturelle et variations non-dirigées.

Définitions et notations : Soit à optimiser une fonction $\mathcal{F}$ à valeurs réelles définie sur un espace Ω, supposé muni d'une distance d. Le parallèle avec l'évolution naturelle a entraîné l'apparition d'un vocabulaire spécifique (et qui peut paraître légèrement ésotérique) :

- La fonction objectif $\mathcal{F}$ est appelée fonction *performance*, ou fonction *d'adaptation* (*fitness* en anglais) ;
- Les points de l'espace de recherche Ω sont appelés des *individus* ;
- Les P-uplets d'individus sont appelés des *populations* ;

– On parlera d'une *génération* pour la boucle principale de l'algorithme ;

Dans la méthode de Monte Carlo la fonction est évaluée en un grand nombre de points choisis suivant une distribution de probabilité fixée à l'avance (en général uniforme sur l'espace de recherche). Dans le cadre de l'optimisation de paramètres multiples, les méthodes Monte Carlo sont intéressantes si le nombre de paramètres reste très petit. Elles ont alors l'avantage de la simplicité. De plus, elles permettent d'avoir un bon aperçu de l'ensemble de l'espace de recherche : les nuages de points que l'on peut en tirer sont souvent riches en information. Si le nombre de paramètres est très grand, les méthodes de Monte Carlo peuvent être utilisées dans un espace de recherche restreint. Appliquées localement, elles permettent par exemple d'évaluer la sensibilité d'un dispositif aux incertitudes technologiques.

Dans la méthode du Recuit Simulé [8], on effectue des déplacements aléatoires à partir du point courant. Si un déplacement mène à une meilleure valeur de la fonction f, il est accepté et devient un point courant. Sinon, il peut quand même être accepté avec une probabilité :

$$p = exp(-\frac{\Delta\mathcal{F}}{kT}) \tag{1.7}$$

où $\Delta\mathcal{F}$ est la variation de la fonction $\mathcal{F}$, T est assimilé à une température qui décroît au cours du temps et k est une constante. Cette méthode est basée sur une analogie avec les processus de recuit utilisés en métallurgie et qui visent à atteindre une configuration d'énergie minimale : en chauffant un système, puis en le refroidissant très lentement, il va naturellement atteindre des niveaux d'énergie très bas.

Dans le cas des algorithmes évolutionnaire, le problème qu'on cherche à résoudre est vu comme un environnement responsable de la sélection «naturelle». La pression de l'«environnement» est simulée à l'aide de la fonction d'adaptation $\mathcal{F}$, et le principe darwinien «*Les plus adaptés survivent et se reproduisent*» est implémenté de la manière suivante :

- **Initialisation** de la *population* Π_0 en choisissant P individus dans Ω, généralement par tirage aléatoire avec une probabilité uniforme sur Ω ;
- **Évaluation** des individus de Π_0 (i.e. calcul des valeurs de $\mathcal{F}$ pour tous les individus) ;
- La génération i construit la population Π_i à partir de la population Π_{i-1} :
  - **Sélection** des individus les plus performants de Π_{i-1} au sens de $\mathcal{F}$ (*les plus adaptés se reproduisent*) ;

- Application (avec une probabilité donnée) des **opérateurs de variation** aux *parents* sélectionnés, ce qui génère de nouveaux individus, les *enfants*; on parlera de *mutation* pour les opérateurs unaires, et de *croisement* pour les opérateurs binaires (ou n-aires) ;
- **Évaluation** des enfants;
- **Remplacement** de la population Π_i par une nouvelle population créée à partir des enfants et/ou des «vieux parents» de la population Π_i au moyen d'une sélection darwinienne (*les individus les plus adaptés survivent*).

– L'évolution stoppe quand le niveau de performance souhaité est atteint, ou qu'un nombre fixé de générations s'est écoulé sans améliorer l'individu le plus performant.

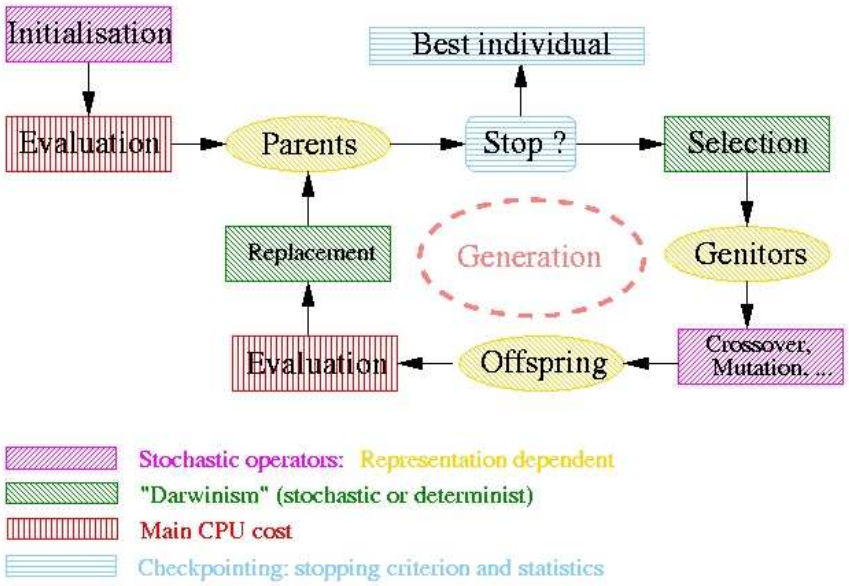


FIG. 1.1 – *Squelette d'un algorithme évolutionnaire*

Il est important de noter que, dans les applications à des problèmes numériques réels, l'essentiel du coût en temps-calcul de ces algorithmes provient de l'étape d'évaluation (calcul des performances) : les tailles de populations sont de l'ordre de quelques dizaines, le nombre de générations de quelques centaines, ce qui donne lieu à quelques dizaines de milliers de calculs de $\mathcal{F}$.

Un point important dans la recherche d'un optimum global par tout algorithme, mais par les algorithmes évolutionnaires en particulier, est l'équilibre

entre exploration (de régions de l'espace de recherche non encore visitées) et l'exploitation (des meilleurs individus rencontrés jusque-là), autour desquels se trouvent peut-être des individus encore meilleurs. En général, on privilégie l'exploration dans la phase initiale de la recherche, et l'exploitation lors de la phase finale.

Du point de vue du dilemme exploration/exploitation, par exemple, nous pouvons dire que les méthodes Monte Carlo permettent une bonne exploration, puisque tout point a une probabilité identique d'être atteint, mais qu'il n'y a aucune exploitation des résultats déjà obtenus. Inversement, avec les méthode déterministes énumérées section 1.3, l'exploration de l'espace n'est pas globale, mais par contre l'exploitation des données précédentes se fait de manière «optimale» par l'intermédiaire des informations relatives au gradient, ce qui permet une bonne recherche locale. Les Algorithmes Évolutionnaires sont censés offrir un bon compromis entre exploration et exploitation [138].

1.4.2 Les Stratégies d'Évolution

Il existe de nombreuses variantes d'algorithmes évolutionnaires – on consultera pour plus de détails le livre récent *Introduction to Evolutionary Computing* [7]. L'algorithme le plus adapté aux problème à variables continues est sans conteste l'algorithme des *Stratégies d'Évolution* (ES en anglais, acronyme que nous utiliserons ici). L'opérateur principal de cet algorithme est la mutation, qui est réalisée en ajoutant une variable aléatoire gaussienne au vecteur courant. Les travaux de cette thèse n'utiliseront que cette variante des algorithmes évolutionnaires, c'est pourquoi nous allons maintenant la détailler.

Les différentes variantes des Stratégies d'Évolution diffèrent par la manière de régler les paramètres de la mutation gaussienne. Prenons l'exemple d'une seule variable réelle : la mutation est alors définie par la variance de la distribution normale centrée dont une réalisation va être ajoutée à la valeur courante. Une grande variance favorise l'exploration, alors qu'une petite variance ne pourra faire que de l'exploitation. Il est donc important de bien régler la variance, de manière dynamique puisque les objectifs d'exploitation et d'exploration varient en cours d'évolution.

Les premières méthodes, basées sur des études théoriques sur quelques fonctions simples (voir [9, 19]) préconisaient d'augmenter la variance lorsque trop de mutations donnaient un enfant meilleur que le parent, et de la diminuer dans le cas contraire. Cette fameuse *règle des 1/5* est malheureusement prise en défaut dans de nombreux cas.

L'idée fondamentale qui a permis l'essor des ES est celle de l'auto-adaptation : les paramètres de la mutation sont associés à chaque individu, et modifiés par mutation également. Bien que la sélection soit ensuite effectuée sur les valeurs de la fitness, dans lesquelles les paramètres de la mutation n'interviennent pas, l'idée sous-jacente qui fait que l'auto-adaptation fonctionne est que les individus ayant des «bonnes» valeurs de paramètres de mutation vont petit à petit dépasser ceux qui ont de «mauvaises» valeurs. Les résultats obtenus en terme de vitesse de convergence montrent la validité de ces idées, tant de manière expérimentale [23] que théorique [2].

En dimension $n > 1$, il existe trois variantes de Stratégies d'Évolution auto-adaptatives, suivant les hypothèses simplificatrices faites sur la matrice de covariance de la distribution utilisée. La variable aléatoire gaussienne la plus générale en dimension n a pour densité de probabilité

$$p(\mathbf{z}) = \frac{1}{(2\pi)^{n/2} \det \mathbf{C}} \exp\left(-\frac{1}{2}\mathbf{z}^T \mathbf{C}^{-1} \mathbf{z}\right). \tag{1.8}$$

oú C est la matrice (symétrique définie positive) de covariance, et la mutation gaussienne correspondante sera notée

$$X := X + \mathcal{N}(0, C) \tag{1.9}$$

Mutation auto-adaptative isotrope

On suppose dans ce modèle que $C = \sigma I_n$. Il y a donc un unique paramètre σ à ajuster. Un individu est alors composé d'un vecteur X (les n variables du problème) et d'un réel σ. La mutation auto-adaptative d'un tel individu se fait alors en deux étapes :

$$\begin{cases} \sigma := \sigma exp(\tau z), & z = \mathcal{N}(0,1) \\ X_i := X_i + \sigma z_i, & z_i = \mathcal{N}(0,1) \end{cases}$$

La mutation de la variance σ est une mutation log-normale, respectant la positivité de σ, et «multiplicativement symétrique» autour de 1, la variance étant ensuite utilisée pour la mutation des variables de manière multiplicative. La variance de cette mutation, τ, doit être définie par l'utilisateur, mais la valeur de $\frac{1}{\sqrt{2n}}$ recommandée par Schwefel [19] d'après une étude sur la fonction sphère, semble très robuste.

Il est cependant clair que dans de nombreux cas, les "bonnes" valeurs de la variance sont différentes dans différentes directions. C'est ce qui justifie l'utilisation d'une variance par dimension.

Mutation auto-adaptative non-isotrope

Dans cette variante, l'individu se compose du vecteur des variables X et d'un vecteur de variances σ. La mutation a là aussi lieu en 2 temps :

$$\begin{cases} \sigma_i := \sigma_i \ \exp(\tau' z) \ \exp(\tau z_i) & z, z_i = \mathcal{N}(0,1) \\ X_i := X_i + \sigma_i z'_i, & z'_i = N(0,1) \end{cases}$$

On remarquera les deux termes log-normaux de la mutation des σ_i : les deux facteurs τ et τ', pour lesquels Schwefel recommande les valeurs $\frac{1}{\sqrt{2n}}$ et $\frac{1}{\sqrt{2\sqrt{n}}}$ respectivement, correspondent d'une part à un ordre de grandeur de modification global à tous les σ_i et d'autre part à des variations différentes pour chacune des coordonnées, mais de moindre importance.

C'est la stratégie d'évolution la plus utilisée du fait de sa simplicité et des bonnes performances qu'elle permet d'obtenir [14, 13].

Mutation auto-adaptative corrélée

Mais bien entendu, l'utilisation d'une matrice de covariance diagonale (mutation auto-adaptative non-isotrope ci-dessus) fait que l'algorithme est incapable de prendre en compte les corrélations entre les différentes variables. La mutation auto-adaptative la plus complète consiste donc à associer une matrice symétrique définie positive complète à chaque individu.

Cependant, il est très difficile de garantir la symétrie et la positivité de la matrice après mutation de ses termes, et c'est pourquoi on utilise plutôt une représentation sous forme du produit d'une matrice diagonale (les termes diagonaux sont les variances suivant les directions principales) et de $n(n-1)/2$ matrices de rotations dans des plans définis par les couples de vecteurs de base. De plus, cela rend l'échantillonnage de la distribution particulièrement simple, puisqu'il suffit de tirer n échantillons indépendants avec chacune des n variances principales, et d'appliquer ensuite les rotations au vecteur ainsi obtenu – sans inversion de matrice. Toute matrice symétrique définie positive peut se représenter de la sorte, et il n'y a donc aucune perte de généralité :

$$C(\sigma, \alpha) = \left(\prod_{i=1}^{n-1} \prod_{j=i+1}^{n} R(\alpha_{ij}) \right) \cdot diag(\sigma_i^2) \left(\left(\prod_{i=1}^{n-1} \prod_{j=i+1}^{n} R(\alpha_{ij}) \right) \cdot diag(\sigma_i^2) \right)^t$$

où les matrices $R(\alpha_{ij})$ sont définies par

$$R(\alpha_{ij}) = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & \cdots & \cdots & 0 \\ \cdots & \cos\alpha_{ij} & \cdots & -\sin\alpha_{ij} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \sin\alpha_{ij} & \cdots & \cos\alpha_{ij} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix}$$

La mutation est alors similaire au cas non-isotrope, auquel il faut bien entendu ajouter la mutation (gaussienne) des angles α_{ij} :

$$\begin{cases} \sigma_i := \sigma_i \ \exp(\tau' z) \ \exp(\tau z_i) & z, z_i = \mathcal{N}(0,1) \\ \alpha_{ij} := \alpha_{ij} + \beta z_{ij}, & z_{ij} = \mathcal{N}(0,1) \\ X := X + Z, & Z = \mathcal{N}(0, C(\sigma, \alpha)) \end{cases}$$

Les valeurs α_{ij} sont prises dans l'intervalle $[-\pi, \pi]$, les valeurs de τ et τ' sont celles de la mutation non-isotrope, et la valeur préconisée par Schwefel pour β est d'environ 5 degrés. Toutes ces valeurs sont rappelées dans le tableau 1.1.

Paramètres	Valeurs standards	Commentaire
τ'	$1/\sqrt{2n}$	valeur heuristique
τ	$1/\sqrt{2\sqrt{n}}$	valeur heuristique
β	0.0873	valeur heuristique
ϵ	10^{-30}	borne inférieur pour σ

TAB. 1.1 – Les paramètres définissant les lois d'adaptation du ES standard et ES corrélées.

Adaptation de la matrice de covariance (CMA)

Relativement récemment est apparue la méthode CMA (pour *Covariance Matrix Adaptation*, Adaptation de la Matrice de Covariance). L'article décrivant de manière exhaustive et synthétique l'ensemble de la méthode date de 2001 [12], même si les premiers travaux des auteurs remontent à quelques années (1996) [10, 11]. L'idée de base est que l'auto-adaptation peut être trop lente dans certains cas de fonctions dans lesquelles les variables sont très corrélées, mais que par contre l'histoire de l'évolution peut permettre de pressentir quelles sont les bonnes directions.

Dans l'algorithme de CMA, la matrice de covariance C décrite dans l'équation (1.8) est adaptée de manière déterministe. On considère une matrice $\mathbf{B}$, qui vérifie $\mathbf{C} = \mathbf{B}\mathbf{B}^t$ (c'est cette matrice qui est d'ailleurs utilisée pour transformer un vecteur de coordonnées aléatoires gaussiennes indépendantes en un vecteur qui est $N(\mathbf{0}, \mathbf{C})$ distribué). Les transformations des variables peuvent être écrites de la manière suivante :

$$\mathbf{x} := \mathbf{x} + \delta\,\mathbf{B}\,\mathbf{z}, \quad z_i \sim N(0,1), \text{indépendants} \tag{1.10}$$

L'adaptation de la matrice de covariance se produit en deux étapes, de nouveau ramenée à des moyennes sur plusieurs générations

$$\mathbf{s} := (1-c)\,\mathbf{s} + c_u\,\mathbf{B}\,\mathbf{z} \tag{1.11}$$

$$\mathbf{C} := (1-c_{cov})\,\mathbf{C} + c_{cov}\,\mathbf{s}\mathbf{s}^t \tag{1.12}$$

Les paramètres de cette méthode ont été définis après de nombreux essais de telle sorte que la matrice de covariance reste la plus proche possible de l'identité sur la fonction sphère.

L'adaptation sur plusieurs générations dans l'équation (1.11) peut être interprétée comme la somme des différents vecteurs. Ces vecteurs sont les résultats des mutations sur différentes générations.

Pour déterminer la matrice $\mathbf{B}$ ($\mathbf{C} = \mathbf{B}\mathbf{B}^t$) on procède comme suit : toute matrice $\mathbf{C}$ symétrique est diagonalisable dans une base orthonomée :

$$\mathbf{C}\mathbf{X}_i = \lambda_i \mathbf{X}_i \quad i = 1, \ldots, n$$

avec $\mathbf{X}_i^t \mathbf{X}_j = \delta_{i,j}$, et la matrice s'écrit

$$\mathbf{C} = \sum_{i=1}^{n} \lambda_i \mathbf{X}_i \mathbf{X}_i^t$$

La matrice $\mathbf{C}$ étant positive, on a $\lambda_i \geq 0$. En définissant la iéme colonne de la matrice $\mathbf{B}$ comme suit :

$$\mathbf{B}_i = \sqrt{\lambda_i}\mathbf{X}_i$$

on a

$$\mathbf{C} = \sum_{i=1}^{n} \mathbf{B}_i \mathbf{B}_i^t = \mathbf{B}\mathbf{B}^t$$

Pour finir, il faut adapter le pas global δ. L'adaptation est liée au déplacement dans l'espace normé. Soit $\mathbf{B}_0$ la matrice dont la ième colonne s'écrit :

$$\mathbf{B}_{0i} = \mathbf{X}_i$$

Si $\mathbf{s}_\delta$ est le déplacement dans l'espace normé alors

$$\mathbf{s}_\delta := (1-c)\,\mathbf{s}_\delta + c_u\,\mathbf{B}_0\,\mathbf{z} \tag{1.13}$$

$$\delta := \delta \cdot \exp\left(\beta\left(||\mathbf{s}_\delta|| - \hat{\chi}_n\right)\right) \tag{1.14}$$

Le tableau 1.2 expose les valeurs heuristiques initialisant les paramètres d'adaptation [12].

Paramètres	Valeurs standards	Commentaires
c	$1/\sqrt{n}$	valeur heuristique
c_{cov}	$2/n^2$	valeur heuristique
β	$1/n$	valeur heuristique
c_u	$\sqrt{c(2-c)}$	normalisation de la variance
$\hat{\chi}_n$	$\sqrt{n}\left(1-\frac{1}{4n}+\frac{1}{21n^2}\right)$	valeur estimée de χ_n

TAB. 1.2 – Les paramètres définissant les lois d'adaptation de CMA et leurs valeurs standards.

1.4.3 Domaines d'applications des algorithmes évolutionnaires

Les applications des algorithmes évolutionnaires sont multiples : optimisation de fonctions numériques difficiles (discontinues, multimodales, bruitées), traitement d'image (alignement de photos satellites, reconnaissance de suspects), optimisation d'emplois du temps, optimisation de design, contrôle de systèmes industriels [138], apprentissage des réseaux de neurones [139], etc.

Les AG peuvent être utilisés pour contrôler un système évoluant dans le temps (chaîne de production, centrale nucléaire) car la population peut s'adapter à des conditions changeantes. En particulier, ils supportent bien l'existence de bruit dans la fonction à optimiser. Ils peuvent aussi servir à déterminer la configuration d'énergie minimale d'une molécule ou à modéliser le comportement animal. Les AG sont également utilisés pour optimiser des réseaux (câbles, fibres optiques, mais aussi eau, gaz), des circuits VLSI [138], des antennes [140] Ils peuvent être utilisés pour trouver les paramètres d'un modèle petit-signal à partir des mesures expérimentales [141]. Des commutateurs optiques adiabatiques ont été optimisés à l'aide des Stratégies d'évolutions (autres AE) chez SIEMENS AG [142]. On envisage aussi l'intégration d'algorithmes génétiques dans certaines puces électroniques (de type FPGA) afin qu'elles soient capables de se re-configurer automatiquement en fonction de leur environnement (Evolving Hardware en anglais).

1.5 Méthodes d'apprentissages

Nous nous intéressons dans ce chapitre au problème d'apprentissage de fonctions à partir d'exemples, aussi appelé problème de régression, ou problème d'approximation de données (*data fitting*).

Le problème d'apprentissage de base s'écrit :

A partir des exemples $(x_i, y_i)_{i\in[1,N]} \in \mathbb{R}^d \times \mathbb{R}$*, l'objectif est de trouver une fonction* f *telle que :*

$$f(x_i) = y_i \quad i = 1..N \tag{1.15}$$

Le problème 1.15 en général n'ayant pas de solution alors l'apprentissage est mis sous forme de régression. Cela consiste à résoudre le problème dit des moindres carrés :

$$f = \text{Argmin}\{\sum_{i=1}^{N}(f(x_i) - y_i)^2\} \tag{1.16}$$

D'autre part, en l'absence de contraintes supplémentaires sur la fonction f, le problème ainsi défini est mal posé (il possède une infinité de solution dans l'ensemble des fonctions de $\mathbb{R}^d$ dans $\mathbb{R}$). Il faut donc ajouter à cette définition certaines hypothèses sur la fonction f, c'est-à-dire restreindre la recherche à certains espaces de fonctions.

Le problème (1.15) devient alors :

Étant donné $C \in \mathbb{R}_+^*$*, trouver* $f \in \mathcal{H}$ *tel que*

$$\begin{array}{l} f(x_i) = y_i \quad i = 1..N \\ ||f||_{\mathcal{H}} \leq C \end{array} \tag{1.17}$$

pour un espace fonctionnel donné $\mathcal{H}$, et dans lequel la constante $C > 0$, donnée par l'utilisateur, permet de régler la régularité de f. Cependant, le problème (1.17) peut également ne pas avoir de solution. On transforme alors le problème d'interpolation en un problème d'approximation au sens des moindres carrés :

$$\begin{array}{c} f = \text{Argmin}\{\sum_{i=1}^{N}(f(x_i) - y_i)^2\} \\ ||f||_{\mathcal{H}} \leq C \end{array} \tag{1.18}$$

Une autre possibilité est de formuler le problème sous contrainte (1.18) sous forme relaxée(régularisée), ce qui donne

$$f = \text{Argmin}\{\sum_{i=1}^{N}(f(x_i) - y_i))^2 + \lambda||f||^2_{\mathcal{H}}\} \tag{1.19}$$

où le paramètre de relaxation (régularisation) λ permet de régler l'importance relative de la qualité de l'approximation (l'erreur au sens des moindres carrés) et de la régularité de la fonction (norme dans $\mathcal{H}$). Ce dernier problème est bien posé : c'est un problème de minimisation d'une fonction quadratique définie positive dans un espace de Hilbert, il a donc une solution unique.

Nous allons nous intéresser aux méthodes RBF et SVM qui utilisent cette formulation. La méthode SVM a spécialement retenu notre attention du fait qu'elle est particulièrement adaptée aux grandes dimensions.

Cependant, il existe d'autres méthodes d'interpolation classiques, comme les techniques inspirées des Éléments finis, les approximations basées sur les fonctions Splines, les approximations par fractions rationnelles (e.g. Padé), qui ne sont pas basés ni sur la théorie de régularisation ni sur la projection. Mais elles ne sont utilisées qu'en petite dimension. Nous les citons pour leur importance en analyse numérique sans trop nous y attarder.

Nous allons donc commencer par exposer les méthodes classiques. Ensuite nous parlerons des méthodes à noyaux, notamment les méthodes RBF (*Radial Basis Functions*). Ensuite nous présentons le Krigeage qui est une méthode de régression et nous terminons avec la méthode SVM.

1.5.1 Méthodes d'approximations classiques

Méthodes d'éléments finis

La méthode des éléments finis est d'abord une méthode d'approximation de fonctions avant d'être une méthode de résolution d'EDP. Le principe est relativement simple : on décompose le domaine en sous-domaines de formes géométriques simples (segments, triangles, tétraèdres...), avec certaines contraintes sur le type de jonctions qu'elles peuvent avoir, puis on considère des fonctions dont la restriction à chaque sous-domaine est un polynôme de bas degré. Suivant la régularité on impose ensuite des conditions de raccord aux interfaces. En dimension supérieure à 1, on impose rarement plus que la continuité. Le côté pratique des éléments finis est la relative facilité d'avoir une base. Dans chaque sous-domaine (élément) on définit ce

qu'on appelle des degrés de liberté qui déterminent de façon unique le polynôme dans l'élément. Par exemple, pour les éléments de type Lagrange, un degré de liberté est la valeur de la fonction en un point de l'élément. Pour des éléments triangles et un degré 1, les degrés de liberté sont les valeurs aux sommets. Ceci assure automatiquement la continuité au passage d'un triangle à l'autre. En dimension inférieure ou égale à 3, il existe des outils, appelé «mailleurs», qui construisent de façon plus ou moins automatique la décomposition en sous-domaine, ou maillage. En dimension supérieure à 3, il n'existe pas de tels outils et de toute façon la méthode ne serait pas trop efficace !

Ils existent deux types de grilles. Les grilles structurées : elles sont rectangulaires parallèles aux axes de coordonnées. Si on a n_i points par dimension alors le nombre de points total est $\prod n_i$. Ainsi, le nombre de points augmente exponentiellement avec la dimension de l'espace. Ces grilles sont donc à utiliser en petites dimensions. Les grilles non-structurées : leur maillage est de type éléments finis quelconques. Les points sont disposés d'une manière à respecter la régularité de la fonction inconnue qu'on cherche.

La régularité de la fonction n'est pas connu à l'avance. Différentes méthodes sont utilisées pour adapter le modèle.

Méthodes multi-grilles : il s'agit de chercher à approcher une fonction de manière itérative en utilisant tantôt une description sur une grille grossière, tantôt une grille fine. L'idée étant que les variations rapides de la fonction à approcher (i.e. les hautes fréquences spatiales) convergent rapidement dans le processus itératif alors les variations lentes (basses fréquences spatiales) convergent lentement. Or une grille grossière suffit pour décrire des comportements lents. On filtre alors les variations rapides, et on itère sur la grille grossière, ce faisant on diminue sensiblement la dimension de l'espace de recherche. Puis on rajoute le comportement rapide qu'on a enlevé et on itère sur la grille fine. Au bout de plusieurs aller-retours entre les deux grilles avec régularisation et projection à chaque fois, on arrive à convergence. Au total, on aura fait beaucoup moins d'itérations sur la grille fine et ainsi beaucoup réduit le temps de calcul.

Méthodes avec raffinement adaptatif Le raffinement du maillages EF s'effectue d'une manière itérative en se basant sur des estimateurs de l'erreur.

Méthodes spectrales : Exemple des polynômes de Legendre

Alors que les méthodes d'éléments finis utilisent des polynômes de bas degré et améliorent la qualité de l'approximation en prenant des éléments de plus en plus petits, les méthodes spectrales utilisent peu d'éléments et des

polynômes de haut degré. Elles sont donc applicables à des dimensions plus importantes que les méthodes d'éléments finis, mais sans toutefois pouvoir être utilisées dans les tailles d'espace envisagées dans cette thèse (quelques dizaines). Nous allons néanmoins présenter dans ce paragraphe un exemple simple que nous avons utilisé en petite dimension.

Pour des raisons pratiques, on utilise des bases orthogonales pour le produit scalaire L^2 avec poids, ce qui cantonne ces méthodes à des domaines de type hypercube. Pour des domaines plus généraux, il faut les coupler à d'autres méthodes comme les éléments finis. Le principal avantage de ces méthodes est qu'elles convergent exponentiellement vite pour des fonctions régulières. Un traitement spécial doit être fait pour le cas des fonctions singulières.

Nous voulons approcher une fonction par une somme de polynômes en minimisant l'erreur quadratique (norme L^2). Pour cela, nous aurons besoin d'une formule d'intégration numérique pour approcher les intégrales. Les polynômes de Legendre associés aux points de Gauss-Legendre se prêtent bien à une telle approche. Cela va nous conduire à évaluer la fonction à interpoler en des points spéciaux.

Rappelons que les polynômes de Legendre, qu'on note P_n, $n = 0, 1, \ldots$ sont des polynômes qui forment une base orthogonale de $L^2(]-1, 1[)$:

$$\int_{-1}^{1} P_n(x) P_m(x) dx = \frac{2}{2n+1} \delta_{n,m}$$

$\deg P_n = n$. La récurrence suivante permet de les calculer

$$(n+1) P_{n+1}(x) = (2n+1) x P_n(x) - n P_{n-1}(x)$$

avec $P_0(x) = 1$ et $P_1(x) = x$.

Le polynôme P_n possède n zéros réels dans l'intervalle ouvert $]-1, 1[$ qu'on note x_i^n, $i = 0, \ldots, n-1$. Si on note

$$\omega_i^n = \frac{2}{(1 - (x_i^n)^2)(P_n'(x_i^n))^2}$$

alors la formule d'intégration numérique suivante dite de Gauss-Legendre :

$$\int_{-1}^{1} f(x) dx \approx \sum_{i=0}^{n-1} \omega_i^n f(x_i^n) \tag{1.20}$$

intègre exactement les polynômes de degré inférieur ou égal à $2n-1$.

Projection L^2 Étant donnée une fonction $f : [-1,1] \longrightarrow \mathbb{R}$, on cherche le polynôme p, de degré inférieur ou égal à $N-1$, le plus proche de f au sens de la norme $L^2(]-1,1[)$:

$$p = \text{argmin}_{q \in \mathbb{R}_{N-1}[X]} \|f - q\|$$

p est donc la projection orthogonale de f sur l'espace $\mathbb{R}_{N-1}[X]$ des polynômes de degré inférieur ou égal à $N-1$, et admet donc le développement suivant :

$$p(x) = \sum_{n=0}^{N-1} f_n P_n(x)$$

les coefficients f_n pouvant être approchés par :

$$\frac{2}{2n+1} f_n = \int_{-1}^{1} f(x) P_n(x) dx \approx \sum_{n=0}^{N-1} \omega_i^N f(x_i^N) P_n(x_i^N)$$

La projection p peut donc être approchée de la manière suivante :

$$p(x) \approx \Pi^N f(x) = \sum_{n=0}^{N-1} f(x_i^N) \phi_i^N(x) \tag{1.21}$$

avec

$$\phi_i^N(x) = \omega_i^N \sum_{n=0}^{N-1} \frac{2n+1}{2} P_n(x_i^N) P_n(x) \tag{1.22}$$

Pour connaître $\Pi^n f$, il faut connaître f aux zéros x_i^N, $i = 0, \ldots, N-1$, du polynôme de Legendre de degré N. Un simple changement de variable nous permet de traiter le cas d'un intervalle $[a,b]$ quelconque.

On peut également généraliser cette formule au cas d'une fonction définie dans un hypercube en dimension d en utilisant les produits tensoriels.

1.5.2 Méthodes à noyaux : RBF

Dans les méthodes à noyaux, on cherche des approximations de la fonction f de la forme :

$$\widetilde{f}(x) = \sum_{j=1}^{n} b_j K(x_j, x)$$

On note

- K la matrice des noyaux : $K_{ij} = K(x_j, x_i)$

- $b = (b_i)$ le vecteur des coefficients
- $y = (y_i)$ les valeurs observées aux points x_i.

En général, on utilise des noyaux symétriques, *i.e.* $K(x, y) = K(y, x)$, ce qui conduit à une matrice K symétrique.

En traduisant

$$\widetilde{f}(x_i) = y_i \quad i = 1, \ldots, n$$

on aboutit au système

$$Kb = y$$

En général, ce problème est mal posé. On peut le résoudre au sens des moindres carré :

$$b = K^+ y$$

où K^+ est la pseudo-inverse de la matrice K. Ceci équivaut à minimiser l'erreur quadratique

$$b = \arg\min_{x \in \mathbb{R}^n} \|y - Kx\|^2$$

Pour améliorer la stabilité du système, on peut utiliser à nouveau la régularisation du problème de moindre carré :

$$b = \arg\min_{x \in \mathbb{R}^n} \|y - Kx\|^2 + \lambda \|x\|^2$$

Ceci revient à modifier le système à résoudre en

$$(K + \lambda I)b = y$$

Dans la pratique, le paramètre de régularisation est difficile à choisir. On préfère la solution consistant à rajouter des contraintes ce qui nous ramène à un système de type point selle :

$$\begin{bmatrix} K & D \\ D^t & 0 \end{bmatrix} \begin{bmatrix} b \\ a \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix}$$

Il suffit alors que K soit inversible sur le noyau de la matrice D^t.

En fait, dans ce cas, on cherche une approximation de f de la forme :

$$\widetilde{f}(x) = \sum_{i=1}^{n} b_i K(x_i, x) + \sum_{i=1}^{p} a_i d_i(x)$$

où les d_j sont des polynômes. D est la matrice $D_{ij} = d_j(x_i)$. Les coefficients des polynômes apparaissent comme les multiplicateurs de Lagrange associés aux contraintes. En général, les noyaux sont localisés et les polynômes d_j donnent donc la tendance globale de f. Sous certaines hypothèses que verrons plus loin, on peut démontrer que le système est inversible.

Voici quelques exemples noyaux utilisés :

- Noyau linéaire : pas très satisfaisant.
- Noyau gaussien : $K(y,z) = \exp(-\frac{\|y-z\|^2}{\sigma^2})$
- Noyau sigmoïdal : $K(y,z) = \tanh(s * y.z + r)$
- Noyau polynômial : $K(y,z) = (s * y.z + r)^2$

Dans le cas particulier où le noyau est isotrope, *i.e.*

$$K(x,y) = k(\|x-y\|) \text{ pour une fonction de variable réelle } k$$

on parle de *Radial Basis Function* ou RBF.

1.5.3 Méthodes de moindres carrées

Il s'agit d'un cadre général dans lequel le terme de régularisation évoqué plus haut devient l'objet de la minimisation, alors que la qualité de l'approximation devient elle une contrainte sur la minimisation de la norme. Concrètement, cela revient à rechercher une projection sur un sous-espace C d'un espace Hilbert $\mathcal{H}$. Le problème s'écrit sous la forme suivant :

$$\min_{x \in C \subset \mathcal{H}} \quad \mathcal{L}(x) = \|y-x\|^2 = \langle y-x, y-x \rangle \tag{1.23}$$

La solution x_* vérifie alors l'équation de la normale :

$$\langle x_* - y, \nu \rangle = 0 \quad \forall \nu \in C \tag{1.24}$$

Régression : fonctions paramétriques

Considérons des couples $(x_i, y_i) \in \mathbb{R}^p \times \mathbb{R}$, $i = 1, \ldots, n$, (point,valeur au point) et considérons une famille de fonctions $(F_j)_{1 \leq j \leq k}$, par exemple une base de polynômes. On cherche une fonction f combinaison linéaire des F_j qui passe à peu près par les points : on cherche $(\beta_j)_{1 \leq j \leq k}$ tel la fonction

$$f(x) = \sum_{j=1}^{k} \beta_j F_j(x)$$

vaut «à peu près» y_i en x_i :

$$y_i = f(x_i) + \epsilon_i$$

ce qui s'écrit matriciellement

$$y = X\beta + \epsilon$$

Avec

- y est le vecteur $n \times 1$ des valeurs aux points
- X est la matrice $n \times k$ des $F_j(x_i)$
- β est le vecteur $k \times 1$ les paramètres recherchés
- ϵ est le vecteur $n \times 1$ des erreurs résiduelles. On suppose que $\epsilon \in N(0, \Sigma)$.

L'objectif est d'estimer β minimisant les erreurs résiduelles.

$H = \mathbb{R}^n$ muni du produit scalaire $\langle u, v \rangle_\Sigma = u^T \Sigma^{-1} v$ est un espace de Hilbert. Nous pouvons alors déterminer β en résolvant le problème suivant :

$$\min_{\beta \in \mathbb{R}^k} \quad g(\beta) = \|y - X\beta\|_\Sigma^2 \tag{1.25}$$

Si $X_1, ..., X_k$ sont les colonnes de X, alors :

$$X\beta = \beta_1 X_1 + ... + \beta_k X_k$$

Notons $col(X) = Vec(X_1, ..., X_k)$. Alors le problème (1.25) est équivalent à

$$\min_{x \in col(X)} \quad h(x) = \|y - x\|_\Sigma^2 \tag{1.26}$$

On voit alors que le problème (1.26) est identique au problème de projection (1.23) avec $H = \mathbb{R}^k$ et $C = col(X)$.

Le problème (1.26) est connu sous le nom de problème des moindres carrés généralisé ou GLS (*Generalized Least Squares*). Quand $\Sigma = \sigma^2 I$, c'est la méthode des moindres carrés ordinaire.

Si les vecteurs X_i sont linéairement indépendants alors nous pouvons déterminer un estimateur $\hat{\beta}$ de β en résolvant les équations normales d'écrites dans l'équation (1.24). Ils s'écrivent matriciellement sous la forme :

$$X_i^T \Sigma^{-1}(y - \hat{y}) = 0 \quad \text{pour tout } i = 1, \ldots, k$$

avec $\hat{y} = X\hat{\beta}$, qui se réécrit :

$$X^T \Sigma^{-1}(y - \hat{y}) = X^T \Sigma^{-1} y - X^T \Sigma^{-1} X \hat{\beta} = 0$$

$\hat{\beta}$ est donc solution du système linéaire (dit système des équations normales) :

$$X^T \Sigma^{-1} X \hat{\beta} = X^T \Sigma^{-1} y$$

Dans le cas où les vecteurs $X_1, ..., X_k$ sont linéairement indépendantes, $X^T \Sigma^{-1} X$ est une matrice $k \times k$ symétrique définie positive et donc inversible. Nous pouvons alors écrire :

$$\hat{\beta} = (X^T \Sigma^{-1} X)^{-1} X^T \Sigma^{-1} y$$

Dans le cas d'une régression avec une fonction quadratique, elle s'écrive sous la forme :

$$\hat{f}(x) = \beta_0 + \sum_{1 \leq j \leq n} \beta_j x_j + \sum_{1 \leq j \leq k \leq n} \beta_{n-1+j+k} x_j x_k$$

En dimension 2, la matrice X s'écrit sous la forme :

$$X = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & x_{1,1}^2 & x_{1,2}^2 & x_{1,1}x_{1,2} \\ 1 & x_{2,1} & x_{2,2} & x_{2,1}^2 & x_{2,2}^2 & x_{2,1}x_{2,2} \\ & & & . & & \\ & & & . & & \\ 1 & x_{n,1} & x_{n,2} & x_{n,1}^2 & x_{n,2}^2 & x_{n,1} \; x_{n,2} \end{bmatrix}$$

Dans la pratique, pour des raisons de stabilité numérique, on ne résout pas les équations normales même si les vecteurs sont linéairement indépendant. En effet, le conditionnement de la matrice $X^T\Sigma^{-1}X$ est bien plus mauvais que le conditionnement de la matrice X ! Nous utilisons la décomposition en valeurs singulière (SVD, *Singular Value Decomposition*) de la matrice X pour calculer la pseudo-inverse X^+ de X. Rappelons brièvement de quoi il s'agit : Étant donnée une matrice A $m \times n$, il existe des matrices orthogonales U $m \times m$ et V $n \times n$ et une matrice diagonale $\Sigma = diag(\sigma_1, \ldots, \sigma_r)$ avec $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_r > 0$, tel que :

$$A = U \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} V^T$$

les vecteurs colonnes de V (qui forment une base orthonormée de $\mathbb{R}^n$) sont dits *vecteurs singuliers de A à droite*, les vecteurs colonnes de U (qui forment une base orthonormée de $\mathbb{R}^m$) sont dits *vecteurs singuliers de A à gauche*, les σ_k sont dits valeurs singulières de A, r est le *rang de A*. C'est la décomposition en valeurs singulière de A. Le conditionnement de A est donné par :

$$cond(A) = \frac{\sigma_r}{\sigma_1}$$

La matrice pseudo-inverse est donnée par

$$A^+ = V \begin{bmatrix} \Sigma^{-1} & 0 \\ 0 & 0 \end{bmatrix} U^T$$

Quand A est une matrice carrée inversible, $A^+ = A^{-1}$. La solution du problème de moindres carrés :

$$\min_x \|Ax - b\|$$

est donnée par

$$x = A^+ b$$

On démontre que si A est de rang plein, c'est l'unique solution du problème de moindres carrés, et sinon, c'est la solution de plus petite norme.

Numériquement, le rang n'est pas toujours déterminé clairement. On introduit la notion de rang numérique à ε près : on se donne $\varepsilon > 0$ petit qui représente le degré de précision qu'on a sur les données, et on considère qu'une valeur singulière σ_i calculée est nulle si

$$\sigma_i < \varepsilon \sigma_1$$

ε est ainsi une espèce de zéro numérique laissé au choix de l'utilisateur. Quand la matrice est très mal conditionnée, augmenter la valeurs de ε régularise (et stabilise donc) le problème de moindres carrés : le conditionnement est saturé par $1/\varepsilon$.

1.5.4 La méthode Krigeage

C'est une méthode d'interpolation spatiale. Le Krigeage est la méthode optimale, au sens statistique du terme, d'estimation de paramètres. Elle est utilisée autant pour l'interpolation que l'extrapolation. Le Krigeage porte le nom de son précurseur, l'ingénieur minier sud-africain D.G. Krige. Dans les années 50, Krige a développé une série de méthodes statistiques empiriques afin de déterminer la distribution spatiale de minerais à partir d'un ensemble de forage [102]. C'est cependant le français Matheron [103] qui a formalisé l'approche en utilisant les corrélations entre forages pour estimer la répartition spatiale. Elle est connue au nom d'"interpolation optimale" en météorologie et "méthode d'interpolation de Gauss-Markov".

Quelques définitions :

Une variable aléatoire, $F(\mathbf{x})$ où $\mathbf{x} \in \mathbb{R}^n$, est une fonction qui prend un ensemble de valeurs ou de réalisations selon une distribution de probabilité quelconque.

Une fonction aléatoire est un ensemble de variables aléatoires définies sur une région d'intérêt : $\{F(x), x \in \mathbb{R}^n\}$. Une réalisation de F est $f = \{f(x) : x \in \mathbb{R}^p\}$ une fonction déterministe.

Les moments importants d'une fonction aléatoire sont :

1. La moyenne : $m(x) = E\{F(x)\}$.
2. La variance : $Var\{F(x)\} = E\{[F(x) - m(x)]^2\}$
3. La covariance : $C(u, v) = E\{[F(u) - m(u)].[F(v) - m(v)]\}$.

Le Krigeage simple :

Nous supposons tout d'abord que f est une réalisation d'un processus stochastique F dont la moyenne $m(x) = 0$. Soit les points d'apprentissage $x_1, ...x_n \in \mathbb{R}^p$. Nous supposons que la fonction covariance $C(.,.)$ est connue et que la matrice $C = [C(x_i, x_j)] = Cov(F(x_i), F(x_j))]$ est définie positive.

Soit

$$y = \begin{bmatrix} f(x_1) \\ . \\ . \\ . \\ f(x_n) \end{bmatrix}$$

Pour $x \in \mathbb{R}^p$ nous cherchons une fonction approchée sous la forme

$$\hat{f}(x) = \sum_{i=1}^{n} b_i f(x_i) = \sum_{i=1}^{n} b_i y_i = y^t b,$$

On choisit $b(x) \in \mathbb{R}^p$ minimisant l'erreur quadratique :

$$\begin{aligned} E[y^t b - F(x)]^2 &= E[\textstyle\sum_{i=1}^{n} b_i F(x_i) - F(x)]^2 \\ &= E[\textstyle\sum_{i=1}^{n} b_i F(x_i)]^2 - E[2\sum_{i=1}^{n} F(x_i)F(x)] + E[F(x)]^2 \\ &= b^t C b - 2b^t D(x) + E[F(x)]^2 \\ &= b^t C b - 2b^t D(x) + E[F(x)]^2 \end{aligned}$$

Avec $D(x) = [C(x_1, x), ..., C(x_n, x)]^t$. C'est une fonction quadratique de b, avec comme minimum :

$$D(x) = Cb.$$

Notons que le b optimal dépends de x.

Si C est définie positive, alors on peut résoudre la dernière équation en utilisant la factorisation de Cholesky. Nous écrivons alors

$$b = C^{-1} D(x)$$

ce qui donne au final :

$$\hat{f}(x) = y^t C^{-1} D(x) \tag{1.27}$$

Cette solution s'appelle la *best linear unbiased predictor* (BLUP) de $f(x)$. En pratique la matrice C est très mal conditionnée nous utilisons alors la pseudo-inverse.

Remarquons que cette fonction interpole les points. En effet,

$$D(x_i) = [Cov(x_1, x_i), ..., Cov(x_n, x_i)]^t = C_i$$

C_i étant la i^{me}colonne de C, alors

$$C^{-1}D(x_i) = C^{-1}C_i = I_i$$

donc

$$\hat{f}(x_i) = y^t C^{-1} D(x_i) = y^t I_i = y_i = f(x_i)$$

Le Krigeage :

Jusqu'ici nous avons supposé que le processus stochastique était connu, de moyenne nulle. Nous supposons maintenant que F est un processus Gaussien avec la fonction moyenne $m(x) = a(x)^t\beta$ et comme fonction de covariance $c(u, v) = \sigma^2 r_\theta(u, v)$. Nous supposons que la fonction a est connue (e.g. a quadratique), que β est un vecteur inconnu, que σ^2 est inconnue et que $r_\theta(.,.)$ est un élément inconnu d'une famille connue de fonctions de corrélation.

