Optimisation de dispositifs électromagnétiques dans un contexte d'analyse par la méthode des éléments finis

Mauricio Caldora Costa

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

|_|_|_|_|_|_|_|_|_|_|_|_|

THÈSE

pour obtenir le grade de

DOCTEUR DE L'INPG

Spécialité: Génie Électrique

préparée au
Laboratoire d'Électrotechnique de Grenoble

dans le cadre de l'École Doctorale
EEATS - Électronique, Électrotechnique, Automatique, Télécommunications, Signal

présentée et soutenue publiquement par

Maurício CALDORA COSTA

le 28 juin 2001

Titre:

OPTIMISATION DE DISPOSITIFS ÉLECTROMAGNÉTIQUES DANS UN CONTEXTE D'ANALYSE PAR LA MÉTHODE DES ÉLÉMENTS FINIS

Directeur de Thèse: Jean-Louis COULOMB, Professeur à l'INPG

JURY

Monsieur	Pascal BROCHET	Président et Rapporteur
Monsieur	Laurent NICOLAS	Rapporteur
Monsieur	Silvio IKUYO NABETA	Examinateur
Monsieur	Yves MARÉCHAL	Examinateur
Monsieur	Jean-Louis COULOMB	Examinateur

À Simone et
à mes parents

Remerciements

Remerciements

Je tiens tout d'abord à remercier Jean-Louis COULOMB, Professeur à l'Institut National Polytechnique de Grenoble, qui a assuré l'encadrement de mon travail avec une formidable compétence. Pendant ces trois années passées ensemble, il a été une source de motivation et d'encouragement. Je conserverai un très bon souvenir de nos discussions, de sa façon de chercher de nouvelles solutions et de toujours m'avoir soutenu pendant ces années de recherche au LEG.

Je tiens aussi à remercier Yves MARÉCHAL, Maître de Conférences à l'Institut National Polytechnique de Grenoble, qui a également encadré ce travail avec extrême compétence. Grâce à ses conseils et son intérêt, il m'a été possible de mener cette thèse à son terme.

J'adresse également mes sincères remerciements:

À Monsieur Pascal BROCHET, Professeur à l'École Centrale de Lille, pour avoir accepté d'être rapporteur de ce mémoire et aussi président du jury.

À Monsieur Laurent NICOLAS, Directeur de Recherche à l'École Centrale de Lyon, qui a également accepté d'être rapporteur de ce mémoire.

À Monsieur Silvio IKUYO NABETA, Maître de Conférences à l'Université de São Paulo, pour l'honneur qu'il m'a fait en acceptant de participer à ce jury.

À tous les permanents du LEG et du LMN, en particulier: Patrice LABIE, Patrick EUSTACHE, Gérard MEUNIER, Patrick GUILLOT, Etiennette CALLEGHER et Gilles CAUFFET.

Aux thésards avec qui j'ai eu le plaisir de travailler