Soit A la matrice $n \times q$ $[a_j(x_i)]$, $R(\theta)$ la matrice symétrique $n \times n$ $[r_\theta(x_i, x_j)]$ et soit

$$r(x;\theta) = \begin{bmatrix} r_\theta(x_1, x) \\ . \\ . \\ . \\ r_\theta(x_n, x) \end{bmatrix}$$

Alors, pour β et θ fixé, le BLUP de $f(x)$ est

$$\hat{f}(x) = a(x)^t\beta + (y - A\beta)^t R(\theta)^{-1} r(x;\theta) \tag{1.28}$$

Ainsi, en changeant β et θ, nous définissons une famille de fonction interpolatrice à partir de laquelle nous pouvons choisir un $\hat{f}$ modèle spécifique à f.

Enfin, pour un θ fixé le "maximum likelihood estimates(MLEs)" de β et σ^2 s'exprime explicitement :

$$\hat{\beta}(\theta) = \left[A^t R(\theta)^{-1} A\right]^{-1} A^t R(\theta)^{-1} y$$

et

$$\hat{\sigma}^2(\theta) = \frac{1}{n}[y - A\hat{\beta}(\theta)]^t R(\theta)^{-1}[y - A\hat{\beta}(\theta)]$$

Finalement pour calculer $\hat{\theta}$, le MLE de θ, il est courant de minimiser :

$$nlog\hat{\sigma}^2(\theta) + logdetR(\theta) \tag{1.29}$$

Finalement l'erreur au sens des moindres carrés de (1.28) de $\hat{f}(x)$ est :

$$M\hat{S}E(\hat{f}(x)) = \sigma^2 \left(1 - \begin{bmatrix} a(x)^t & r(x,\theta)^t \end{bmatrix} \begin{bmatrix} 0 & A^t \\ A & R(\theta) \end{bmatrix}^{-1} \begin{bmatrix} a(x) \\ r(x;\theta) \end{bmatrix} \right) \tag{1.30}$$

tel que la définissent Sacks, Welch, Mitchell et Wynn en (1989). Cette dernière joue un rôle majeur dans différentes approches en optimisation.

1.5.5 Supports Vecteurs Machines : SVM

Les *Support Vector Machines* (SVMs) sont des algorithmes d'apprentissage qui s'inspirent de techniques déjà utilisées dans le domaine de l'apprentissage automatique depuis les années 1960. Mises aux points au début des années 90 par Vapnik (voir le papier séminal par Boser, Guyon et Vapnik [123]) pour résoudre originellement des problèmes de classification binaire, leur adaptation aux problèmes de régression scalaire a été réalisée par Vapnik en 1995, puis systématisée dans son livre [137].

L'algorithme de classification binaire le plus simple est la séparation linéaire : on cherche le meilleur hyperplan séparant deux classes (+ et -). On cherche l'hyperplan séparateur qui passe «au milieu» de ces deux classes, c.à.d. dont la distance aux points les plus proches est maximale. Ces exemples les plus proches qui suffisent à déterminer cet hyperplan sont appelés veteurs de support, ou encore exemples critiques. La distance séparant l'hyperplan de ces points est appelée «marge». Plus la marge est grande et plus l'apprentissage est sûr : avec de petites variations, une petite variation ne modifiera pas la classification d'un exemple si sa distance à l'hyperplan est grande. Bien entendu, les limites de cet algorithme sont bien connues, et la plupart des problèmes réels ne sont pas linéairement séparables. L'idée forte des SVM est de plonger le problème dans un espace de dimension (beaucoup) plus grande dans lequel il devient linéairement séparable. C'est pourquoi certains auteurs ont détourné l'acronyme en Séparateurs à Vaste Marge ! Et ce qui a permis d'en faire une méthode utilisable en pratique est ce qu'on a ensuite appelé le *kernel trick* qui permet, par l'utilisation de noyaux bien choisis, de résoudre le problème d'optimisation linéaire dans l'espace de grande dimension en n'ayant à calculer que des quantités dans l'espace original.

Nous allons dans cette section détailler tout d'abord les deux ingrédients (Régression linéaire et *kernel trick*), puis présenter l'utilisation des SVMs pour les problèmes de régressions, qui est celle qui en sera faite dans cette thèse.

Il s'agit d'effectuer une régression linéaire dans un espace de caractéristiques de grande dimension induit par un noyau, en résolvant en dualité un programme d'optimisation quadratique sous contraintes d'inégalité. **Les points de l'ensemble d'apprentissage qui correspondent aux contraintes actives sont appelés Vecteurs de Support.**

Les avantages de cette approche sont multiples. Tout d'abord, elle s'appuie sur des bases théoriques solides : théorie de l'apprentissage statistique, théorie de l'optimisation (la condition d'optimalité de Kuhn et Tucker), théorie des noyaux de Mercer. La théorie permet d'obtenir des estimations d'erreur sur la généralisation. Les Vecteurs de Support constituent une représentation "creuse" de la solution, qui fait que les SVMs peuvent être considérées comme des schémas de compression. Les SVMs réalisent un apprentissage linéaire efficace dans un espace de caractéristiques de grande dimension. Enfin, on peut traiter des échantillons de grande taille grâce à diverses heuristiques.

Régression linéaire de $\mathbb{R}^d$ dans $\mathbb{R}$

Le problème de régression linéaire s'énonce de la façon suivante : étant donné un échantillon ou ensemble d'apprentissage

$$S = ((x_1, y_1), ..., (x_l, y_l)) \in (\mathbb{R}^d \times \mathbb{R})^l$$

on cherche une fonction affine :

$$\hat{f}(x) = \langle w, x \rangle + b$$

qui approche les exemples au sens des moindres carrés tout en gardant une régularité minimale, ce qui est assuré en pénalisant l'erreur de moindres carrés par une norme de la fonction (Hoerl et Kennard, 1970) [107], par exemple la norme L^2 ce qui amène au problème suivant :

$$\min_{(w,b)} \quad \mathcal{L}(w, b) = \lambda \|w\|^2 + \sum_{i=1}^{l} (y_i - \langle w, x_i \rangle - b)^2 \tag{1.31}$$

où $\lambda \geq 0$ est le paramètre de régularisation.

La condition d'optimalité

$$\frac{\partial \mathcal{L}}{\partial b} = 0$$

donne immédiatement

$$b = \bar{y} - \langle w, \bar{x} \rangle$$

où $\bar{x}$ et $\bar{y}$ sont les moyennes des x_i et y_i respectivement. On pose :

$$\tilde{x}_i = x_i - \bar{x}, \quad \tilde{y}_i = y_i - \bar{y}$$

$$\tilde{X} = \begin{pmatrix} \tilde{x}_1^t \\ \vdots \\ \tilde{x}_l^t \end{pmatrix}, \quad \tilde{y} = \begin{pmatrix} \tilde{y}_1 \\ \vdots \\ \tilde{y}_l \end{pmatrix}$$

La fonctionnelle L se réécrit

$$\mathcal{L}(w, b) = \lambda \|w\|^2 + (\tilde{y} - \tilde{X}w)^t(\tilde{y} - \tilde{X}w)$$

et nous pouvons maintenant écrire les autres conditions d'optimalité sous la forme

$$0 = \frac{\partial \mathcal{L}}{\partial w} = (\tilde{X}^t\tilde{X} + \lambda I_d)\, w = \tilde{X}^t\tilde{y}$$

ce qui donne facilement w

$$w = (\tilde{X}^t\tilde{X} + \lambda I_d)^{-1}\tilde{X}^t\tilde{y}$$

D'un point de vue pratique, par contre, la matrice $d \times d$ du système ci-dessus est pleine, et donc de plus en plus difficile à inverser quand la dimension croît.

On peut reformuler le problème (1.31 en un problème d'optimisation quadratique sous contraintes en introduisant les variables d'erreur ξ_i de la manière suivante :

$$\begin{array}{ll} \min\limits_{(w,b,\xi)} & \lambda \|w\|^2 + \sum_{i=1}^{l} \xi_i^2 \\ \text{sous les contraintes} & y_i - \langle w, x_i \rangle - b = \xi_i, \quad i = 1, ..., l \end{array} \tag{1.32}$$

On introduit les multiplicateurs de Lagrange β_i, et le Lagrangien :

$$\mathcal{L}(w, b, \xi, \beta) = \quad \lambda \|w\|^2 + \sum_{i=1}^{l} \xi_i^2 + \sum_{i=1}^{l} \beta_i\, h_i(w, b, \xi)$$

Avec :

$$h_i(w, b, \xi) = y_i - \langle w, x_i \rangle - b - \xi_i$$

Les conditions nécessaires et suffisantes d'optimalité s'écrivent :

$$\left\{ \begin{array}{rcl} \dfrac{\partial \mathcal{L}}{\partial (w, b, \xi)}\,(w^*, b^*, \xi^*, \beta^*) & = & 0 \\ & & b \\ \dfrac{\partial \mathcal{L}}{\partial \beta}\,(w^*, b^*, \xi^*, \beta^*) & = & 0 \end{array} \right.$$

Le problème dual s'écrit :

$$\max_{\beta} \quad \theta(\beta)$$

Où :

$$\theta(\beta) = \inf_{(w,b,\xi)} (\mathcal{L}, b, \xi, \beta)$$

Réécrivant les conditions d'optimalité $\frac{\partial L}{\partial(w,b,\xi)} = 0$, il vient

$$\left(w = \frac{1}{2\,\lambda} \sum_{i=1}^{l} \beta_i \, x_i, \quad \xi_i = \frac{1}{2}\,\beta_i, \quad \sum_{i=1}^{l} \beta_i = 0 \right)$$

D'où :

$$\theta(\beta) = \langle \beta, y \rangle - \frac{1}{2}\,\beta' \,(\frac{1}{2\,\lambda}\,G + \frac{1}{2}\,I_l)\,\beta, \quad \text{où} \quad G_{ij} = \langle x_i, x_j \rangle$$

qui atteint son maximum en :

$$\beta^* = 2\,\lambda\,(G + \lambda\,I_l)^{-1}\,y$$

ce qui donne

$$w^* = X'(G + \lambda\,I_l)^{-1}\,y$$

et donc :

$$\hat{f}(x) = y'\,(G + \lambda\,I_l)^{-1} \begin{pmatrix} \langle x_1, x \rangle \\ \vdots \\ \langle x_l, x \rangle \end{pmatrix} + b \tag{1.33}$$

Par ailleurs, les contraintes donnent :

$$b = \frac{1}{l} \sum_{i=1}^{l} y_i - \langle w^*, x_i \rangle = \frac{1}{l} \sum_{i=1}^{l} y_i - y'\,(G + \lambda\,I_l)^{-1} \begin{pmatrix} \langle x_1, x_i \rangle \\ \vdots \\ \langle x_l, x_i \rangle \end{pmatrix} \tag{1.34}$$

Certes, la représentation reste toujours "pleine". Par contre la solution ne fait intervenir que des produits scalaires de la forme $\langle x_i, x_j \rangle$ ou $\langle x_i, x \rangle$ – et c'est ce qui va permettre l'utilisation des noyaux (*kernel trick*).

Transformation non-linéaire : Les noyaux

Nous allons donc tout d'abord considérer une transformation de l'espace de recherche initial, appelé espace des *attributs* en un espace de (beaucoup) plus grande dimension dont les coordonnées seront appelées *caractéristiques*.

Passage de l'espace des *attributs* à l'espace des *caractéristiques* :

$$\begin{array}{rcl} X \subset \mathbb{R}^d & \longrightarrow & F \\ x = (x_1, ..., x_n) & \longmapsto & \Phi(x) = (\Phi_1(x), ..., \Phi_N(x)) \end{array}$$

Le choix de la fonction Φ est crucial, et pour ne pas avoir à utiliser explicitement les Φ_i dans les calculs, on utilisera des noyaux définis par

Définition 1. *Un noyau K sur un ensemble X est une fonction de $X \times X$ dans $\mathbb{R}$ telle qu'il existe un espace pré-Hilbertien F et une fonction $\Phi : X \to F$ tels que :*

$$\forall\ (x, z) \in X \times X, \quad K(x, z) = \langle \Phi(x), \Phi(z) \rangle_F$$

où F est l'espace de caractéristiques induit par le noyau K.

On voit que l'utilisation de noyaux permet le calcul des produits scalaires $\langle \Phi(x), \Phi(z) \rangle_F$ dans l'espace des attributs initial sans connaître explicitement Φ. Ce sera le cas dans les équations (1.33) et (1.34).

La théorie de Mercer caractérise les noyaux. Rappelons les propriétés principales des noyaux puis donnons quelque exemples.

Propriétés de structure
Si K_1 et K_2 sont des noyaux, alors les K définis ci-dessous le sont également :

$$\begin{array}{rcl} K(x, z) & = & K_1(x, z) + K_2(x, z) \\ K(x, z) & = & a\, K_1(x, z), \quad a > 0 \\ K(x, z) & = & K_1(x, z)\, K_2(x, z) \\ K(x, z) & = & exp(K_1(x, z)) \end{array}$$

Quelques exemples de noyaux classiques
Les noyaux suivants sont parmi les plus utilisés

$K(x, z)$	$=$	$x^t\, B\, z,$	Noyau quadratique (B symétrique positive)
$K(x, z)$	$=$	$tanh(s * \langle x, z \rangle + t)$	Noyau sigmoïdal
$K(x, z)$	$=$	$(\langle x, z \rangle + c)^d$	Noyau polynomial
$K(x, z)$	$=$	$exp(-\|x - z\|^2/\sigma^2)$	Noyau gaussien

La régression linéaire dans l'espace des caractéristique s'écrit de manière

très simple

$$\begin{aligned}\hat{f}(x) &= y'\,(K+\lambda\,I_l)^{-1}\begin{pmatrix}\langle\Phi(x_1),\Phi(x)\rangle\\ \vdots\\ \langle\Phi(x_l),\Phi(x)\rangle\end{pmatrix}+b\\ &= y'\,(K+\lambda\,I_l)^{-1}\begin{pmatrix}K(x_1,x)\\ \vdots\\ K(x_l,x)\end{pmatrix}+b\end{aligned}$$

avec :

$$K_{ij}=\langle\Phi(x_i),\Phi(x_j)\rangle=K(x_i,x_j)$$

et :

$$\begin{aligned}b &= \tfrac{1}{l}\textstyle\sum_{i=1}^{l}y_i-\langle w^*,x_i\rangle\\ &= \tfrac{1}{l}\textstyle\sum_{i=1}^{l}y_i-y'\,(K+\lambda\,I_l)^{-1}\begin{pmatrix}K(x_1,x_i)\\ \vdots\\ K(x_l,x_i)\end{pmatrix}\end{aligned}$$

Nous allons maintenant présenter les deux extensions qui ont été proposées pour faire de la régression dans le contexte des SVMs. Dans les deux cas, on repart de la formulation avec variables d'erreur (1.32). Toutefois, au lieu de minimiser, au travers des variables ξ_i, les erreurs $|\hat{f}(x_i)-y_i|,\quad i=1,...,l$ on cherchera à minimiser les *fonctions de perte ε-insensibles* $|\hat{f}(x_i)-y_i|_\varepsilon,\quad i=1,...,l$, où $|z|_\varepsilon=\max(0,|z|-\varepsilon)$.

Les deux méthodes qui vont être décrites ne diffèrent que par la manière de prendre en compte ces fonctions de perte, quadratique ou linéaire.

Régressions à fonction de perte quadratique

L'approche quadratique consiste à prendre en compte l'erreur ε-insensible de la manière suivante :

$$\begin{array}{ll}\min\limits_{(w,b,\xi)} & \|w\|^2+C\sum_{i=1}^{l}(\xi_i^2+\hat{\xi}_i^2)\\ \text{sous} & \langle w,\Phi(x_i)\rangle+b-y_i\le\varepsilon+\xi_i,\quad i=1,...,l\\ & y_i-\langle w,\Phi(x_i)\rangle-b\le\varepsilon+\hat{\xi}_i,\quad i=1,...,l\\ & \xi_i\ge 0,\ \hat{\xi}_i\ge 0,\quad i=1,...,l\end{array}$$

Il s'agit d'un problème de régression régularisé, où la régularisation est contrôlée par le paramètre $C>0$. Notons que C agit comme $\frac{1}{\lambda}$(voir 1.32).

Ce problème est un programme quadratique convexe, qui est donc équivalent au problème Lagrangien dual, qui, après quelques simplifications, peut

s'écrire :

$$\begin{array}{ll} \max\limits_{(\alpha,\hat{\alpha})} & \sum_{i=1}^{l} y_i(\hat{\alpha}_i - \alpha_i) - \varepsilon \sum_{i=1}^{l}(\hat{\alpha}_i + \alpha_i) \\ & \qquad -\frac{1}{2}\sum_{i,j=1}^{l}(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j)(K(x_i, x_j) + \frac{1}{C}\delta_{ij}) \\ \text{sous} & \sum_{i=1}^{l}(\hat{\alpha}_i - \alpha_i) = 0, \quad i = 1, ..., l \\ & \alpha_i \geq 0,\ \hat{\alpha}_i \geq 0, \quad i = 1, ..., l \end{array}$$

où l'on remarque ici encore que les points x_i n'apparaissent que dans les produits de noyau $K(x_i, x_j)$.

Les conditions de complémentarité de Karush-Kuhn-Tucker [98] s'écrivent :

$$\begin{array}{l} \alpha_i^* \left(\langle w^*, \Phi(x_i)\rangle + b^* - y_i - \varepsilon - \xi_i^*\right) = 0, \quad i = 1, ..., l \\ \hat{\alpha}_i^* \left(y_i - \langle w^*, \Phi(x_i)\rangle - b^* - \varepsilon - \hat{\xi}_i^*\right) = 0, \quad i = 1, ..., l \end{array}$$

On note aussi que :

$$\xi_i^* \hat{\xi}_i^* = 0 \quad \text{et} \quad \alpha_i^* \hat{\alpha}_i^* = 0, \quad i = 1, ..., l$$

Les conditions d'optimalité donnent la relation :

$$w^* = \sum_{i=1}^{l} (\hat{\alpha}_i{}^* - \alpha_i^*)\, \Phi(x_i)$$

Les points x_i tels que $(\hat{\alpha}_i{}^* + \alpha_i^*) > 0$ sont les Vecteurs de Support.

On trouve :

$$\hat{f}(x) = \langle w^*, \Phi(x)\rangle + b^* = \sum_{i=1}^{l} (\hat{\alpha}_i{}^* - \alpha_i^*)\, K(x_i, x) + b^*$$

où b^* est choisi tel que $h(x_i) - y_i = -\varepsilon - (\hat{\alpha}_i{}^* - \alpha_i^*)/C$ pour tout i tel que $(\hat{\alpha}_i{}^* + \alpha_i^*) > 0$.

Régressions à fonction de perte linéaire

Dans cette approche, on cherche à minimiser la somme des erreurs ε-sensibles, ce qui se traduit par le problème régularisé suivant :

$$\begin{array}{ll} \min\limits_{(w,b,\xi)} & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \hat{\xi}_i) \\ \text{sous} & \langle w, \Phi(x_i)\rangle + b - y_i \leq \varepsilon + \xi_i, \quad i = 1, ..., l \\ & y_i - \langle w, \Phi(x_i)\rangle - b \leq \varepsilon + \hat{\xi}_i, \quad i = 1, ..., l \\ & \xi_i \geq 0,\ \hat{\xi}_i \geq 0, \quad i = 1, ..., l \end{array}$$

Le problème Lagrangien dual peut se simplifier en éliminant certains multiplicateurs, ce qui conduit à la formulation suivante :

$$\begin{array}{ll} \max\limits_{(\alpha,\hat{\alpha})} & \sum_{i=1}^{l} y_i(\hat{\alpha}_i - \alpha_i) - \varepsilon \sum_{i=1}^{l}(\hat{\alpha}_i + \alpha_i) \\ & \qquad -\frac{1}{2}\sum_{i,j=1}^{l}(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j)K(x_i, x_j) \\ \text{sous} & \sum_{i=1}^{l}(\hat{\alpha}_i - \alpha_i) = 0, \quad i = 1, ..., l \\ & 0 \leq \alpha_i \leq C,\ 0 \leq \hat{\alpha}_i \leq C, \quad i = 1, ..., l \end{array}$$

Les conditions de complémentarité de Karush-Kuhn-Tucker donnent :

$$\begin{array}{ll} \alpha_i^* \left(\langle w^*, \Phi(x_i)\rangle + b^* - y_i - \varepsilon - \xi_i^*\right) = 0, & i = 1, ..., l \\ \hat{\alpha}_i^* \left(y_i - \langle w^*, \Phi(x_i)\rangle - b^* - \varepsilon - \hat{\xi}_i^*\right) = 0, & i = 1, ..., l \\ (\alpha_i^* - C)\,\xi_i^* = 0, & i = 1, ..., l \\ (\hat{\alpha}_i^* - C)\,\hat{\xi}_i^* = 0, & i = 1, ..., l \end{array}$$

Avec encore :

$$\xi_i^* \hat{\xi}_i^* = 0 \quad \text{et} \quad \alpha_i^* \hat{\alpha}_i^* = 0, \quad i = 1, ..., l$$

Les conditions d'optimalité permettent de montrer :

$$w^* = \sum_{i=1}^{l} (\hat{\alpha}_i{}^* - \alpha_i^*)\, \Phi(x_i)$$

Les points x_i tels que $(\hat{\alpha}_i{}^* + \alpha_i^*) > 0$ sont les Vecteurs de Support.

On trouve :

$$\hat{f}(x) = \langle w^*, \Phi(x)\rangle + b^* = \sum_{i=1}^{l} (\hat{\alpha}_i{}^* - \alpha_i^*)\, K(x_i, x) + b^*$$

où b^* est choisi tel que $h(x_i) - y_i = -\varepsilon$ pour tout i tel que $0 < \alpha_i^* < C$, et tel que $h(x_i) - y_i = \varepsilon$ pour tout i tel que $0 < \hat{\alpha}_i^* < C$.

Implémentation des SVM de régressions

Il existe de nombreux algorithmes permettant de résoudre des programmes quadratiques duaux. La plupart des méthodes de maximisation de formes quadratiques convexes, comme les méthodes de gradient, peuvent s'appliquer au cas des SVMs. Cependant elles nécessitent en général le stockage de la matrice des $K(x_i, x_j)$, qui est une matrice pleine. Ceci limite, sur un PC récent «musclé», la taille des ensembles d'apprentissage à quelques milliers d'exemples.

Les méthodes de *décomposition* permettent quant à elles de traiter des échantillons de grande taille en tentant de limiter le problème d'optimisation aux seules contraintes actives. Ces méthodes reposent sur l'application itérative d'heuristiques de sélection d'«ensembles de travail», qui visent à identifier les contraintes actives. La convergence des méthodes de décomposition est en général assurée par la réduction de l'écart de faisabilité, ou bien l'accroissement de la fonction objective duale. La plus célèbre d'entre elles est sans doute l'algorithme SMO (Sequential Minimal Optimisation) de Platt [134].

Nous avons utilisé un autre algorithme de décomposition, appelé *SVMTorch*, et dù à R. Collobert et S. Bengio [124]. *SVMTorch* s'inspire de l'algorithme *SVM-Light* de Joachims [131] pour la classification, en l'adaptant aux problèmes de régression à fonction de perte linéaire. Une implémentation en C++ est disponible sur le site `www.torch.ch`.

1.6 Méthodes hybrides

On appelle hybridation le fait de combiner plusieurs méthodes distinctes en vue de la résolution d'un même problème. Les différentes approches peuvent constituer des étapes successives dans la résolution du problème donné, mais peuvent aussi interagir de manière beaucoup plus imbriquée, une méthode en appelant une autre plusieurs fois lors de son exécution.

Dans le cadre de cette thèse, pour résoudre un problème d'optimisation globale en grande dimension, nous considérerons des hybridations entre des approches stochastiques d'optimisation globale et des approches déterministes d'optimisation locale et des méthodes d'apprentissage.

Des travaux antérieurs ont proposé des hybridations entre des méthodes d'optimisation stochastiques et des méthodes d'optimisation déterministes : La manière la plus évidente d'effectuer une telle hybridation consiste à effectuer une recherche globale à l'aide de la méthode stochastique, l'approche déterministe étant chargée du raffinement local. Nous pouvons cité la plus simple et connue sous le nom de "Multistart"(plusieurs tirage) [4]. Dans le cas d'une fonction "légèrement" multimodale, lancer un algorithme d'optimisation déterministe à partir de points échantillonnés de manière aléatoire sur le domaine de recherche peut être vu comme de degré 0 de telles hybridations, la méthode stochastique étant vue comme une méthode de Monte-Carlo. Cette approche devient bien sûr problématique s'il y a beaucoup de minima locaux, et/ou en grande dimensions. On peut alors remplacer l'échantillonnage naïf par un algorithme stochastique plus sophistiqué, de type évolutionnaire

par exemple. On obtient ce que l'on appelle alors un algorithme *mimétique*. Notons que ce type d'approche, qui a permis d'obtenir des résultats exceptionnels dans le cadre de l'optimisation combinatoire [3], sont rapidement insuffisants dans le cadre de l'optimisation continue, et surtout ne couvrent pas le cas de fonctions chères en grande dimensions.

D'autres travaux ont proposé d'hybrider une méthode d'optimisation avec une méthode d'apprentissage. La méthode d'apprentissage ajuste un modèle de la fonction objectif, qui est alors optimisé par la méthode d'optimisation, en lieu et place de la fonction objectif initiale.

Certaines fonctions éventuellement fortement multimodales peuvent ainsi être approchées sans changer la nature du problème ni modifier la localisation de l'optimum global. Une méthode déterministe pure pourrait tomber dans un minimum local en travaillant sur la fonction objectif initiale, alors qu'une fonction approchée remplaçant la fonction objectif permet, en lissant les optima locaux, d'aller de manière déterministe au minimum global.

Dans le cas d'une fonction très fortement multimodale, l'utilisation d'un algorithme déterministe de recherche locale pour échantillonner l'ensemble des minima locaux, combiné avec une méthode d'approximation qui est appliqué à cet ensemble de minimaux locaux peut également permettre de localiser le minimum global.

Cependant, la motivation première de l'hybridation avec une méthode d'approximation reste la diminution du coût dans le cas d'une fonction objectif chère : l'évaluation de la fonction approchée est en général négligeable par rapport à l'évaluation de la fonction objectif initiale. Nous nous placerons dans ce cadre dans toute la suite de la thèse.

Nous pouvons classer les méthodes de ce type proposées dans la littérature en deux familles :

La première famille est celle des algorithmes manipulant un point unique : c'est le cas de toutes les méthodes déterministes, ainsi que des algorithmes stochastiques ne manipulant qu'un unique itéré. L'hybridation dans cette famille porte le nom de SAO (*Sequential Approximate Optimization*).

La deuxième famille est celle basée sur l'utilisation en tant qu'algorithmes d'optimisation des algorithmes évolutionnaires, à base de population, que nous regroupons sous le nom de EOA (*Evolutionary Optimization Approximation*).

Dans les deux cas, différentes méthodes d'approximation, locales ou globales, seront utilisées, selon que nous disposons ou pas des dérivées de la fonction objectif.

Nous allons passer en revue les travaux antérieurs dans les deux familles, après avoir au préalable commencé par exposer la problématique d'une hy-

bridation avec les méthodes d'approximation. Nous insisterons plus spécialement sur les méthodes EOA, cadre de la présente thèse.

1.6.1 Problématique de l'apprentissage

Nous allons examiner dans cette section les deux principaux problèmes qui se posent lorsque l'on veut utiliser une méthode d'apprentissage au sein d'un algorithme hybride.

Premièrement, se pose la question de la mise à jour du modèle. D'une part, toutes les méthodes d'approximation ne se prêtent pas facilement à une mise à jour incrémentale avec de nouveaux points. D'autre part, tant que le modèle peut être considéré comme une bonne approximation de la fonction objectif, il est inutile de le remettre à jour. En général la mise à jour s'effectue quand le modèle s'éloigne de l'original, ou dès que l'on dispose d'un nombre de points suffisant. À noter qu'un problème annexe du problème de la mise à jour est celui du réglage des paramètres de l'algorithme d'apprentissage lui-même, souvent appelés hyper-paramètres. Le chapitre 3 est entièrement consacré à ce problème dans le cadre de l'hybridation avec une méthode stochastique.

Deuxièmement, se pose la question du nombre de points à utiliser pour la construction du modèle, et de la méthode de génération de points. En particulier, les points rencontrés durant les étapes précédentes de l'optimisation sont-ils suffisants pour définir un bon modèle ? De plus, si les points disponibles sont trop proches, nous pourrons avoir du mal à définir un modèle dans un temps raisonnable. Si les points sont mal disposés, nous risquons de ne pas détecter toutes les variations de la fonction d'origine. Parmi les méthodes utilisées dans la littérature pour générer des points, citons : "Latin hypercube sampling" [5], points sur un maillage, Orthogonal Array [6], les opérateurs génétiques (mutation gaussienne par exemple), ou tout simplement générés uniformément dans un hypercube, dans un ellipse (e.g. dans la méthodes CMA) ou dans un sous-espace. Le nombre de points utilisés est parfois considéré comme variable avec des variations dynamiques selon l'erreur obtenue sur le modèle, mais la plupart du temps il est maintenu fixe.

Un autre point important qui sera détaillé dans cette section est le type de la méthode d'approximation qu'on choisit. L'approche est différente selon qu'on utilise une approximation locale ou globale. Une approximation locale est d'ordre 1 autour d'un point si l'approximation passe par les points et utilise les gradients.

Une approximation locale est d'ordre 2 si l'approximation passe par les points et vérifie l'égalité des gradients et des matrices hessiennes. Une ap-

proximation locale est souvent hybridée avec la méthode de "la région de confiance" dans l'approche "Sequential Approximate Optimization"(SAO).

Une approximation globale est une approximation d'ordre zéro. Il est courant d'utiliser dans ce cas ce que l'on appelle la "Fonction mérite".

Les deux approches région de confiance et fonction de mérite vont maintenant être détaillées.

Région de confiance

Les méthodes locales sont basées sur la connaissance en au moins un point, de la vraie valeur de la fonction objectif, de son gradient et si possible de sa matrice Hessienne. Lorsque l'on dispose de ces connaissances, on peut construire une approximation quadratique efficace de la fonction objectif. Dans le cas où nous ne disposons pas des dérivée seconde, il est courant d'utiliser comme matrice hessienne le résultat d'une récurrence du type de celle utilisée dans la méthode BFGS (voir section 1.3) qui tend vers la matrice hessienne lorsqu'on s'approche de l'optimum.

Pour plus de détails sur le principe de la méthode de "région de confiance" dans le cas général, voir la section 1.3.3.

Nous allons décrire une variante de cette méthode, décrite dans [55], qui s'applique dans le cas d'une approximation quadratique d'ordre 0 :

La région de confiance est une boule de rayon δ_k (à déterminer à l'itération k) centrée sur l'itéré courant. On choisit des constantes positives $r_1 < r_2 < 1$ et $c_1 < 1$, $c_2 > 1$ qui vont réguler les expansions et les contractions de la région de confiance.

On note r le rapport entre la diminution prédite et la diminution effective et $\hat{f}$ le modèle quadratique.

$$r = \frac{f(x_k) - f(x_k + s)}{f(x_k) - \hat{f}(x_k + s)}$$

On adapte la région de confiance de la manière suivante :

$$\delta_{k+1} = \begin{cases} c_1||s|| & \text{si} \quad r < r_1 \\ min(c_2||\delta_k||, \Delta^*) & \text{si} \quad r > r_2 \\ ||s|| & \text{sinon,} \end{cases} \tag{1.35}$$

Δ^* : une borne supérieure pour δ.

Et on décide si on accepte l'étape s :

$$x_{k+1} = \begin{cases} x_k + s & \text{si} \quad f(x_k + s) < f(x_k) \\ x_k & \text{sinon,} \end{cases} \tag{1.36}$$

Algorithme

On choisit $x_0 \in R^n$ et $\delta_0 > 0$.

Pour $k = 0, 1, ...$ jusqu'à convergence faire

1. Trouver une solution s_k du sous-problème :

$$\min_{||s|| \leq \delta_k.} \hat{f}_k(x_k + s)$$

2. On compare la diminution actuelle avec celle prédite : r.
3. On met à jour x_k suivant l'équation (1.36) et δ_k suivant l'équation (1.35).
4. On met à jour le modèle $\hat{f}_k$ avec le nouveau point déjà évalué pour devenir $\hat{f}_{k+1}$.

La différence entre un algorithme classique de région de confiance utilisant une fonction quadratique d'ordre 1 est que le modèle quadratique d'ordre 0 est mis à jour à chaque fois qu'on dispose de nouveau point.

Fonction de mérite

Les méthode d'approximation globales sont des projections sur des espaces hilbertiens utilisant des points évalués avec la fonction objectif, sans passer forcément par les points ni les gradients, pré-requis pour la méthode de région de confiance !

L'utilisation de l'algorithme de la région de confiance devient alors problématique. En effet, la convergence en nombre fini d'itérations qui vient du fait que localement le modèle colle au mieux à la fonction objectif n'est plus assurée. Une alternative est alors la méthode de la fonction de mérite.

La fonction de mérite est une pénalisation pour la prise en compte de contraintes dans l'optimisation de la fonction modèle. C'est une optimisation avec contrainte ! L'idée de départ est d'optimiser la fonction modèle dans un voisinage des points d'apprentissage. Dans l'approche Krigeage, il est possible de calculer une approximation du $\hat{MSE}$(Mean Squared Error). Cette fonction donne une idée en un point x quelconque de l'écart type de l'erreur du modèle par rapport à $f(x)$. Si la valeur $\hat{MSE}$ en un point est grande alors le modèle en ce point peut être loin de la fonction objectif. Au contraire si $\hat{MSE}$ est petite alors la valeur du modèle est probablement proche de la fonction objectif. $\hat{MSE}$ aux points d'apprentissages est nulle et plus on s'éloigne des points d'apprentissage $\hat{MSE}$ est grand.

Dans le cas d'un noyau gaussien, $\hat{MSE}$ est reliée à la densité des points. Cela rejoint l'idée de départ (rester près des points d'apprentissage).

La fonction de mérite qui a été étudiée par Sacks et al. en 1989 est sous la forme :

$$S(x) = \hat{f}(x) - w\sqrt{\hat{MSE}(\hat{f}(x))}$$

L'adaptation du poids w permet d'optimiser sur un domaine plus ou moins grand et permet ainsi une exploration dans des régions éloignées des points. Dans le cas des problèmes d'optimisation avec contrainte cela permet aussi d'équilibrer entre une exploration et la recherche dans un espace faisable.

En 1998 Virginia Torczon et Michael W. Trosset dans [58] étendent l'approche à d'autres méthodes d'approximation.

Soit

$$d(x) = \min \|x - x_i\|_2$$

alors la fonction de mérite s'écrit :

$$S(x) = \hat{f}(x) - wd(x)$$

Hyoung-Seog Chung et Alonso(2000) [83] suggèrent dans le cas d'une fonction quadratique la fonction de mérite suivante :

$$S(x) = \hat{f}(x) - (wd(x))^2$$

Christopher M. Siefert [65] dans le cadre des SAO a proposé d'adapter w dynamiquement de la manière suivante :

$$w_{k+1} = \begin{cases} m_g.w_k, & \text{si} \quad |f(x_k) - \hat{f}_k(x_k)| < \dfrac{|f(x_{best}) - \hat{f}_k(x_k)|}{2} \\ m_b.w_k & sinon \end{cases} \tag{1.37}$$

où x_{best} est le meilleur individu déjà trouvé. Expérimentalement, il a utilisé $m_g = 0.75$ et $m_b = 1.25$ qui donnent de bons résultats.

1.6.2 SAO : Sequential Approximate Optimization

Nous considérons dans cette section des algorithmes hybrides séquentiels qu'on appelle SAO. Ils ne rentrent pas dans le cadre des algorithmes évolutionnaires (voir la section 1.4) .

Ces algorithmes suivent le schéma suivant :

On commence par générer des points dans l'espace de recherche avec une méthode de génération de points (voir 1.6.1), et à chaque itération

1. on définit une fonction approchée avec les points déjà évaluées ;
2. on adapte le domaine de recherche utilisant les optima de la fonction approchée et éventuellement on évalue ces optima ;
3. si on trouve un point acceptable, on le stocke à part ou bien on le considère comme l'itéré suivant ;
4. on génère à nouveau des points dans le nouveau domaine de recherche avec le même générateur de points et on boucle.

Remarquons que pour des fonctions objectif chères, l'évaluation des différents points peut se faire en parallèle.

Par exemple l'algorithme "Pattern Search Methods" [65] itère des points suivant des directions orthogonales avec des pas adaptatifs. La mise à jour de l'itéré courant ne s'effectue que si l'on trouve un nouveau point meilleur ! Christopher M. Siefert et Virginia Torczon [64] ont démontré la convergence de cet algorithme dans le cas d'une fonction objectif convexe. De plus, ils ont constaté expérimentalement que s'ils appliquent une méthode d'approximation utilisant les points tirés précédemment et en évaluant les optima de la fonction approchée, cela accélère la convergence.

La convergence globale de ces algorithmes est «démontrée» expérimentalement.

Cette section n'a pas de vocation de citer tous les algorithmes déterministes ou stochastique itérant sur un point, mais de mettre en relief les choix des auteurs pour s'en inspirer dans les algorithmes évolutionnaires.

Optimisation différentiable

La première approche suppose qu'on utilise les dérivées de la fonction objectif, et on définit un modèle qui a la propriété d'être local autour de l'itéré courant. Dans la littérature, il est courant de recourir à la méthode de la région de confiance pour définir le nouvel itéré. La méthode BFGS est une méthode de mise à jour de la matrice hessienne qui permet de définir un modèle au moins d'ordre un. Le modèle est mis à jour à chaque itération (voir section 1.6.1 et 1.3.3). La méthode SQP (Sequential Quadratic Programming) en est un exemple type. Mais en grande dimension, une interpolation quadratique devient irréaliste et on utilisera plus naturellement des méthodes comme RBF, Krigeage et SVM qui ont été mis au point pour se libérer des limitations pour les grandes dimensions dont souffre la régression quadratique ! Pour une comparaison systématique entre une régression quadratique et une méthode de Krigeage, le lecteur intéressé pourra consulter Simpson et al. [44].

Pérez et Renaud (2000-2002) [69, 70] ont hybridé la SAO avec la méthode de région de confiance appliquée aux problèmes avec contraintes(voir la section 1.6.1). La méthode d'approximation utilisé est la régression quadratique d'ordre 1 (le gradient du Lagrangien est connu) autour de l'itéré courant (voir la section 1.5.3). En effet les contraintes sont aussi différentiables. Le modèle est alors définis a travers les $\frac{n(n+1)}{2}$ points projetés dans l'espace faisable linéarisé.

Dans [71, 72] les mêmes auteurs hybrident la SAO avec la méthode de la région de confiance(voir la section 1.6.1) et la méthode BFGS (voir la section 1.3). La matrice BFGS, qui est mise à jour avec le gradient, approche les directions propres de la matrice hessienne. Pour approcher les valeurs propres de la matrice, ils utilisent une régression quadratique, de matrice hessienne diagonale, dans l'espace définis par les directions propres. Ainsi le modèle local est mis à jour tous les n points au lieu de $\frac{n(n+1)}{2}$. Ils étudient cette approche plus en détails dans l'article [73].

Hyoung-Seog Chung et Alonso (2000-2003) [82, 80, 81] comparent entre une approximation quadratique d'ordre 1 au point itéré courant soit $\frac{n(n+1)}{2}$ points, une approximation quadratique utilisant le gradient en tout point et la méthode appelée "Cokriging", qui est une généralisation de la méthode de Krigeage utilisant les gradients. La complexité de régression "Cokriging" dépend alors quadratiquement en fonction de la dimension. La dernière méthode s'est avérée intéressante car elle permet de détecter les minima locaux. Les résultats trouvés sont meilleurs qu'une méthode de Krigeage d'ordre zéros. Le gradient est calculé avec la méthode du problème adjoint. Ils ont appliqué cette méthode pour optimiser l'aérodynamique d'un avion supersonique d'affaire (5 et 14 variables).

Optimisation non différentiable

La deuxième approche consiste à utiliser un modèle d'ordre 0. Dans ce cadre, l'hybridation avec la méthode de la région de confiance n'est pas courante. Il est plus adapté d'utiliser la fonction de Mérite (voir section 1.6.1).

Citons quelques travaux :

Virginia Torczon et al (1995-2001) se sont concentrés avant tout à démontrer la convergence de l'algorithme Stochastique "Pattern Search". Dans les articles suivants [55, 58, 56, 57, 54, 65, 64] ils ont proposé une hybridation avec différentes méthodes d'approximation, région de confiance et fonction de mérite. Ils ont proposés une fonction de mérite utilisée dans le

cas de modèle quadratique. Ils ont aussi proposé des fonctions quadratiques d'ordre 1 dans le cas où nous disposons du gradient. Anthony A. Guinta et al [66] détaillent l'implémentation de la méthode de région de confiance dans DAKOTA, leur algorithme SAO. Dans le cas d'une approximation d'ordre zéro, ils utilisent la méthode de la région de confiance en calibrant avec les nouveaux points générés en adaptant la région de confiance(voir la section 1.6.1 et 1.6.1). La convergence n'est alors pas garantie en général, mais expérimentalement les résultats sont plutôt bons.

Hyoung-Seog Chung et Alonso(2000) ont étudié l'hybridation de SAO avec la fonction de mérite dans [83]. Ils ont modifié la fonction Mérite définie par Virginia Torczon(voir 1.6.1). Ils ont alors comparé entre une approximation quadratique, soit $\frac{(n+2)(n+1)}{2}$ points nécessaires pour définir le modèle, et la méthode de Krigeage sur un problème d'aérodynamique à 14 variables. Ils ont utilisé un noyau gaussien non isotrope avec une matrice de corrélation diagonale. Les résultats dépendent beaucoup des valeurs de corrélation. Le critère MLE 1.29 n'a pas d'optimum si nous ne disposons pas suffisamment de point. Ils proposent alors d'utiliser un critère de validation pour choisir une valeur optimale du vecteur corrélation, où d'augmenter le nombre de point. Ils ont aussi remarqué que le critère avec la méthode de "Cokrigeage" en dimension 1, que le critère MLE comporte des optima même avec peu de points. Ces remarques ont été constatées expérimentalement en dimension 1. Pour leur application, ils ont optimisé manuellement les paramètres de corrélation en dimension 14 en utilisant le critère de validation !

G. Gary Wang et T. W. Simpson(2002) : Dans un algorithme SAO classique et dans le cadre d'une hybridation avec la méthode de région de confiance, le tirage des points est dans un hypercube carrée de côté lié à la région de confiance trouvée précédemment. Ce choix rend l'algorithme local. Dans [53], les auteurs proposent un algorithme plus global. Ils recourent à la méthode de Krigeage pour définir un modèle global. A chaque itération ils raffinent le modèle en gérant des points nouveaux. En définissant un seuil dynamique sur la fonction $\hat{f}$, et en tirant des points aléatoirement dans tout l'espace, ils décident d'évaluer avec la fonction objectif uniquement les points dépassant le seuil. Ainsi cela définit une méthode de réduction l'espace de recherche. Le domaine de recherche est défini à travers les courbes de niveaux. Dans le cadre d'une fonction non convexe, le domaine est alors l'union de plusieurs domaines convexes.

1.6.3 EOA : Evolutionary Optimization Approximation

Nous considérons dans cette section l'hybridation des algorithmes Évolutionnaires avec des méthodes d'approximation(voir section 1.4).

Ces algorithmes sont réputés pour être globaux, au prix parfois d'une convergence locale très lente. L'hybridation avec une méthode d'approximation doit permettre en premier lieu d'accélérer la convergence locale. Mais les convergences prématurées sont déjà courantes dans un algorithme EA, et l'hybridation pourrait accentuer ce phénomène ! Dans la section suivante 1.6.3 nous développons quelques méthodes utilisées dans la littérature pour contrôler aussi l'équilibre entre exploration et exploitation.

Deux grandes approches ont été proposées dans la littérature : La première consiste à remplacer la fonction objectif (ou fitness, ou performance) par un modèle approché au sein même de l'algorithme évolutionnaire non modifié. C'est ce qu'on appelle l'«approche dynamique». La deuxième approche consiste à agir à travers les opérateurs. C'est l'approche appelée des «opérateurs informés» (*Informed Operator*).

Approche dynamique

L'idée est donc de remplacer localement la fonction objectif par le modèle appris durant un certain nombre de générations, puis de réapprendre un nouveau modèle. Dans une génération, on peut avoir une partie des individus évalués avec la fonction objectif et l'autre partie avec des modèles de la fonction objectif. On parle d'optimisation dynamique car le modèle s'adapte, se calibre au cours de l'évolution. On peut voir ça comme l'optimisation d'une fitness qui varie au cours du temps, et utiliser les techniques qui ont été mises au point dans la communauté évolutionnaire pour ce cadre-là [25].

Les techniques spécifiques à l'optimisation dynamique consistent à maintenir une diversité plus importante que dans le cadre statique, afin de permettre à l'algorithme une nouvelle phase d'exploration après chaque changement de fitness.

Keane , P. B. Nair et R. P. Shimpi (1998) [30] partent d'un algorithme génétique standard. La population entière est évaluée en utilisant un modèle interpolé, qui est mis à jour tous les ng générations. Le paramètre ng est dynamique, et son adaptation suit l'algorithme de la région de confiance (section 1.6.1). Les individus servant à améliorer le modèle et qui sont évalués par la vraie fonction objectif, sont choisis en cours de route : c'est par exemple à chaque génération le barycentre de la population, les points de grande concentration ou enfin les individus qui atteignent les meilleures per-

formances selon le modèle. La dernière stratégie donne les meilleurs résultats.

La deuxième idée de ces auteurs, pour éviter la convergence prématurée, est d'introduire un opérateur de remplacement spécifique, lui aussi adaptatif, appelé *Adaptive Tournament*, qui modifie de manière dynamique le paramètre s de pression de remplacement :

Ils choisissent aléatoirement s individus avec $s = 1, 2, 3$ ou 4, les classent et retiennent le meilleur. Plus s augmente, plus la pression augmente et donc les individus moyens ou mauvais ont moins de chance d'être sélectionnés. L'adaptation de s est faite en fonction de l'écartement de la population par rapport aux points d'apprentissage courants qui ont servi à construire le modèle.

Notant X_{ap} l'ensemble de points d'apprentissage, ils définissent la moyenne de la distance des points se trouvant dans une génération courante aux points d'apprentissage :

$$Avg(\Delta X) = \frac{1}{m} \sum_{i=1}^{m} \|X^i - X_{ap}\|$$

où m est la taille de la population.

La pression de sélection s est choisie de la manière suivante :

- Si $Avg(\Delta X) > \frac{p^{1/2}}{2}$ alors $s = 4$
- Si $\frac{p^{1/2}}{5} < Avg(\Delta X) < \frac{p^{1/2}}{2}$ alors $s = 3$
- Si $Avg(\Delta X) < \frac{p^{1/2}}{5}$ alors $s = 2$

L'idée est de concentrer la recherche autour des meilleurs individus seulement si l'on s'éloigne du dernier ensemble d'exemples ($s = 4$), et au contraire de permettre plus d'exploration si on constate que l'on reste très près des exemples courants.

Dans ce schéma, le point négatif est *a priori* que l'exploration est très locale, ce qui tend à pénaliser cette approche pour des problèmes fortement multimodaux. Ils ont appliqué cet algorithme à un problème d'optimisation de forme (The optimal design problem of 10 bar truss structure).

Y. Jin, M. Olhofer et B. Sendhoff (2001) [35] ont modifié cette approche et l'ont rebaptisée l'approche "évolution contrôlée" [35]. Ils l'utilisent avec un algorithme de type CMA-ES. Ils distinguent deux stratégies, le "contrôle par génération" et le "contrôle par individu". La première est l'approche de Keane détaillée précédemment. Il s'agit de contrôler (évaluer avec la fonction objectif) la population tous les ng générations. La deuxième stratégie contrôle certains individus à toutes les générations. Le problème est alors dans le choix des points à contrôler et de leur nombre. *neval* individus de la population de taille λ sont choisis pour être évalués avec la vraie fonc-

tion objectif. Le paramètre *neval* est adapté dynamiquement, inversement proportionnel à l'erreur normalisée qu'effectue le modèle sur la population contrôlée. En ce qui concerne la méthode d'approximation, ils ont utilisé des réseaux de neurones (Multi-layer Perceptrons (MLP) et Radial Basis Function Networks (RBFN). Ils résolvent un problème de moindres carrés pondérés par la matrice de covariance du meilleur. Cela revient à ne prendre compte dans l'apprentissage que des points se trouvant dans l'ellipsoïde défini par CMA.

Th. Bäck et al. (2002) [40] présentent l'algorithme MAES : cette approche ressemble à l'approche dite de "contrôle par individu" décrite ci-dessus (certains points sont évalués avec la vraie fonction objectif toutes les générations). Par contre, contrairement à Jin et al., le choix des points évalués avec la vraie fonction est effectué en utilisant la fonction de Mérite (section 1.6.1) avec poids constant. Cela permet de contrôler la distance par rapport aux points d'apprentissage. Une autre idée de ce travail est d'éviter la convergence prématurée en utilisant une stratégie d'évolution avec des individus de durée de vie limitée dans le temps, paramétrés par le nombre de générations. Ainsi un individu ne peut vivre qu'un nombre donné de générations.

D. Buche, N.N. Schraudolph et P. Koumoutsakos (2004) [42] reprennent les idées de Jin et al. Ils utilisent comme méthode d'approximation une méthode à noyaux, le *Gaussian Process Model*. A la différence de Y. Jin et al., le domaine d'approximation est toujours centré autour du meilleur individu mais au lieu de tirer les points d'apprentissage utilisant la densité de probabilité de l'algorithme CMA sous-jacent, les points sont tirés dans le plus petit hypercube incluant tous les individus de la population courante qu'ils aient été évalués avec le modèle ou avec la fonction objectif. Ils définissent également une autre approche qu'ils baptisent *Surrogate Approach*. Cette dernière fera partie des approches exposées dans le paragraphe suivant.

Opérateurs informés

L'approche *Informed Operators* consiste à modifier les opérateurs de variation (croisement, mutation) en utilisant des informations données par le modèle de la fonction objectif.

L'algorithme n'est pas modifié, et utilise la fonction objectif originale. Seuls les opérateurs prennent en compte la fonction approchée.

X. Yao et al. (1999-2000) [91] construisent un opérateur de mutation

et un opérateur de croisement à partir d'approximations quadratiques de la fonction objectif (voir 1.5.3) :

- en partant de l'idée que le modèle prédit la zone où l'optimum pourrait se trouver, la mutation part du minimum du modèle quadratique pour lancer une recherche locale sur la fonction objectif.
- l'opérateur de croisement part de trois individus alignés, et en définit explicitement un quatrième qui est l'extremum de la fonction polynomiale passant par ces trois points.

L'inconvénient de cette méthode est l'utilisation d'une approximation quadratique, qui peut être assez éloignée de la fonction originale.

K. Rasheed et al.(2000-2003) [88] sont les inventeurs de l'appellation *Informed Operator*, concept qu'ils implémentent dans leurs code GADO. À l'initialisation : pour tirer un individu, on commence par tirer plusieurs aléatoirement et on choisit le meilleur selon le modèle à chaque fois pour être évalué par la fonction objectif. Pour les opérateurs de variation, on en applique plusieurs et on garde celui qui donne le meilleur résultat sur le modèle. Par exemple, l'opérateur de mutation est composé de plusieurs opérateurs de mutation, chacun avec ses paramètres. Quand un individu est à muter, les différents opérateurs lui sont appliqués (choix de l'opérateur de mutation et de ses paramètres aléatoire), et les enfants résultants sont alors testés sur le modèle dont l'évaluation est beaucoup plus rapide. Le meilleur opérateur (avec ses paramètres) selon le modèle est choisi pour la suite de l'évolution. L'opérateur de croisement agit d'une manière similaire à partir de deux parents pour choisir parmi un ensemble de croisements possibles (avec leurs paramètres).

Au cours de ces trois dernières années, ils ont testé différentes méthodes d'approximation : QNN (*Quickprop Neural Networks*), RBFNN (*Radial Basis Function Neural Networks*) et QLS (*Quadratic Least Squares*) sur différents problèmes. Ils ont cependant toujours négligé le temps de l'algorithme de régression des différentes méthodes par rapport aux problèmes réels testés (problèmes de conception d'avions de transport). D'un point de vue algorithmique, les travaux de cette équipe semblent privilégier la régression quadratique, qui a toujours donnés les meilleurs résultats sur les problèmes les plus difficiles parmi les applications qu'ils ont testées.

1.6.4 Conclusion

Nous remarquons que les algorithmes évolutionnaires s'inspirent fortement des algorithmes déterministes et stochastiques antérieurs comme par exemple, la méthode de région de confiance et l'utilisation de la fonction de

mérite. Nous pouvons également citer l'algorithme CMA qui lui s'inspire de la méthode de BFGS.

Les paramètres internes sont les paramètres des opérateurs de l'algorithme AG. Dans les deux approches de EOA («Approche dynamique» et «opérateurs informés») la problématique commune est la suivante : Comment contrôler les paramètres internes des opérateurs en vue d'accélérer la convergence en utilisant des informations données par le modèle approché ?

Si l'on utilise une approche dynamique de l'hybridation, l'adaptation des paramètres, comme par exemple la matrice de covariance dans CMA, pourrait se faire soit entièrement sur le modèle, dans le cadre du contrôle par génération, soit sur la fonction dynamique, si on choisit le contrôle par évaluation. Le challenge est de pouvoir doser entre les deux approches.

Dans le cas de l'approche *Informed Operators*, les opérateurs sont adaptés entièrement sur le modèle. On pourrait penser que cette approche est complètement tributaire du modèle, mais il ne faut pas oublier qu'une fois les paramètres des opérateurs de variation adaptés, ils vont être appliqués sur la fonction objectif. L'adaptation des paramètres est ainsi validée sur les résultats trouvés après évaluation avec la vraie fonction objectif.

Nous ne pouvons pas dire d'une manière précise quelle approche est la meilleure mais nous devons au cas par cas mettre en relief les différents mécanismes qui jouent sur l'exploration et l'exploitation. De plus, les cas-tests présentés dans les différents articles sont tous différents, ce qui rend la comparaison très difficile.

Dans le chapitre 5, notre approche qui est de type approche «Opérateurs Informés», sera comparée à l'approche MAES, l'une des plus récente dans le domaine des algorithmes hybrides, et qui semble posséder une légère avance sur les autres, si l'on en croit les travaux publiés par Bäck et al. [40].

1.7 Les fonctions de tests

Certaines fonctions de tests se trouvent dans Dixon et Szego (1978).

Pour l'étude nous regroupons les fonctions en plusieurs classes qui sont représentatives des fonctions et des difficultés rencontrées dans les problèmes réels(figure 1.7). Le domaine de recherche est $[-100, 100]^d$.

Classe A : La classe A correspond aux fonctions convexes qui ont un seul minimum et qui sont bien conditionnées. Nous étudierons au sein de

cette classe la fonction Sphère d'équation (1.38).

$$Sphre(\mathbf{X}) = \sum_{i=1}^{n} x_i^2 \tag{1.38}$$

Classe B : La classe B correspond aux fonctions qui ont un comportement global appartenant à la classe A mais comportant beaucoup de minima locaux. Le lissage de ces fonctions rend la convergence plus facile et plus rapide. Nous étudions de cette classe la fonction Griewank d'équation (1.39).

$$Griewank(\mathbf{X}) = \frac{\sum_{i=1}^{n} x_i^2}{4000} + (1 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}})) \tag{1.39}$$

Classe C : La classe C est composée de fonctions qui ont un nombre fini de minima. Un un algorithme déterministe simple avec différentes initialisations suffit pour retrouver tous les optima.

Classe D : La classe D est constituée de fonctions mal (voir très mal) conditionnées. Nous étudierons la fonction Ellipse d'équation (1.40).

$$Ellipse(\mathbf{X}) = \sum_{i=1}^{n} i^{2.5} x_i^2 \tag{1.40}$$

Classe E : La classe E sont les fonctions dont le comportement global appartient à la classe D. Nous étudions la fonction GriewankE d'équation (1.41). La fonction GriewankE étant une fonction difficile, nous allons l'étudier sur différents intervalles. Nous parlerons de GriewankE0.4-0.5 si le domaine de recherche est $[100*(2*0.4-1), 100*(2*0.5-1)]^d$ et de GriewankE0.4-0.6 si le domaine est $[100*(2*0.4-1), 100*(2*0.6-1)]^d$.

$$GriewankE(\mathbf{X}) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j}{4000} + (1 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}})) \tag{1.41}$$

Classe F : La classe F représente les problèmes avec pic dont la localisation est totalement imprévisible. Le comportement global ne donne pas d'information sur sa localité. Nous n'étudierons pas cette classe.

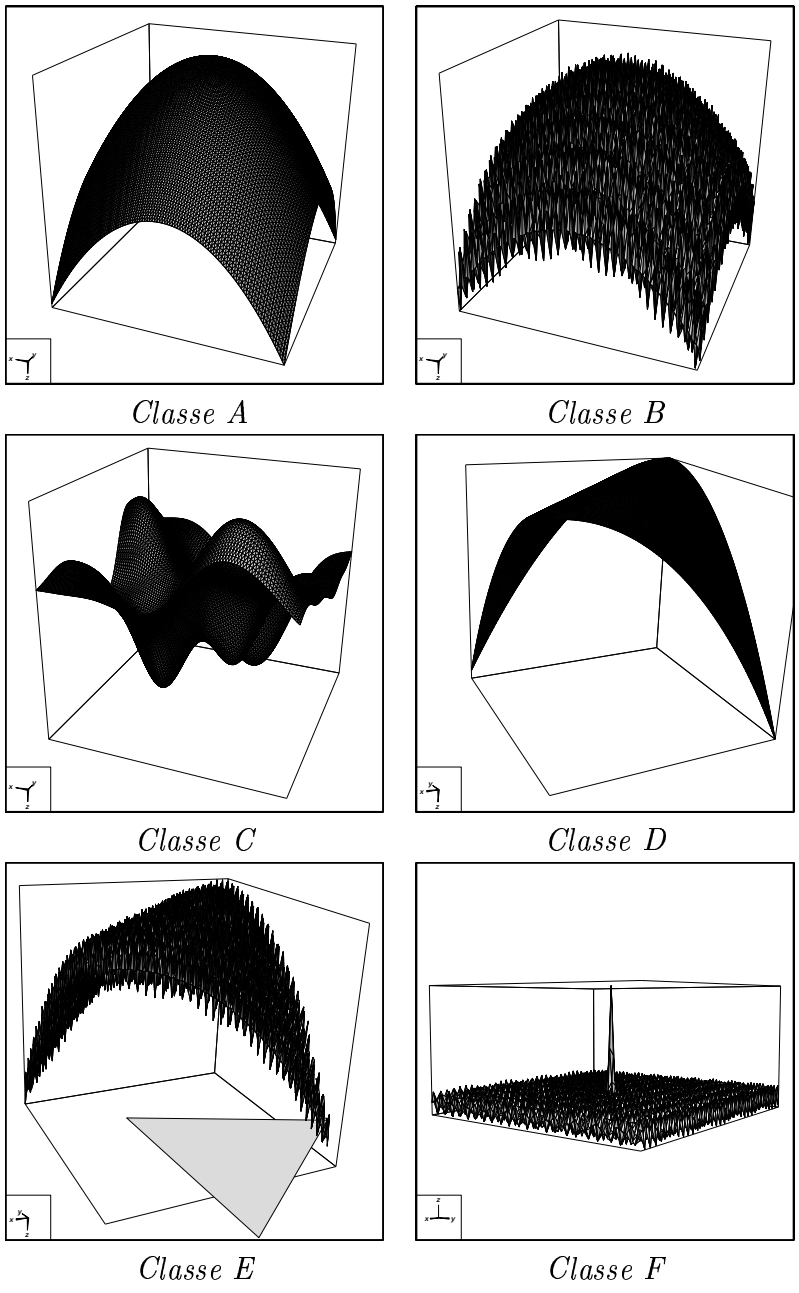


FIG. 1.2 – **A** : fonction uni-modale comme la sphère. **B** : Globalement c'est une fonction uni-modale mais admettant beaucoup de minima locaux. **C** : Fonction avec nombre fini de minimum. **D** : Fonction mal conditionnée dont la matrice hessienne n'est pas définie négative. **E** : Globalement, c'est une fonction mal conditionnée mais avec les petites oscillations, elle admet un seul maximum global. **G** : L'optimum est localisé et le comportement global ne donne pas d'informations sur sa localité

Chapitre 2

Approximation & Optimisation

2.1 Introduction

Dans ce chapitre, nous nous proposons d'étudier la variabilité des deux méthodes d'apprentissage présentées au chapitre précédent, SVM et Krigeage, par rapport aux paramètres définissant leurs conditions d'application (voir section 1.5.5), en particulier par rapport aux paramètres de régularité. Le but est la mise au point d'une procédure automatique de réglage de ces paramètres utilisant elle-même un algorithme d'optimisation. Les questions auxquelles nous allons essayer de répondre dans ce chapitre sont

- Quelles sont les paramètres des méthodes d'apprentissage qu'il est important de régler dynamiquement pour chaque nouvel apprentissage, et quels sont ceux au contraire qu'il semble que nous pouvons fixer une fois pour toute ?
- Quel critère utiliser pour trouver dynamiquement les meilleurs paramètres, sachant que nous ne cherchons pas forcément à trouver la meilleure approximation possible de la fonctions objectif, mais à trouver une approximation qui donnera, par minimisation, une bonne idée de la localisation du minimum de la fonction initiale.

Notre étude expérimentale se portera sur la fonction sphère, en tant que représentante caractéristique des fonctions régulières.

Bien que leurs origines théoriques soient similaires, les méthodes SVM et Krigeage ont des caractéristiques différentes en ce qui concerne leurs paramétrages : SVM permet un meilleur contrôle direct de l'erreur d'approximation, avec pour conséquence la possibilité de proposer des modèles intermédiaires. Par contre la méthode de Krigeage qui définit un modèle passant le plus près possible par les points d'apprentissage, semble a priori plus contraignante. Par contre, les deux méthodes ont en commun les paramètres des noyaux qui

sont liés à la régularité recherchée.

En règle générale nous ne possédons pas d'hypothèse sur la régularité de la fonction qu'on optimise. Pour choisir des "bons" paramètres nous devons procéder par essais/erreurs.

Le point de départ de ce chapitre est donc la donnée d'un certain nombre de points pour lesquels nous connaissons la valeur exacte de la fonction cible. À partir de ces points, nous allons construire un modèle de cette fonction en utilisant une méthode d'apprentissage (SVM ou Krigeage). Nous chercherons ensuite le minimum de ce modèle en utilisant une méthode déterministe (LBFGS – voir section 1.3). Nous étudierons expérimentalement, sur la fonction sphère, l'influence des divers paramètres de la méthode d'apprentissage utilisées sur la qualité du minimum du modèle par rapport au minimum réel de la fonction initiale.

Le but de ce chapitre est double : tout d'abord, étudier le comportement des méthodes d'apprentissage par rapport à leurs paramètres, en terme de localisation de l'optimum de la fonction initiale ; d'autre part, proposer un algorithme de réglage des paramètres des méthodes d'apprentissage qui pourra ensuite être utilisé lors de son hybridation avec un algorithme évolutionnaire pour maximiser les chances d'amélioration de l'algorithme hybride.

Plus précisément, nous allons tenter dans ce chapitre de répondre aux questions suivantes :

1. Quelle est l'influence du nombre de points sur l'apprentissage ? En particulier, toujours compte-tenu de notre objectif, un nombre "petit" de points, qui serait notoirement insuffisant pour une bonne approximation, pourrait-il suffire ici ? Plus précisément, comment se comporte de nombre de points lorsque la dimension augmente ?
2. Nous supposerons initialement une distribution uniforme des exemples d'apprentissage. Mais quelle influence la distribution des points a-t-elle sur le modèle final ?
3. Comment se comporte chaque méthode d'apprentissage par rapport au but fixé en fonction des valeurs de ses différents paramètres ? En particulier, certains paramètres peuvent-ils être fixés une fois pour toute, allégeant d'autant la procédure d'apprentissage lors de l'hybridation.
4. Quel doit être le critère d'optimalité permettant de comparer les résultats de méthodes d'apprentissage ? En particulier, notre objectif premier n'étant pas d'approcher la fonction objectif mais de localiser son minimum, le critère usuel de minimisation de l'erreur empirique n'est pas forcément le bon ici.

En particulier, des critères de généralisations sont utilisés dans la littérature de l'apprentissage dans le cas de grands nombres d'exemples. Sont-ils également applicables dans notre cadre ?

5. Quel algorithme (d'optimisation ?) utiliser pour trouver les meilleurs paramètres, sachant que ceci sera fait ensuite de très nombreuses fois lorsque l'apprentissage sera intégré au processus complet d'optimisation ?

   Ou alors, existe-t-il des règles heuristiques robustes pour le choix des paramètres cruciaux que sont le choix du noyau (gaussien ou sigmoïde) et de ses paramètres ?

Le chapitre est organisé comme suit :

La première section propose un certains nombre de critères de comparaison pour les paramétrages des méthodes d'apprentissage.

Les deux sections suivantes sont consacrées respectivement à la méthode SVM et à la méthode de Krigeage. Chaque méthode est étudiée expérimentalement, entre autre pour déterminer ses paramètres les plus influents, puis l'utilité de chacun des critères proposés dans la première section est discutée.

Le chapitre se conclut en dégageant les procédures les plus efficaces qui seront ensuite utilisées lors de l'hybridation avec les algorithmes évolutionnaires.

2.2 Critères de qualité des méthodes d'apprentissage

Le premier point important agissant directement sur la qualité de la méthode d'approximation est le choix des points et leurs nombre qui seront utilisés pour apprendre le modèle. D'autre part le problème de l'apprentissage – trouver une fonction passant par des points donnés – est un problème mal posé auquel il convient de rajouter des contraintes de régularité (voir section 1.5). Les paramètres mettant en jeu cette régularité sont spécifiques à chaque méthode d'apprentissage, et constituent l'essentiel des leviers dont nous disposons pour ajuster l'apprentissage à notre but.

Notre objectif est d'adapter le modèle de façon à ce que son optimum s'approche de l'optimum de la fonction objectif. Mais bien entendu, en situation réelle, cet optimum est inconnu. Nous allons donc tenter de trouver un critère utilisable durant l'apprentissage lui-même permettant d'atteindre au mieux ce but.

2.2.1 Critères de choix des paramètres

Notations

Nous supposons donc ici que nous disposons d'une fonction f, de $\mathbb{R}^n$ dans $\mathbb{R}$ et de l exemples $(x_i, y_i = f(x_i))_{i=1\ldots l}$ de valeurs prises par f. Nous notons

- $F = min(\{y_i, i = 1 \ldots l\}$: la meilleure valeur de f sur l'ensemble d'apprentissage. Nous supposons de plus que ce minimum est atteint pour $i = 0$, *i.e.* $F = y_0$;
- $\hat{f}$: la fonction approchée définie par application de la méthode d'apprentissage aux exemples $(x_i, y_i = f(x_i))_{i=1\ldots l}$.
- $\hat{x}$ le minimum de $\hat{f}$ trouvé par la méthode L-BFGS-B partant de x_0
- $dF = F - f(\hat{x})$: le gain sur la fonction obtenu par apprentissage et optimisation du modèle appris. Son calcul suppose de recalculer $f(\hat{x})$.

Nous pouvons alors définir les critères suivants :

Critère Objectif

$$CO(p) = \frac{dF}{F}; \tag{2.1}$$

Ce critère représente directement la quantité que nous voulons maximiser. Malheureusement, il demande un calcul de la fonction exacte f. D'autre part, il est non-différentiable par rapport aux paramètres de l'apprentissage, puisque $\hat{x}$ est solution (numérique !) d'un problème de minimisation.

Critère Erreur

$$CE(p) = \sqrt{\sum_{i=1}^{l} (\hat{f}(x_i) - y_i)^2} \tag{2.2}$$

Il s'agit du critère classique de minimisation de l'erreur empirique : Optimiser ce critère, c'est trouver des paramètres tel que le modèle associé passe au mieux par les points d'apprentissage.

Critère de validation

Ce critère utilise la procédure classique en apprentissage pour tester la capacité de généralisation d'une méthode d'apprentissage (appelé *Leave-One-Out*) : on effectue en fait l régressions en retirant successivement un des exemples de la base d'apprentissage ; on obtient ainsi l modèles $\hat{f}_i$, chaque modèle $\hat{f}_i$ étant obtenu par apprentissage sur l'ensemble d'exemples $(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_l)$.

Le critère est alors défini comme la somme des erreurs de chaque modèle sur l'exemple qu'il n'a pas vu lors de l'apprentissage :

$$CV(p) = \frac{1}{l} \sum_{i=1}^{i=l} (\hat{f}_i(x_i) - y_i)^2 \tag{2.3}$$

Si ce critère ne nécessite pas le calcul de nouvelles valeurs de la fonction objectif, il demande par contre l régressions, si l est le nombre de points exemples.

Critère de généralisation

Le critère de généralisation a pour but de parvenir à fitter les futures observations grâce aux échantillons limités de données. La question centrale de l'apprentissage est donc de savoir comment utiliser les données pour avoir une bonne performance en généralisation. Rappelons que le but n'est pas la généralisation mais la recherche d'un modèle qui a un optimum meilleur que les points d'apprentissage. Dans le cas de la méthode SVM, adapter les conditions d'arrêts de l'algorithme de régression associé, à travers les paramètres ε_{reg}, ε_{fin}, nous éloigne de la généralisation : on ne passe plus par les points d'apprentissage ! Mais cela permet de proposer des solutions en temps négligeable et nous permet de trouver un meilleur individu. Pour étudier le critère de généralisation avec SVM, il faut prendre de très petites valeurs pour ε_{reg} et ε_{fin} et $C = \infty$, on passe ainsi par les points d'apprentissage et on se ramène ainsi au problème de Krigeage. C'est pourquoi le critère de généralisation sera étudié plus loin (section 2.4) uniquement dans le cas du Krigeage. Le critère de généralisation pour le Krigeage est énoncé dans l'état de l'art section 1.5.4 :

$$CG(p) = n \log \hat{\sigma}^2 + \log \det R(p) \tag{2.4}$$

R la matrice des noyaux $R_i j = K_p(x_j, x_i)$.

Discussion

Notre but étant la minimisation de la fonction objectif, il s'agit bien de trouver des paramètres qui minimisent le critère (CO). Mais ce dernier est coûteux, puisqu'il nécessite une évaluation de la fonction objectif. Nous allons donc dans la suite étudier les autres critères, *a priori* moins coûteux, pour tenter de les substituer à (CO). A noter toutefois que le coût du critère (CV) (une régression par point d'apprentissage) dépend, lui, du coût d'une régression, et donc de la méthode d'apprentissage utilisée – et des coûts respectifs d'une évaluation de la fonction objectif et d'une régression.

2.2.2 Procédure expérimentale

Le but premier des expériences est de comparer les différents critères. Le but ultime est de définir des règles, éventuellement heuristiques, pour le choix des paramètres d'apprentissage pour chacune des méthodes SVM et Krigeage.

La procédure expérimentale de validation et choix des différents critère va être la suivante : Pour chaque méthode d'apprentissage, et pour différents nombre de points d'apprentissage (voir plus loin), nous allons optimiser successivement les différents critères énumérés dans la section précédente en fonction des paramètres de la méthode d'apprentissage, et nous analysons le comportement du modèle résultant.

Dimension et nombre de points

Dans les expériences qui vont être décrites dans les section suivantes, les exemples vont être tirés uniformément dans $[0,1]^{dim}$.

Mais il ne faut pas séparer le choix du nombre de points de la dimension de l'espace dans lequel est défini f. En effet, plus on augmente la dimension, plus il faut de points pour échantillonner correctement le voisinage d'un point de l'espace. Cependant, pour des raisons de volume de calculs, nous avons envisagé les situations suivantes : Pour étudier l'influence de la dimension nous fixons le nombre de points à $\texttt{Npt} = 50$ et nous étudions les dimensions suivantes $dim = 2, 4, 8, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100$. Pour étudier l'influence du nombre de points sur les paramètres et sur la variation de f nous fixons la dimension à $dim = 100$ et nous étudions les différents paramètres en fonction du nombre de point $\texttt{Npt} = 10, 20, 30, 40, 50, 100, 200, 300, 400, 500$.

Optimisation des critères CO, CV, CE, et CG.

L'optimisation des différents critères se fera en utilisant un algorithme (3+6)-ES classique (voir section 1.4). Expérimentalement, nous avons fixé à 300 le nombre maximal d'évaluations des critères, ce qui équivaut à 50 générations maximum. Ce nombre, qui peut paraître insuffisant pour une recherche stochastique, a semblé suffisant dans le cas de la fonction sphère.

Certes, comme il a été discuté plus haut, le coût de cette optimisation et difficile à prendre en compte hors du contexte global de l'algorithme hybride. Mais après de nombreux essais, il semble qu'un prix de 300 évaluations du critère représente un bon compromis entre coût et efficacité.

Comparaison des différents critères

En premier lieu nous allons établir les profils des différents critères en fonction de la variation des principaux paramètres de la méthode d'apprentissage, pour avoir une idée des difficultés à les optimiser.

En deuxième lieu nous allons étudier les paramètres optimaux trouvés par l'algorithme (3+6)-ES associé aux différents critères.

Nous effectuons plusieurs tirages uniformes des exemples, et étudions les variations des moyennes de chaque critère. Pour une comparaison effective entre les critères, nous devons prendre en compte le temps de calcul que prend chaque critère par rapport au temps d'évaluation de la fonction objectif. Si l'unité de temps est le temps d'évaluation de la fonction objective alors combien coûterait l'adaptation en nombre d'évaluations. Pour chaque critère *a priori* nous ne pouvons pas savoir combien nous devons l'optimiser pour une variation optimale de f. Donc dans ce chapitre la seule comparaison sera sur la variation de f avec comme nombre maximal d'évaluation des critères fixés à 300.

Une comparaison plus précise en terme de coût sera faite dans les chapitres suivants, une fois que l'apprentissage d'apprentissage adaptatif du modèle aura été intégré dans un algorithme d'optimisation évolutionnaire.

2.3 SVM

L'objet de cette section est d'étudier l'adaptabilité de la méthode SVM, dans le cas des noyaux gaussiens et sigmoïdes, sur la fonction sphère, dans le contexte défini dans la section précédente. Nous allons étudier principalement le cas des noyaux gaussiens, les plus utilisés en pratique. Dans la dernière section, nous évoquerons les noyaux sigmoïdes, pour les éliminer de notre étude du fait des mauvaises qualités de robustesse en grande dimension qu'ils présentent.

2.3.1 Influence du paramètre de régularisation σ

Le paramètre le plus important dans la méthode des SVM a noyaux Gaussiens et sans conteste le paramètre σ_{reg}, qui impose la régularité de la solution (voir section 1.5.5). Nous allons donc étudier en premier lieu l'influence de ce paramètre sur les différents critères. Nous fixons pour cela les autres paramètres de la méthode aux valeurs robustes (voir plus loin) $C_{reg} = 50$, $\epsilon_{reg} = 1e-10$ et $\epsilon_{fin} = 1e-10$.

Pour un jeu de `Npt` points tirés uniformément dans $[0,1]^{100}$ nous traçons les différents critères. La figure 2.3.1 donne une idée des allures typiques que

nous avons obtenues.

Le critère (CO)

La figure 2.3.1-a montre le profil pour le critère (CO) : Il a un seul minimum local et semble très régulier par rapport au paramètre σ_{reg}. Mais, comme il a été signalé, il n'est pas possible de définir une différentielle du critère (CO). Par contre, la régularité suggère l'utilisation d'une méthode de Legendre pour déterminer l'optimum associé – dans le cas oú σ_{reg} est le seul paramètre en jeu.

A noter sur la figure illustre parfaitement ce qui se passe pour les valeurs extrêmes de σ_{reg} : Un petit σ_{reg} permet en général une meilleure erreur empirique, au détriment de l'erreur en généralisation : le modèle approché est une somme de pics très étroits, et le gain en minimum de la fonction est nul. Au contraire, un grand σ_{reg} donne un fonction très lisse : le modèle est proche de la droite de régression linéaire de exemples, et le "gain" par rapport au minimum est même négatif!

Le critère (CE)

Il a toujours présenté des profils irrégulier comme celui de la figure 2.3.1-b. Les optima globaux sont en général situé dans la même région que ceux du critère (CO), mais l'optimisation de ce critère est à priori plus difficile, du fait de la grande variabilité de ce critère.

Le critère (CV)

Le profil de ce critère est très proche de celui du critère (CO) – en particulier les minima globaux des 2 critères sont voisins. Comme le critère (CO), ce critère semblerait être régulier par rapport à σ_{reg}. La régularité suggère de même l'utilisation de Legendre comme interpolation en dimension 1, pour optimiser la minimisation.

Règle heuristique

Lors de l'utilisation de noyaux Gaussiens, le modèle approché est une combinaison de $l \leq$ `Npt` gaussiennes, centrées aux points d'apprentissage, et d'écart type σ_{reg}. Il faut à la fois que les différentes gaussiennes interagissent, pour éviter l'effet "pics" déjà mentionné – mais il faut aussi que σ_{reg} ne soit pas trop grand pour laisser suffisamment de degrés de liberté au modèle. Si

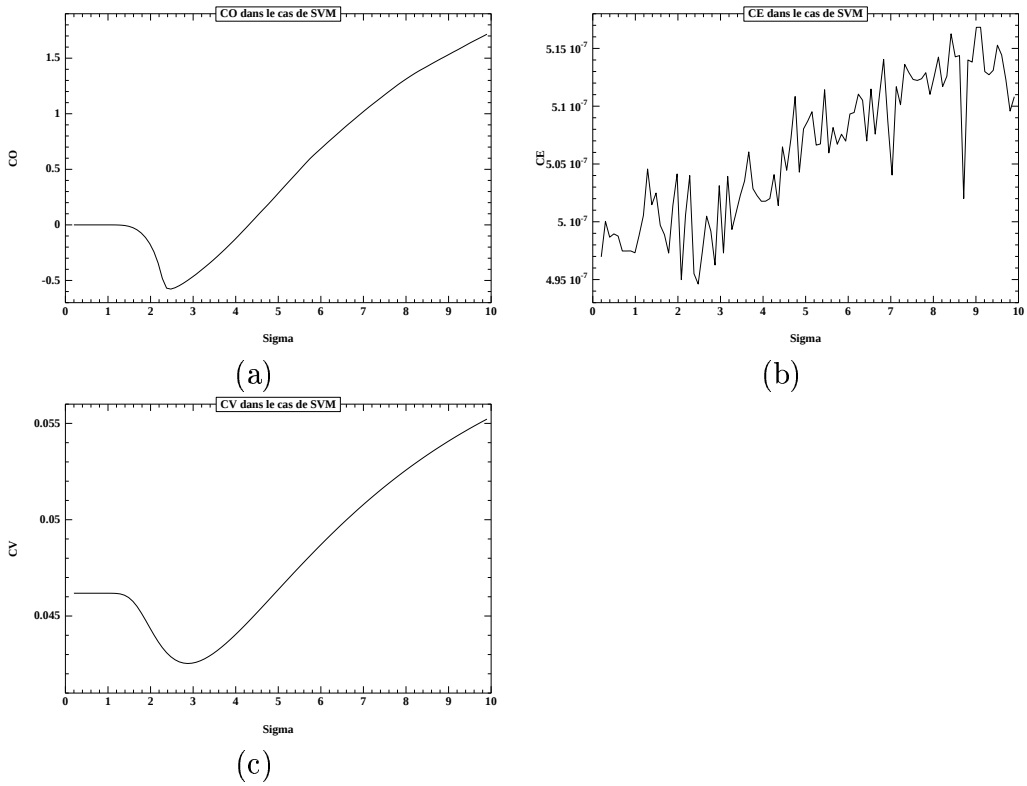


FIG. 2.1 – Le profil des différents critères en fonction du paramètre σ_{reg}. (a) le critère (CO), (b) le critère (CE) et (c) le critère (CV). Les données d'apprentissage sont 50 points tirés uniformément dans le domaine $[0,1]^{100}$ et évalués sur la fonction Sphère.

on suppose que nos points sont disposés uniformément, une valeur de σ pourrait être de l'ordre du rayon maximal des boules qui permette d'en placer exactement Npt dans l'hypercube $[0,1]^{Dim}$ soit :

$$\sigma(\texttt{Npt}, Dim) = \frac{\texttt{Npt}}{(\texttt{Npt} * S)^{1/Dim}} \tag{2.5}$$

oú $S_{\texttt{Npt}}$ est l'aire de la sphère de rayon 1 ($\frac{r^{2p}2^{p+1}\pi^p}{1.3.5...(2p-1)}$ si $Dim = 2p$ et $\frac{2r^{2p+1}\pi^p}{p!}$ si $Dim = 2p+1$).

Il faut préciser que cette règle ne donne qu'un ordre de grandeur de la valeur de σ_{reg}. Néanmoins, elle va nous permettre de réduire l'intervalle de recherche de l'inconnue σ_{reg}, par exemple à $[\frac{\sigma(\texttt{Npt},Dim)}{3}, 3 * \sigma(\texttt{Npt}, Dim)]$.

2.3.2 Dimension et nombre de points

Quelle que soit la dimension, on trouve des valeurs moyennes des σ_{reg} optimaux, pour les différents critères, raisonnablement proches de ceux de la courbe heuristique (voir Figure 2.2). Une des causes probable de l'écart entre la règle heuristique et l'optimum constaté vient du faible nombre de points(voir Figure 2.3), entraînant un écart qui reste toute fois raisonnable par rapport à la distribution uniforme idéale. On voit que la règle heuristique donne une bonne idée de l'ordre de grandeur de la valeur de σ_{reg} optimal de (CO). La dispersion dans les valeurs de σ_{reg} vient du fait que la distribution des points n'est pas la même d'un cas à l'autre. L'heuristique faciliterait la recherche du sigma optimal.

F et dF

La figure (2.4) illustre un phénomène connu : F décroît avec Npt, suivant une loi connue, la statistique d'ordre de la loi de la norme de (x, y) quand x et y sont tirés avec une loi uniforme. Inversement, l'apprentissage est de plus en plus efficace, dF rejoignant presque F (ce qui correspondrait à la découverte du minimum après apprentissage). L'apprentissage dans le cas de la sphère, qui représente les fonctions régulières, améliore de beaucoup la valeur minimale.

Le même phénomène est illustré plus en détail sur la figure 2.3.2 : pour les petites valeurs de Npt (en haut), dans de nombreux cas le gain est nul. Par contre, quand le Npt devient grand, le nuage de points se positionne autour de la diagonale : on arrive presque à l'optimum après apprentissage.

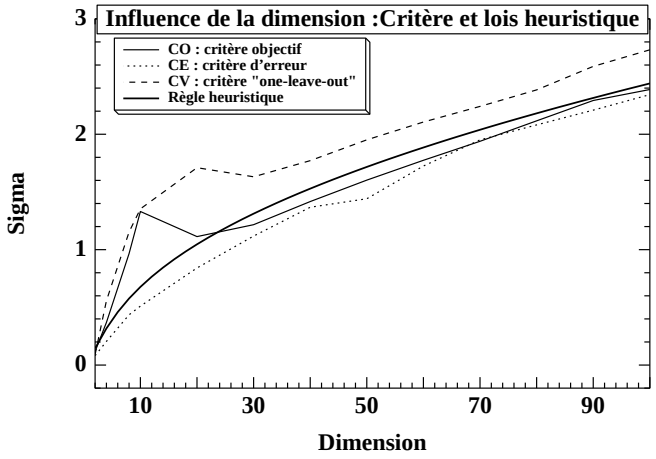


FIG. 2.2 – En abscisse, la dimension de l'espace. En ordonnée, la moyenne des valeurs des σ_{reg} optimaux pour les différents critères, moyennées sur 100 tirages uniformes de 50 points dans le domaine $[0,1]^{Dim}$ et évalués avec la fonction sphère.

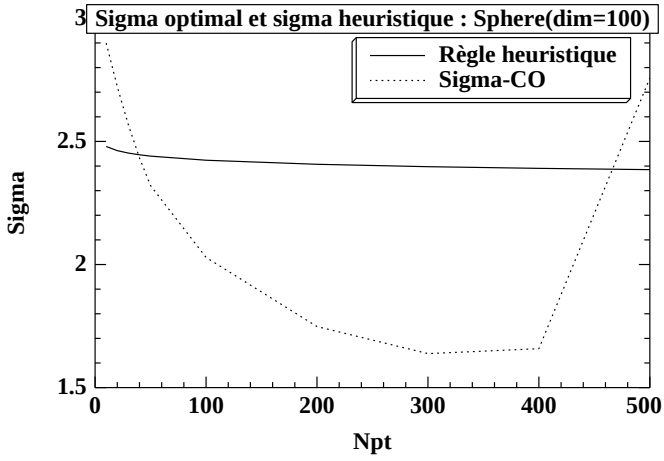


FIG. 2.3 – En abscisse le nombre de points d'apprentissage `Npt` $= 10, 20, 30, 40, 50, 100, 200, 300, 400, 500$. En ordonnée la moyenne, sur 1000 tirages uniformes de `Npt` points, de la valeur optimale du paramètre σ_{CO} trouvée en optimisant le critère (CO) avec (3+6)-ES en 50 générations. La deuxième courbe correspond à la valeur donnée par la règle heuristique 2.5.

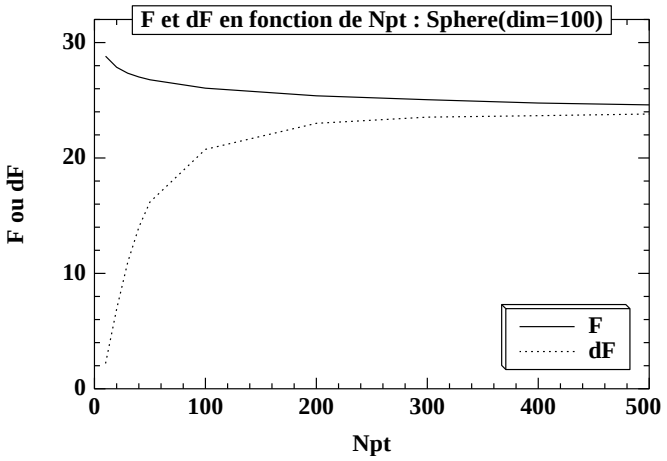


FIG. 2.4 – Influence du nombre de points `Npt` sur F et dF.

σ et dF

Lorsqu'on augmente le nombre de points, il faut s'attendre à ce que la valeur de σ_{reg} optimal diminue puisqu'on dispose de plus de centres de gaussiennes. Cependant, on constate sur la figure 2.3.2 que pour `Npt` $= 500$ il apparaît deux valeurs d'ordre de grandeur différents de σ. L'espace de recherche étant $[0,1]^{100}$ les cotés sont de longueur égal à 1 et la diagonale est de longueur $\sqrt{100} = 10$. Si les points se trouvent disposés suivant la diagonal alors on se ramène à un problème à un dimension et les valeurs de σ_{reg} sont alors de l'ordre 10. Si les points sont disposés suivant les cotés alors les valeurs de σ_{reg} sont alors de l'ordre de 1.

σ et d

Le déplacement d (la distance entre x_0 et $\hat{x}$) est de l'ordre de σ presque dans tous les cas. Le modèle est une somme de fonctions gaussiennes centrées respectivement aux points d'apprentissage. Le chemin optimisant parcouru par LBFGS est probablement définis par un petit nombre de points La position des points est très importante. Plus on augmente le nombre de points et plus on a de chances que les quelques points soient bien disposés.

2.3.3 Les autres paramètres, ϵ_{reg}, C_{reg} et ϵ_{fin}

Nous nous intéressons maintenant aux autres paramètres qui entrent en jeu dans la méthode SVM, et qui ont été décrit section 1.5.5. Nous allons optimiser les critères (CO) et (CE) par rapport aux quatre paramètres que

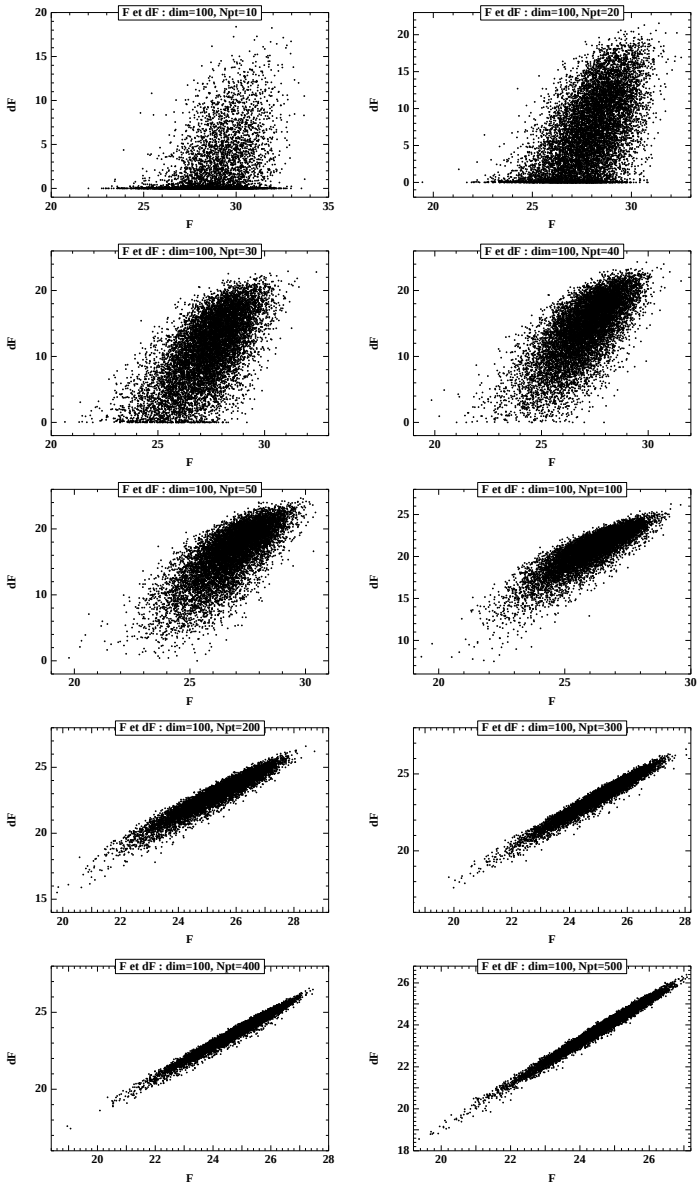


FIG. 2.5 – Chaque point correspond à un tirage de `Npt` $= 10, 20, 30, 40, 50, 100, 200, 300, 400, 500$ points de $[0,1]^{100}$ et évalués avec la fonction sphère. En abscisse, F, la valeur minimum de f trouvée parmi les `Npt` points. En ordonnée, dF, le gain sur f du à l'apprentissage (critère (CO)).

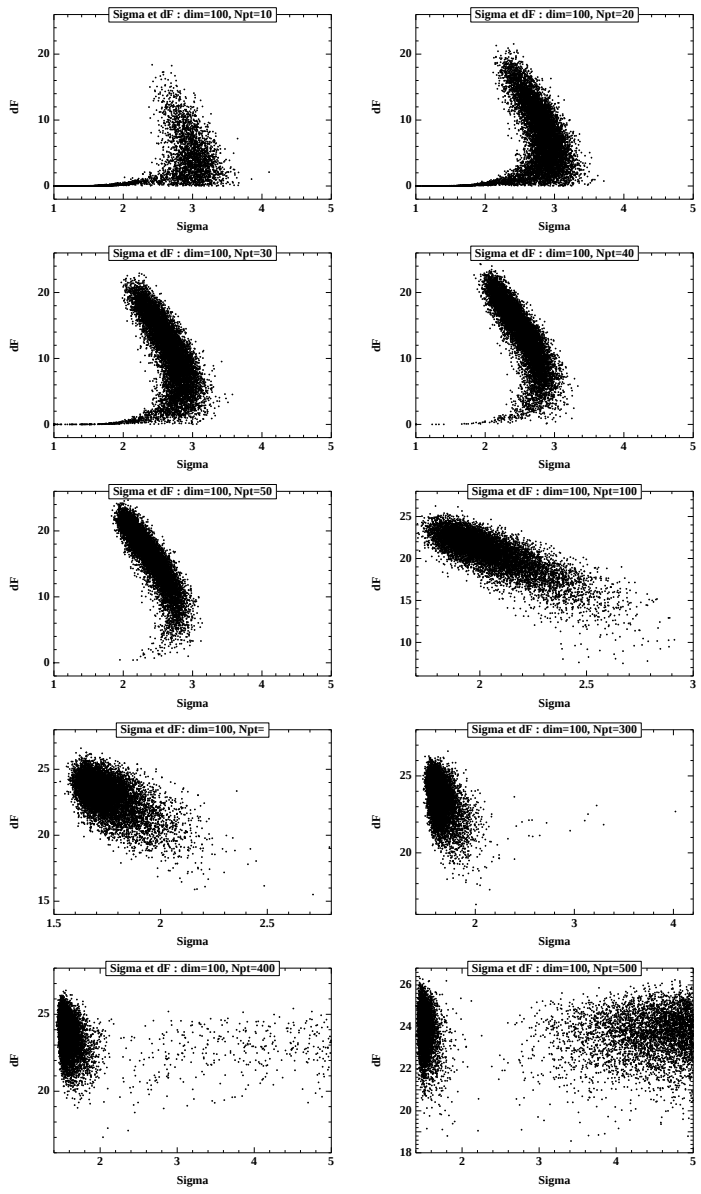


FIG. 2.6 – Chaque point correspond à un tirage de `Npt` $= 10, 20, 30, 40, 50, 100, 200, 300, 400, 500$ points de $[0, 1]^{100}$. En abscisse σ_{CO} la valeur optimale. En ordonnée, la variation de f entre l'optimum trouvé avec l'optimisation de (CO) et F.

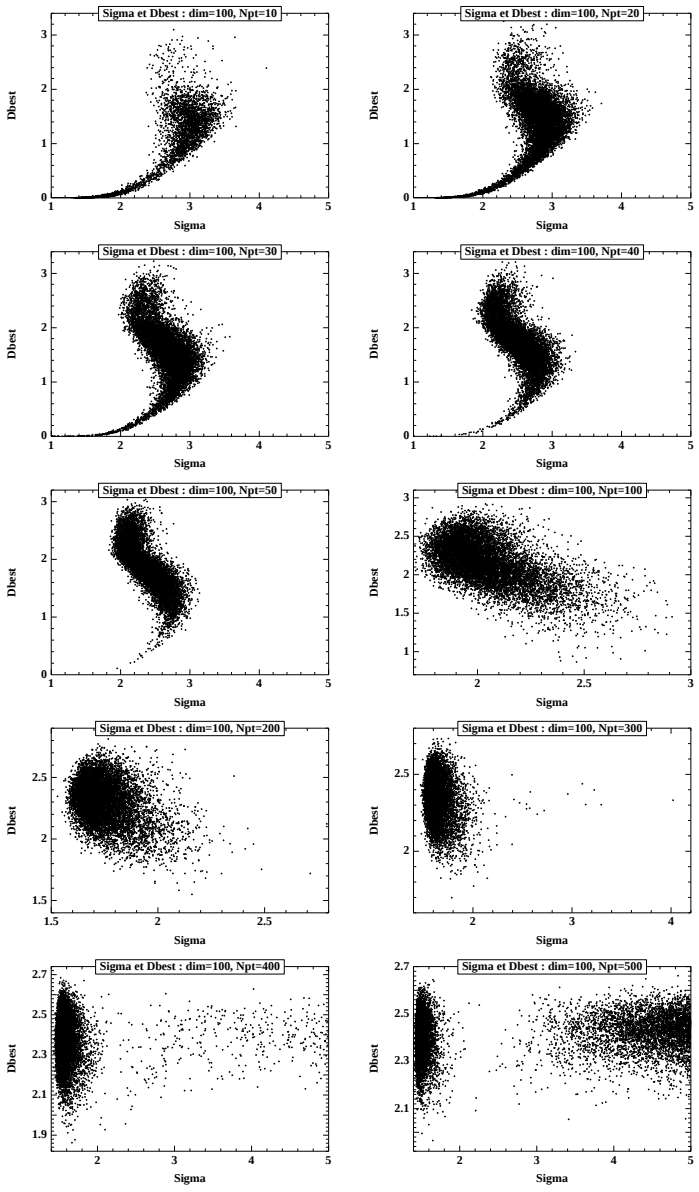


FIG. 2.7 – Chaque point correspond à un tirage de `Npt` $= 10, 20, 30, 40, 50, 100, 200, 300, 400, 500$ points de $[0, 1]^{100}$ et évalués avec la fonction sphère. En abscisse σ_{CO} la valeur optimale. En ordonnée, d, la distance entre l'optimum trouvée avec l'optimisation de (CO) et x_0, le meilleur parmi les points d'apprentissage.

sont σ_{reg}, le paramètre de régularisation étudié en détail dans la section précédent, C_{reg} et ε_{reg} qui servent à faire le poids entre la fonction de perte à ε_{reg}-insensible et le nombre de vecteurs machines. La régularité est entièrement imposée à travers les paramètres du noyau et ϵ_{fin} qui contrôle l'erreur et donc le nombre d'itérations de la régression. Nous chercherons le minimum des critères (CO) et (CE) en limitant les paramètres aux intervalles suivants : $\sigma_{reg} \in [1, 10]$, $C_{reg} \in [1, 100]$, $\epsilon_{reg} \in [1e-10, 1e-1]$ et $\epsilon_{fin} \in [1e-10, 1e-1]$.

Le critère CO

Milles tirages uniformes de 50 points ont été réalisés, et pour chacun d'eux, le critère (CO) a été optimisé par une (3+6)-ES. Chaque courbe de la figure 2.3.3 représente le gain obtenu sur F en ordonnée, avec l'un des 4 paramètres étudiés en abcisse.

Il semblerait qu'aucune tendance particulière ne puisse se dégager en ce qui concerne les paramètres C_{reg} et ϵ_{fin}. Cela veut dire qu'il faudra les optimiser au coup par coup, sans pouvoir a priori restreindre la recherche par des considérations générales.

Le paramètre ϵ_{reg} indique la marge d'erreur tolérée pour l'approximation. Il est clair que les paramètres optimaux se situent vers les petites valeurs, comme on peut le vérifier sur la courbe correspondante de la figure 2.3.3. Les valeurs obtenues sont concentrées dans $[1e-3, 1e-2]$.

Finalement, en ce qui concerne le paramètre σ_{reg}, on remarque que les valeurs se situent dans l'intervalle $[2, 3]$. N'oublions pas toutefois que c'est la fonction sphère, qui est de régularité constante, qu'on approche. Le paramètre optimal dépend aussi de la distribution des points.

Le critère CE

Encore une fois le paramètre C_{reg} n'a pas d'influence dans l'optimisation du critère (CE).

Les deux paramètres ϵ_{reg}, ϵ_{fin} ont les mêmes influences. En effet, il s'agit ici de minimiser l'erreur sur les points. Pour minimiser l'erreur sur les points, il faut absolument une petite valeur de ϵ_{fin}. Pour minimiser aussi l'erreur et ainsi rendre la résolution du problème de régression facile il faut aussi réduire le paramètre ϵ_{reg}. En effet, si ϵ_{reg} est grand, le nombre de support vecteur est petit. La fonction modèle s'écrira en nombre limité de vecteur support. Plus la valeur de ϵ_{reg} est grande et plus la fonction approchée devient pauvre en noyau et plus il est difficile de résoudre le problème de régression. Rappelons que nous utilisons un noyau gaussien. Loin des points vecteurs supports la fonction est constante.

La non influence du paramètre σ_{reg} dans la minimisation de (CE) était

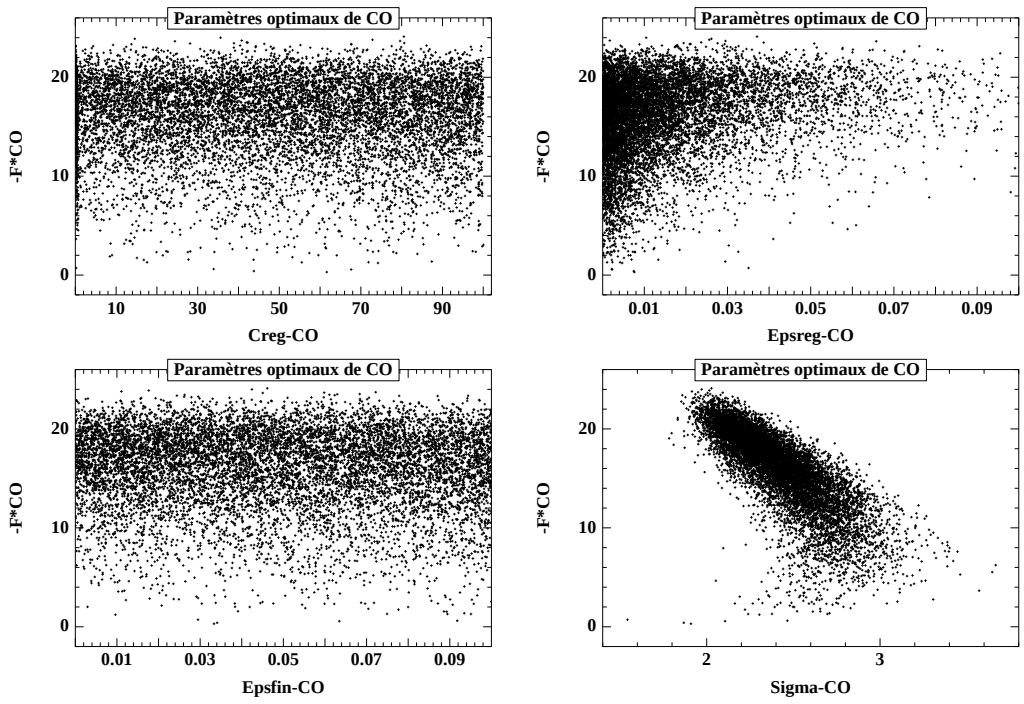


FIG. 2.8 – Chaque point correspond à un tirage de `Npt` $= 50$ point $[0,1]^{100}$ et évalués avec la fonction sphère. En abscisse un des paramètres σ,ϵ_{reg}, C_{reg} et ϵ_{fin}. Les paramètres sont optimaux pour le critère (CO). En ordonnée, la variation $dF = -F * (CO)$.

prévisible. En effet l'interpolation est possible quelle que soit la régularité qu'on impose. Si on évalue le critère (CO) avec les paramètres optimaux de (CE), on se rend compte que la valeur σ_{reg} qui avait été choisie arbitrairement dans la minimisation de l'erreur donne des résultats répartis suivant une droite. Le paramètre σ_{reg} pourrait donc être choisi a priori en fonction d'hypothèses sur la régularité de la fonction f.

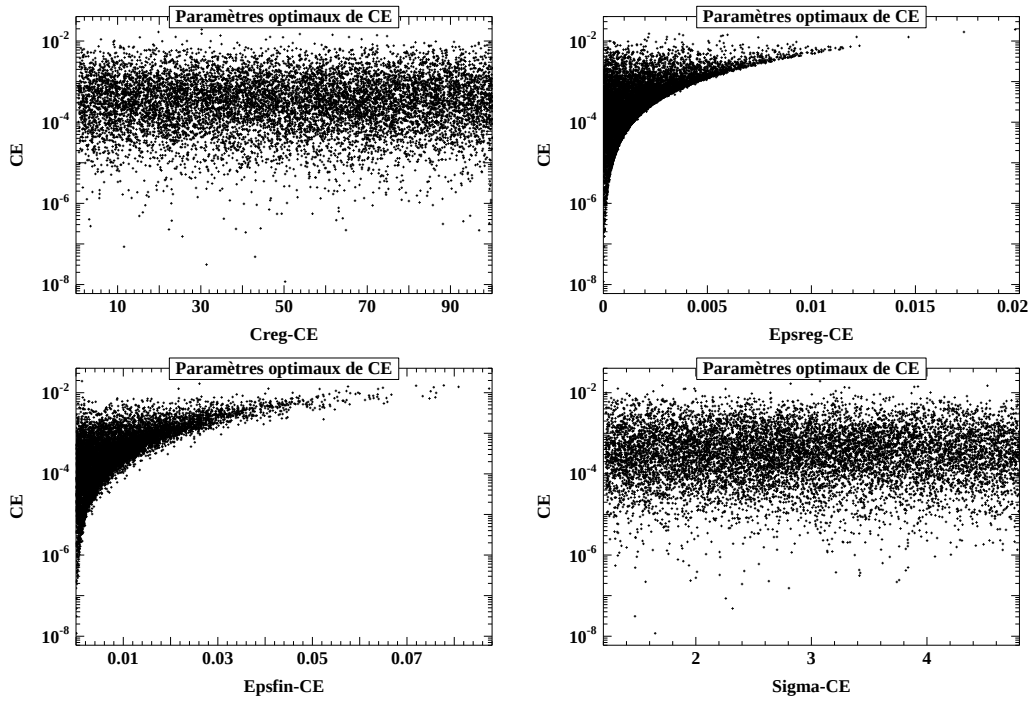


FIG. 2.9 – Chaque point correspond à un tirage de `Npt` $= 50$ point $[0,1]^{100}$ et évalués avec la fonction sphère. En abscisse un des paramètres σ,ϵ_{reg}, C_{reg} et ϵ_{fin}. Les paramètres sont optimaux pour le critère (CE). En ordonnée, la valeur de (CE).

2.3.4 Noyaux gaussiens : Discussion

Le critère **(CO)** donne sans conteste les meilleurs résultats, que ce soit avec une optimisation de 10 ou 50 générations du ES (voir 2.12). Malheureusement, dans le cas où la fonction est coûteuse, son utilisation pourrait devenir problématique puisqu'il requiert des évaluations de la fonction exacte supplémentaires.

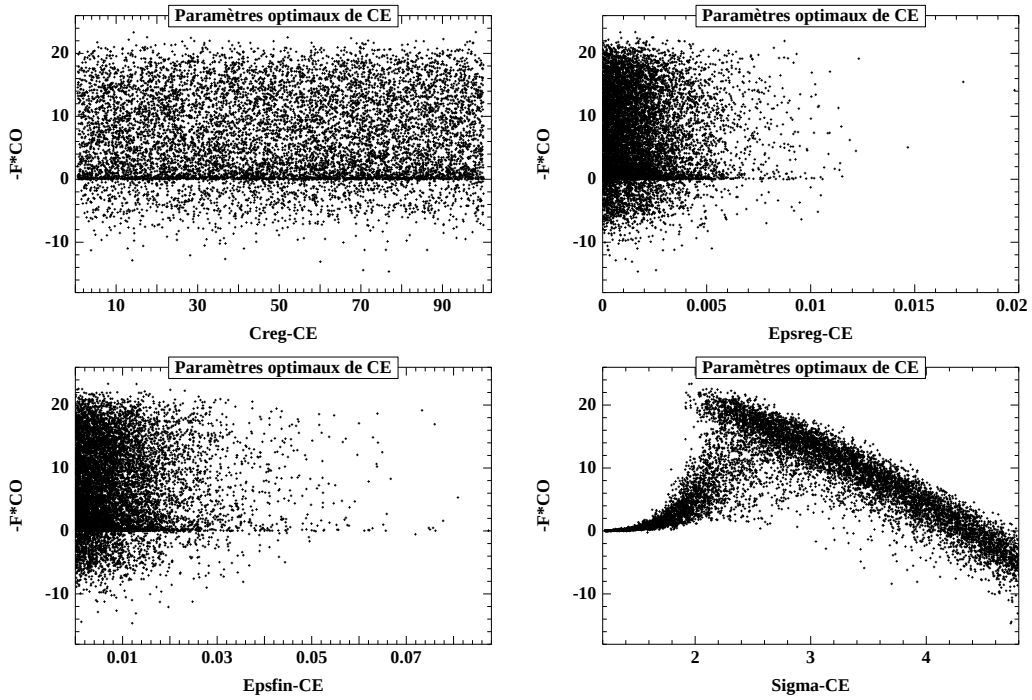


FIG. 2.10 – Chaque point correspond à un tirage de `Npt` $= 50$ point $[0,1]^{100}$ et évalués avec la fonction sphère. En abscisse les valeurs optima des paramètres σ,ϵ_{reg}, C_{reg} et ϵ_{fin} par rapport au critère (CE). En ordonnée, la valeur associée à $-F*CO$.

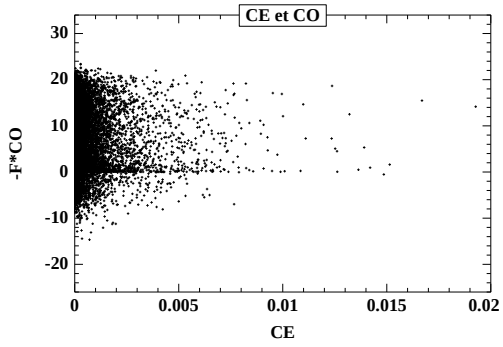
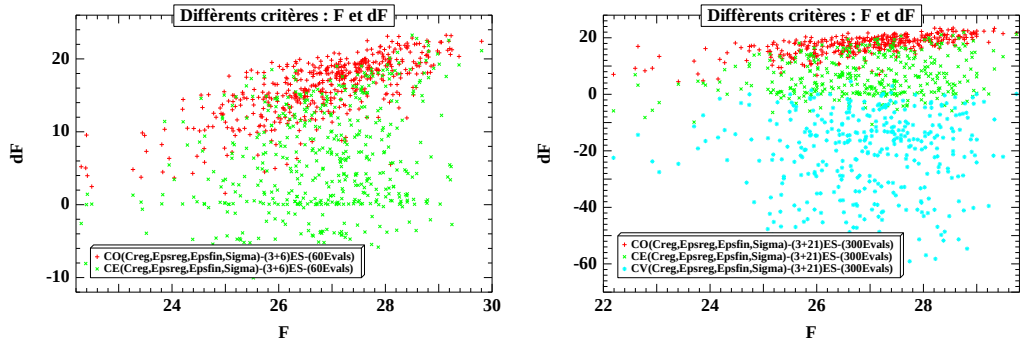


FIG. 2.11 – En abscisse la valeur optimale du critère (CE). En ordonnée, la valeur associée à $-F*CO$.

Le critère **(CE)** donne en général de mauvais résultats, qui sont surtout peu fiables. Cela veux dire que minimiser l'erreur empirique n'est pas compatible avec notre but d'optimisation.

Le critère **(CV)** sur la sphère ne donne pas de résultats si on considère les paramètres ϵ_{reg} et ϵ_{fin} comme variables. (voir Figure 2.12 et la section 2.2.1 concernant la généralisation).

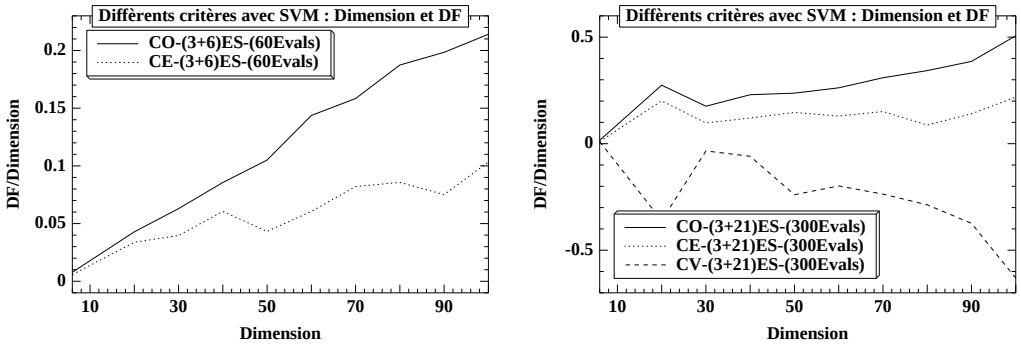


(a) (b)

FIG. 2.12 – A chaque point correspond un tirage uniforme de 50 points en dimension 100. L'axe des abscisse correspond à la valeur minimal de f parmi les 50 points. L'ordonnée correspond à la variation entre le meilleur parmi les 50 points et l'optimum du modèle obtenu après adaptation avec les différents critères. Pour (a) l'adaptation s'effectue avec (3+2*3)ES en 10 générations et pour (b) 50 générations. Les paramètres sont limités aux intervalles suivants : $\sigma_{reg} \in [1, 10]$, $C_{reg} \in [1, 100]$, $\epsilon_{reg} \in [1e-10, 1e-1]$ et $\epsilon_{fin} \in [1e-10, 1e-1]$.

2.4 Le Krigeage

L'approche de la méthode de Krigeage est complètement différente de celle des SVM, puisqu'elle ne vise quà approcher le mieux possible les exemples d'apprentissage. En particulier, les paramètres disponibles pour contrôler la variabilité ne sont que les paramètres des noyaux – en l'occurrence ceux des noyaux gaussiens et sigmoïdaux. En grande dimension, il serait illusoire d'utiliser du Krigeage quadratique avec peu d'exemples. De plus, le temps de régression est trop élevé et rend l'approche de l'adaptation problématique.



(a) (b)

FIG. 2.13 – En abcisses la dimension de l'espace. En ordonnée la moyenne de σ_{XX} sur 100 tirages uniformes de 50 points dans le domaine $[0,1]^{Dim}$ et évalués avec la fonction sphère. Pour (a) l'adaptation s'effectue avec (3+2*3)ES en 10 génération et pour (b) sur 50 générations. Les paramètres sont limités aux intervalles suivants : $\sigma_{reg} \in [1,10]$, $C_{reg} \in [1,100]$, $\epsilon_{reg} \in [1e-10, 1e-1]$ et $\epsilon_{fin} \in [1e-10, 1e-1]$.

2.4.1 Fonction de corrélation Gaussienne

Nous allons maintenant reprendre la teneur des expériences réalisées pour la méthode SVM dans la section précédente. Nous pouvons cependant prédire, compte-tenu de la remarque ci-dessus, que le critère (CO) sera très probablement le seul vraiment efficace dans le cadre du Krigeage.

Profils des différents critères

Critère d'erreur (CO) : comme dans le cas des SVM, le critère comporte un minimum, et est régulier. Nous pourrons donc accélérer l'optimisation en utilisant la méthode de Legendre.

Critère d'erreur (CE) : Il faut remarquer que les valeurs de ce critère quelles que soient les valeurs du paramètre σ_{reg} sont très petites (de l'ordre $1e-20$). Même si, visuellement, on peut identifier un minimum proche de la valeur donnant un optimum du critère (CO) (puisque les très petites valeurs de σ_{reg} sont de toute façon à éviter, l'optimisation de ce critère est numériquement inconcevable.

Critère d'erreur (CV) : Il est régulier et comporte un minimum. Notons que le minimum du critère (CV) est diffèrent de celui de (CO).

Critère d'erreur (CG) Le critère (CG) certes est minimal pour les valeurs de σ_{reg} de l'ordre des optima du critère (CO), mais l'énorme plateau quasi-horizontal risque de rendre le problème difficile à optimiser.

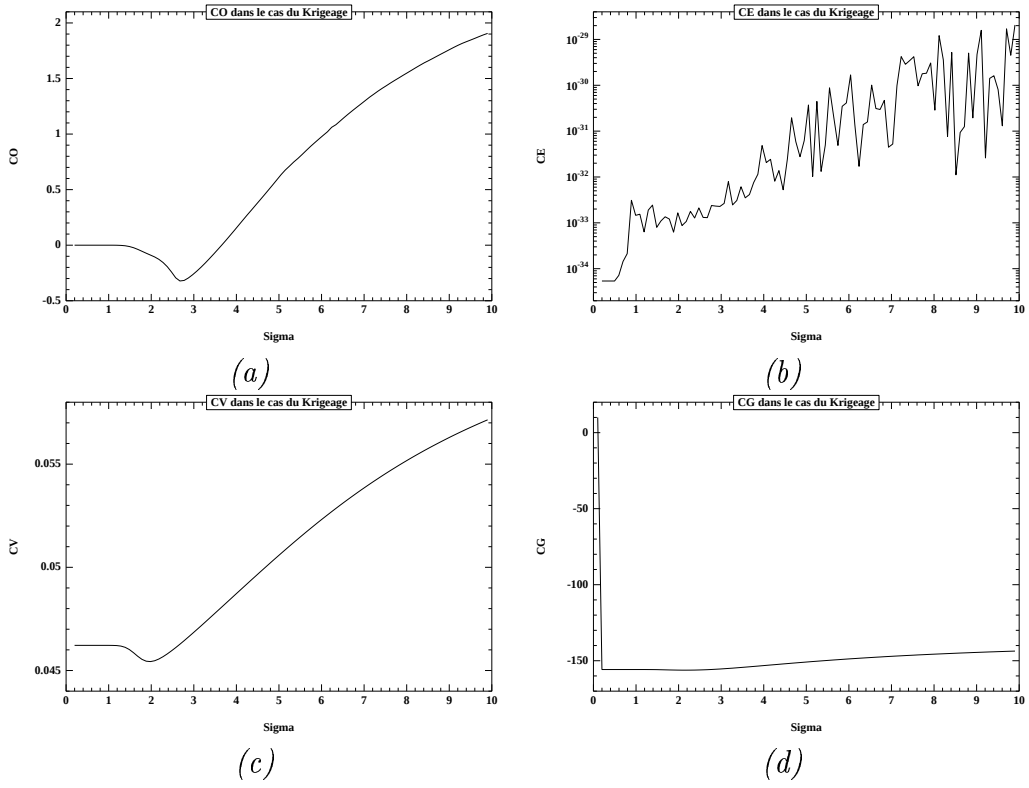


FIG. 2.14 – Le profil des différents critères en fonction du paramètre σ_{reg}. (a) le critère (CO), (b) le critère (CE) , (c) le critère (CV) et (d) le critère (CG). Les données d'apprentissage sont 50 points tirés uniformément dans le domaine $[0,1]^{100}$ et évalués avec la fonction Sphère.

Influence de la dimension

Critère d'erreur (CO) Comme prévu, ce critère est toujours le meilleur.

Critère d'erreur (CV) Dans le cas de Krigeage c'est le critère (CV) qui détrône le critère (CE). Contrairement au cadre SVM, la régression s'apparente à une interpolation. Optimiser le critère (CV) revient à chercher le modèle qui généralise le mieux sachant qu'il passe déjà par les points d'apprentissages – condition qui n'est pas imposée dans le cas SVM. Il faut dans le cas de SVM imposer aux paramètres qui contrôlent la précision d'être petits. Les temps de régression de SVM rejoignent alors les temps de régression de ceux du Krigeage. Ce point sera développé dans le chapitre suivant.

Critère d'erreur (CE) Il est clair que minimiser l'erreur sur tous les points ne correspond pas ici à l'objectif final de notre étude.

Critère de généralisation (CG) Cela consiste à minimiser la variance sur les points. On remarque qu'à partir de la dimension 20 et au delà, la performance de ce critère diminue fortement. Il faudrait probablement augmenter le nombre de points proportionnellement à la dimension – mais le temps de régression deviendrait alors trop important.

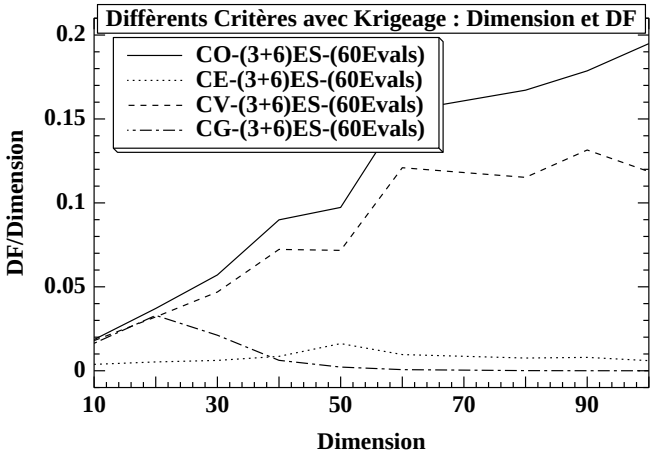


FIG. 2.15 – En abcisses la dimension de l'espace. En ordonnées la moyenne de σ_{XX} sur 100 tirages uniformes de 50 points dans le domaine $[0,1]^{Dim}$ et évalués avec la fonction sphère.

Quel critère choisir ?

Le coût du critère (CG) est celui d'une régression : ainsi, si on ne veut pas payer le sur-coût imposé par le critère (CO), il est certainement le plus intéressant en petites dimensions, puisque ses performances sont comparables au critère (CV) pour un coût moindre. Reste la question des grandes dimensions ...

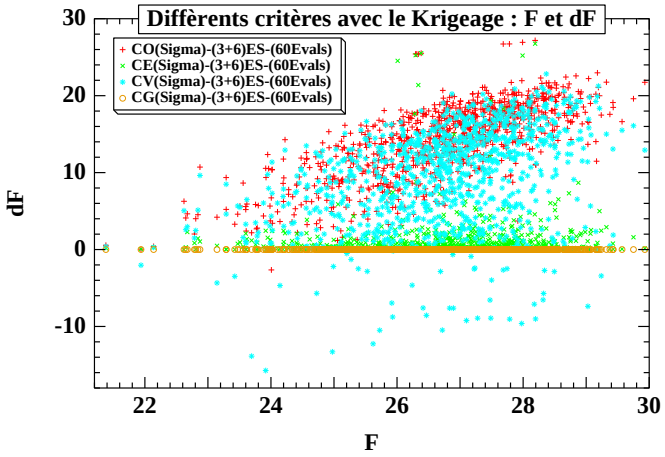


FIG. 2.16 – A chaque point correspond un tirage uniforme de 50 points en dimension 100. L'axe des abscisses correspond à la valeur minimal de f parmi les 50 points. L'ordonnée correspond à la variation entre le meilleur parmi les 50 points et l'optimum du modèle obtenu après adaptation avec les différents critères. L'adaptation s'effectue avec (3+2*3)ES en 10 générations.

2.5 Conclusion

2.5.1 SVM ou Krigeage ?

C'est évidemment une des questions de cette thèse ! Pour l'instant, les éléments de réponse apportés par ce chapitre sont que la méthode SVM semble plus avantageuse que la méthode Krigeage du fait qu'elle offre la possibilité de donner des solutions intermédiaires (c'est-à-dire ne passant pas forcément par les points d'apprentissage) en un temps relativement raisonnable. Nous allons tenter maintenant de résumer les enseignements de ce chapitre sur les deux méthodes.

Méthode SVM

- Le noyau sigmoïde ne semble pas satisfaisant dans le cas d'une fonction régulière. En effet nous disposons d'un petit nombre de points pour l'apprentissage. Optimiser une telle fonction à l'aide de noyaux sigmoïdes est problématique. Il faut probablement ajouter un critère géométrique pour dimensionner correctement les paramètres de noyau. Néanmoins, en attendant des études plus spécifiques sur les noyaux sigmoïdes (voir chapitre 7), tous les résultats qui suivent ont été obtenus ne utilisant des noyaux gaussiens.
- Nous avons remarqué que le paramètre C_{reg}, qui est un paramètre de borne de régularité pour le problème de régression, ne semble pas avoir d'influence dans l'optimisation des critères (CO), (CE) et (CV). Il ne sera donc plus considéré comme un paramètre dans la suite.
- Optimiser le critère (CV) simultanément par rapport aux paramètres ϵ_{reg}, ϵ_{fin} et σ_{reg} ne donne pas de résultats. Il serait raisonnable de fixer les paramètres ϵ_{reg} et ϵ_{fin} assez petits et de considérer comme seul paramètre σ_{reg}. Malheureusement, plus les paramètres ϵ_{reg} et ϵ_{fin} sont petits et plus le temps de régression est grand. Nous devons prendre en compte le facteur temps et par exemple diminuer progressivement les paramètres de précision tout en testant différents paramètres σ_{reg}. L'optimisation des paramètres par un ES classique n'est alors plus adaptée.
- La minimisation de (CO) est la plus coûteuse, mais nous avons pu constater qu'en minimisant légèrement le paramètre ϵ_{reg} qui contrôle la précision, on peut trouver à l'aide d'une stratégie d'évolution différentes valeurs de σ_{reg} acceptables.
- Dans l'optimisation de (CE) le paramètre σ_{reg} n'a pas d'influence. Néanmoins, σ_{reg} et la valeur de CO sont corrélées lorsque (CE) diminue. De combien faut-il minimiser (CE) pour se retrouver en situation

de corrélation, sachant que réduire les ϵ_{fin} et ϵ_{reg} augmente le temps de régression ?

L'objectif qui est de trouver un modèle avec un temps raisonnable dont l'optimum minimise la fonction objective, c'est à dire la fonction (CO), est atteint. Cet objectif est atteint avec des évaluations supplémentaires pour l'adaptation du modèle. Ces évaluations sont nécessaires car nous n'avons pas une idée de la régularité. Dans le cas de la sphère le nombre d'évaluations n'est pas grand ; il est de l'ordre 10 à 15 évaluations par régression. Mais bien entendu, cette approche n'aura d'intérêt que si le gain en nombre total d'évaluations compense ce sur-coût.

Par contre, il en ressort des résultats de ce chapitre qu'il n'est pas forcément nécessaire d'approcher très précisément la fonction objectif pour l'optimiser – ce que permet la méthode SVM.

Le Krigeage

- La fonction de corrélation sigmoïde ne donne pas non plus avec le Krigeage des résultats significatifs. Nous travaillons alors par la suite avec la fonction de corrélation gaussienne.
- L'optimisation de (CE) n'est pas significative. En effet, le Krigeage est sensé approcher au mieux les données, et les résultats obtenus ne montrent pas de corrélation en la minimisation de (CO) et celle de (CE).
- L'optimum de (CO) ne coïncide pas non plus avec celui de (CV). Il se peut qu'en général l'optimisation de (CV) ne soit pas profitable.
- Le critère (CG) n'est pas satisfaisant en grande dimension. Par contre, il est couramment utilisé en petite dimension. Il est alors plus intéressant (car bien moins cher) que le critère objectif.
- Le Krigeage quadratique n'est pas envisageable du fait que le temps de régression est trop grand. A noter que nous n'avons pas pris en considération explicitement dans les essais présentés le temps de régression, mais il est largement supérieur pour la méthode de Krigeage aux temps de régression de la méthode SVM. (CV) est raisonnable mais lourd en temps de calcul.

2.5.2 Discussion

Nombre de points

De combien de points avons-nous besoin pour que l'apprentissage fonctionne bien ? Il est clair que c'est une direction de descente qu'on cherche,

et pas forcément **la** direction de descente optimale (d'autant qu'il est bien connu que la direction de descente optimale n'est pas forcément la direction de l'optimum !) et donc on peut probablement se contenter de petit nombre de points, mais bien disposés dans l'espace de recherche.

Surcoût du critère (CO) ?

Dans le cas de la sphère, l'optimisation du critère (CO) est facile et le surcoût relativement peu élevé. La situation sera sans doute identique pour les fonctions régulières. Mais encore une fois, notons que l'objectif final n'est pas d'optimiser la partie apprentissage, mais d'optimiser la fonction objective, et par conséquent même pour les fonction chahutées, on pourra se contenter d'une approximation peu précise du moment que la régularité à gros grain est capturée par l'approximation. Cela devrait pouvoir se faire également avec peu de calculs supplémentaires.

Règle heuristique

On a proposé une heuristique pour σ_{reg} dans le cadre du noyaux gaussiens qui donne un ordre de grandeur physiquement valable : on veut que les boules centrés sur les points d'apprentissage de rayon σ_{reg} se touchent. En fait le σ_{reg} dépend plus de la disposition des points que du nombre de points : pour chaque distribution de points il faut calculer la distance moyenne aux voisins et en tirer σ_{reg}. Mais on serait obligé de calculer toutes les distances et ce serait trop coûteux. On a utilisé une formule qui donne cette moyenne en toute dimension. Il y a là deux remarques à faire : d'une part avec un petit tirage, on ne pourrait jamais prétendre approcher cette moyenne par des expériences numériques en grandes dimensions : en dimension 100, il faut 10^{100} points pour mailler l'hypercube avec 10 points dans chaque direction ! Donc les moyennes qu'on peut calculer avec 500 points sont un peu aléatoires. D'autre part, on aurait pu changer le critère pour que les boules se chevauchent à 10, 20 ou 30 % du volume... Encore une fois, le critère heuristique donne juste l'ordre de grandeur et permet ainsi de restreindre l'intervalle de recherche pour trouver le σ_{reg} optimal : on cherche ainsi des sigmas 3 fois plus petits à 3 fois plus grands. Vu sous cet angle, les résultats expérimentaux sont très bons.

Par contre, dans le cas de la sigmoïde, la caractérisation géométrique est plus compliquée. Il faut tenir compte des intersections entre les différents plans. Il est nécessaire de les établir pour bien définir les bornes de recherche des paramètres de la sigmoïde pour rendre l'optimisation des critères (CX) envisageable avec un algorithme évolutionnaire.

Portée des résultats

Peut-on généraliser les résultats obtenus dans ce chapitre sur toutes les fonctions unimodales ? Sur des fonctions multi-modales ? Nous verrons dans le chapitre suivant des tests sur des fonctions plus complexes.

Ce chapitre a montré que l'adaptation des paramètres des méthodes d'approximations pouvait se révéler profitable pour l'objectif final d'optimisation de fonctions. Nous allons nous pencher dans le chapitre suivant sur l'utilisation de cette approche dans un processus itératif. Sur la foi des résultats de ce chapitre, nous proposerons une méthode dynamique pour réaliser pendant l'optimisation elle-même l'adaptation des paramètres de l'apprentissage.

Chapitre 3

Algorithme déterministe & algorithme stochastique

3.1 Introduction

Dans le chapitre précédent, nous avons étudié différents paramètres des méthodes d'apprentissage, et examiné les différents critères qui doivent nous permettre de régler ces paramètres lors de l'utilisation de ces méthodes au sein d'algorithmes hybridés avec des algorithmes évolutionnaires. Toutes les expériences concernaient des essais statiques des méthodes d'apprentissage, effectuées une seule fois sur des ensembles d'exemples indépendants, choisis uniformément dans un domaine donné. Or lors de l'hybridation avec un algorithme évolutionnaire, ces méthodes d'apprentissage seront appliquées de manière itérative, sur des ensembles d'exemples qui ne seront plus tirés uniformément sur le même ensemble de $\mathbb{R}^n$.

Il nous a donc semblé opportun, avant de se lancer dans l'hybridation elle-même, d'examiner de plus près le comportement itératif de la technique "apprentissage + optimisation du modèle appris" aux fins d'optimisation de la fonction objectif. En particulier, un des problèmes importants à résoudre concerne l'aspect local des approximations qui seront effectuées lors de l'hybridation : dans quel voisinage du meilleur individu courant faudra-t-il choisir les exemples d'apprentissage pour effectivement accélérer la convergence de l'algorithme évolutionnaire ?

Dans le but d'étudier la dynamique du comportement itératif de l'apprentissage couplé avec une optimisation déterministe du modèle, en essayant de comprendre la cause (apprentissage ou optimisation déterministe) de ce comportement, nous nous proposons dans ce chapitre de comparer les quatre algorithmes suivants, qui mettent en jeu les aspects stochastiques et déter-

ministes à des degrés divers :

- le premier algorithme, `AlgoAlea`, consiste à tirer aléatoirement `Npt` points dans un voisinage bien choisi du point courant, et à prendre le meilleur point obtenu comme nouveau point courant ;
- le deuxième algorithme, `AlgoDet`, rajoute au premier une étape d'optimisation déterministe sur la fonction objectif. Cet algorithme convergera évidemment rapidement dans le cas de fonctions uni-modales, mais il sera intéressant de l'étudier sur des fonctions multimodales ;
- le troisième algorithme, `AlgoApp`, remplace l'optimisation déterministe sur la fonction objectif du second algorithme par l'optimisation déterministe d'un modèle de la fonction objectif, comme décrit au chapitre précédent. En particulier, les paramètres de la méthode d'apprentissage choisie (SVM ou Krigeage) seront ajustés par rapport aux critères évalués dans ce même chapitre ;
- un quatrième algorithme, `AlgoAppBis` sera proposé, dans lequel l'optimisation déterministe du modèle appris sera remplacée par une optimisation évolutionnaire. En effet, le modèle appris n'est pas forcément uni modal. Et même si nous sommes a priori plutôt intéressés par un optimum local proche du meilleur individu courant, il peut être avantageux d'essayer de regarder plus loin.

La base stochastique de tous ces algorithmes sera donc un algorithme de type (1+`Npt`)-ES, qui, à partir d'un itéré, génère un nouveau point en tirant `Npt` points uniformément dans la boule centrée sur l'itéré courant et de rayon ε (et non pas suivant une distribution gaussienne, comme l'algorithme ES classique). Un des points cruciaux étudiés dans ce chapitre concernera, comme il a été dit, l'évolution du voisinage dans lequel les points d'apprentissage seront tirés : les effets de différente mises à jour du rayon ε de la boule-échantillon seront soigneusement étudiés, au travers d'expériences sur la convergence de ces quatre algorithmes sur les quatre différents type de fonction définis au Chapitre 2 : les fonctions Sphère et Ellipse pour la convergence locale, et les fonctions Griewank et GriewankE pour la convergence globale.

Ce chapitre n'a donc pas pour but d'opposer algorithmes stochastiques et déterministes en général. Au contraire, tout au long du chapitre, nous mettrons en relief les points forts et faible des deux familles d'algorithme, ainsi que les similarités et les différences dans leurs comportements.

Le premier algorithme AlgoAlea est stochastique pur. Le deuxième du fait de l'utilisation du gradient est un algorithme hybride mais nous verrons qu'il se comporte en fait comme un algorithme à dominance déterministe. Enfin, les deux derniers algorithmes, AlgoApp et AlgoAppBis, utilisant l'opérateur d'apprentissage, retrouvent un caractère stochastique.

Les points suivants seront donc considérés pour chacun des 4 algorithmes :
- Le plus important est l'aspect local de la sélection des points, comme il a été discuté plus haut ;
- L'algorithme déterministe utilisé (BFGS) est déjà un algorithme hybride. En effet, la recherche linéaire est hybridé avec une interpolation avec des splines en dimension (voir l'état de l'art plus d'explications).
- Un point souvent négligé, mais qui va s'avérer des plus importants dans le cadre hybride, est celui des conditions d'arrêts. Les algorithmes déterministes s'arrêtent lorsque les conditions d'optimalité sont satisfaites, dans le cas des fonctions régulières. Ainsi à l'optimum, le gradient doit être nul et la matrice hessienne définie positive (pour un minimum). Du côté des algorithmes stochastiques, par contre, on ne peut que demander une solution "acceptable", et la condition d'arrêt est basée sur un seuil de la valeur f ou de df. Ici nous allons montrer que la valeur ε pourrait être utilisée comme test d'arrêt.
- Le dernier point porte sur les fonctions non convexes comportant plusieurs minima locaux. Un algorithme déterministe pur converge en général vers le premier minimum rencontré. Mais la recherche linéaire ou curviligne de BFGS permet de faire des bonds pour éviter ces minima locaux. Le modèle quadratique avec recherche curviligne et l'interpolation lissent le problème, permettant éventuellement de dépasser les minima locaux. Ces méthodes peuvent converger vers le minimum global si au niveau global, la fonction a un comportement régulier respectant les conditions d'optimalité à l'optimum global. Nous montrerons que l'apprentissage introduit dans ces algorithmes a aussi ces propriétés.

Nous allons maintenant introduire et étudier successivement les quatre algorithmes esquissés ci-dessus.

3.2 L'algorithme AlgoAlea

3.2.1 Définition

A chaque itération, cet algorithme tire uniformément `Npt` points autour du point courant x_k dans une boule de rayon ε_k, choisit pour x_{k+1} le meilleur des $\texttt{Npt} + 1$ points ainsi obtenus et met à jour ε_k. Plus précisément :

Initialisation :
- Tirer `Npt0` points uniformément sur le domaine de recherche.
- Le meilleur est le point de départ de l'algorithme :
  $x_0 = ArgMax\{f(x_i), i \in [1, \texttt{Npt0}]\}$

– Initialiser les paramètres : $k = 0$, ε_0 donné

Itération :

1. Tirer uniformément `Npt` points autour de x_k :
   $x_k^i = U[B(x_k, \varepsilon_k)], i \in [1, \texttt{Npt}]$
2. Choisir le meilleur (en incluant x_k) :
   $x_{k+1} = ArgMax\{f(x_k), f(x_k^i), i \in [1, \texttt{Npt}]\}$
3. Mettre à jour ε :
   $\varepsilon_{k+1} = \max(\alpha * \varepsilon_k, \beta * d_k)$

où α, β sont des paramètres à déterminer, et oú $d_k = d(x_{k+1}, x_k)$ est la distance euclidienne entre les itérés précédents.

Paramètres de l'algorithme

L'algorithme dépend donc des paramètres suivants :

– `Npt` : Nombre de points tirés aléatoirement à chaque itération.
– $0 < \alpha < 1$ et $1 < \beta$: paramètres pour l'adaptation du domaine.
– `Npt0` : Nombre de points tirés uniformément initialement sur tout le domaine de recherche. En général, `Npt0` sera pris égal à Npt dans un souci de diminution du nombre de paramètres (quelques expériences préliminaires n'ont pas montré une grande influence de ce paramètre sur les résultats).
– ε_0 : Rayon initial du domaine de recherche pour définir l'itéré suivant.
– Le nombre maximal d'évaluations $Neval_{max}$

Si à l'étape k aucun individu n'est meilleur que x_k, alors d_k vaut zéro et du fait que α est inférieur à 1, le domaine de recherche diminue pour tenter d'améliorer les chances de convergence de l'algorithme. Dans le cas contraire, la diminution due à α est contrecarrée par le terme $d_k * \beta$, pour éviter de diminuer le domaine de recherche trop rapidement, et être sûr de continuer à échantillonner autour de x_k **et de** x_{k+1}.

3.2.2 Étude de convergence qualitative

Le choix des paramètres α et β a une grande influence sur la qualité de l'algorithme tant sur la vitesse de convergence que sur la capacité à converger vers la solution optimale.

Étudions l'effet de chacun des paramètres. Pour cela, introduisons tout d'abord la notion de distance totale parcourue (DTP), définie comme la

somme des déplacements depuis le point initial (x_0) jusqu'à l'arrêt de l'algorithme :

$$DTP = \sum d(x_k, x_{k+1})$$

Commençons par l'étude de l'influence du paramètre α, et prenons $\beta = 0$. Le paramètre α est responsable de la convergence de l'algorithme, diminuant progressivement le domaine de recherche. En effet, pour $\beta = 0$, la distance totale parcourue vérifie, quelle que soit le nombre de générations, l'inégalité

$$DTP = \sum d_k \leq \sum_{k=0}^{\infty} \varepsilon_k \leq \varepsilon_0 \sum_{k=0}^{\infty} \alpha^k = \frac{\varepsilon_0}{1-\alpha} \tag{3.1}$$

En d'autres termes, si la solution est à une distance supérieure à $\frac{\varepsilon_0}{1-\alpha}$ du point de départ, l'algorithme n'a aucune chance de la trouver si $\beta = 0$. Le choix de β doit donc permettre d'éviter une telle convergence prématurée de l'algorithme.

Mais il ne suffit pas de choisir $\beta > 1$. En effet, à l'itération k, on tire `Npt` points à l'intérieur de la boule de centre x_k et de rayon ε_k. Si l'optimum est situé à la surface de la boule (cas loin de l'optimum), la déplacement optimal sera au mieux égal à ε_k. Statistiquement, la distance

$$d(x_k, x_{k+1}) = \tau \varepsilon_k$$

$\tau < 1$ étant un facteur statistique dépendant de Npt qui peut être calculé avec les statistiques d'ordre de `Npt` points de la boule.

On en déduit alors que :

$$d_{k+1} \leq \varepsilon_{k+1} \leq \beta d_{k-1} \leq \tau\beta . d_k \tag{3.2}$$

Pour éviter une possible convergence prématurée, il faudrait au moins que (cf. le raisonnement fait sur α) :

$$\tau\beta > 1$$

τ dépend de Npt, de la dimension de l'espace et de la densité de probabilité. On pourrait calculer à l'aide de statistiques d'ordre de la distribution uniforme (ou numériquement) une valeur de β en dessous de laquelle il ne faut pas descendre au risque d'avoir une convergence prématurée. Mais cela ne donnerait de toute façon qu'une borne inférieure et l'étude expérimentale qui suit serait quand même nécessaire pour en tester la finesse.

3.2.3 Étude expérimentale

Conditions expérimentales

Les expériences qui vont être présentées maintenant concernent des fonctions en dimension 100. Tous les résultats sont des moyennes sur 10 lancements indépendants pour chaque jeu de paramètres. Dans les tests, nous ne distinguons pas les deux paramètres `Npt` et $\texttt{Npt}_0$. Les résultats sont limités par le nombre maximum d'évaluations : nous avons choisi $Neval_{max} = 10^5$ Nous avons également décidé d'arrêter l'algorithme après 50 itérations durant lesquelles on ne constate plus d'amélioration. Dans certains tableaux de résultats, quand c'est nécessaire nous préciserons aussi le nombre d'itérations et le nombre d'évaluations qui ont suffi à la convergence de l'algorithme. Les résultats sont présentés en échelle logarithmique sauf indication du contraire.

En premier lieu nous allons fixer les valeurs α et β. Nous prendrons $\alpha = 0.9$ pour ne pas diminuer trop vite le domaine de recherche ε et $\beta = 2$ pour permettre de l'augmenter. Nous étudierons plus en détails l'influence de ces deux paramètres ultérieurement, et nous montrerons que ces valeurs peuvent être considérées comme raisonnables, bien que pas forcément optimales.

Influence de `Npt`

Il est clair qu'à chaque itération, plus le nombre de points tirés aléatoirement est grand, et plus la probabilité d'amélioration est grande. Nous avons testé différentes valeurs de `Npt` inférieures à 50. Ce nombre est arbitraire mais nous permet la comparaison entre les différents algorithmes.

Les résultats (log de la meilleure valeur obtenue, moyennée sur 10 essais) sont donnés dans la Table 3.1. Tout d'abord, même sur la fonction sphère, on remarque que prendre un trop petit nombre de points (≤ 10) peut conduire à faire converger prématurément l'algorithme `AlgoAlea`, et ce malgré un α proche de 1. Selon la loi qui régit l'évolution de ε, et suivant la discussion de la section 3.2.2, la diminution est principalement déterminée par le paramètre β – et le déplacement a d'autant plus de chance d'être non nul que le nombre de points tirés est grand. Le paramètre β doit donc *a priori* dépendre du nombre de points.

Le phénomène est encore plus frappant sur la fonction Griewank, comme on pouvait s'y attendre vu les nombreux optima locaux.

Avec la fonction Ellipse, les résultats ne sont pas étonnants, mais le phénomène est amplifié. En effet, il ne faut pas oublier qu'on est en dimension 100

et qu'on est en train de tirer aléatoirement 50 points pour un problème mal conditionné. En terme numérique, une petite erreur par rapport à la direction propre associée à la plus grande valeur propre, risque de détériorer la valeur de la fonction objective plutôt que de l'améliorer. Sans un apprentissage adéquat, on ne peux donc pas espérer converger proprement vers l'optimum.

`Npt`	5	10	15	20	25	30	35	40	45	50
Sphère										
$Neval_{max} = 10^4$	0.3	-4.9	-16	-13	-11	-8.1	-7.3	-6.2	-5.4	-4.7
$Neval_{max} = 10^5$	0.3	-4.9	-29	-29	-29	-29	-30	-30	-30	-29
Ellipse										
$Neval_{max} = 10^4$	4.3	3.8	3.8	4.1	4.6	4.5	4.9	5.0	5.2	5.2
$Neval_{max} = 10^5$	4.3	3.7	3.4	3.3	3.1	3.1	3.0	3.0	3.0	2.9
Griewank										
$Neval_{max} = 10^4$	0.7	-0.66	-1.6	-2.4	-3.5	-4.3	-5.0	-2.1	-6.6	-5.9
$Neval_{max} = 10^5$	0.7	-0.66	-1.6	-2.4	-3.5	-4.3	-5.1	-2.1	-14	-14

TAB. 3.1 – Résultats de `AlgoAlea` (log de la meilleure valeur obtenue moyennée sur 10 essais) pour les fonctions Sphère, Ellipse et Griewank en fonction de `Npt` ($\varepsilon_0 = 1$, $\alpha = 0.9$ et $\beta = 2$). Les valeurs sont données après 10^4 et 10^5 générations.

La situation est encore plus délicate sur les fonctions Griewank modifiées (Table 3.2), qui sont elles aussi mal conditionnées. Mais la pire situation est celle de la fonction GriewankE0.4-0.6 : on tombe facilement dans des minima locaux si on ne prend pas suffisamment de points on tend vers d'autre minima.

Influence de ε_0

Rappelons que le point de départ est choisi après un tirage aléatoire dans tout le domaine de recherche. Le paramètre ε_0 définit alors le domaine de recherche initial, centré sur ce point de départ.

Des essais systématiques ont été effectués en fixant les valeurs `Npt` $= 50$ et $Neval_{max} = 10^5$, et en gardant $\alpha = 0.9$ et $\beta = 2$. ε_0 variait dans $[0.1, 1]$ par pas de 0.1.

Npt	10	20	30	40	50
GriewankE0.4-0.5					
$Neval_{max} = 10^5$	2.3	2.3	2.3	2.3	2.3
$Neval_{max} = 10^5$	2.3	2.3	2.3	2.3	2.3
GriewankE0.4-0.6					
$Neval_{max} = 10^5$	0	0	0	0	-0.0017

TAB. 3.2 – Résultats de `AlgoAlea` (log de la meilleure valeur obtenue moyennée sur 10 essais) pour les fonctions Griewank modifiées, en fonction de `Npt` ($\varepsilon_0 = 1$, $\alpha = 0.9$ et $\beta = 2$). Même pour $Neval_{max} = 10^5$, il n'y a pas convergence sur la plus difficile.

L'influence de ε_0 sur les résultats de l'algorithme `AlgoAlea` semble, au vu des résultats de ces essais, complètement négligeable – ce qui peut s'expliquer par l'influence uniquement "multiplicative" de ε_0, comparées aux influences exponentielles de α et β, voire de `Npt`, paramètres qui interviennent à chaque étape de l'algorithme.

Quelques rares cas de convergence prématurée ont été observées sur la fonction Griewank, pour les très petites valeurs de ε_0.

Enfin, les fonctions Griewank modifiées sont difficiles quel que soit ε_0. Pour `Npt` $= 50$ nous assistons inévitablement à une convergence prématurée de ε vers 0, de manière exponentielle, quel que soit la valeur de départ.

Influence de α et β

La figure 3.2.3 présente les résultats (fitness en échelle log sur l'axe des z après $1e5$ évaluations) obtenus par l'algorithme `AlgoAlea` sur les différentes fonctions tests pour de nombreuses valeurs de $\alpha \in [0.1]$ et $\beta \in [1, 3]$.

Sur les fonctions Sphère, Griewank et Elliptique (figure 3.2.3-a, b et c), on observe très nettement deux situations différentes, selon la valeur de β : Lorsque $\beta < 2$, la convergence est globalement très mauvaise, et est légèrement améliorée pour les valeurs de α proches de 1 : la valeur de β n'est pas suffisamment grande pour rétablir l'équilibre.

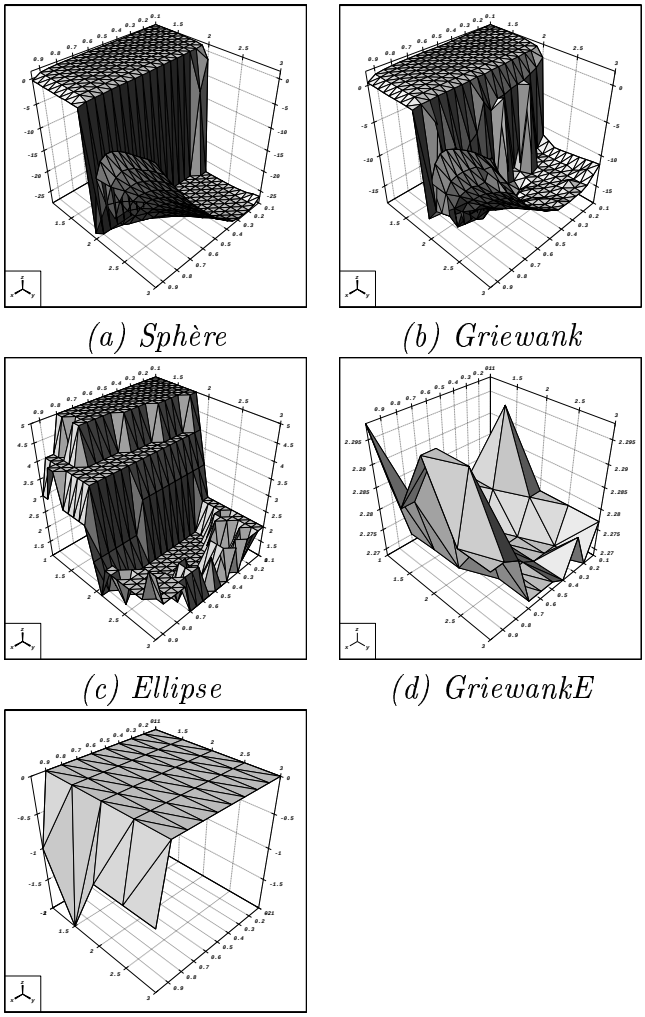
Lorsque $\beta > 2$ par contre, les résultats sont très nettement meilleurs, mais se dégradent pour les valeurs de α proches de 1 – et sont également légèrement détériorés pour les petites valeurs de α. Une valeur optimale surtout visible sur la fonction Griewank semble être autour de 0.3.

Le cas des fonctions Griewank modifiées est en revanche beaucoup moins tranché, et il ne semble pas y avoir d'effet clair des différentes valeurs possibles

de α et β.

Dans le cas de GriewankE0.4-0.6, on peut toutefois noter que le fait de prendre $\alpha = 0.99$ ralentit suffisamment la convergence de ε pour offrir la possibilité de tirer à nouveau des points dans presque dans tout le domaine. Cela revient quasiment à augmenter le nombre de points `Npt` qui sont tirés à chaque itération.

En conclusion, il est clair qu'il faut choisir des valeurs de β supérieures à 2 – pas trop grandes tout de même (tendance à la détérioration pour $\beta = 3$ sur Griewank et Ellipse). Les petites valeurs de α sont préférables en ce qu'elles accélèrent la convergence, mais avec le risque de convergence prématurée pour de trop petites valeurs. Par contre, de trop grandes valeurs peuvent ralentir la recherche suffisamment pour l'arrêter complètement avant le minimum global, même dans le cas de la fonction sphère.



(a) Sphère *(b) Griewank*

(c) Ellipse *(d) GriewankE*

(e) GriewankEbis

FIG. 3.1 – AlgoAlea (dim=100). Influence des paramètres α et β sur la convergence. Sur l'axe des x, à droite, α de 0 à 1 de gauche à droite. Sur l'axe des y, à gauche, β entre 1 et 3 de gauche à droite. Sur l'axe z, le logarithmique de la meilleure fitness après 100000 évaluations avec $\varepsilon_0 = 1$ et $\texttt{Npt} = 50$.

3.3 AlgoDet

3.3.1 Définition

A chaque itération, cet algorithme tire uniformément `Npt` points autour du point courant x_k dans une boule de rayon ε_k, puis applique un algorithme d'optimisation déterministe (ici, BFGS) partant du meilleur des `Npt` points et restreint à la boule $B(x_k, \varepsilon_k)$, et choisit pour x_{k+1} le résultat de cette optimisation. ε_k est ensuite mis à jour comme dans l'algorithme précédent. Plus précisément :

Initialisation :

- Tirer `Npt0` points uniformément sur le domaine de recherche.
- Le meilleur est le point de départ de l'algorithme :
  $x_0 = ArgMax\{f(x_i), i \in [1, \texttt{Npt0}]\}$
- Initialiser les paramètres : $k = 0$, ε_0 donné

Itération :

1. Tirer uniformément `Npt` points autour de x_k :
   $x_k^i = U[B(x_k, \varepsilon_k)], i \in [1, \texttt{Npt}]$
2. Choisir le meilleur (en incluant x_k) :
   $y_k = ArgMax\{f(x_k), f(x_k^i), i \in [1, \texttt{Npt}]\}$
3. Lancer l'algorithme BFGS partant de y_k dans le domaine $B(x_k, \varepsilon_k)$. x_{k+1} est le résultats de cette optimisation.
4. Mettre à jour ε :
   $\varepsilon_{k+1} = \max(\alpha * \varepsilon_k, \beta * d_k)$

où α, β sont des paramètres à déterminer, et oú $d_k = d(x_{k+1}, x_k)$ est la distance euclidienne entre les itérés précédents.

Paramètres de l'algorithme

De même que `AlgoAlea`, l'algorithme dépend donc des paramètres suivants :

- `Npt` : Nombre de points tirés aléatoirement à chaque itération.
- $0 < \alpha < 1$ et $1 < \beta$: paramètres pour l'adaptation du domaine.
- `Npt0` : Nombre de points tirés uniformément initialement sur tout le domaine de recherche. En général, `Npt0` sera pris égal à Npt dans un souci de diminution du nombre de paramètres (quelques expériences préliminaires n'ont pas montré une grande influence de ce paramètre sur les résultats).

– ε_0 : Rayon initial du domaine de recherche pour définir l'itéré suivant.
– Le nombre maximal d'évaluations $Neval_{max}$

Remarque : Cet algorithme demande la calcul du gradient de la fonction à optimiser, et le cadre général de cette thèse suppose que le gradient est a priori non disponible. Il reste toujours les options d'utiliser la dérivation automatique du code, qui peut être délicate pour les gros codes, ou d'utiliser le gradient numérique, avec les instabilités numériques près d'éventuels points critiques, et un coût d'ordre n fois un calcul de la fonction objectif, n étant la dimension de l'espace de recherche. Si la fonction à optimiser résulte de la résolution d'un problème d'EDP, le calcul du gradient peut ne coûter qu'environ 2 fois le coût d'un calcul de la fonction objectif, à condition d'être capable de résoudre le problème adjoint.

3.3.2 Étude expérimentale

Cette section étudie l'influence des paramètres de l'algorithme `AlgoDet` sur son comportement. Les conditions expérimentales sont les même que dans la section précédente (voir sous-section 3.2.3).

Rappelons que l'utilisation de la méthode déterministe BFGS est restreinte à la boule $B(x_k, \varepsilon_k)$ afin de se placer dans des conditions similaires à celles de l'hybridation du chapitre suivant – ceci explique que même sur la fonction sphère, BFGS ne trouve pas le minimum en une seule itération.

Influence de ε_0

Les résultats des tables 3.3 à 3.7 d'une part confirment que une valeur trop petite de ε_0 peut entraîner un ralentissement de convergence (voire un défaut de convergence) de l'algorithme, mais que par contre des valeurs entre 0.5 et 1.0 ne modifient pas sensiblement le comportement de l'algorithme.

Sphère ; Eps	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
$Evals_{fit=0}$	2277	1215	861	861	406	507	507	507	507	507
$Iter$	6	3	2	2	1	1	1	1	1	1

TAB. 3.3 – AlgoDet sur la fonction Sphère. Influence du domaine initial ε_0 sur la convergence). On a `Npt` $= 50$, $\alpha = 0.9$ et $\beta = 2$.

Ellipse ; Eps	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
$Log(fit_{Evals=1e5})$	4.5	-8.4	-7.1	-7.6	-7.9	-6.5	-7.2	-7.2	-7.4	-7.4
$Iter$	4	3	2	2	1	1	1	1	1	1

TAB. 3.4 – AlgoDet sur la fonction Ellipse. Influence du domaine initial ε_0 sur la convergence. On a `Npt` $= 50$, $\alpha = 0.9$ et $\beta = 2$.

Griewank ; Eps	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
$fit_{Evals=4e4}$	0.012	0.64	15.0	1.42	0	0	0	0	0	0
$Eval_{final}$	1e5	9350	850	8200	400	500	500	500	500	500
f_{final}	0.012	0.64	15.0	1.42	0	0	0	0	0	0
$Iter$	16	3	2	2	1	1	1	1	1	1

TAB. 3.5 – AlgoDet sur la fonction Griewank. Influence du domaine initial ε_0 sur la convergence. On a `Npt` $= 50$, $\alpha = 0.9$ et $\beta = 2$.

GriewankE0.4-0.5 ; Eps	0.1	0.3	0.5	0.7	0.9
$Evals$	100019	2076	1064	558	558
fit	9.80283	0	0	0	0
$Iter$	560	8	4	2	2

TAB. 3.6 – AlgoDet sur la fonction GriewankE0.4-0.5. Influence du domaine initial ε_0 sur la convergence. On a `Npt` $= 50$, $\alpha = 0.9$ et $\beta = 2$.

GriewankE0.4-0.6 ; Eps	0.1	0.3	0.5	0.7	0.9
$Log(fit_{Evals=1e5})$	0	0	0	0	-1.63
$Iter$	135	148	163	238	39

TAB. 3.7 – AlgoDet sur la fonction GriewanKam0.4-0.6. Influence du domaine initial ε_0 sur la convergence. On a `Npt` $= 50$, $\alpha = 0.9$ et $\beta = 2$.

Influence de `Npt`

Nous avons maintenant fixé $\varepsilon_0 = 1$ afin de mettre en lumière les performances de l'algorithme déterministe. Les essais effectués avec des valeurs de `Npt` variant de 5 à 50 par pas de 5 n'ont pas montré une influence quelconque

sur la convergence (voir tables 3.8 à 3.12). Il en résulte qu'il est alors plus intéressant de prendre un petit nombre de points puisque cela entraînera un plus petit nombre d'évaluations de la fonction objectif pour le même nombre d'itérations.

Sphere ; Npt	5	10	15	20	25	30	35	40	45	50
$Evals_{fit=0}$	417	427	437	447	457	467	477	487	497	507

TAB. 3.8 – AlgoDet sur la fonction Sphère. Influence de `Npt` sur la convergence. On a $\varepsilon_0 = 1$, $\alpha = 0.9$ et $\beta = 2$.

Ellipse ; Npt	5	10	15	20	25	30	35	40	45	50
$Log(fit_{Evals=1e5})$	-7	-7.8	-7.2	-8.6	-7.2	-7.2	-7.3	-7.9	-7.8	-8.1

TAB. 3.9 – AlgoDet sur la fonction Ellipse. Influence de `Npt` sur la convergence. On a $\varepsilon_0 = 1$, $\alpha = 0.9$ et $\beta = 2$.

Griewank ; Npt	5	10	15	20	25	30	35	40	45	50
$Evals_{fit=0}$	417	427	437	447	457	467	477	487	497	507

TAB. 3.10 – AlgoDet : Influence de `Npt` sur la fonction Griewank(dim=100). $\varepsilon_0 = 1$, $\alpha = 0.9$ et $\beta = 2$.

GriewankE0.4-0.5 ; Npt	10	20	30	40	50
$Evals_{fit=0}$	225	245	265	285	305

TAB. 3.11 – AlgoDet sur la fonction GriewankE0.4-0.5. Influence de `Npt` sur la convergence. On a $\varepsilon_0 = 1$, $\alpha = 0.9$ et $\beta = 2$.

Influence de α et β

Du fait de la convergence en une seule itération de l'algorithme dans le cas oú ε_0 est supérieur à 0.5, les paramètres α et β n'entrent pas en jeu dans

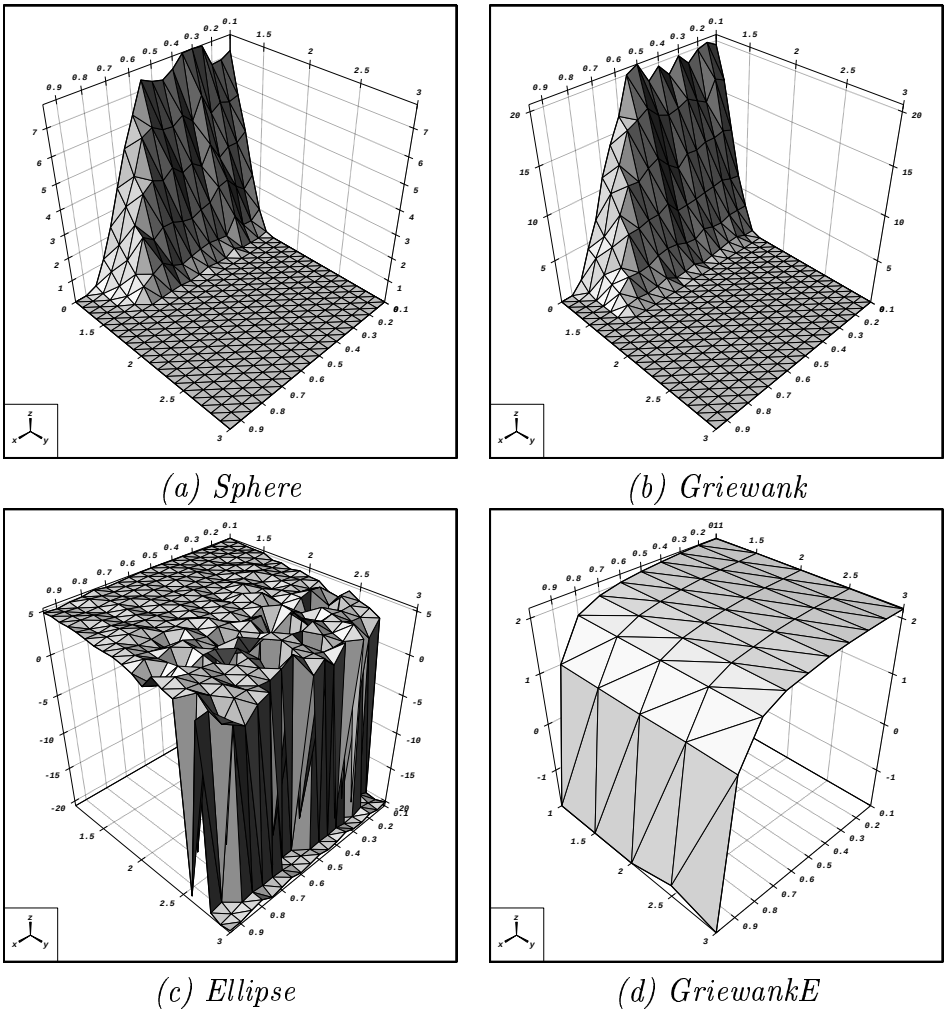
GriewankE0.4-0.6 ; Npt	10	20	30	40	50
$Log(fit_{Evals=1e5})$	-0.7	-2.3	-2.4	-2.4	-2.4

TAB. 3.12 – AlgoDet sur la fonction GriewankE0.4-0.6. Influence de `Npt` sur la convergence. On a $\varepsilon_0 = 1$, $\alpha = 0.9$ et $\beta = 2$.

les résultats. Nous avons donc examiné leur influence uniquement dans le pire cas relevé précédemment, le cas $\varepsilon_0 = 0.1$.

voient bloquées et donc l'algorithme de base opère mieux puisque le nombre d'itérations augmente comme on voit dans le cas de Griewank dans le tableau 3.5 en 16 itérations pour $\varepsilon_0 = 0.1$ au lieu 1 pour des valeurs de $\varepsilon_0 > 0.5$. Dans ce cadre les paramètres jouent sur la convergence de l'algorithme.

Dans le cas de la sphère par exemple, figure 3.3.2-a, on retrouve toujours la limite de 2 pour la valeur de β, et deux situations en ce qui concerne α : dans le cas $\beta < 2$, on a intérêt à prendre des grandes valeurs de α, et si $\beta > 2$, on converge vers 0 quelle que soit la valeur de α – même situation pour la fonction Griewank (figure 3.3.2-b). Dans le cas de l'ellipse 3.3.2-c on retrouve la limite de 2 pour β mais la convergence est meilleure pour les grandes valeurs de β. Dans le cas de la fonction GriewankE (figure 3.3.2-d) la convergence dans des minimaux locaux est trop rapide pour que les paramètres α et β interviennent.



(a) Sphere *(b) Griewank*

(c) Ellipse *(d) GriewankE*

FIG. 3.2 – AlgoDet sur la fonction Sphère (dim=100). Influence des paramètres α et β sur la convergence. L'axe des x c'est le paramètre α. L'axe des y c'est le paramètre β. L'axe z c'est la fonction objective en échelle logarithmique après $1e5$ évaluations. $\varepsilon_0 = 0.1$, $\texttt{Npt} = 50$.

3.4 AlgoApp

Dans cette section nous allons introduire un algorithme d'apprentissage au sein de l'algorithme itératif d'optimisation.

3.4.1 Définition

A chaque itération, cet algorithme tire uniformément `Npt` points autour du point courant x_k dans une boule de rayon ε_k. Il construit ensuite une approximation de la fonction objectif à partir de ces points à l'aide d'une des méthodes étudiées dans le chapitre 3. Cette approximation est ensuite optimisée par un algorithme déterministe (BFGS), et le point x_{k+1} est défini comme soit le meilleur du résultat de cette optimisation et de x_k. Le paramètre ε_k est alors mis à jour comme dans les algorithmes précédents.

Initialisation :
- Tirer `Npt0` points uniformément sur le domaine de recherche.
- Le meilleur est le point de départ de l'algorithme :
  $x_0 = ArgMax\{f(x_i), i \in [1, \texttt{Npt0}]\}$
- Initialiser les paramètres : $k = 0$, ε_0 donné

Itération :

1. Tirer uniformément `Npt` points autour de x_k :
   $x_k^i = U[B(x_k, \varepsilon_k)], i \in [1, \texttt{Npt}]$
2. Construire une approximation $\hat{f}$ de f à partir de ces `Npt` points. Les paramètres de l'algorithme d'apprentissage sont optimisés suivant l'un des critères discutés au chapitre 3.
3. Soit y_k l'optimum de $\hat{f}$ dans $B(x_k, \varepsilon_k)$ (obtenu par la méthode BFGS).
4. $x_{k+1} = ArgMax\{f(x_k), f(y_k)\}$
5. Mettre à jour ε :
   $\varepsilon_{k+1} = \max(\alpha * \varepsilon_k, \beta * d_k)$

où α, β sont des paramètres à déterminer, et oú $d_k = d(x_{k+1}, x_k)$ est la distance euclidienne entre les itérés précédents.

Paramètres de l'algorithme

L'algorithme dépend donc des mêmes paramètres que les deux algorithmes décrits au début de ce chapitre :
- `Npt` : Nombre de points tirés aléatoirement à chaque itération.
- $0 < \alpha < 1$ et $1 < \beta$: paramètres pour l'adaptation du domaine.

- `Npt0` : Nombre de points tirés uniformément initialement sur tout le domaine de recherche. En général, `Npt0` sera pris égal à Npt dans un souci de diminution du nombre de paramètres (quelques expériences préliminaires n'ont pas montré une grande influence de ce paramètre sur les résultats).
- ε_0 : Rayon initial du domaine de recherche pour définir l'itéré suivant.
- Le nombre maximal d'évaluations $Neval_{max}$

Mais l'algorithme dépend de plus des paramètres choisis pour l'algorithme d'apprentissage, ainsi qu'il a été décrit au chapitre 3. En particulier, ces paramètres seront optimisés dynamiquement à chaque itération de `AlgoApp` suivant l'un des critères proposés section 2.2. Nous allons donc commencer par étudier l'influence des différents critères sur la convergence de notre algorithme sur les fonctions tests.

3.4.2 Les paramètres de régularités

Nous allons tout d'abord étudier l'influence des critères sur le choix des paramètres de régularité dans le cadre itératif, dans le cas des méthodes SVM et Krigeage, avant d'examiner les possibilités d'ajustement dynamique des paramètres de régularité eux-mêmes durant le déroulement de l'algorithme.

Choix du critère

Nous allons passer en revue les critères définis au chapitre 3. Il savaient été comparés sous l'angle de l'erreur d'approximation par rapport au nombre de calculs de la vrai fonction qu'ils demandaient dans le cas d'un unique apprentissage. Nous allons ici affiner ces comparaisons dans le cadre itératif. Comme dans le chapitre 3, les méthodes d'apprentissage seront les SVM et la méthode de Krigeage. La figure 3.3 montre une situation typique de comparaison des critères encore en lisse pour les deux méthodes, sur la fonction sphère.

- **SVM** : On a vu dans le chapitre précèdent que le critère de validation (CV) est à éviter, puisqu'il ne permet pas de trouver de bonne approximation sur la fonction sphère (voir figures 2.12). Deux critères restent donc en course, dans le cas d'une fonction d'évaluation coûteuse, le critère d'erreur (CE) et le critère objectif (CO). Dans le cadre itératif qui nous occupe, les résultats sur la sphère montrent que le coût supplémentaire du critère (CO) n'est pas compensé par une plus grande rapidité de convergence, et que le critère (CE) reste plus intéressant.
- **KRIG** : Les tests effectués dans le chapitre précèdent ont montré que le critère de généralisation (CG) obtient de bons résultats dans le cas

de petite dimensions sur la sphère, mais ne permet pas de traiter le cas des grandes dimensions (> 30). Par contre, contrairement à SVM, le critère de validation (CV) a donner des résultats comparables au critère objectif (CO) – et ce sont donc les critères (CO) et (CV) qui sont comparés ici. Le résultat est alors clair sur la figure 3.3 : le critère (CV) donne les meilleurs résultats. Toutefois, on notera qu'aucun des deux critères ne permet à la méthode de Krigeage d'atteindre des précisions importantes sur les résultats (nous sommes en dimension 100).

La méthode SVM a donné de loin de meilleur résultats que la méthode Krigeage en dimension 100. C'est pourquoi dans la suite de ce chapitre nous n'étudierons plus que la méthode SVM pour des problèmes en dimension 100.

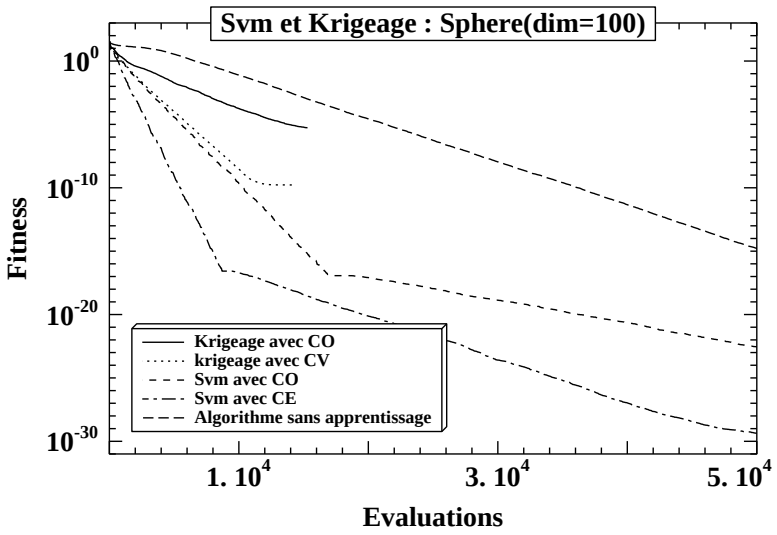


FIG. 3.3 – Influence des différents critères avec SVM et KRIGEAGE sur la convergence de l'algorithme sur la fonction sphère

Paramètres dynamique

Ne perdons pas de vue que notre problème n'est pas d'optimiser les paramètres de l'apprentissage *per se*, mais la fonction objectif de départ.

Rappelons qu'à chaque itération, on essaie de trouver avec un algorithme de type ES les paramètres de la méthode d'apprentissage qui optimisent le critère retenu. L'initialisation de l'algorithme ES, habituellement uniforme sur le domaine de recherche, pourrait ici utiliser la population finale trouvée

dans la recherche à l'étape précédente – nous parlerons alors d'une initialisation dynamique, par opposition à l'initialisation aléatoire usuelle.

La figure 3.4.2 compare, sur la fonction sphère, les deux types d'initialisation, et montre l'influence du nombre d'évaluations autorisés pour cette étape, de 6 à 30 par pas de 6. Les deux figures du haut concernent le critère (CE) et les deux du bas le critère (CO). On remarque tout de suite le peu d'influence du nombre d'évaluations autorisées, sauf dans le cas (CE) dynamique, pour lequel les résultats sont très instables. Mais souvenons-nous qu'à l'issue du chapitre 3 nous avions donné des bornes pour les paramètres de régularité déjà restreintes et dans lesquelles l'apprentissage semblait assez robuste : les résultats de la figure 3.4.2 confirme cela, en ce sens que très peu d'évaluations sont nécessaires. De plus, comme 6 évaluations ne permettent pas vraiment de distinguer une optimisation évolutionnaire d'une recherche aléatoire, la conclusion de ces résultats semble être qu'il est finalement préférable d'utiliser des paramètres constants.

Mais ces résultats ont été obtenus sur ls fonction sphère, de "régularité constante". En comparant maintenant sur les autres fonctions test les trois approches (paramètres constants, initialisation aléatoire, initialisation dynamique) avec un algorithme sans apprentissage (AlgoAlea) (voir figure 3.4.2), on remarque que l'optimisation des paramètres de régularité de SVM ne semble pas intéressant pour les fonctions sphère et Griewank (sans toutefois détériorer les performances !), mais améliore légèrement les performances de AlgpApp sur la fonction ellipse, et de manière spectaculaire sur la fonction Griewank modifiée. Dans le cadre d'une fonction aux caractéristiques inconnues, il semble donc raisonnable de conserver cette étape et d'optimiser de manière dynamique les paramètres de la méthode SVM avec comme critère le critère (CO).

3.4.3 Les paramètres de l'algorithme

Nous étudions maintenant l'influence des autres paramètres de l'algorithme AlgoApp, d'une manière similaire à celle utilisée pour les algorithmes AlgoAlea et AlgoDet.

Influence de `Npt`

L'algorithme utilise à chaque itération une méthode d'apprentissage, et il est légitime de penser que, plus on utilise de points pour l'approximation et meilleure sera la convergence. Effectivement, c'est ce que nous pouvons constater sur les résultats donnés dans les tables 3.13 à 3.17.

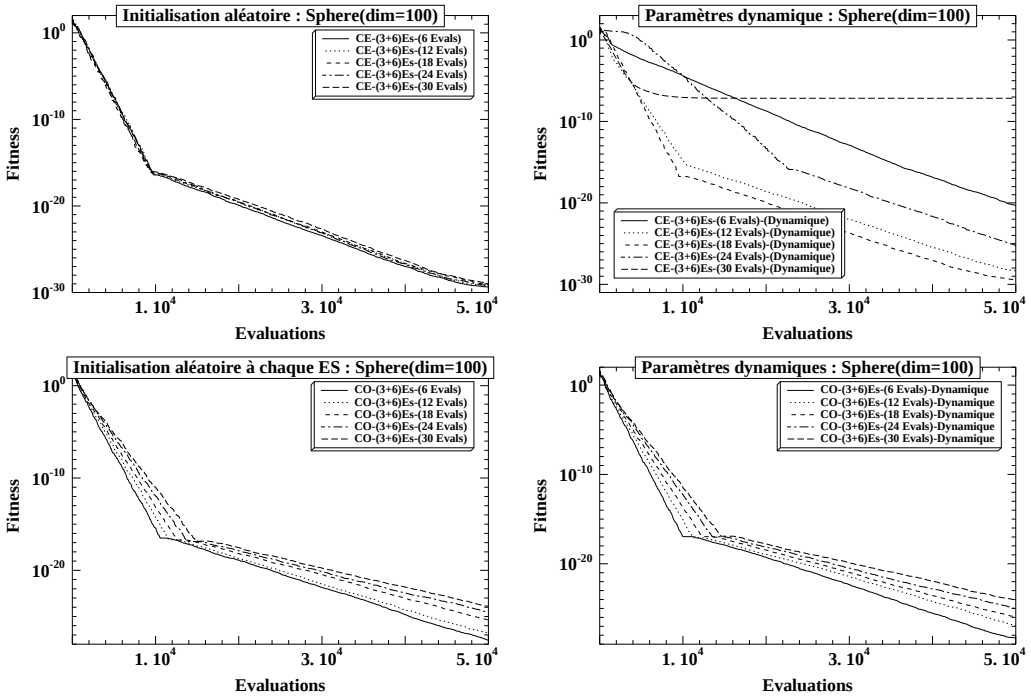


FIG. 3.4 – Influence sur la convergence, du nombre d'évaluation maximal du critère (CX) dans le cas d'une initialisation dynamique de la population dans le sous -ES

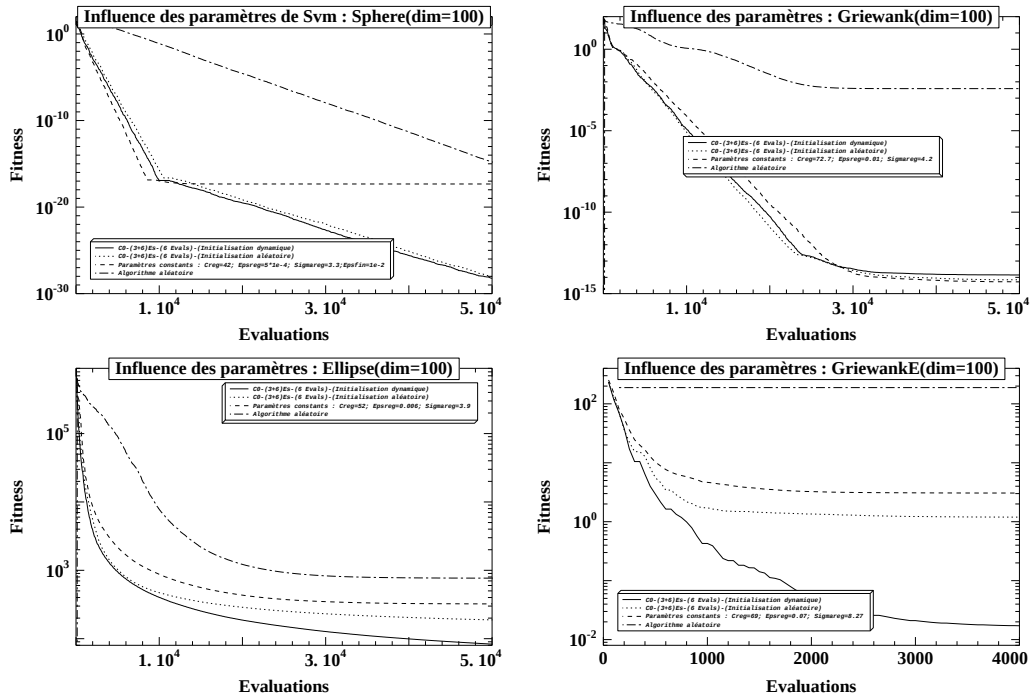


FIG. 3.5 – Comparaison entre une initialisation aléatoire ou dynamique de la population dans le sous -ES sur (CO), des paramètres constants et l'algorithme aléatoire pur sur les fonctions sphère, Griewank, Ellipse et GriewankE

Sur la fonction sphère, au delà de 20 points, les résultats ne s'améliorent plus. Par contre, sur les autres fonctions (sauf GriewankE0.4-0.6 pour laquelle le nombre de points n'a aucune influence, l'algorithme reste bloqué au même endroit), les performances augmentent avec le nombre de points.

Log(f) Npt	10	20	30	40	50
$Log(fit_{Evals=40000})$	-6.9	-14	-16	-17	-16
$Log(fit_{Evals=100000})$	-17	-28	-29	-29	-28

TAB. 3.13 – AlgoApp sur la fonction Sphère. Influence de `Npt` sur la convergence. On a $\varepsilon_0 = 1$, $\alpha = 0.9$ et $\beta = 2$.

Log(f) Npt	10	20	30	40	50
$Log(fit_{Evals=4000})$	4.4	3.9	3.8	3.7	3.7
$Log(fit_{Evals=100000})$	2.9	2.5	2.3	2.1	1.6

TAB. 3.14 – AlgoApp sur la fonction Ellipse. Influence de `Npt` sur la convergence. On a $\varepsilon_0 = 1$, $\alpha = 0.9$ et $\beta = 2$.

Log(f) Npt	10	20	30	40	50
$Log(fit_{Evals=40000})$	-2.1	-8	-10	-12	-12
$Log(fit_{Evals=100000})$	-2.1	-8.3	-11	-13	-14

TAB. 3.15 – AlgoApp sur la fonction Griewank. Influence de `Npt` sur la convergence. On a $\varepsilon_0 = 1$, $\alpha = 0.9$ et $\beta = 2$.

Log(f) Npt	10	20	30	40	50
$Log(fit_{Evals=1000})$	1.8	1.3	0.45	-0.65	-6.4
$Log(fit_{Evals=100000})$	1.8	1.3	0.45	-0.65	-7.6

TAB. 3.16 – AlgoApp sur la fonction GriewankE0.4-0.5. Influence de `Npt` sur la convergence. On a $\varepsilon_0 = 1$, $\alpha = 0.9$ et $\beta = 2$.

Log(f) Npt	10	20	30	40	50
$Log(fit_{Evals=100000})$	0	0	0	0	0

TAB. 3.17 – AlgoApp sur la fonction GriewankE0.4-0.6. Influence de `Npt` sur la convergence. On a $\varepsilon_0 = 1$, $\alpha = 0.9$ et $\beta = 2$.

Influence de ε_0

L'influence de ε_0 est ici très faible, contrairement au cas de l'algorithme AlgoDet. Elle est même carrément inexistante pour la fonction sphère (table 3.18), et très peu visibles sur les fonctions Ellipse (table 3.19) et Griewank (table 3.19). Par contre, et contrairement à l'algorithme AlgoDet, on constate sur la fonction GriewankE0.4-0.6 une détérioration des performances pour les petites valeurs de ε_0, pour lesquelles l'apprentissage ne capture que les tendances locales de la fonction – alors que les valeurs plus grandes lui permettent d'éviter une partie de ces optima locaux en approximant la fonction à un niveau de granularité plus important. Toutefois, ce mécanisme ne suffit plus si l'on rajoute un mauvais conditionnement (fonction Griewank modifiée, table 3.7.

Log(f) Eps	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
$Log(fit_{Evals=20000})$	-17	-17	-16	-16	-16	-17	-17	-17	-17	-17
$Log(fit_{Evals=100000})$	-28	-28	-28	-28	-28	-28	-28	-28	-28	-28

TAB. 3.18 – AlgoApp sur la fonction Sphère. Influence du domaine initial ε_0 sur la convergence. On a `Npt` $= 50$, $\alpha = 0.9$ et $\beta = 2$.

Log(f) Eps	0.1	0.3	0.5	0.7	0.9
$Log(fit_{Evals=100})$	3.7	3.7	3.7	3.7	3.6
$Log(fit_{Evals=100000})$	2.3	2.2	2.1	1.6	1.9

TAB. 3.19 – AlgoApp sur la fonction Ellipse. Influence du domaine initial ε_0 sur la convergence. On a `Npt` $= 50$, $\alpha = 0.9$ et $\beta = 2$..

Log(f) Eps	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
$Log(fit_{Evals=40000})$	-12	-12	-12	-12	-12	-12	-11	-11	-12	-11
$Log(fit_{Evals=100000})$	-14	-14	-14	-12	-14	-12	-14	-14	-14	-9.8

TAB. 3.20 – AlgoApp sur la fonction Griewank. Influence du domaine initial ε_0 sur la convergence. On a `Npt` $= 50$, $\alpha = 0.9$ et $\beta = 2$.

Log(f) Eps	0.1	0.3	0.5	0.7	0.9
$Log(fit_{Evals=500})$	2.2	1.9	1.1	0.26	0.37
$Log(fit_{Evals=40000})$	1.8	-0.26	-4.6	-1.4	-1.1
$Log(fit_{Evals=100000})$	1.8	-0.26	-4.6	-1.4	-1.1

TAB. 3.21 – AlgoApp sur la fonction GriewankE0.4-0.5. Influence du domaine initial ε_0 sur la convergence. On a `Npt` $= 50$, $\alpha = 0.9$ et $\beta = 2$.

Log(f) Eps	0.1	0.3	0.5	0.7	0.9
$Log(fit_{Evals=300})$	0	0	0	0	0
$Log(fit_{Evals=100000})$	0	0	0	0	0

TAB. 3.22 – AlgoApp sur la fonction GriewankE0.4-0.6. Influence du domaine initial ε_0 sur la convergence. On a `Npt` $= 50$, $\alpha = 0.9$ et $\beta = 2$.

Influence de α et β

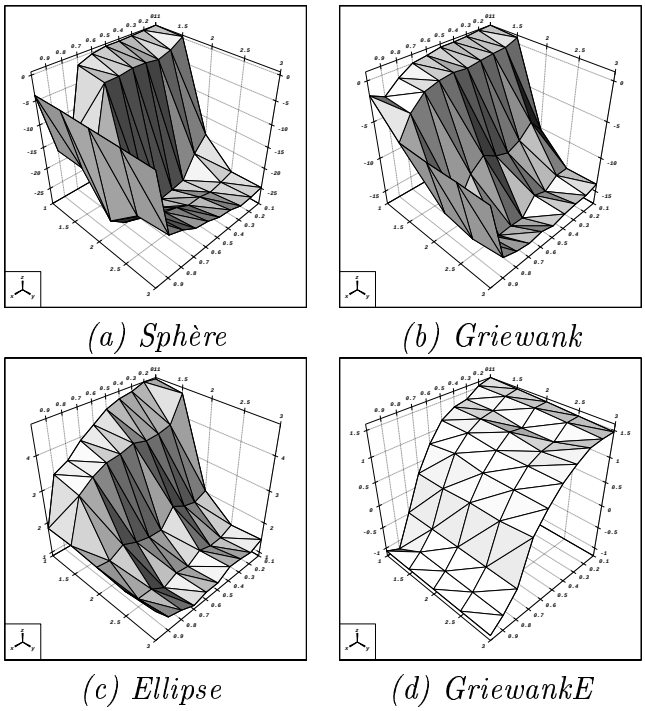
C'est maintenant que l'apprentissage entre en jeu que nous allons pouvoir effectivement constater l'influence de la loi de variation de ε. L'idée est qu'on voudrait diminuer le domaine d'approximation si l'approximation n'est pas bonne et l'augmenter si l'approximation est satisfaisante.

Si l'approximation est bonne, et ceci est matérialisé dans notre contexte de manière très pragmatique par le fait que l'on améliore la valeur de la fonction objectif, alors le déplacement effectué, multiplié par β, devrait augmenter le domaine d'approximation pour l'itération suivante. Au contraire, si le déplacement est nul (ou petit), le rayon du domaine va diminuer. Les

paramètres α et β jouent un rôle important dans l'adaptation du domaine vers un domaine "efficace".

Les résultats montrent que tout se passe effectivement plus ou moins comme prévu (figure 3.4.3), et ce manière similaire avec la situation pour les algorithmes précédents de ce chapitre :

En premier lieu, on retrouve sur la sphère la valeur charnière de $\beta = 2$, et le contrôle de la vitesse de convergence par α dans le cas $\beta > 2$. De même, le cas de Griewank suit encore à peu près celui de la sphère. Enfin, le cas de l'ellipse montre que les trop petites valeurs de α empêchent une bonne convergence, phénomène encore accentué sur la fonction GriewankE, pour laquelle on ne retrouve même pas la valeur critique $\beta = 2$, mais pour laquelle les meilleurs résultats sont obtenus pour les plus grandes valeurs de α.



(a) Sphère *(b) Griewank*

(c) Ellipse *(d) GriewankE*

FIG. 3.6 – AlgoApp sur la fonction Sphère (dim=100). Influence des paramètres α et β sur la convergence. L'axe des x c'est le paramètre α. L'axe des y c'est le paramètre β. L'axe z c'est la fonction objective en échelle logarithmique après $1e5$ évaluations. $\varepsilon_0 = 1$, $\texttt{Npt} = 50$.

3.5 AlgoAppBis : Variante d'AlgoApp

3.5.1 Définition

L'algorithme AlgoAppBis est défini de manière analogue à l'algorithme AlgoApp (section 3.4), à la différence près que c'est un algorithme de type SA-ES non-isotrope qui est appelé pour l'optimisation du modèle $\hat{f}$ en lieu et place de l'algorithme L-BFGS-B. De plus, l'adaptation du domaine ne se fait plus avec la loi α-β. On prend comme domaine le plus petit hypercube qui contient la population et les enfants à la génération courante.

Quatres questions supplémentaires se posent alors :

1. Comment initialiser la population (et les écart-types associés) de l'algorithme SA-ES qui optimise le modèle appris ?
2. Comment régler les conditions d'arrêts de l'algorithme SA-ES ?
3. Quels sont les points qu'il est nécessaire d'évaluer avec la vraie fonction objectif ?
4. Dans AlgoApp nous avons adapté les hyper-paramètres du modèle avec le critère (CO). Le modèle optimal selon (CO) est-t-il toujours optimal pour la recherche globale avec SA-ES ? Comment règle-t-on les paramètres ?

Il a finalement été décidé d'utiliser la population finale de l'algorithme SA-ES ayant servi à optimiser $\hat{f}_{k-1}$ comme population initiale pour l'optimisation de $\hat{f}_k$ – après réévaluation par $\hat{f}_k$ bien entendu – avec les valeurs de σ_i associées. Après quelques tests préalable, nous avons décidé de choisir 10 générations à chaque phase d'optimisation avec SA-ES. La population initiale pour l'optimisation de $\hat{f}_0$ est, elle, tirée au hasard dans $B(x_0, \varepsilon_0)$, les σ_i étant pris, comme recommandé par Schwefel, égaux à $0.3 * \varepsilon_0$.

3.5.2 Les paramètres de régularité

Le remplacement de L-BFGS-B par un SA-ES pour l'optimisation de $\hat{f}$ n'entraîne ainsi pas de paramètre supplémentaire, et est susceptible d'apporter un élément de recherche plus globale lors de l'optimisation du modèle appris.

Cependant, les tendances constatées sur l'algorithme AlgoApp ont rapidement semblées respectée pour AlgoAppBis, et nous ne présenterons pas tous les résultats en détail ici.

Le seul point remarquable concerne le choix des paramètres de la partie apprentissage. En effet rien ne garantissait que le modèle défini à travers les paramètres optimaux du critère (CO) soit adapté pour une recherche globale

avec SA-ES. Nous avons toujours l'instabilité dans la convergence sur la sphère par rapport au critère (CE) et une certaine stabilité par rapport au critère (CO).

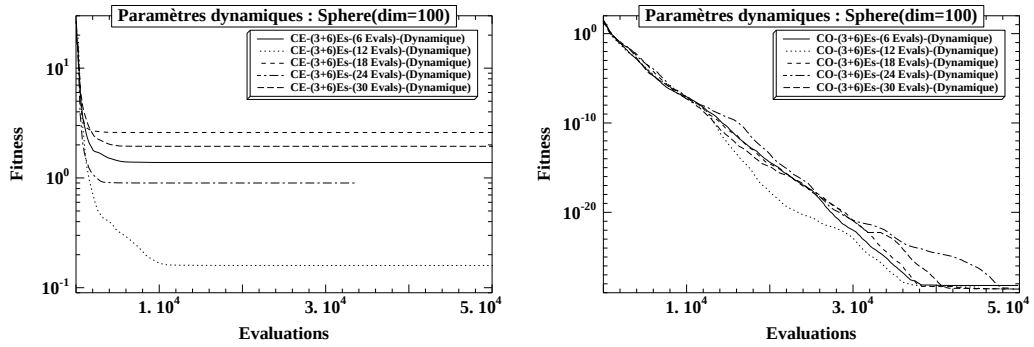


FIG. 3.7 – Influence sur la convergence, du nombre d'évaluation maximal du critère (CX) dans le cas d'une initialisation dynamique de la population dans le sous-ES

3.6 Conclusion

Nous terminons ce chapitre par une comparaison globale de tous les algorithmes que nous avons définis et étudiés. La figure 3.6 montre les performances médianes de ces quatre algorithmes, utilisant chacun les valeurs des paramètres considérées comme donnant les meilleurs résultats sur la foi des études expérimentales présentées plus haut. Le grand vainqueur de cette comparaison est sans conteste l'algorithme AlgoApp.

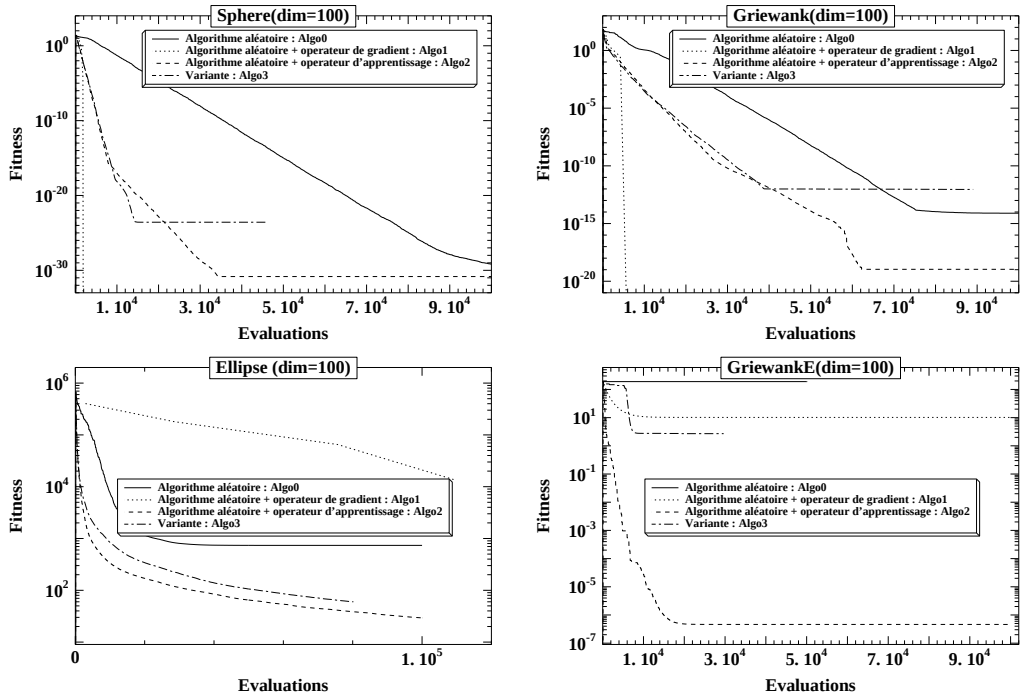


FIG. 3.8 – Comparaison des différents algorithmes itératifs considérés dans le chapitre 4. Chaque courbe représente la médiane sur 10 lancements. Dans le cas du gradient on initialise $Eps0 = 0.1$ pour inhiber la recherche linéaire.

Faisons maintenant quelques commentaires résumant les contributions de ce chapitre :

- L'approche itérative de l'apprentissage est validée. En effet l'amélioration apportée dans la convergence locale par rapport a l'algorithme aléatoire est significative. De plus, par rapport à l'algorithme déterministe, l'apprentissage permet une recherche plus globale en évitant quelques-uns des minima locaux.
- La méthode SVM semble la seule apte à travailler en grandes dimensions. Nous continuerons à étudier le Krigeage pour les petites dimen-

sions, mais nous concentrerons sur la méthode SVM dans le prochain chapitre.

- Le choix des paramètres des méthodes d'apprentissage dynamique permet d'améliorer la convergence. En effet, ceci a permis d'utiliser le critère CO tout en diminuant son coût, qui est essentiellement dû à une régression et une évaluation de la (vraie) fonction objectif.
  Point négatif : si les paramètres des mutations gaussienne de l'algorithme ES utilisé pour optimiser le critère (CO) et déterminer les paramètres de l'apprentissage convergent vers zéro alors l'adaptation dynamique de ces paramètres devient problématique. Il faudrait sans doute changer la borne inférieure de ces paramètres.
- Le choix du domaine d'approximation, qui est défini à travers le paramètre ε, est crucial, et semble directement lié à la difficulté de la fonction. La double loi de mise à jour de ε au travers des paramètres α et β nous semble à même de capturer les différents types de paysages à condition de prendre $\beta > 2$, et α assez petit.
- Dans le cas de la Sphère nous relevons un problème numérique dans la convergence locale. Il est visible dans le premier algorithme itératif et l'est un peu moins dans la variante. Cela suggère que le problème est d'ordre numérique. Il est probable que le problème pourra être éviter avec des normalisations adéquates !
- La variante AlgoAppBis évite les problèmes numérique rencontrer sur la sphère avec AlgoApp. En effet dans la partie optimisation du modèle le fait de choisir 10 générations uniquement ammène à éviter les minimas locaux artificiels en optimisant partiellement la fonction modèle. Sur les autres fonctions AlgoAppBis ne parvient pas à atteindre des bonnes précisions et est dépassée de manière consistante par AlgoApp. Cela provient peut-être du fait que les enfants générés au sein de l'algorithme SA-ES qui optimise $\hat{f}$ le sont par des mutations gaussiennes d'écart-type σ_i auto-adaptatifs qui ne prennent en particulier pas directement en compte la régularité constatée de la fonction objectif (au contraire du mécanisme d'adaptation de ε). Ainsi la convergence de ε suit la convergence des σ_i qui convergent rapidement vers zéro, entraînant une convergence prématurée dans la plupart des cas.
- D'après les résultats du chapitre précèdent, il semble clair que le nombre de points `Npt` à une influence. Cependant, au sein d'un algorithme itératif, un trop grand nombre de point pourrait vite devenir très coûteux par rapport au gain espéré. Comment évaluer le gain apporté ? Le chapitre suivant aura pour but de répondre à ces questions.

L'algorithme itératif utilisant l'apprentissage AlgoApp pourrait être vu

comme une hybridation de l'apprentissage avec l'algorithme aléatoire AlgoAlea. Cependant, ce dernier n'est pas "suffisamment global" pour que cette hybridation permette d'obtenir des résultats intéressants. Et même la convergence locale pourrait être améliorée. Ce sont ces deux points qui nous ont motivés pour hybrider l'apprentissage avec un algorithme réellement global qui soit de plus muni de bonnes propriétés de convergence local, tel l'algorithme ES. L'hybridation s'effectuera à travers un opérateur qu'on appellera SDM et qui sera présenté dans le chapitre suivant. Nous présenterons une autre approche qui est MAES que nous l'adaptons à la méthode SVM et dans le cas des fonctions à grande dimension.

Chapitre 4

Surrogate Deterministic Mutation(SDM) & Metamodel-Assisted Evolution Strategies(MAES)

4.1 Introduction

Nous avons observé dans le chapitre précèdent que l'approche itérative de l'apprentissage (AlgoApp) accélère la convergence locale dans le cas d'une fonction convexe (comme la sphère) par rapport à l'algorithme ES.

Mais AlgoApp a un comportement principalement déterministe. Or, la convergence dans le cas des fonctions multimodales n'est pas toujours assurée avec les algorithmes déterministes. Dans ces cas, ES, bien qu'il converge lentement localement, possède un avantage par rapport à AlgoApp grâce à ses propriétés globales.

D'où l'idée d'hybrider de pousser plus loin l'hybridation en hybridant l'apprentissage avec les Stratégies d'Évolution : profiter des propriétés constatées sur AlgoApp pour la convergence locale et de celles de ES pour la recherche globale. Pour cela, nous allons introduire un opérateur de mutation que l'on va appeler **SDM** (pour *Surrogate Deterministic Mutation*). Chaque mutation correspondra à une itération élémentaire de l'algorithme AlgoApp. Contrairement à AlgoApp qui lui est caractérisé par la densité de probabilité uniforme dans un rayon ε adaptatif, le choix des points de SDM est effectué selon une densité de probabilité plus complexe, en particulier, le paramètre ε devient propre à l'opérateur SDM et la mise à jour s'effectue à chaque mutation. Mais l'adaptation garde le même esprit que dans AlgoApp.

Plusieurs points nécessiteront notre attention :

1. Avec quelle probabilité, et sur quel(s) individu(s) devons nous appliquer l'opérateur de mutation SDM ?
2. Dans l'algorithme itératif du chapitre précèdent, les points étaient tirées uniformément dans le domaine D_ε centrés autour de l'itéré courant et de rayon ε. Dans SDM, les points sont tirés autant que possible selon la loi de probabilité gaussienne utilisée au sein de ES. Il faudra vérifier que les règles d'adaptation du domaine de recherche local au travers des paramètres α et β sont encore valables.
3. Si nous décidons de rechercher localement à travers l'opérateur de mutation, nous devons alors envisager d'avoir à créer dans certains cas de nouveaux points pour améliorer l'apprentissage localement. Comment tirer ces points ?
4. L'hybridation se fera avec un ES isotopique. Quelle valeur donner au σ de l'individu trouvé par apprentissage ? Quelle stratégie adopterons-nous avec le muté ? Devons-nous l'ajouter à la population ou le substituer au meilleur muté ?
5. L'application de l'opérateur SDM ne garantit pas de trouver un individu meilleur à chaque essai de mutation. L'amélioration effectuée dans une recherche locale peut aussi devenir insignifiante. Sachant que l'application de l'opérateur SDM exige des évaluations supplémentaires, nous devons poser une condition d'arrêt à l'application de l'opérateur SDM localement, et, plus généralement, décider d'une réinitialisassions de l'opérateur. Quelle stratégie choisir pour réinitialiser et/ou arrêter l'opérateur ?
6. A l'arrêt de l'opérateur, l'algorithme redevient un algorithme ES classique, “initialisé” avec la population finale trouvée par l'algorithme hybride ES-SDM. Que doit vérifier cette population pour continuer la recherche efficacement ?

L'approche choisie dans l'hybridation de SDM avec ES correspond aussi à une approche proche de celle dite des "Informed Operators" (voir le chapitre 2). Nous validerons notre approche par une comparaison avec une autre approche, MAES (voir également le chapitre 2).
L'approche **MAES** [40], consiste à remplacer complètement la fonction objectif f par un modèle $\hat{f}$ appris à l'aide de la méthode de Krigeage, sauf pour un certain nombre de points par génération, choisis selon le critère de la fonction de mérite. La méthode MAES entre dans le cadre du “contrôle par évaluations” (voir chapitre 2 section 1.6.3). Pour compléter notre approche,

nous envisagerons enfin une approche dynamique, basé sur le "Contrôle par Générations" (voir chapitre 2, section 1.6.3).

Mais quelle que soit la méthode utilisée, nous devrons faire face à des problèmes numériques lors de l'application de la méthode d'apprentissage choisie. Ce sera surtout vrai pour le méthode de Krigeage, qui est au départ une méthode d'interpolation et non d'approximation. Les points d'apprentissage peuvent, dans certains cas, se retrouver confinés dans une petite région de l'espace de recherche. Ceci peut alors poser des problèmes lors de l'inversion (ou la pseudo-inversion) de la matrice de covariance, qui prend en compte les distances relatives. Ces matrices non inversibles, ou très mal conditionnées, peuvent mener à des résultats instables, non convergents. Ce problème apparaîtra surtout avec la méthode de Krigeage. Un autre avantage de la méthode SVM est la possibilité d'obtenir, à l'aide d'un choix raisonné des paramètres de la méthode, des "modèles intermédiaires"(c'est-à-dire ne passant pas forcément par les points d'apprentissage).

Le chapitre est organisé de la sorte : En premier lieu, nous décrirons en détail l'opérateur SDM et nous discuterons des problématiques de l'hybridation avec ES. Ensuite, nous exposerons en détail MAES, et nous proposons une fonction mérite dans le cas de la méthode d'approximation SVM (seul le Krigeage a été envisagé sous cet aspect dans la littérature). Nous comparerons enfin les différentes approches sur les quatre types de problèmes test que nous avons déjà utilisé jusqu'à présent : Sphère, Griewank, Ellipse et GriewankE.

4.2 Surrogate Deterministic Mutation (SDM)

L'opérateur de mutation **SDM** agit au sein d'un algorithme ES classique, et est basé sur une méthode d'apprentissage combinée avec une méthode d'optimisation classique, dans la lignée de ce qui a été proposé au chapitre 4 pour définir l'algorithme AlgoApp. La méthode d'apprentissage utilisée, SVM ou Krigeage, va exploiter des points déjà évalués avec la "vraie" fonction objectif.

4.2.1 L'algorithme

On dispose donc d'un algorithme de type Stratégie d'Évolution auto-adaptative isotrope, c'est-à-dire dans lequel chaque individu a un paramètre de mutation qui lui est propre noté σ. La matrice de covariance de la mutation est alors σI_n (voir chapitre 2 section 1.4).

Initialisation

L'archive $\mathcal{A}_0$ de points déjà évalués est initialisée avec l'ensemble de la population. On initialise les paramètres k_{std} et ε_0.

L'opérateur de mutation

À la génération k, on applique l'opérateur SDM une fois, de la manière suivante :

1. Mettre à jour ε :
$$\varepsilon_k := \max(\alpha * \varepsilon_{k-1}, \beta * d_k)$$
où α, β sont des paramètres, et oú $d_k = d(x_{k-1}, x_k)$ est la distance euclidienne entre les **meilleurs individus successifs** de la population lors des étapes précédentes de l'algorithme.
2. Soit x_k le meilleur individu de la population courante ;
3. Récupérer tous les individus déjà évalués se trouvant dans une boule de centre x_k et de rayon ε_k :
$$(x^1, x^2, \ldots, x^l) = A_k \bigcap B(x_k, \varepsilon_k)$$
4. Compléter éventuellement par des points tirés uniformément dans $B(x_k, \varepsilon_k)$ jusqu'à disposer de `Npt`points dans cette boule. Évaluer les nouveaux points à l'aide de la "vraie" fonction objectif :
$$x^i = U[B(x_k, \varepsilon_k)], i \in [l+1, \texttt{Npt}]$$

5. Construire une approximation $\hat{f}$ de f à partir de ces `Npt` points $x^1, \ldots, x^{Npt}$. Les paramètres de l'algorithme d'apprentissage sont optimisés suivant la procédure mise au point et discutée au chapitre 4.
6. Soit y_k l'optimum de $\hat{f}$ dans $B(x_k, \varepsilon_k)$ (obtenu par la méthode L-BFGS-B).
7. $\hat{x}_{k+1} = ArgMax\{f(x_k), f(y_k)\}$ est l'enfant résultat de la mutation SDM.

Par rapport à l'algorithme AlgoApp, les étapes ci-dessus présentent quelques différences, dues à la présence de l'algorithme ES sous-jacent, qui appellent quelques commentaires :

- Le choix de n'appliquer l'opérateur SDM qu'une fois par génération, et au meilleur individu de la population uniquement peut sembler arbitraire, et pourrait nuire au bon déroulement de la recherche globale en cas de début de convergence dans un optimum local. Mais nous verrons que les paramètres de l'algorithme peuvent être ajustés pour minimiser ce risque ;
- La mise à jour de ε_k se fait de manière analogue à l'étape correspondante de AlgoApp, mais elle met en jeu maintenant la distance "parcourue" par l'algorithme complet lors de sa dernière itération, et non plus simplement le gain apporté par le couplage apprentissage – optimisation déterministe. C'est le retour d'information de l'algorithme global vers SDM ;
- Dans AlgoApp, tous les points nécessaires étaient générés uniformément dans $B(x_k, \varepsilon_k)$. Ici, les points déjà évalués vont être autant que faire se peut utilisés afin de minimiser le nombre total d'évaluations de la "vraie" fonction objectif.

Reprise de l'algorithme ES

Avant que l'algorithme ES n'effectue une nouvelle itération, le point $\hat{x}_{k+1}$ doit être remis dans la population courante s'il est différent du parent x_k. Deux stratégies vont être étudiées :

- SDM-Plus, dans laquelle $\hat{x}_{k+1}$ est ajouté à la population courante ;
- SDM-Comma dans laquelle $\hat{x}_{k+1}$ remplace le parent x_k.

Lorsque $\hat{x}_{k+1}$ est réintroduit dans la population (et quelle que soit la stratégie de remplacement retenue), il faut lui donner un écart-type propre σ_k (l'ES utilisé est un ES isotrope). Il a été choisi de prendre un écart-type proportionnel à ε_k afin d'avoir un retour d'information de SDM vers l'algorithme complet, contrepartie de l'utilisation de la distance entre deux

meilleurs successifs de l'algorithme complet dans l'ajustement de ε. Ainsi nous choisirons

$$\sigma_k = k_{std}\varepsilon_k \tag{4.1}$$

Nous étudierons dans la section suivante l'influence de k_{std} sur les performances de l'algorithme complet.

Paramètres de SDM

Les paramètres mis en jeu par l'opérateur SDM décrit ci-dessus comprennent les paramètres analogues à ceux de AlgoApp plus les paramètres spécifiques à l'hybridation de SDM avec ES :

- `Npt`le nombre de points minimal dont il faut disposer pour lancer l'apprentissage
- $0 < \alpha < 1$ et $1 < \beta$: paramètres pour l'adaptation du domaine.
- ε_0 : Rayon initial du domaine de recherche pour choisir les points d'apprentissage autour du meilleur individu ;
- k_{std} : facteur de proportionnalité entre l'écart-type propre de l'enfant résultant de la mutation et le paramètre ε de SDM.

4.2.2 Influence des paramètres `Npt`, α et β

Le chapitre 4 avait étudié en détail, dans des situations de plus en plus proches de celle de SDM, l'influence des paramètres `Npt`(nombre de point requis pour lancer l'apprentissage), α et β (qui règlent la mise à jour de la taille du domaine de recherche local ε).

Il était à prévoir que l'influence de `Npt`dans le cadre de SDM serait identique à ce qui a été constaté dans le cadre de l'algorithme AlgoApp (voir section 3.4), et c'est effectivement le cas : un nombre de points de 50 semble tout à fait satisfaisant,du moins jusqu'à la dimension 100.

Il s'est avéré également que le réglage optimal des paramètres α et β pour SDM est comparable à celui mis au point sur AlgoApp, et ce malgré la différence de mode d'échantillonage. Cependant, l'examen des figures 4.2.3 , 4.2.3, 4.2.3, 4.2.3 discutée dans la section suivante, permet de proposer une première explication : de fait, l'utilisation de SDM entraîne un échantillonnage supplémentaire de points autour de x_k lorsque l'archive des points déjà évalués ne contient pas suffisamment de points dans $B(x_k, \varepsilon)$ – et ces points sont tirés uniformément dans $B(x_k, \varepsilon)$. D'autre part, en grande dimensions, la loi des grands nombres fait que les distances moyennes entre un parent et son enfant par mutation gaussienne et par mutation uniforme ne sont pas très

différentes, ce qui expliquerait également qu'on retrouve les mêmes résultats concernant l'influence de α et β dans SDM et dans AlgoApp.

Quoi qu'il en soit, nous ne présenterons pas ici les résultats détaillés de l'influence des trois paramètres, renvoyant aux section 3.4 du chapitre 4.

4.2.3 Interaction SDM-ES – Influence du paramètre k_{std}

Discussion

L'interaction entre SDM et ES s'effectue d'une part à travers le paramètre k_{std}, qui détermine le paramètre σ de l'enfant résultant de la mutation, et d'autre part par la rétroaction de l'algorithme ES sur ε_k lorsqu'un nouveau meilleur individu a été trouvé par une mutation gaussienne et non par SDM, au travers des paramètres α et β. Cette section va se concentrer sur l'influence de k_{std}.

k_{std} a un effet direct sur l'aspect local ou global de la recherche : le pas de la mutation du nouveau meilleur individu de la population étant proportionnel à k_{std} (équation (4.1)), de petites valeurs de k_{std} intensifient la recherche locale autour du meilleur individu alors que de grandes valeurs diversifient la recherche, augmentant son côté exploratoire. On serait tenté d'ajuster k_{std} en fonction du caractère multimodal de la fonction objectif – mais ce caractère est généralement inconnu.

Remarquons d'autre part que l'équation (4.1) rend de fait σ_k proportionnel à d_{k-1} et que par conséquent l'ajustement du pas σ peut être considéré ici comme adaptative et non plus auto-adaptative : après un déplacement important, le pas de la mutation va immédiatement être augmenté, et ce même s'il s'agit d'une mutation chanceuse (i.e. de grande amplitude malgré un petit σ) : en ce sens, le mécanisme SDM se rapproche du mécanisme originale de CMA-ES (voir section 1.4.2) qui remplaçait la mise à jour stochastique du σ par une mise à jour déterministe.

Une autre remarque concerne la différence entre les remplacements SDM-Plus et SDM-Comma. En effet, dans le cas de SDM-Plus, l'enfant SDM est ajouté à la population, risquant d'accélérer une convergence éventuellement prématurée.

Notons enfin que, comme pour les ES, l'application de l'opérateur de croisement sur les paramètres de mutation est nécessaire afin de re-mélanger les valeurs des σ dans toute la population et d'éviter que ces valeurs ne diminuent trop rapidement, après une mutation chanceuse par exemple.

Étude expérimentale

Tous les résultats présentés dans la fin de ce chapitre ont utilisé une (7+49)ES isotrope au sein de laquelle SDM était implémenté. Comme dans les chapitres précédents, nous étudierons la convergence sur les quatre fonctions Sphère, Griewank, Ellipse et GriewankE.

Les premières courbes présentées (figures 4.2.3, 4.2.3, 4.2.3 et 4.2.3) concernent un essai représentatif du comportement de l'algorithme SDM complet et ont pour but d'étudier en détail le gain apporté par SDM (quand est-ce que c'est la mutation SDM qui génère le meilleur individu) et le sur-coût nécessaire à l'obtention de ce gain. Sont tracés, en fonction du nombre d'évaluations de la fonction objectif (incluant, bien entendu, les évaluations supplémentaires nécessaires à l'application de SDM) :

- dF_{Sdm} : La variation de la fonction objectif entre le parent SDM et l'optimum du modèle calculé de manière déterministe ;
- dF_{Es} : La variation de la fonction objectif due aux opérateurs de variations standard de l'ES (mutations gaussiennes et opérateurs de croisements) ;
- $dEval_{SDM}$: Le nombre d'évaluations supplémentaires du fait de l'utilisation de l'opérateur SDM, que ce soit dans les cas oú l'apprentissage ne disposait pas des `Nptpoints` nécessaires, ou lors du calibrage de la méthode d'apprentissage ;
- $Fitness$: La performance globale, telle que la voit l'utilisateur.

Dans le cas de la fonction sphère 4.2.3, nous observons deux phases, indépendamment de la stratégie SDM-Plus ou SDM-Comma. La première phase correspond à l'amélioration due à SDM, qui est alors très proche de l'algorithme AlgoApp (section 3.4). La deuxième phase correspond au ralentissement de la convergence de SDM – les mutations gaussiennes de ES prennent alors le relais et e sont elles qui améliorent la performance. Notons que dans le cas de SDM-Plus, pour des grandes valeurs de k_{std} (de l'ordre de 1.0, nous observons des détérioration de la convergence de la deuxième phase. En effet, dans le cas de la sphère, une recherche purement locale suffit. Nous pouvons relever de plus que, dans le cas de SDM-Plus, nous assistons à un gel total de la population. En effet l'amélioration avec l'opérateur SDM-Plus est due à l'évaluation, en dehors de l'algorithme et de manière systématique, le même nombre d'individu ($dEval_{SDM}$). Dans ce cas, l'algorithme SDM est très proche de l'algorithme AlgoDet (section 3.3). Dans la deuxième phase nous remarquons que dans la plupart des cas, le nombre $dEval_{SDM}$ fluctue, renforçant l'idée de la participation des mutations gaussiennes à la convergence locale.

Le cas de la fonction Griewank 4.2.3 montre que les mutations gaussiennes sont actives pour $k_{std} = 0.1$. Il faut avoir à l'esprit que cette fonction a un comportement global quadratique. La recherche globale par ES est alors en fait peu utile. Ainsi en augmentant les valeurs des σ, nous n'améliorons pas pour autant la convergence. Comme dans le cas de la sphère, les grandes valeurs de k_{std} diminuent l'activité de ES. Les individus évalués provenant des mutations gaussiennes ne participent pas beaucoup dans l'élaboration de l'opérateur SDM qui lui s'applique sur les individus localisés autour du meilleur. Dans le cas SDM-Comma, l'interaction ES finit par rattraper SDM et la convergence sera portée principalement par les mutations ES. On observe bien les deux phases pour l'opérateur SDM. La première phase SDM est plus rapide que les mutations ES : les individus utilisés sont principalement évalués en dehors de ES. Lors de la deuxième phase, les mutations ES "rattrapent" le meilleur et contribuent au résultat final.

Dans le cas de la fonction Ellipse 4.2.3, la mutation ES isotrope utilisée (matrice de covariance σI_n) n'est pas adaptée aux cas des fonctions fortement elliptiques. Ainsi la valeur de ε est amenée à s'aligner sur la petite valeur propre. Les sauts observés à chaque amélioration par les mutations ES viennent en fait des ré-initialisations du paramètre ε. Il faudrait peut-être songer à utiliser un algorithme ES non-isotrope – la difficulté est alors dans la mise à jour des variables σ de l'individu créé par SDM ...

Le même problème se retrouve dans le cas de la fonction GriewankE 4.2.3. On constate pour cette fonction que les grandes valeurs de k_{std} qui vont augmenter les σ dans l'ensemble de la population ne sont pas efficaces pour la recherche de l'optimum global. Si on compare la convergence par rapport à celle d'un algorithme ES pur, ou celle d'AlgoApp, nous remarquons que l'hybridation ne stagne pas. En effet le paramètre ε est un indicateur de la convergence de SDM. Si ε tend vers zéro, alors SDM devient inactive et comme les mutations ES n'améliorent pas non plus la performance, l'algorithme s'arrête. Ici, par contre, le paramètre ε diminue moins vite, et ce grâce aux déplacements de l'optimum dus aux mutation gaussiennes standards Nous verrons dans le paragraphe suivant plus en détails la modification apportée à la relation de récurrence de la mise à jour de ε, qui crée un dynamisme évitant les convergences prématurés.

4.2.4 Un algorithme dynamique

Les paramètres de SDM sont dynamiques, c'est-à-dire que leurs valeurs sont héritées puis modifiées d'une itération à l'autre. Ceci permet, au travers

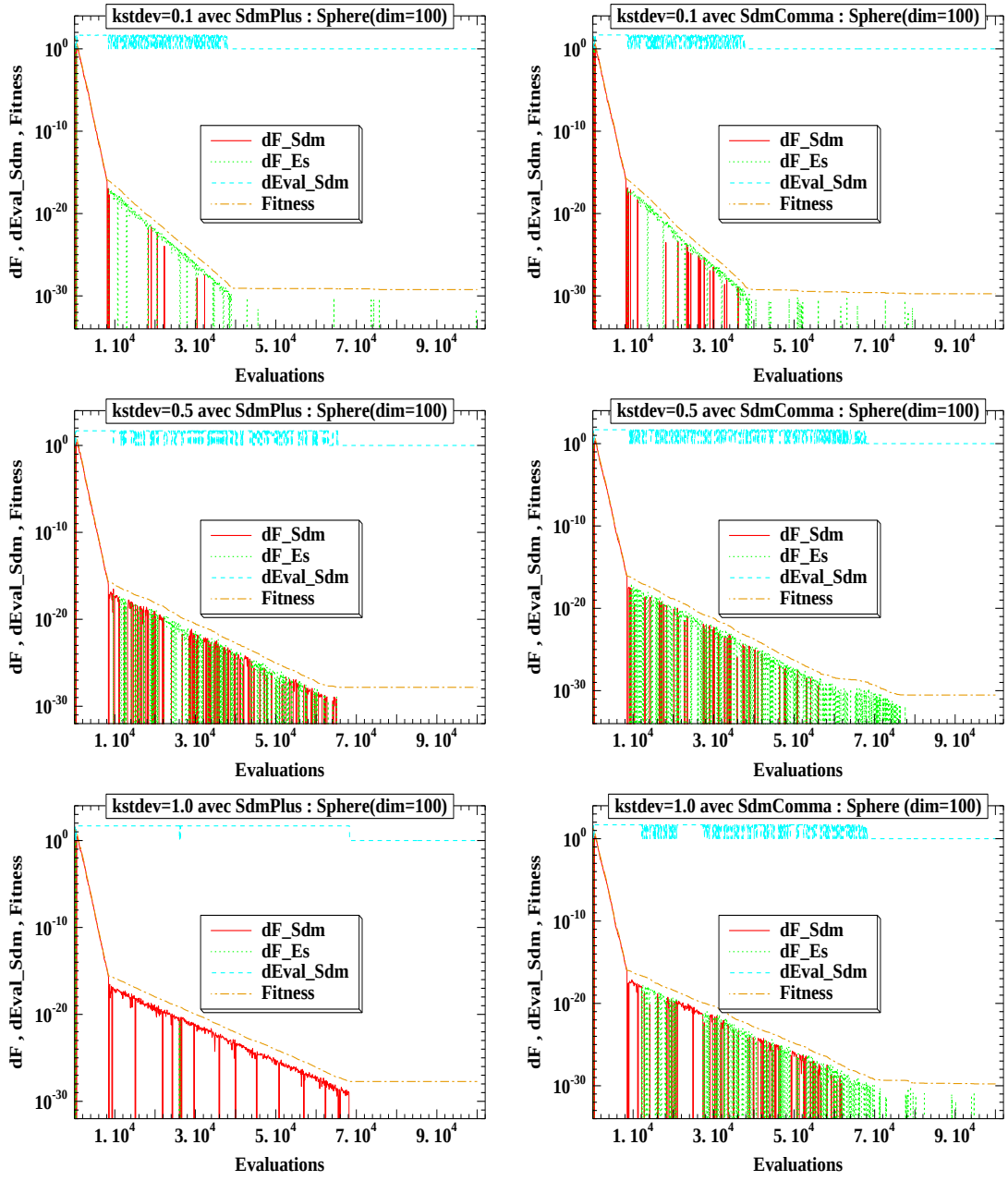


FIG. 4.1 – Influence de k_{std} dans les deux approches Sdm-Plus et Sdm-Comma sur la fonction sphère.

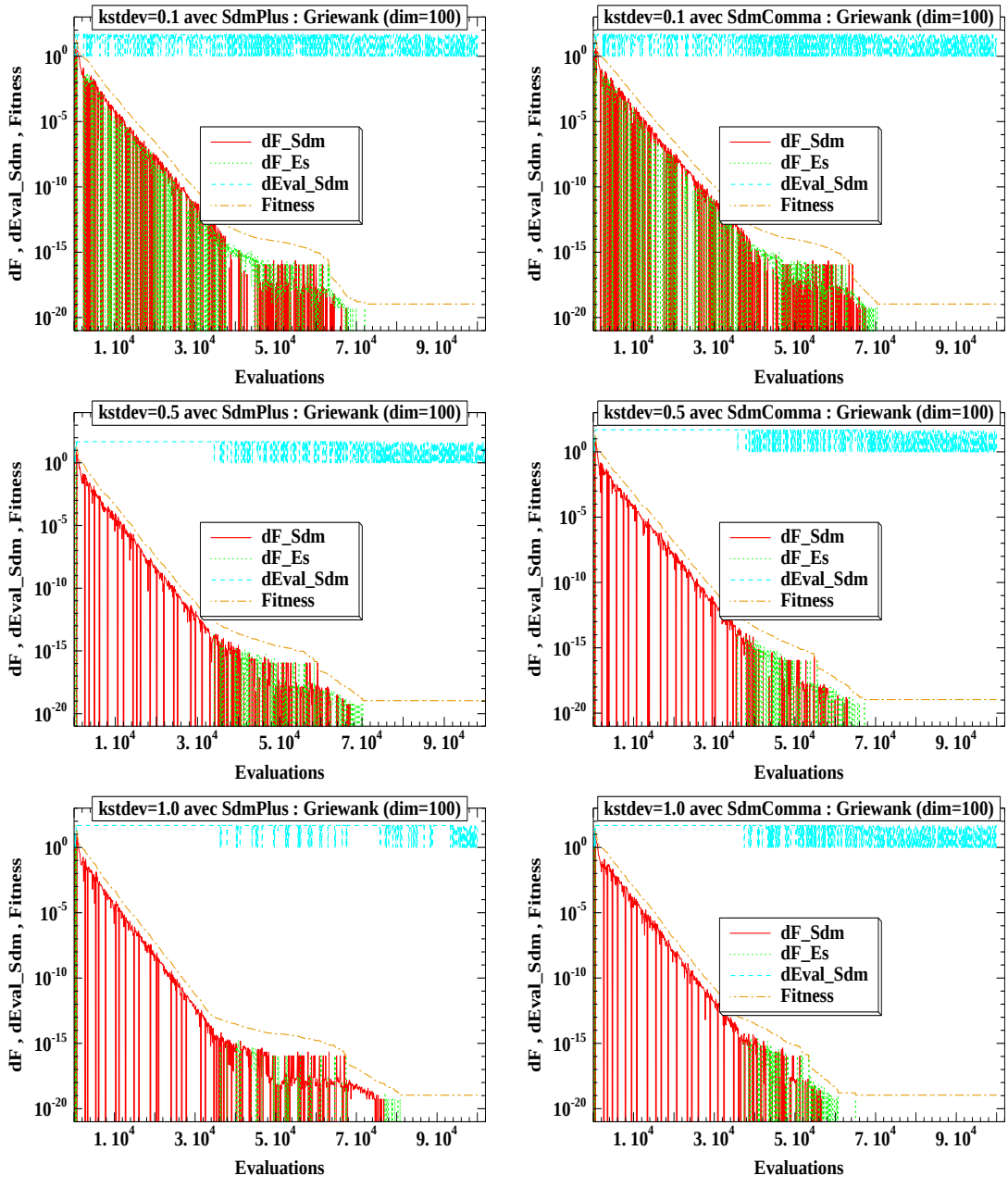


FIG. 4.2 – Influence de k_{std} dans les deux approches Sdm-Plus et Sdm-Comma sur la fonction Griewank.

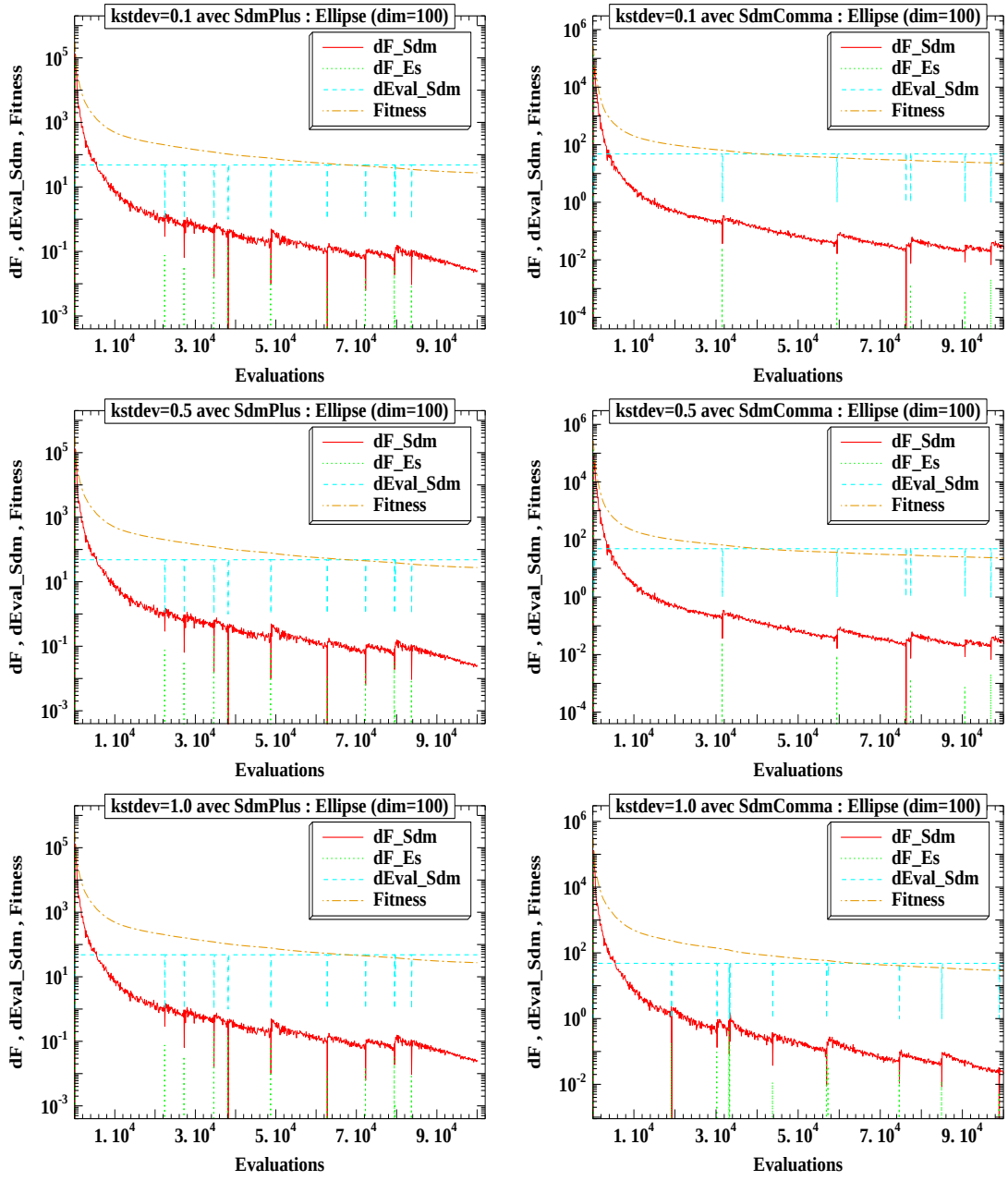


FIG. 4.3 – Influence de k_{std} dans les deux approches Sdm-Plus et Sdm-Comma sur la fonction Ellipse.

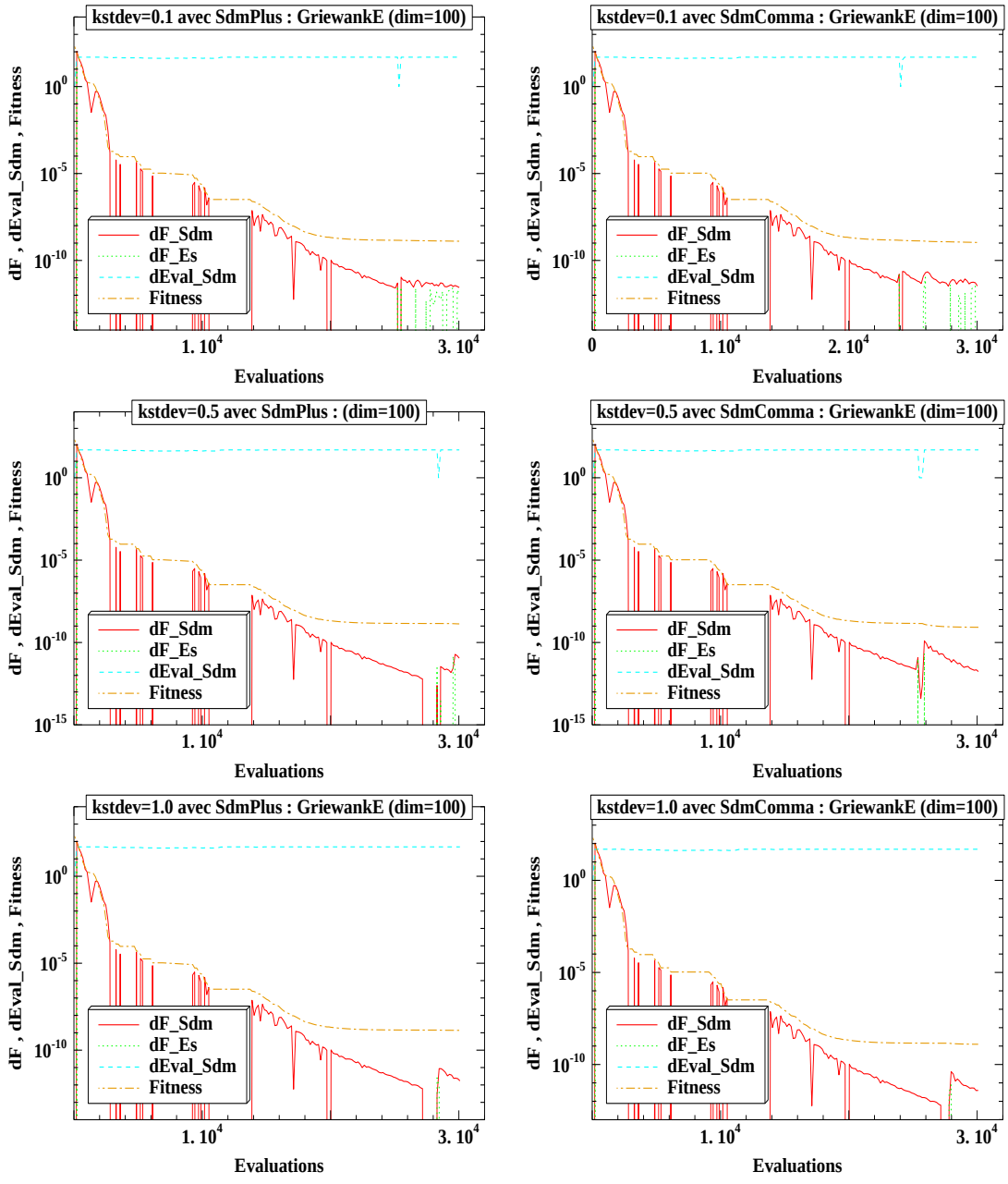


FIG. 4.4 – Influence de k_{std} dans les deux approches Sdm-Plus et Sdm-Comma sur la fonction GriewankE.

de ces paramètres, de contrôler le compromis exploration-exploitation, et de tenter d'éviter aussi les convergences prématurées.

Paramètre de régularité : Comme dans AlgoApp nous avons opté pour le critère objectif (CO) pour la recherche des paramètres adéquats (section 3.4). Les paramètres de la régularité de l'algorithme d'apprentissage sont hérités d'une itération à l'autre. En fait, ce ne sont pas directement les paramètres qui sont hérités, mais leur recherche qui s'effectue à partir de la population finale trouvée dans la recherche précédente, au moyen d'un algorithme ES partiel. A travers l'héritage de la population d'une itération à l'autre nous différons l'optimisation des paramètres de régularité sur plusieurs génération ! Cet héritage est bien sûr d'autant plus justifié que la régularité de la fonction objectif est relativement stable (d'une génération à la suivante).

Le domaine d'interpolation : le paramètre ε est adapté pour refléter la difficulté à définir une approximation dans le domaine D_ε sachant que le nombre de point est fixé à l'avance. Il s'adapte à travers les paramètres α et β. Une différence essentielle avec l'algorithme AlgoApp est que le déplacement utilisé par SDM n'est pas forcément la distance entre le parent et l'enfant généré par SDM puisque c'est la distance entre les deux meilleurs successifs qui est utilisée. Ainsi, si les mutations ES trouvent un individu meilleur que celui ajouté par SDM, alors la multiplication par β crée une sorte de ré–initialisation de l'opérateur SDM, ce qui est tout à fait justifié du fait que l'on a changé de région de l'espace de recherche, et que la régularité de la fonction objectif peut avoir elle aussi changé.

Vers un k_{std} dynamique ? : Comme nous l'avons vu plus haut, l'interaction SDM – ES s'effectue aussi à travers le paramètre k_{std}. Au vi des résultats précédents, on est tenté de définir une loi dynamique pour k_{std} ?

Il est courant de mettre en relation l'écart-type Δ de la population courante avec un paramètre contrôlant l'exploration. Par exemple on pourrait procéder ainsi : si Δ est "petit" alors on choisit un k_{std} "grand" sinon on choisit un k_{std} "petit". Des contraintes de temps nous ont empêché de tester ces idées, et dans la suite de l'étude, nous ne considérerons que le cas k_{std} constant.

Les tests d'arrêt de SDM : L'application de la mutation SDM peut quelquefois s'avérer chère, sans amélioration notable de la performance. Il est donc judicieux d'arrêter SDM assez tôt sans pour autant arrêter l'algorithme ES. Nous pouvons imaginer plusieurs modes d'arrêt de l'opérateur SDM :

– Un seuil sur la valeur de ε pourrait constituer une condition d'arrêt.

C'est d'ailleurs cette condition d'arrêt qui est utilisée dans l'algorithme AlgoApp.

- Nous pouvons aussi considérer un seuil sur la valeur de f. Observons la figure 4.2.4. Nous remarquons que l'arrêt brutal de la mutation SDM peut provoquer des convergences prématurées. En effet dans la convergence locale, si SDM s'arrête avec des σ encore élevés dans la population, alors ES nécessite un certain temps de relaxation et d'adaptation pour continuer seul efficacement. Nous remarquons par la même occasion, que dans la stratégie SdmPlus la saturation est plus affectée que dans le cas de SdmComma. Dans le cas de la fonction GriewankE l'arrêt est systématique. Cela conforte l'idée qu'un ES pur n'est pas l'algorithme le plus adapté à la situation.
- Le dernier test d'arrêt que nous envisagerons est un test adaptatif, dynamique, basé sur l'observation de l'évolution de la performance. Il prend en compte la nature de l'hybridation. En effet, revenons aux deux algorithmes qui sont ici hybridés, l'ES isotrope et AlgoApp. Si oublions un instant les interactions entre ces deux algorithmes, nous pouvons comparer leurs performance respectives. Nous définissons pour cela les rapports suivants :

$$\rho_{SDM} = \frac{\Delta f_{SDM}}{\Delta n_{SDM}}, \rho_{ES} = \frac{\Delta f_{ES}}{\Delta n_{ES}}$$

Δf représente l'amélioration de la performance effectuée en évaluant Δn points de la fonction objectif.
Ces taux varient bien sûr beaucoup d'une génération à l'autre, et nous les lissons sur plusieurs générations (toutes les 5 générations dans nos expériences). Nous décidons alors l'arrêt de SDM dès que la condition $\rho_{ES} > \rho_{SDM}$ est vérifiée. Si nous observons maintenant la figure 4.2.4, nous remarquons que l'influence du paramètre k_{std} dans le cadre d'un arrêt dynamique a diminué par rapport à un arrêt brutal, pour les fonctions de tests choisis à l'exception de la fonction sphère avec la stratégie SdmPlus! Cela ne doit pas nous conduire à minimiser pour autant l'influence du paramètre k_{std}, mais cela met en relief l'utilité d'un critère d'arrêt de SDM adapté à la situation courante.

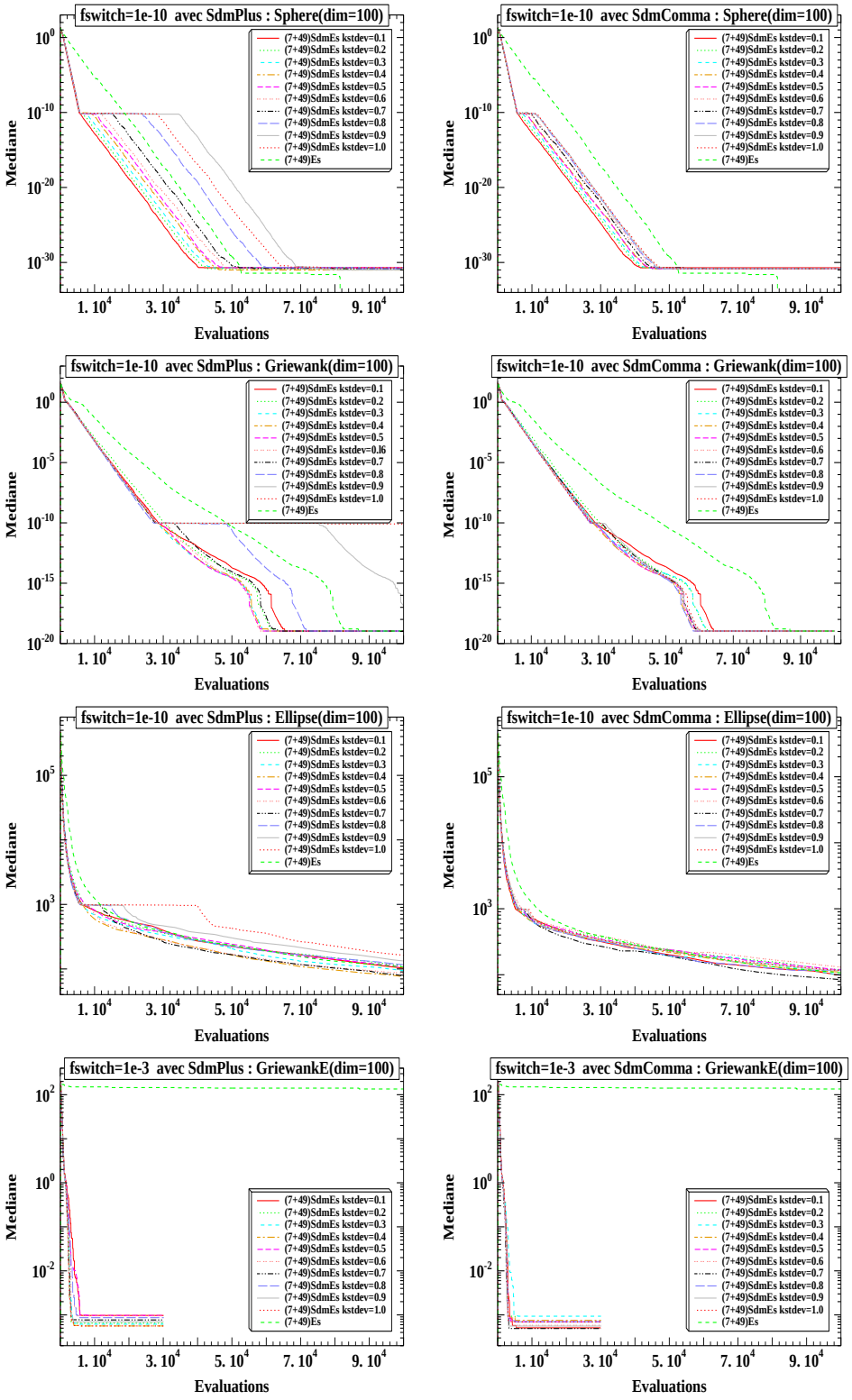


FIG. 4.5 – Arrêt de SDM selon un seuil f_{switch}. Fonctions sphère ($f_{switch} = 1e-10$), Griewank ($f_{switch} = 1e-10$), Ellipse ($f_{switch} = 1e-3$) et GriewankE ($f_{switch} = 1e-3$).

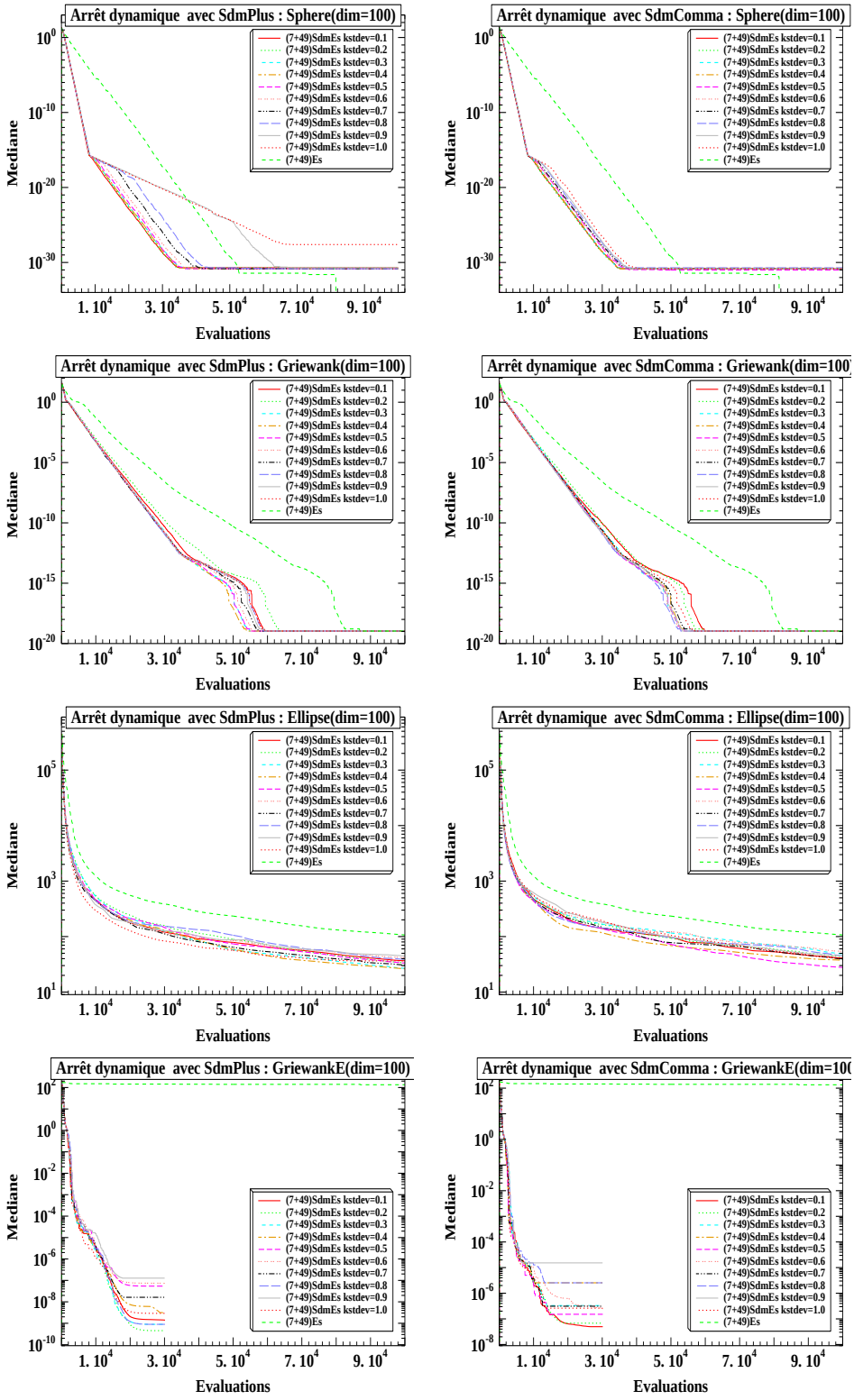


FIG. 4.6 – Arrêt dynamique de SDM

4.3 Metamodel Assisted Evolution Strategies

Dans cette section, nous allons comparer notre approche SDM avec une autre utilisation de l'apprentissage au sein des algorithmes évolutionnaires proposée initialement par A. J. Keane au sein d'un AG classique dans [30] et repris par Yaochu Jin [35] au sein d'un CMA-ES et finalement elle sera hybridée avec l'approche de la fonction de mérite (voir section 1.6.1) avec Thomas Bäck dans [40] au sein d'un algorithme ES(voir section 1.6.3). Cependant, l'outil d'approximation utilisé était dans ces travaux soit les réseaux de neurones soit le Krigeage et soit la régression quadratique, qui tous les trois posent des problèmes en grandes dimensions. Nous allons donc tout d'abord proposer et implémenter la méthode MAES basée sur l'utilisation de SVM comme technique d'apprentissage.

4.3.1 Construction de MAES

Le principe

La **MAES** est une approche dynamique de l'hybridation avec une méthode d'approximation (voir l'état de l'art, chapitre : 1).
Il s'agit, après avoir appris une fonction $\hat{f}$ approchant la fonction objectif, de minimiser au sein d'un algorithme ES standard la fonction suivante :

$$\hat{f}(x) = \begin{cases} f(x) & \text{si x vérifient un critère } \mathcal{S} \\ \hat{f}(x) & \text{si non} \end{cases}$$

Les points vérifiant le critère $\mathcal{S}$ sont les *npteval* meilleurs points selon la fonction de mérite utilisée dans la méthode de Krigeage, *npteval* étant le nombre maximal de points à évaluer avec la fonction vraie objectif f.

$$S_w(x) = \hat{f}(x) - w\sqrt{M\hat{S}E(x)} \tag{4.2}$$

En particulier, si on choisit $w = 0$, l'approche ci-dessus revient à évaluer avec la vraie fonction objectif les *npteval* points les meilleurs selon le modèle $\hat{f}$.

L'algorithme

On dispose toujours d'un algorithme de type Stratégie d'Évolution auto-adaptative isotrope, c'est-à-dire dans lequel chaque individu a un paramètre de mutation qui lui est propre noté σ. La matrice de covariance de la mutation est alors σI_n (voir chapitre 2 section 1.4).

Initialisation

On tire au hasard `Npt0` points dans tout le domaine de recherche et on les évalue avec f. L'archive $\mathcal{A}_0$ de points déjà évalués est initialisée avec ces `Npt0`. On définit avec les `Npt` meilleurs points une fonction approchée. On initialise ES avec une population formée par les meilleurs parmi les `Npt` points de l'ensemble d'apprentissage.

L'opérateur MAES

À la génération k, on applique l'opérateur MAES, de la manière suivante :

1. On évalue npt points au maximum parmi les enfants avec la fonction objectif f les autres sont évalués avec $\hat{f}$. Les points à évaluer avec la fonction f sont les npt meilleurs selon le critère S.
2. On stocke les points évalués dans A_k. Si le nombre total de points évalués atteint `Npt`, on construit alors une approximation $\hat{f}$ de f à partir de ces derniers `Npt` points évalués $x^1, \ldots, x^{Npt}$. Les paramètres de l'algorithme d'apprentissage sont optimisés suivant la procédure mise au point et discutée au chapitre 4.
3. La stratégie de remplacement est modifiée pour conserver un rapport constant entre points évalués avec la fonction objectif f et points évalués avec la fonction approchée $\hat{f}$.

Remarques :

Le modèle est mis à jour toutes les $ngop$ générations. Les points évaluées au cours de ces $ngop$, qui sont au nombre de $\texttt{Npt} = ngop * npteval$, serviront à construire le nouveau modèle. L'utilisation d'un noyau gaussien dans le cas de SVM ou d'une fonction de corrélation dans le cas du Krigeage donne un bon modèle là où la densité des points est élevée. Rappelons que les points tirés par ES, évalués avec f et se trouvant isolés par rapport aux points d'apprentissage ne servent à rien dans l'approximation de la fonction.

Le critère $\mathcal{S}$ est défini pour la méthode d'approximation Krigeage (fonction de mérite). Nous proposons dans la suite un critère pour la méthode d'approximation SVM basé sur la même idée.

4.3.2 Interaction Modèle - ES

L'interaction est évaluée à travers les paramètres des mutations gaussiennes des individus σ_i. Rappelons premièrement que ES-MAES converge

si ces derniers tendent vers zéro. Deuxièmement, la lenteur des algorithmes évolutionnaires en général est intimement liée à l'adaptation des paramètres internes qui sont auto-adaptatifs : cette adaptation s'effectue en effet par essais et erreurs. Nous retrouvons la problématique qui a donné naissance à CMA-ES.

Dans l'algorithme MAES la convergence est elle aussi liée aux valeurs σ_i. Or contrairement à SDM, l'hybridation dynamique MAES contrôle indirectement ces valeurs. En effet, les individus choisis pour remplacer la population courante sont en partie seulement évalués avec la vraie fonction objectif, mais en majorité avec la fonction modèle. Ainsi les paramètres σ_i sont principalement hérités des tests sur la fonction modèle. Si le modèle comporte des minima locaux artificiels alors les paramètres σ_i auront tendance à converger vers zéro, et ce malgré l'opérateur de croisement qui a tendance à les uniformiser dans la population. C'est ce que nous avons remarqué de façon criante sur la fonction Griewank par exemple.

Ces problèmes sont similaire aux problèmes rencontrés avec AlgoAppBis (Section 3.5). En effet, AlgoAppBis, au lieu d'optimiser la fonction modèle avec un algorithme déterministe, utilise un algorithme ES. À chaque mise à jour du modèle, nous initialisons ES avec la population courante et les paramètres σ_i hérités dans la précédente recherche.

Alors qu'avec SDM, nous avions un contrôle direct sur les paramètres σ_i à travers le paramètre k_{std} et donc sur l'exploration/exploitation alors qu'avec MAES nous ne pouvons l'avoir que de manière indirecte à travers l'auto-adaptation de l'évolution sur le modèle. La solution pourrait être de considérer le critère $\mathcal{S}$(La fonction mérite) avec un w dynamique. En effet, l'adaptation du poids w permet d'optimiser sur un domaine plus ou moins grand, équilibrant ainsi entre exploration et exploitation.

Le critère $\mathcal{S}$

La méthode de Krigeage offre naturellement (et "gratuitement") l'erreur MSE en un point quelconque ce qui permet de l'utiliser comme critère de choix des individus qu'il faudra réévaluer avec a vraie fonction objectif. Ainsi, en optimisant la fonction avec la contrainte de rester dans la zone où le modèle est meilleur (en tant qu'approximation), les chances trouver un individu meilleur qui soit aussi meilleur pour la fonction objectif sont accrues.

Dans le cas d'une corrélation gaussienne, la MSE est petite là ou la densité de points est grande et si on s'éloigne des points d'apprentissage, la valeur de MSE augmente. Cela suggère dans le cas de SVM avec noyau gaussien un critère sous la forme suivante :

$$\mathcal{S}^{SVM}(x) = \hat{f}(x) - w * d(x, APP).$$

Avec $d(x, APP)$ la distance du point x aux points d'apprentissage.

Pour des w grands cela permettrait de choisir des points éloignés des points d'apprentissage. Si le remplacement préserve ces individus dans la population suivante, alors leurs σ_i (qui ont toutes les chances d'être grands) seront préservés et la convergence repoussée. Sinon, nous risquons de tomber dans des minima locaux artificiels du modèle, et cela causerait la convergence prématurée.

La diversité dans la population

L'hybridation dynamique est intéressante dans la mesure où elle permet d'adapter les paramètres internes de l'algorithme évolutionnaire rapidement.

Il faut bien avoir à l'esprit que l'on mélange des individus évalués avec la fonction objectif et des individus évalués avec la fonction modèle. Mais rien ne garantit avec un opérateur de remplacement usuel que l'on garde des individus évalués avec la fonction objectif!

Les auteurs de MAES n'ont pas prévus ce point. A. J. Keane ([30], et voir section 1.6.3) relève le problème et propose un remplacement dynamique. Le paramètre s de remplacement garantit la préservation d'individus évalués selon la vraie fonction objectif.

Sachant que le nombre de points évalués est le même à chaque génération, nous avons alors décidé de garder un rapport constant entre les deux catégories (points évalués par le modèle et par la vraie fonction) lors du remplacement.

4.3.3 Le dynamisme de MAES

Comme dans SDM, l'algorithme MAES contient des paramètres dynamiques.

Paramètres de régularités

Rappelons que dans le cas de AlgoAppBis nous avons utilisé l'algorithme déterministe pour définir le critère objectif (CO).

Nous procédons de la même manière avec MAES, en donnant simplement rapidement les résultats avec les différents critères.

- **CE** :
  De même que AlgoAppBis défini dans le chapitre précèdent, nous remarquons une instabilité par rapport au nombre d'évaluations du critère (CE).
- **CO** :
  Le critère (CO) est plus stable. En effet l'optimisation déterministe du modèle permet de trouver des paramètres avec peu d'oscillations. Nous avons dans le cas de la sphère un résultat meilleur que SDM. C'est un problème numérique affectant la phase d'optimisation déterministe lors de l'évaluation du critère CO.
- **CV** :
  Le critère (CV) est trop cher pour les tests mais pourrait devenir compétitif dans le cas où la fonction objectif est aussi extrêmement chère.
- **CG** :
  Le critère (CG) est défini pour la méthode d'approximation Krigeage. Il donne de mauvais résultats au-delà de la dimension 30 !

Le domaine d'interpolation

Dans le cas de SDM, le domaine d'interpolation est défini dans l'hypercube de côté D_ε et centré autour du meilleur. Ici le domaine est implicitement défini par l'ensemble des points d'apprentissage. Dans le cas de SDM, le domaine était adapté selon une loi déterministe et l'adaptation était liée au résultat de la mutation SDM. Ici la contraction est implicitement contrôlée par le paramètre w. Cela expliquerait l'intérêt d'une lois dynamique pour w. Il faut donc mettre en relation w avec l'amélioration effectuée sur la fonction objectif après optimisation du critère $\mathcal{S}$.

Vers un w dynamique

Nous pourrons nous inspirer de la loi dynamique proposée par Christopher M. Siefert [65] dans le cadre SAO.

Dans le cadre de EOA-dynamique, rappelons que si w est choisie grand alors les meilleurs individus classés selon le critère $\mathcal{S}$ et qui seront évalués avec la fonction objectif, sont parmi les enfants les plus éloigné des points d'apprentissage. On est alors dans la phase exploration. A l'inverse si w est choisi petit alors les points à évaluer avec la fonction objectif, seront choisis parmi les individus proches des points d'apprentissage. On est alors dans la phase exploitation.

Un test d'arrêt de MAES

Comme dans le cadre SDM, il est utile de considérer un test d'arrêt pour l'application de la méthode d'approximation. Ainsi le problème redevient statique sur la fonction objectif. Aucun auteur ayant travaillé sur MAES n'ayant envisagé la questions, nous nous sommes inspirés des tests d'arrêt utilisés mis au point sur SDM, et les critères suivants ont été envisagés :

- Un seuil sur la valeur du diamètre du domaine d'interpolation défini par l'ensemble des points d'apprentissage. Cette condition d'arrêt est aussi utilisée dans l'algorithme AlgoAppBis.
- Un seuil sur f.

A noter que nous n'avons pas, comme dans le cas de SDM-ES, la possibilité de comparer entre deux algorithmes. Un arrêt dynamique doit se porter uniquement sur l'amélioration de l'algorithme lui-même et cela nous ramène à un choix de seuil.

4.4 Résultats et Conclusions

4.4.1 Choix du modèle

Pour définir un modèle nous devons bien choisir les paramètres de régularité. Dans le cas de grande dimension supérieure à 30 les critères de généralisation du Krigeage ne se sont pas révélés satisfaisants.

Le critère objectif (CO), malgré son sur-coût, nous apparaît incontournable, pour les deux méthodes SVM et Krigeage en grande dimension. Nous avons alors testé l'idée d'étaler la recherche des paramètres tout au long de l'optimisation : on fait quelques itérations à chaque génération.

Enfin, la méthode SVM est moins contraignante que la méthode Krigeage du fait qu'elle propose des modèles intermédiaires, c'est-à-dire obtenus sans pousser complètement l'aspect approximation de la méthode : en effet notre but est bien de trouver de bons candidats à l'optimalité de la vraie fonction objectif, et pas forcément de l'approcher.

Nous concluons que la méthode SVM est plus adaptée que la méthode de Krigeage en grande dimension.

4.4.2 SDM et MAES

L'idée centrale est d'accélérer la convergence non seulement en proposant des nouveaux points mais également en agissant sur les paramètres internes de l'algorithme ES : les σ_i des individus sont déterminés d'une manière directe dans le cas SDM, alors que dans le cas de MAES, l'optimisation du modèle avec ES les adapte comme d'habitude mais suivant le modèle, on peut parler d'influence indirecte.

Un deuxième point important est de contrôler les convergences prématurées. On peut ainsi utiliser des techniques dynamiques d'optimisation qui tendent vers une hybridation stable. La diversité entre individus évalués avec le modèle et ceux évalués avec la fonction objectif doit être aussi contrôlée. Dans le cas de SDM, le contrôle direct demande aussi une bonne adaptation des paramètres contrôlant l'hybridation.

Comparaison expérimentale

Nous allons maintenant présenter des résultats expérimentaux qui porteront en quelque sorte la conclusion de cette thèse. Nous allons comparer, toujours sur les mêmes fonctions test, et du point de vue purement pragmatique de la performance en fonction du nombre d'évaluations, les algorithmes

SDM-ES, MAES avec comme critère soit la fonction modèle $\hat{f}$ (qui correspond au cas $w = 0$), soit le critère $\mathcal{S}^{SVM}$ décrit dans la section précédente, et enfin bien entendu l'algorithme ES de base. La figure 4.4.2 donne l'ensemble des résultats.

Les résultats sur la fonction sphère montrent que la vitesse de convergence est du même ordre entre les deux approches SDM et MAES. Dans le cas de Griewank nous remarquons que SDM est meilleur que MAES. Les différences sont principalement dues à l'arrêt dynamique de l'opérateur SDM.

Sur la fonction Ellipse, SDM l'emporte sur MAES grâce à la relation dynamique mettant à jour le paramètre de recherche ε, le rayon du domaine d'interpolation. En effet, comme nous l'avons déjà discuté (section 4.2), cela s'apparente à des ré–initialisations et des re–calibrages des paramètres σ_i induites par ε.

Finalement la différence sur la fonction GriewankE est flagrante cela vient du déplacement de ES sans forcement améliorer la performance. Ce dynamisme crée une instabilité dans SDM (AlgpApp). Mais les deux algorithmes ES et SDM travaillant ensemble permettent de faire converger l'ensemble.

4.4.3 Perspectives

Hybridation avec CMA

Nous nous sommes intéressés au cas sphérique et avons pointé du doigt les problèmes rencontrés sur des fonctions mal conditionnées. Cela ne couvre pas, bien entendu l'ensemble des difficultés qu'on peut rencontrer dans les cas de fonctions réelles.

Dans cette perspective, il est aujourd'hui assez naturel de s'intéresser à l'algorithme CMA-ES (voir [12], ou section 1.4.2). L'hybridation de CMA-ES avec une méthode d'apprentissage parait au premier abord assez facile à réaliser. En effet il devrait suffire de remplacer la matrice de covariance par l'inverse de la matrice hessienne de la fonction modèle. Cependant, rappelons que la fonction modèle, définie à travers les points exemples, est loin d'être locale, et en particulier les dérivés ne sont pas forcement bien approchées.

Yaochu Jin, Markus Olhofer et Berhard Sendhoff (2001)[35] ont utilisé une approche dynamique avec CMA, exempte de tout calcul de la matrice Hessienne. Cette optimisation avec "Contrôle par évaluation" donne une vision globale d'une fonction certes bruitée, mais ne s'éloignant pas de l'original, bon candidat pour une hybridation dans un cadre similaire à celui de SDM.

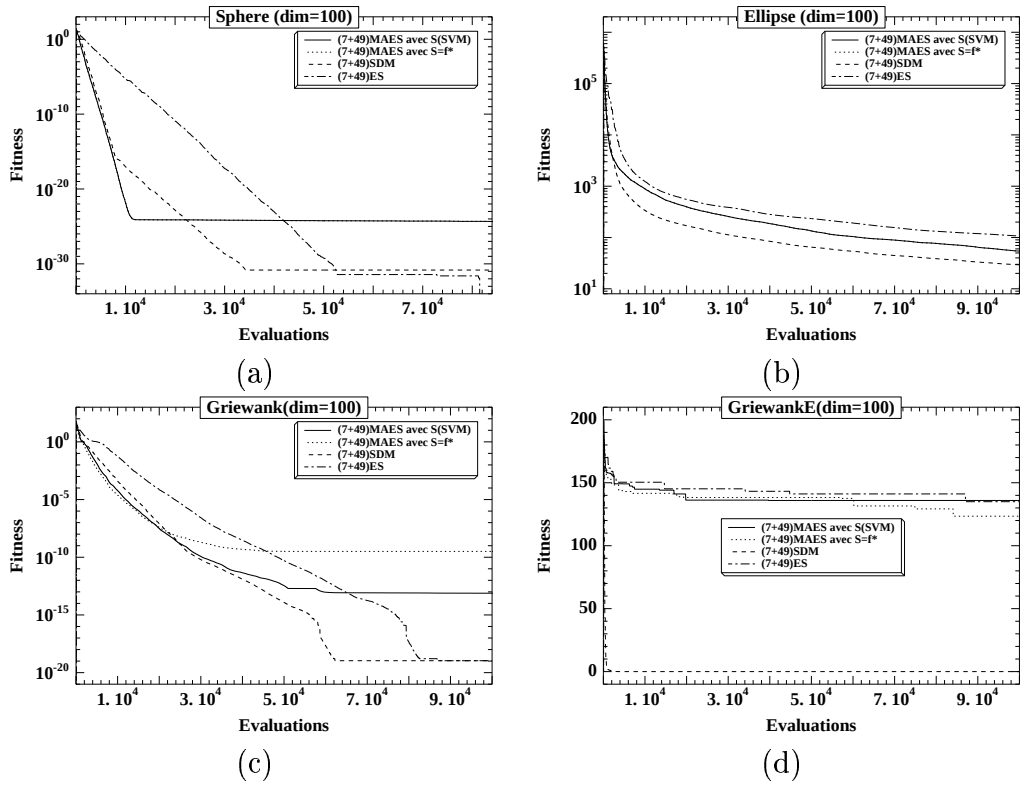


FIG. 4.7 – Comparaison entre SDM, MAES($S = \hat{f}$), MAES($S = S^{SVM}$) et ES en dimension 100. (a) Fonction Sphère, (b) Fonction Ellipse, (c) Fonction Griewank, (d) Fonction GriewankE

Le dynamisme de MAES

De même que l'on peut envisager l'adaptation dynamique du paramètre k_{std} de SDM, on peut aussi envisager de modifier le poids w de la fonction de mérite au cours de l'évolution. L'importance de ce paramètre rend à la fois cette piste intéressante, mais peut-être délicate.

Chapitre 5

Problème de calibration

Ce problème m'a été soumis par la société ITO33 spécialisée dans la finance quantitative et développement de logiciels de pricing et de calibration. Je remercie Élie Ayache et Philippe Henrotte qui m'ont ainsi donné l'occasion d'appliquer mes travaux à un problème concret très pointu.

5.1 Problématique du smile et marché incomplet

5.1.1 Modèle de Black-Scholes

Le formidable développement qu'ont connu les marchés de produits dérivés depuis environ trente ans n'aurait probablement pas été possible sans la publication en 1973 de l'article de Black-Scholes présentant un modèle d'évaluation d'options. Pour un «call européen», c.à.d. l'option d'achat d'un actif risqué, par exemple une action, à une date future T, l'équation (backward) de Black-Scholes s'écrit :

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + r(t) S \frac{\partial V}{\partial S} = r(t) V \quad \text{pour } S > 0, ; T > t \geq 0 \tag{5.1}$$

Dans cette équation :

- S est le «spot» sous-jacent, l'actif risqué.
- V est le prix de l'option à tout instant t et pour tout spot S.
- σ est la volatilité : plus elle est grande et plus c'est risqué ! En fait, l'équation de Black-Scholes peut être dérivée de l'hypothèse selon laquelle le sous-jacent risqué suit un processus de diffusion lognormal :

$$\frac{dS}{S} = \pi(t)dt + \sigma dW \tag{5.2}$$

où dW est un Brownien.
- $r(t)$ est le taux d'intérêt sans risque.
- T est la «maturité» de l'option.
- On introduit aussi K le prix d'exercice de l'option qu'on appelle aussi «strike» ;

Le spot d'aujourdui (à $t = 0$) vaut S_0. On cherche le prix de l'option d'achat à la maturité $t = T$ de l'actif risqué au prix K. L'option (call) permet ainsi à son acheteur de se garantir le cours d'achat d'un actif dans l'avenir, sans pour autant qu'il s'engage à acheter ou à céder cet actif. Le vendeur d'une telle option s'engage à vendre le sous-jacent au prix fixé, si l'acheteur de l'option exerce son droit. Le prix à la maturité est évident : si l'actif vaut plus que K, le possesseur de l'option exerce son droit d'achat à K et revend immédiatement à S et gagne la différence $S - K$. Par contre, si le prix de l'actif est inférieur à K, l'option n'a pas de valeur ! La condition initiale (finale) pour l'EDP est donc :

$$V(S, T) = \max(S - K, 0)$$

La résolution de cette équation donne le prix de l'option à $t = 0$ (aujourd'hui) et pour le spot S_0 d'aujourd'hui. Des solveurs numériques de cette équation, qu'on appelle **pricers**, sont utilisés tous les jours par les traders.

5.1.2 Volatilité implicite et smile

Le modèle de Black-Scholes permet ainsi de calculer le prix d'un call connaissant le prix du sous-jacent S, K le prix d'exercice de l'option, r le taux d'intérêt sans risque, σ la volatilité, T la maturité et t le temps écoulé depuis l'émission. Tous les paramètres sont connus (K, T, t) ou observables (S, r) sauf la volatilité qui doit être estimée. Le modèle de Black-Scholes suppose une volatilité constante or *cela n'est absolument pas le cas dans la réalité.*

À partir, des prix observées sur le marché dans le passé, on peut déduire une première estimation de la volatilité à partit de l'équation différentielle stochastique (5.2) : si S_t est la prix de l'action à l'instant t, on pose R_t la variation de $\ln S_t$:

$$R_t = \Delta \ln S_t = \ln S_t - \ln S_{t-1} = \ln \frac{S_t}{S_{t-1}}$$

Pour de petites variations, on a

$$R_t \approx \frac{\Delta S_t}{S_t}$$

La volatilité d'une période élémentaire (en général quotidienne) est alors l'écart type de cette valeur R_t. En appelant μ_t la moyenne des n dernières valeurs de R_t :

$$\mu_t = \frac{1}{n}\sum_{k=1}^{n} R_{t-k+1}$$

La **volatilité historique** est définie comme l'écart type des rendements du sous-jacent :

$$\sigma_t^2 = \frac{1}{n}\sum_{t=1}^{n}(R_t - \mu_t)^2$$

On peut estimer la volatilité d'une autre manière en partant de l'EDP. Il est possible, connaissant le prix de marché du call de strike K et de maturité T de déduire une valeur unique pour la volatilité σ par inversion de la formule de Black-Scholes. Cette valeur de volatilité est appelée **volatilité implicite** et elle peut s'écarter notablement de la volatilité historique car elle est censé refléter la volatilité future anticipée par le marché.

De plus, on peut calculer à partir des prix de calls de même maturité, et de prix d'exercice différents calculer les valeurs de la volatilité implicite. On obtient une courbe convexe assez caractéristique que l'on appelle le smile de volatilité. En fait, on constate que la volatilité dépend aussi de la maturité !

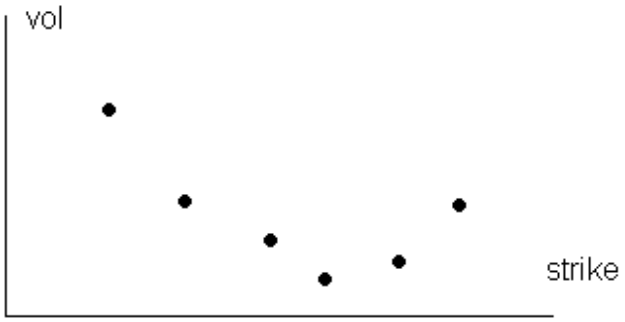


FIG. 5.1 – Smile.

On trace ainsi des surfaces de volatilité (en K et T). Ceci révèle une inconsistance du modèle de Black-Scholes (qui suppose une volatulité constante) que les praticiens et les chercheurs académiques essaient de résoudre depuis une quinzaine d'années. De multiples modèles de smile ont été développés, par exemple les modèles se basant sur une volatilité locale : $\sigma(S,t)$. Dans ce cas, B. Dupire propose une formule explicite pour résoudre le problème

inverse qui permet de déduire la volatilité locale en partant des valeurs de marché de la volatilité implicite (voir [145]). Dans la pratique, cette formule comportant des dérivées secondes de la surface de volatilités implicites n'est pas très utiles : les valeurs de marché sont souvent incomplètes, et l'utilisateur est confronté à un problème d'interpolation et d'extrapolation puis de réglarisation de la surface de valatilités implicites. D'autre part, la notion de volatilité locale n'est pas très parlante ni intuitive pour le praticien. Enfin, cette approche ne rend pas bien compte du smile pour des maturités longues. D'autres approches permettent de résoudre le problème en considérant la volatilité comme stochastique, ou bien en rajoutant des sauts Poissoniens au processus de diffusion du sous-jacent de Black-Scholes (voir Merton [147]). Dans tous ces modèles de smile, on arrive à plus ou moins bien bien calibrer le modèle, mais le problème non-résolu reste la couverture de risque lors de l'utilisation du modèle pour «pricer» un nouvel intrument financier. Récemment, Philippe Henrotte a développé à ITO33, un nouveau modèle de marché incomplet qui permet de décrire le smile et de donner une couverture dynamoque du risque. Il s'agit très grossièrement d'un modèle comportant plusieurs régimes de sauts-diffusion avec sauts entre les régimes. Nous renvoyons le lecteur intéressé à [143] et [146] ainsi que la page web `http://www.ito33.com/theory` pour plus de détails.

5.2 Calibration

La calibration d'un tel modèle avec des données de marché est un problème d'optimisation qui doit être réguliement résolu. Il s'agit d'une part de retrouver la meilleure solution possible et éventuellement en proposer plusieurs au praticien. C'est souvent un problème mal conditionné difficile à résoudre.

La fonction à optimiser est la somme des carrés des différences entre les prix constatés sur le marché et les prix fournis par le modèle au cours du processus d'optimisation des paramètres du modèle :

Étant donnés :

- un ensemble de maturités/strikes $(T_i, K_i)_{i \in I}$,
- un ensemble de prix $(V_i)_{i \in I}$ constatés sur le marché,

trouver un jeu de paramètres $p \in \mathbb{R}^N$ tel que la solution du modèle "collent au mieux" aux prix V_j pour les maturités/trikes (T_j, K_j).

On introduit la fonction coût :

$$\mathcal{J}(p) = \sum_{i \in I} w_i (V(T_i, K_i) - V_i)^2$$

w_i étant des poids positifs dépendants du degré de confiance dans les données.

On résout le problème de minimisation :

$$\inf_p \mathcal{J}(p)$$

Nous avons testé nos méthodes hybrides SDM-ES et SDM-CMA sur un cas avec 18 paramètres.

5.3 Résultats

L'algorithme SDM-ES est nettement meilleur qu'un ES pure. Sachant qu'on pourrait facilement paralléliser SDM-ES nous pourrons sans aucun doute concidérer que l'approche SDM-ES est plus prometteuse que d'utiliser un algorithme deterministe tel L-BFGS-B !

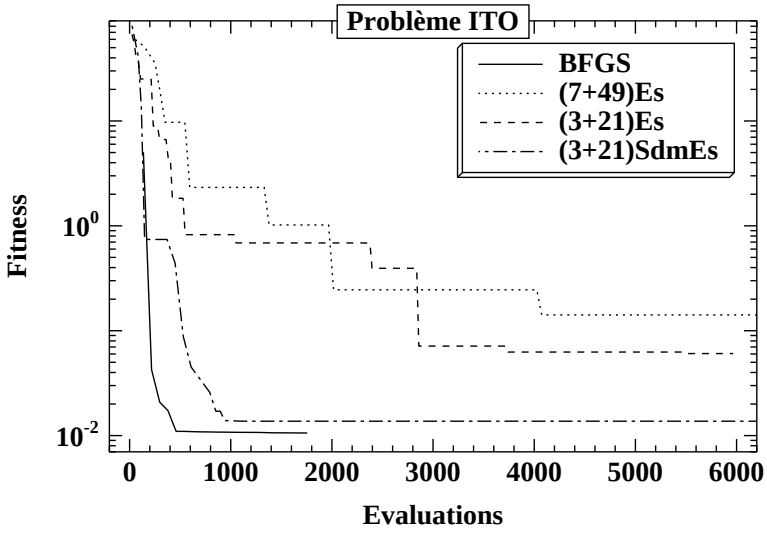


FIG. 5.2 – Comparaison entre ES isotropique, BFGS(le gradient = 2* évaluation) et SDM

Chapitre 6

Conclusion et perspectives

6.1 Discussion

Dans cette thèse, nous avons étudié l'hybridation de méthodes d'approximation et d'optimisation déterministes avec des méthodes d'optimisation stochastiques. Étant donnés des points (x_i) et des valeurs $f(x_i)$ (éventuellement $f'(x_i)$) en entrée, l'algorithme déterministe commence par proposer une approximation $\tilde{f}$ de la fonction f dans un domaine localisé autour des points d'apprentissage, basée sur les méthodes SVM ou Krigeage, puis trouve le minimum de $\tilde{f}$. Cette procédure est utilisée comme mutation déterministe, que nous avons baptisé SDM (*Surrogate Deterministic Mutation*, au sein d'un algorithme évolutionnaire comme les Stratégies d'Évolution (ES). On parle d'un algorithme hybride SDM-ES.

Nous avons commencé par étudier l'opérateur SDM isolément, puis au sein d'un algorithme itératif de base pour essayer de dégager quels sont les propriétés importantes de la méthode d'approximation, et essayer des heuristiques pour le choix des paramètres qui y interviennent. Lors de l'hybridation avec les algorithmes évolutionnaires, l'adaptation des paramètres se fait de façon dynamique à partir de ces heuristiques.

Une première difficulté concernant l'hybridation est l'adaptation de la taille du domaine d'approximation. Nous l'avons lié au déplacement effectué dans la dernière mutation réussie et au rayon du domaine précédent (loi $\alpha-\beta$). Cette approche est bien adaptée pour les fonctions bien-conditionnées. Le déplacement après mutation reflète directement la difficulté dans l'apprentissage. La diminution imposée par la loi d'adaptation ajuste le domaine pour améliorer la partie d'apprentissage. L'interaction avec ES se fait au niveau de l'adaptation du paramètre de mutation de l'algorithme ES. Le choix d'une ES isotropique avec un écart-type de l'ordre du rayon du domaine précédent

a permis d'améliorer les résultats de ES. L'interaction ES-(modèle approché) se fait ainsi premièrement par le choix des points générés avec les mutations ES et deuxièmement en prenant en compte dans la loi d'adaptation du domaine d'approximation le déplacement des itérés successifs d'une génération à l'autre. Ainsi si une mutation ES réussie avec un point loin de la zone où on applique l'apprentissage alors l'adaptation du domaine autour du nouveau point s'apparente à une ré-initialisation du rayon du domaine d'approximation.

La deuxième difficulté concerne la méthode d'approximation. Voulant travailler avec le minimum d'informations sur f, nous avons étudié des approximations globales (ne faisant pas appel au gradient). Voulant travailler en grande dimension d'une part, et prendre en compte les points déjà évalués par ES d'autre part, nous avons supposé dès le départ que les points pouvaient être n'importe où, ce qui élimine pas mal de méthodes d'approximation. Le choix de méthodes à noyaux s'impose de lui-même. La méthode SVM offre plus de liberté que le Krigeage : en proposant une relaxation du problème de régression on peut éventuellement trouver un meilleur optimum. Le noyau Gaussien s'est avéré beaucoup plus facile à utiliser que le noyau sigmoïde. On peut ensuite calibrer le modèle, *i.e.* trouver de «bons» paramètres en fonction de plusieurs critères. Il est apparu que le meilleur critère est celui qui permet la meilleure amélioration de la fonction objectif, mais il peut être coûteux à mettre au point. Pour diminuer le coût de calibration du modèle, il est important de savoir adapter les paramètres d'une itération à l'autre : on cherche ainsi à «hériter» les bons paramètres de l'étape précédente. Nous effectuons de petits déplacements avec la mutation SDM et nous utilisons des points déjà évaluées et utilisés dans l'apprentissage précédent. Les paramètres de régularités et les paramètres du noyau Gaussien hérités de la recherche précédente ont permis d'améliorer considérablement la convergence. D'autre part, les paramètres du noyau gaussien optimaux sont de l'ordre d'une valeur heuristique (le nombre de sphere de rayon sigma qu'on peut placer dans un cube unité), ce qui accélère l'étape de calibration. Cela a facilité considérablement l'adaptation du modèle. La question reste ouverte pour le noyau sigmoïde. Les courbes statistique n'ont pas permis de dégager une loi heuristique. Cela suggère que la distribution des points a une grande influence sur le résultat. Les paramètres hérités n'ont pas permis d'amélioration la recherche suivante.

Une des limites de notre approche c'est la difficulté à optimiser les fonctions mal-conditionnées. La loi adaptation du domaine, si on cherche à approcher la fonction dans un domaine carré, tend à diminuer dramatiquement le rayon ! Et l'algorithme converge prématurément ou devient très lent. D'autre part, l'approximation avec un noyau isotrope, comme le noyau Gaussien,

risque de ne pas de fonctionner, sauf avec petit rayon et beaucoup de points ! Dans ce cas, l'utilisation du noyau sigmoïde peut être intéressante. Une bonne distribution des points d'apprentissage est également cruciale. Pour résoudre ce problème, nous nous sommes penchés sur l'algorithme CMA qui a des performances supérieures aux algorithmes classiques comme ES dans le cas des fonctions mal-conditionnées en dimension raisonnable. L'hybridation de SDM avec CMA parait donc intéressante.

6.2 Vers un SDM non-isotrope

Toutes les méthodes proposées dans cette thèse mettent en jeu des procédures isotropes sur l'espace de recherche : les Stratégies d'Évolution utilisées sont les ES isotropes standard, les noyaux gaussiens des SVM d'interpolation ont une matrice de covariance identité, et les points additionnels éventuellement tirés pour compléter l'ensemble d'apprentissage est fait de maniére isotrope. Or, comme nous l'avons discuté dans la section précédente, dans le cadre de fonctions mal conditionnées, cela peut nuire à la recherche de l'optimum. C'est pourquoi nous proposons plusieurs pistes possibles pour de futures recherches afin de pallier ces problèmes.

Tout d'abord, nous exposons l'hybridation avec l'algorithme CMA-ES et les problémes spécifiques posés – donnant des pistes pour leur résolution.

Ensuite, nous présentons la problématique du choix des paramètres du noyau sigmoïde, beaucoup plus délicat que celui des paramètres du noyau Gaussien utilisé dans cette thèse. En effet, le noyau sigmoïde privilégie naturellement certaines directions : cela le rend certes plus difficile à régler, comme l'ont montré les résultats du chapitre 3, mais cela peut justement devenir un avantage dans le cas des fonctions mal conditionnées.

Enfin, nous discutons des différents méthodes de génération de points supplémentaires, et proposons une méthode possible dans le cadre de l'hybridation avec l'algorithme CMA-ES.

6.2.1 Hybridation CMA-ES

L'hybridation de la mutation SDM avec l'algorithme CMA-ES, qui est en train de prendre le meilleur sur les Stratégies d'Évolution plus classiques (voir section 1.4) semble prometteuse pour plusieurs raisons. La première est évidemment les meilleures performances obtenues par CMA-ES par rapport à ES. Mais dans la perspective particulière de SDM, les premiers tests effectués sur la fonction Rosenbrock en dimension 30 montrent l'intérêt de cette approche. En effet le tirage des points d'apprentissage qui s'effectue à partir de

la matrice de covariance calculée par l'algorithme CMA-ES semble améliorer grandement la qualité du modèle obtenu, et, partant, le comportement global de l'algorithme hybride. Plusieurs problèmes spécifiques se posent dans cette hybridation.

Adaptation de la matrice de covariance

Soient $\mathbf{C} = \mathbf{B}\mathbf{B}^t$ la matrice de covariance de la méthode CMA-ES, B étant la matrice de vecteurs propre de $\mathbf{C}$, de valeurs propres $(\lambda_i)_{i\in[1,n]}$. Si M est un point à muter avec SDM, et $M^{'}$ e résultat de la mutation, alors $\mathbf{MM}^{'}$ représente $\delta\,\mathbf{B}\,\mathbf{z}$, et on a

$$\mathbf{B}^T\,\mathbf{B}\,\mathbf{z} = diag(\lambda_1 ... \lambda_n)\,\mathbf{z}$$

et donc

$$\mathbf{z} = \frac{1}{\delta} diag(\frac{1}{\lambda_1} ... \frac{1}{\lambda_n})\,\mathbf{B}^T\,\mathbf{MM}^{'}$$

On réécrit alors les mises à jour avec les relations décrites dans 1.4.2.

Expérimentalement nous avons remarqué que dans le cadre de l'hybridation de SDM avec CMA-ES, la valeur de δ a tendance à toujours augmenter : en effet, $\mathbf{MM}^{'}$ n'est pas alors une variable aléatoire, mais le résultat d'une mutation déterministe. De plus l'adaptation adopté dans l'algorithme CMA "pur" dépend du déplacement lors de la dernière mutation : si ce déplacement est plus grand que la moyenne de la norme d'une variable aléatoire normale de pas δ, alors on augmente δ avec une loi logarithmique, sinon on le diminue. Cette loi de mise à jour ne prend évidemment pas en compte directement la partie apprentissage. Pour remédier à cela, nous proposons d'imposer une borne supérieur dynamique qui dépendrait par exemple de la moyenne sur quelques itérations précédentes des δ correspondant à des mutations réussies. Il s'agirait alors d'une interaction entre l'opérateur d'apprentissage et l'adaptation de δ voir 6.2.1. Un premier résultat en dimension 50 est très encourageant : voir Figure 6.2.1, oú nous avons choisi le noyau Gaussien avec adaptation dynamique des paramètres en utilisant 10 points tirés par l'algorithme CMA-ES pour la partie apprentissage, et 5 évaluations supplémentaires de la fonction objecif pour la partie calibrage de l'apprentissage.

Remarquons enfin que dans le cadre de l'hybridation SDM-CMS-ES proposée, comme la méthode CMA utilise de manière déterministe le chemin

parcouru par l'algorithme, on peut penser que l'accélération due à l'utilisation de SDM permettra également l'amélioration de l'apprentissage de la matrice de covariance, comme on peut le constater sur la figure 6.2.1. Bien sûr, seule une analyse plus detaillée des résultats, avec en particulier l'observation de la matrice de covariance apprise, pourra permettre de se faire une idée exacte des contributions de chacune des parties de l'algorithme hybride ainsi obtenu.

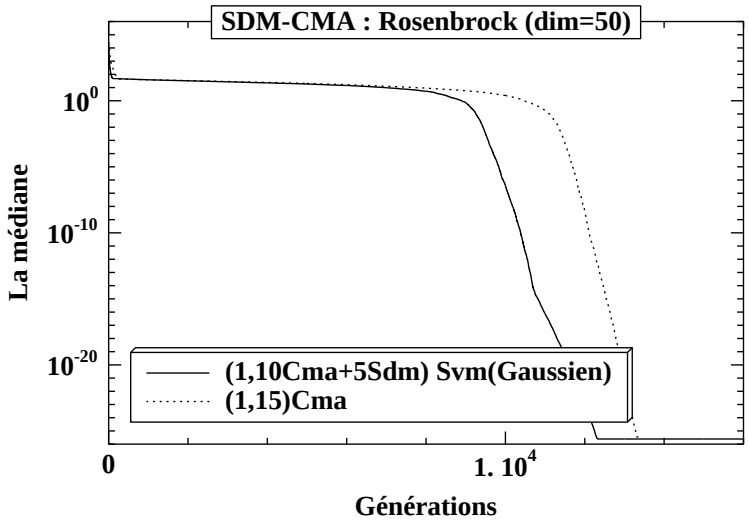


FIG. 6.1 – Couplage SDM-CMA : Adaptation dynamique des paramètres dans le cas d'un noyau gaussien. 10 évaluations sont utilisées pour définir le modèle, et 5 pour le calibrer. Comparaison avec l'algorithme CMA standard.

6.2.2 Noyau Sigmoïde ou noyau Gaussien ?

Le problème de l'isotropie au niveau de l'apprentissage par SVM peut être résolu de deux manières différentes : soit en utilisant des noyaux gaussiens dont la matrice de covariance serait une matrice symétrique définie positive quelconque, soit en utilisant des noyaux sigmoïdaux.

Les noyaux gaussiens non isotropes

Le problème posé par l'utilisation de noyaux gaussiens non isotropes est celui du choix de la matrice de covariance. En effet, le nombre de points d'apprentissage disponibles ne permet par en général, et surtout en grande dimension, d'apprendre correctement des directions privilégiées.

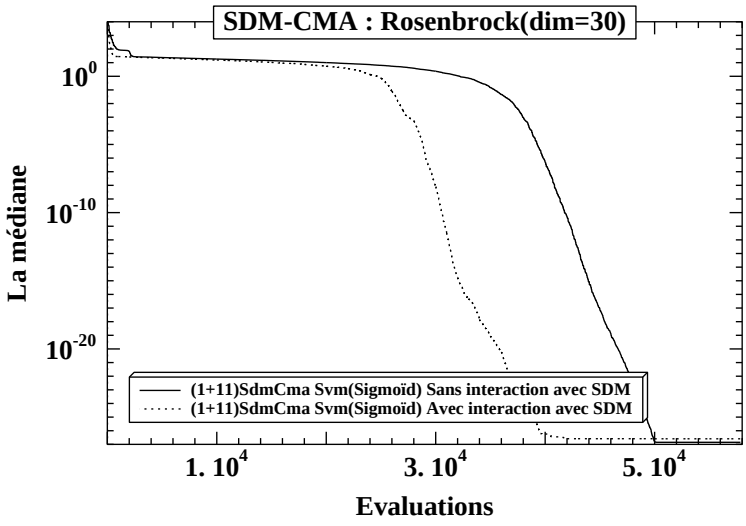


FIG. 6.2 – Importance d'une bonne adaptation du pas δ. La courbe "Sans interaction" utilise la règle logarithmique du CMA pur. La courbe "Avec interaction" est la loi avec une borne supérieur dynamique qui est la moyenne sur les dix dernière itérations des δ de mutation SDM réussie.

Par contre, si l'algorithme évolutionnaire auquel est couplé SDM apprend déjà une matrice de covariance, celle-ci peut alors être utilisée pour les noyaux gaussiens de la méthode SVM. On peut ainsi envisager soit l'utilisation des Stratégies d'Évolution dites "corrélées", dans lesquelles une matrice de covariance complète évolue avec chaque individu (voir chapitre 2), soit le couplage avec la méthode CMA-ES déjà décrit dans la section précédente.

Les noyaux sigmoïdaux

D'autre part, bien que le noyau sigmoïde n'ait pas donné de bons résultats par rapport au noyau gaussien pour des fonctions bien-conditionnées, dans le cas de la fonction Rosenbrock nous remarquons que le noyau sigmoïde pourrait devenir compétitif (voir Figure 6.2.2). En effet, l'accélération constatée pourrait provenir de la faculté des noyaux sigmoïdes à privilégier certaines directions, ce que ne permet pas un noyau isotrope comme la Gaussienne. Les tests effectués sur la fonction Rosenbrock vont dans ce sens, le noyau gaussien est moins adapté que le noyau sigmoïde. Il semble prometteur d'étudier le noyau Sigmoïde plus en détail, en particulier au niveau du réglage dynamique des paramètres pour lequel aujourd'hui nous n'avons pas de méthode à proposer.

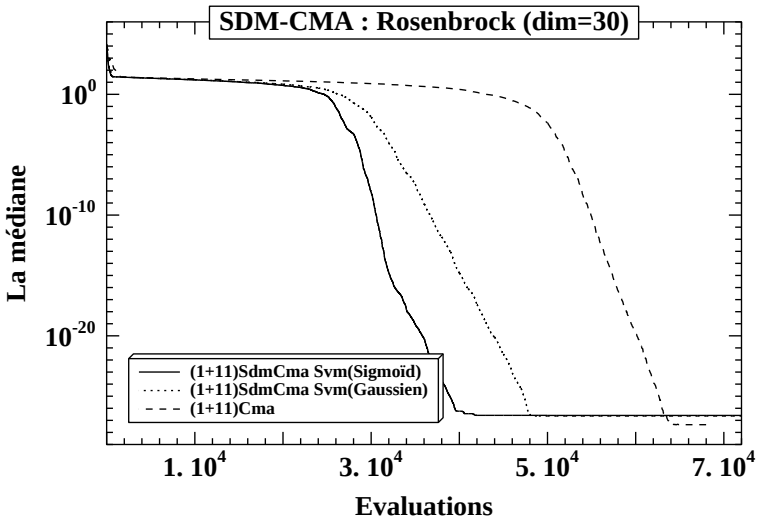


FIG. 6.3 – Le noyau sigmoïde est plus adapté que le noyau gaussien dans le cas des fonctions mal-conditionnée. Les courbes de convergence ne prennent pas en compte les évaluations effectuées pour adapter les paramètres ! En effet c'est pour montrer l'objectif à atteindre en améliorant l'adaptation des paramètres du noyau.

Bibliographie

[1] Auger, Anne - Le Bris, Claude - Schoenauer, Marc Rapport de recherche de l'INRIA-Rocquencourt, Equipe : FRACTALES

[2] A.Auger, C. Le Bris, M.Schoenauer. Dimension-independent convergence rate for non-isotropic (1, ?)-ES. In Erick Cantu-Paz et al. editor, volume 2723, pages 512-524. Springer Verlag, 2003. 21 pages - Août 2003 - Document en anglais

[3] Josselin Garnier, Leila Kallel. Efficiency of Local Search with Multiple Local Optima. SIAM J. Discrete Math. 15(1) : 122-141 (2001)

[4] Fred J. Hickernell, Ya-Xiang Yuan A simple Multistart Algorithm for Global Optimization. Dec, 1997 OR Transactions Vol.1 No.2.

[5] Ramesh A. DANDEKAR, Michael COHEN, Nancy KIRKENDALL Applicability of Latin Hypercube Sampling to Create Multi Variate Synthetic Micro Data.

[6] A.S. Hedayat, Neil J. A. Sloane, John Stufken Orthogonal Arrays : Theory and Applications (Springer Series in Statistics)

[7] Eiben, A.E., Smith, J.E. Introduction to Evolutionary Computing Series : Natural Computing Series 2003, XV, 299 p. 102 illus., Hardcover, ISBN : 3-540-40184-9

[8] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. Science, (220) :671-680, 1983

[9] Rechenberg, Ingo, Evolutions Strategie '94, Friedrich Frommann Holzboog Verlag, 1994.

[10] Nikolaus Hansen and Andreas Ostermeier Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies : The Covariance Matrix Adaption, Proceedings of the 1996 IEEE Intern. Conf. on Evolutionary Computation (ICEC '96) 1996, IEEE Press, 312-317

[11] Hansen, Nikolaus, Verallgemeinerte individuelle Schrittweitenregelung in der Evolutionsstrategie. Eine Untersuchung zur entstochastisierten, koordinatensystemunabhängigen Adaptation der Mutationsverteilung, Mensch und Buch Verlag, ISBN 3-933346-29-0, 1998, PhD. thesis.

[12] Hansen , N. and A. Ostermeier (2001). Completely Derandomized Self-Adaptation in Evolution Strategies. Evolutionary Computation, 9(2), pp. 159-195 ;

[13] Schwefel, Hans-Paul, Evolution and Optimum Seeking, John Wiley & sons, 1995, New York.

[14] Bäck, Thomas, Evolutionary Algorithms in Theory and Practice, Oxford University Press, 1996.

[15] Thomas Bäck and Frank Hoffmeister, Extended Selection Mechanisms in Genetic Algorithms, "92–99","ICGA91","ls11.informatik.uni-dortmund.de icga91-2.ps.gz"

[16] Thomas Bäck" "Evolutionary Algorithms in Theory and Practice", "Doctoral dissertation", "University of Dortmund, Department of Computer Science", "Germany", feb, 1994.

[17] "Andreas Ostermeier", "An Evolution Strategy with Momentum Adaptation of the Random Number Distribution","PPSN92" "197–206"

[18] "Andreas Ostermeier", "A Derandomized Approach to Self Adaptation of Evolution Strategies", "Evolutionary Computation", 1994, 2, 4, "369–380",

[19] "Hans–Paul Schwefel", "Numerische Optimierung von Computer–Modellen mittels der Evolutions strategie", "Birkhäuser", 1977, 26, "Interdisciplinary Systems Research", "Basel, Germany"

[20] "Nikolaus Hansen and Andreas Ostermeier and Andreas Gawelczyk", "On the adaptation of arbitrary normal mutation distributions in evolution strategies : The generating set adaptation", "57–64" "ICGA95"

[21] I. Rechenberg Evolution Strategie : Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution, 1973, Fromman-Hozlboog Verlag, Stuttgart.

[22] H.-P. Schwefel Numerical Optimization of Computer Models 1981, John Wiley & Sons, New-York, 1995 – 2^{nd} edition

[23] Th. Bäck and H. P. Schwefel, Evolutionary Computation : An Overview, ICEC96, 1996, ICEC96editors, IEEE Press, Piscataway NJ, pages 20-29

[24] H.-P. Schwefel, Collective phenomena in evolutionary systems, 1987, 31^{st} annual meeting of the int'l society for general system research, Budapest, volume 2, pages 1025-1033

[25] Jürgen Branke. Evolutionary Optimization in Dynamic Environments. Book series Genetic Algorithms and Evolutionary Computation, Kluwer Academic Publishers, 2001.

[26] N. Saravanan and D. B. Fogel Learning strategy parameters in evolutionary programming : an empirical study, EP94, 1994, EP94editors, San Diego, CA, pages 269-280

[27] N. Saravanan and D. B. Fogel and K. M. Nelson, A Comparison of methods for self-adaptation in Evolutionary Algorithms, Biosystems 1995, volume 36, pages 157-166.

[28] A. Petrowski and S. Ben Hamida, A logarithmic mutation operator to solve constrained optimization problems, GECCO99editors, GECCO99, 1999, Morgan Kaufmann

[29] S. Ben Hamida and A. Petrowski, The need for improving the exploration operators for constrained problems", CEC2000editors, CEC2000, 2000, Springer Verlag, LNCS, San Diego, USA, pages 1176-1183

[30] Nair, P. B., Keane, A. J., and Shimpi, R. P., Combining Approximation Concepts With Genetic Algorithm-Based Structural Optimization Procedures AIAA Paper 98-1912.

[31] M.A. El-Beltagy, P.B. Nair, and A.J. Keane. Metamodeling techniques for evolutionary optimization of computationally expensive problems : Promises and limitations. In D.E. Goldberg & al., editor, *Proceedings of the Genetic and Evolutionary Conference 99*, pages 196–203. Morgan Kaufmann, 1999.

[32] Mohammed A. El-Beltagy and Andy J. Keane Evolutionary Optimization for Computationally expensive problems using Gaussian Processes.

[33] Ong, Y., S., Nair, P., B. and Keane, A., J. Evolutionary optimization of computationally expensive problems via surrogate modeling AIAA Journal

[34] Andras Sobester, Stephen Leary and Andy Keane. A parallel Updating Scheme for Approximating and Optimizing High Fidelity Computer Simulations Third ISSMO/AIAA Internet Conference on Approximations in Optimization, October 14-18, 2002

[35] Yaochu Jin, Markus Olhofer and Bernhard Sendhoff. On evolutionary optimisation with approximate fitness functions. In Proceedings of the Genetic and Evolutionary Computation Conference GECCO, pages 786-793, Morgan Kaufmann, 2000.

[36] Yaochu Jin, Markus Olhofer and Bernhard Sendhoff. A Framework for Evolutionary Optimisation with Approximate Fitness Functions. IEEE Transactions on Evolutionary Computation, 6(5), pages 481-494, 2002.

[37] Lars Willmes, Thomas Bäck, Yaochu Jin and Bernhard Sendhoff. Comparing neural networks and Kriging for fitness approximation in evolu-

tionary optimization. In Congress on Evolutionary Computation (CEC), IEEE Press, 2003.

[38] Michael Husken and Yaochu Jin and Bernhard Sendhoff. Structure Optimization of Neural Networks for Evolutionary Design Optimization. In Proceedings of the Genetic and Evolutionary Computation Conference GECCO - Workshop, Morgan Kaufmann, 2002.

[39] Yaochu Jin, Markus Olhofer and Bernhard Sendhoff. Managing Approximate Models in Evolutionary Aerodynamic Design Optimization. In Congress on Evolutionary Computation (CEC), IEEE Press, volume 2, pages 592-599, IEEE Press, 2001.

[40] Michael Emmerich, Alexios Giotis, Mutlu Ozdemir, Thomas Back, Kyriakos Giannakoglou Metamodel-Assisted Evolution Strategies. PPSN 2002 : 361-370

[41] Ulmer, H., Streichert, F. and Zell, A., Model-Assisted Steady-State Evolution Strategies" Genetic and Evolutionary Computation – GECCO-2003 (Conference paper)

[42] D. Buche, N. N. Schraudolph, and P. Koumoutsakos, Accelerating Evolutionary Algorithms with Gaussian Process Fitness Function Models , IEEE Transactions on Systems, Man and Cybernetics, Part C, Special Issue on Knowledge Extraction and Incorporation in Evolutionary Computation, 2004

[43] Jin, R.; Chen, W.; Simpson, T. W. Comparative Studies of Metamodeling Techniques under Multiple Modeling Criteria. Proceedings of the 8th AIAA, USAF, NASA, ISSMO Multidisciplinary Analysis & Optimization Symposium, AIAA 2000-4801, Long Beach, CA, 6-8 Sept.

[44] Simpson, T.W., Comparison of Response Surface and Kriging Models in the Multidisciplinary Design of an Aerospike Nozzle, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center : Hampton, VA, February 1998.

[45] T. Simpson, T. Mauery, J. Korte, and F. Mistree. Comparison of response surface and Kriging models for multidiscilinary design optimization. Technical Report 98-4755, AIAA, 1998.

[46] Srivastava, A., Hacker, K., Lewis, K., and Simpson, T., 1999, Development of a Kriging Based Surrogate Approximation Method for LargeScale Systems Design Optimization, International Journal for Product and Process Improvement, submitted.

[47] Timothy W. Simpson, Timothy M. Mauery, John J. Korte and Farrokh Mistree Kriging Models for Global Approximation in Simulation-Based

Multidisciplinary Design Optimization AIAA01 Vol. 39, No. 12, December 2001

[48] Srivastava, A., Hacker, K., Lewis, K., Investigation of Different Approximation Techniques in the Design of the High Speed Civil Transport Aircraft, May 1999, The Third World Congress of Structural and Multidisciplinary Optimization, University at Buffalo, Buffalo, NY, to appear in proceedings.

[49] Koch, P. N., Mavris, D., and Mistree, F., 1998, Multi-Level, Partitioned Response Surfaces for Modeling Complex Systems to be presented at 7th Symposium on Multidisciplinary Analysis and Optimization, St. Louis, Missouri, AIAA-98-4958.

[50] Koch, P. N., Simpson, T. W., Allen, J. K., and Mistree, F., 1998 Statistical Approximations for Multidisciplinary Design Optimization : The Problem of Size, AIAA Special MDO Issue, (accepted for publication).

[51] Jay D. Martin and Timothy W. Simpson 2002. Use of adaptive metamodeling for design optimization, AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 2002, Atlanta, Georgia.

[52] Timothy W. Simpson, ndrew J. Booker, Dipankar Ghosh, Anthony A. Giunta, Patrick N. Koch and Ren-Jye Yang. Approximation Methods in Multidisciplinary Analysis and Optimization : A Panel Discussion. Discussion at the Approximation Methods Panel that was held at the 9^{th} AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 2002, Atlanta, Georgia.

[53] G. Gary Wang and Timothy W. Simpson. Fuzzy clustering based hierarchical metamodeling space reduction design optimization Copyright © 2002 by G. Wang and T. Simpson, Published by the American Institute of Aeronautics and Astronautics, Inc 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization 4-6 September 2002, Atlanta, Georgia AIAA 2002-557

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

[55] N. Alexandrov, J.E. Dennis, Jr., R.M. Lewis and V. Torczon, A trust region framework for managing the use of approximation models in optimization. ICASE Report 97-50, ICASE, NASA Langley Research Center, Hampton, 1997.

[56] A. J. Booker, J. E. Dennis, Jr., P. D. Frank, D. W. Moore, and D. B. Serafini. Managing surrogate objectives to optimize a helicopter

rotor design - further experiments. In Proceedings of the Seventh AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 1998.

[57] A.J. Booker, J.E. Dennis, P.D. Frank, D. B. Serafini and V. Torczon. Optimization using surrogate objectives on a helicopter test example. Optimal Design and Control, J.Borggard, E. M. Cliff, and S. Schreck (eds), Birkhauser, Boston, 1998, pp. 49-58.

[58] Virginia Torczon and Michael W. Trosset. Using approximations to accelerate engineering design optimization Copyright © 1998, Appears in the Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, Missouri, September 2-4, 1998, as AIAA Paper 98-4800. Also released as ICASE Technical Report 98-33.

[59] Andrew J. Booker, J. E. Dennis, Jr., Paul D. Frank, David B. Serafini, and Virginia Torczon. Optimization using surrogate objectives on a helicopter test example, Copyright © 1998 Birkhauser,Appears in Computational Methods in Optimal Design and Control, edited by Jeff Borggaard, John Burns, Eugene Cliff, and Scott Schreck, Birkhauser, Boston, 1998, pages 49-58.

[60] Natalia Alexandrov, J. E. Dennis, Jr., Robert Michael Lewis, and Virginia Torczon. A trust region framework for managing the use of approximation models in optimization Copyright © Springer-Verlag 1998, Revision 1.12, 1998/01/13 22 :20 :41, Appears in Structural Optimization, Vol. 15, No. 1, February 1998, pages 16-23. Originally released as ICASE Technical Report 97-50.

[61] Michael W. Trosset and Virginia Torczon, Numerical optimization using computer experiments. Copyright © 1997, ICASE Technical Report 97-38

[62] J. E. Dennis, Jr. and Virginia Torczon Managing approximation models in optimization. Copyright © 1996 American Institute of Aeronautics and Astronautics. All Rights Reserved. Appears as a "Work in Progress" paper in Proceedings of the 6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Bellevue, Washington, September 4-6, 1996.

[63] J. E. Dennis and Virginia Torczon. Managing approximation models in optimization. Copyright © 1997 Society for Industrial and Applied Mathematics. Appears in Multidisciplinary Design Optimization : State of the Art, edited by Natalia M. Alexandrov and M. Y. Hussaini, SIAM, 1997, pages 330-347.

[64] Christopher M. Siefert and Virginia Torczon Model-Assisted Pattern Search Methods for Optimizing Expensive Computer Simulations.

[65] Siefert C. M. MOdel-assisted pattern search. Senior honors thesis, Departement of Computer Science, College of William & Mary, Williamsburg, VA. Available with software at http ://www.cs.wm.edu/ va/.

[66] Anthony A. Giunta and Michael S.Eldred Implementation of a trust region model management strategy in the dakota optimization toolkit. AIAA-2000-4935

[67] John Dennis, Mark Abramson, Charles Audet, Gilles Couture, Optimization Using Surrogates for Engineering Design

[68] Alison L. Marsden, Meng Wang, John E. Dennis, Jr. and Parviz Moin. Optimal aeroacoustic shape design using the surrogate management framework. November 13, 2003

[69] Pérez, V.M., Renaud, J.E., 2000, "Decoupling the Design Sampling Region from the Trust Region in Approximate Optimization", in Proceedings of the International Mechanical Engineering Congress & Exposition , Orlando, FL, November 5-10.

[70] Pérez, V.M., Renaud, J.E., Gano, S.E., 2000, "Constructing Variable Fidelity Response Surface Approximations on the Usable Feasible Region", Proceedings of the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidsciplinary Analysis & Optimization, ,AIAA 2000-4888, Long Beach, CA, September 6-8

[71] Pérez, V. M., Renaud, J.E., Watson, L. T., 2001, "Adaptive Experimental Design for Construction of Response Surface Approximations", AIAA Journal, volume40, No 12, pp. 2495-2503.

[72] Rodriguez, J.F., Pérez, V. M., Padmanabhan,D., Renaud, J.E., 2001 : "Sequential Approximate Optimization Using Variable Fidelity Response Surface Approximations", Structural and Multidisciplinary Optimization , volume22, pp. 24-34. Published by Springer-Verlag, Germany.

[73] Pérez, V. M., Renaud, J.E., Watson, L. T., 2002, "Reduced Sampling for Construction of Quadratic Response Surface Approximations Using Adaptive Experimental Design", Proceedings of the 43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA 2002-1587, Denver, CO, April 22-25.

[74] Victor M. Pérez, Thomas B. Apker and John E. Renaud, PARALLEL PROCESSING IN SEQUENTIAL APPROXIMATE OPTIMIZATION AIAA-2002-1589

[75] G. Gary Wang, Fuzzy Clustering Based Hierarchical Metamodeling For Space Reduction and Design Optimization newblock Journal of Engineering Optimization, accepted in Sept.35,2003.

[76] G. Wang and Z. Dong, Adaptive Response Surface Method - An Efficient Global Optimization Technique for Virtual Prototyping Based Design Optimization, Journal of Engineering Optimization, accepted in April 2001.

[77] G. Wang and Z. Dong, Design Optimization of A Complex Mechanical System Using Adaptive Response Surface Method, Transactions of the CSME, Vol. 24, No. 1B, 2000, pp. 295-306.

[78] G. G. Wang and Z. Dong, An Extension to Design of Experiment for Design Optimization with Implicit Parametric Models and Virtual Prototype, Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics, IEEE, 1998.

[79] G. Gary Wang. Adaptive Response Surface Method Using Inherited Latin Hypercube Design Points Transactions of the ASME, Journal of Mechanical Design, Vol. 125, pp. 210-220, June 2003.

[80] Chung, H.S. and Alonso, J.J., "Design of a Low-Boom Supersonic Business Jet Using Cokriging Approximation Models", 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization AIAA2002-5598, Altanta, Georgia, September 4-6, 2002.

[81] Chung, H.S. and Alonso, J.J., "Using Gradients to Construct Cokriging Approximation Models for High-Dimensional Design Optimization Problems", AIAA 40th Aerospace Sciences Meeting and Exhibit,AIAA-2002-0317, Reno, NV, January 14-17, 2002.

[82] Chung, H.S., and Alonso, J.J., "Using Gradients to Construct Response Surface Models for High-Dimensional Design Optimization Problems", AIAA 39th Aerospace Sciences Meeting and Exhibit,AIAA-2001-0922, Reno, NV, January 8-11, 2001.

[83] Chung, H.S. and Alonso, J.J. "Comparison of Approximation Models with Merit Functions for Design Optimization," AIAA/USAF/NASA/ISSMO 8th Symposium on Multidisciplinary Analysis and Optimization, AIAA-2000-4754, Long Beach, CA, September 2000.

[84] Deepti Chafekar, Liang Shi, Khaled Rasheed and Jiang Xuan. "Constrained Multi-objective GA Optimization Using Reduced Models". Submitted under review to IEEE Transactions on Systems, Man and Cybernetics.

[85] Khaled Rasheed, Xiao Ni and Swaroop Vattam. "Comparison of Methods for Developing Dynamic Reduced Models for Design Optimization". To appear in The Soft Computing Journal, 2003.

[86] Khaled Rasheed, Swaroop Vattam and Xiao Ni. "Comparison of Methods for Using Reduced Models to Speed Up Design Optimization". The Genetic and Evolutionary Computation Conference (GECCO'2002) , 2002.

[87] Khaled Rasheed, Xiao Ni and Swaroop Vattam. "Comparison of Methods for Developing Dynamic Reduced Models for Design Optimization". The Congress on Evolutionary Computation (CEC'2002), 2002.

[88] Khaled Rasheed, Haym Hirsh. "Informed operators : Speeding up genetic-algorithm-based design optimization using reduced models". Genetic and Evolutionary Computation Conference (GECCO'2000), 2000.

[89] Khaled Rasheed. "An Incremental-Approximate-Clustering Approach for Developing Dynamic Reduced Models for Design Optimization". Congress on Evolutionary Computation (CEC'2000).

[90] K.-H. Liang, X. Yao and C. Newton, "Evolutionary search of approximated N-dimensional landscapes," International Journal of Knowledge-Based Intelligent Engineering Systems, 4(3) :172-183, July 2000.

[91] K.-H. Liang, X. Yao and C. S. Newton, "Combining landscape approximation and local search in global optimization," Proc. of the 1999 Congress on Evolutionary Computation, Vol. 2, IEEE Press, Piscataway, NJ, USA, pp.1514-1520, July 1999.

[92] X. Yao, "Global optimisation by evolutionary algorithms," Proc. of the Second Aizu International Symposium on Parallel Algorithm/Architecture Synthesis (pAs-97), Aizu-Wakamatsu, Japan, 17-21 March 1997, IEEE Computer Society Press, pp.282-291.

[93] Alotto, P., Caiti, A., Molinari, G. and Repetto, M. A multiquadrics based algorithm for the acceleration of simulated annealing optimisation procedures. IEEE Trans. Magn., Vol. 32 No. 3, pp. 1198-201.

[94] Farina, M. and Sykulski, J.K. (2000), Comparative study of evolution strategies combined with approximation techniques for practical electromagnetic optimisation problems in press in IEEE Trans. Mag., 2001.

[95] Malik, Z. and Rashid, K. (2000) A comparison of optimisation by response surface methodology with neuro-fuzzy models IEEE Trans. Magn., Vol. 36 No. 1, January, pp. 241-57.

[96] Pahner, U. and Hameyer, K. (1999) Adaptive coupling of differential evolution and multiquadrics approximation for the tuning of the optimization process.

Proceedings of COMPUMAG Sapporo, 25-28 October, pp. 116-17.

[97] J-F.M. Barthelemy and R.T. Haftka. Approximation concepts for optimum structural design - a review. *Structural Optimization*, 5 :129–144, 1993.

[98] F. Bonnans, J.C. Gilbert, C. Lemarechal, and C. Sagastizábal. *Optimisation Numérique, aspects théoriques et pratiques*, volume 23 of *Mathématiques & Applications*. Springer Verlag, 1997.

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

[100] R. H. Byrd, P. Lu, and J. Noceda. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific and Statistical Computing*, 16(5) :1190–1208, 1995.

[101] R. H. Byrd, J. Nocedal, and R. B. Schnabel. Representation of quasi-newton matrices and their use in limited memory methods. *Mathematical Programming*, 63(4) :129–156, 1994.

[102] Krige, D.G. A statistical approach to some basic mine valuation problems on the Witwatersrand. 1951, J, of Chem., Metal. and Mining Soc. of South Africa, 52, 119-139.

[103] Matheron, G. 1963. Principles of Geostatics. Economic Geol., 58, 1246-1268.

[104] R. Collobert and S. Bengio. *Support Vector Machines for Large-Scale Regression Problems*. IDIAP-RR-00-17, 2000.

[105] N. Cristianini and J. Shawe-Taylor. *An introduction to Support Vector Machines*. Cambridge University Press, 2000.

[106] Chapelle, O., V. Vapnik, O. Bousquet and S. Mukherjee : *Choosing Multiple Parameters for Support Vector Machines.* Machine Learning 46(1), 131-159 (2002)

[107] Hoerl, A. E., Kennard, R. W. (1970), Ridge Regression : Biased Estimation for Nonorthogonal Problems, Technometrics, 12, 55-67

[108] P. Galinier and J. Hao. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, 3(4) :379–397, 1999.

[109] J. J. Grefenstette. Predictive models using fitness distributions of genetic operators. In L. D. Whitley and M. D. Vose, editors, *Foundations of Genetic Algorithms 3*, pages 139–161. Morgan Kaufmann, 1995.

[110] J. J. Grefenstette and J. M. Fitzpatrick. Genetic search and approximate function evaluation. In J. J. Grefenstette, editor, *Proceedings of ICGA*, pages 160–168. Laurence Erlbaum Associates, 1985.

[111] P. Husbands, G. Jermy, M. McIlhagga, and R. Ives. Two applications of genetic algorithms to component design. In T. Fogarty, editor, *AISB Workshop on Evolutionary Computing*, pages 50–61. Springer Verlag, 1996.

[112] Y. Jin, M. Olhofer, and B. Sendhoff. On evolutionary optimisation with approximate fitness functions. In D. Whitley & al., editor, *Proceedings of the Genetic and Evolutionary Conference*, pages 786–793, 2000.

[113] P. Merz and B. Freisleben. A genetic local search approach for the QAP. In Th. Bäck, editor, *Proceedings of ICGA'97*, pages 465–470. Morgan Kaufmann, 1997.

[114] P. Merz and B. Freisleben. Genetic local search for the TSP : New results. In T. Bäck, Z. Michalewicz, and X. Yao, editors, *Proceedings of ICEC'97*, pages 159–164. IEEE Press, 1997.

[115] K.-H. Liang, X. Yao, and C. Newton. Combining landscape approximation and local search in global optimization. In Proceedings of the 1999 Congress on Evolutionary Computation, volume 2, pages 1514–1520, Piscataway, NJ, 1999. IEEE Press. http ://citeseer.nj.nec.com/liang99combining.html

[116] P. Merz and B. Freisleben. Fitness landscapes and memetic algorithm design. In David Corne, Marco Dorigo, and Fred Glover, editors, *New Ideas in Optimization*, pages 245–260. McGraw-Hill, London, 1999.

[117] G. Mosetti and C. Poloni. Aerodynamic shape optimization by means of hybrid genetic algorithms. In *3rd International Congress on Industrial and APplied Mathematics*, 1995.

[118] C. Poloni and V. Pediroda. GA coupled with computationaly expensive simulations : tools to iprove efficiency. In *Genetic Algorithms and Evolution Strategies in Engineering and Computer Sciences*, pages 267–288. John Wiley, 1997.

[119] N. J. Radcliffe and P. D. Surry. Formal memetic algorithms. In T.C. Fogarthy, editor, *Evolutionary Computing*, pages 1–16. Springer Verlag LNCS 865, 1994.

[120] A. Ratle. Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In Th. Bäck, G. Eiben, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of the* 5^{th} *Conference on Parallel Problems Solving from Nature*, pages 87–96. Springer-Verlag, LNCS 1498, 1998.

[121] D. Waagen, P. Diercks, and J. McDonnell. The stochastic direction set algorithm : A hybrid technique for finding function extrema. In

D. B. Fogel and W. Atmar, editors, *Proceedings of EP'92*, pages 35–42. Evolutionary Programming Society, 1992.

[122] C. Zhu, R. H. Byrd, and J. Nocedal. L-bfgs-b, fortran routines for large scale bound constrained optimization. *ACM Transactions on Mathematical Software*, 23(4) :550–560, 1997.

[123] B. E. Boser, I. M. Guyon et V. N. Vapnik. A training algorithm fpr optimal margin classifiers. In D. Haussler, éditeur, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, 144-152, ACM Press, 1992.

[124] R. Collobert, S. Bengio. SVMTorch : Support Vector Machines for Large-Scale Regression Problems. *Journal of Machine Learning Research*, **1**(2001), 143-160.

[125] N. Cristianini et J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods.* Cambridge University Press, 2000.

[126] A. Fadda. *Étude de problèmes inverses par algorithmes d'évolution et réseaux de neurones.* Thèse, École Polytechnique, 1998.

[127] A. Fadda, M. Schoenauer. Evolutionary chromatographic law identification by recurrent neural nets. *Proceedings of the 4th Annual Conference on Evolutionary Programming*, MIT Press, mars 1995, 219-235.

[128] F. James. *Sur la modélisation mathématique des équilibres diphasiques de colonnes de chromatographie.* Thèse, École Polytechnique, 1990.

[129] F. James. *Quelques problèmes directs et inverses issus de modèles hyperboliques en chromatographie*, Document de synthèse présenté pour l'habilitation à diriger des recherches de l'Université d'Orléans, 1999.

[130] F. James, M. Sepúlveda. Parameter identification for a model of chromatographic column. *Inverse Problems*, **10**(1994), n°6, 1299-1314.

[131] Making large-scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, et A. J. Smola, éditeurs, *Advances in Kernel Methods - Support Vector Learning*, 169-184, MIT Press, 1999.

[132] *The Visualization Toolkit User's Guide.* K. M. Martin, W. A. Hoffman, C. C. Law. ISBN 1-930934-05-X. Kitware, Inc.

[133] T. M. Mitchell. *Machine Learning.* WCB/McGraw-Hill, 1997.

[134] Fast training of support vector machines using sequential mimimal optimisation. In B. Schölkopf, C. J. C. Burges, et A. J. Smola, éditeurs, *Advances in Kernel Methods - Support Vector Learning*, 185-208, MIT Press, 1999.

[135] M. Sepúlveda. *Identification de paramètres pour un système hyperbolique. Application à l'estimation des isothermes en chromatographie.* Thèse, École Polytechnique, 1992.

[136] *The Visualization Toolkit – An Object-Oriented Approach To 3D Graphics.* W. Schroeder, K. Martin, B. Lorensen. CDROM inclus. ISBN 0-13-954694-4. Prentice Hall.

[137] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag, 1995 .

[138] Beasley D., Bull D.R. et Martin R.R., *An Overview of Genetic Algorithms : Part 1, Fundamentals* University Computing, vol. 15, n2, p. 58-59, 1993a.

[139] Renders J.M., *Algorithmes génétiques et réseaux de neurones.* Paris : Hermés, 1995. BU : 006.3 REN

[140] Reineix A., Eclercy D. et Jecko B., *FDTD/genetic algoritm coupling for antennas optimization* Annales de Télécommunications, vol. 52, n9-10, 1997.

[141] Menozzi R. et Piazzi A., *HEMT and HBT Small-Signal Model Optimization Using a Genetic Algorithm* , EDMO97, Londres, p.13-18, 24-25 novembre 1997.

[142] Moosburger R., Kostrzewa C., Fischbeck G. et Petermann K., *Shaping the Digital Optical Switch Using Evolution Strategies and BPM*, IEEE Photonics Technology Letters, vol. 9, n11, p. 1484-1486, novembre 1997.

[143] E. AYACHE, Ph. HENROTTE, S. NASSAR & X. WANG, « Can anyone solve the smile problem ?», WILMOTT Magazine, pp. 78-96, January 2004.

[144] F. BLACK et M. SCHOLES, « The pricing of options and corporate liabilities », *Journal of Political Economy*, (81) :637, 1973.

[145] B. DUPIRE, « The pricing with a smile », *Risk*, 7(1) :18, 1994.

[146] Ph. HENROTTE, «The Case for Time Homogeneity», WILMOTT Magazine, pp. 71-75, January 2004.

[147] R. MERTON, « Options pricing when underlying stock returns are discontinuous », *Journal of Financial Economics*, 3(1), 1996.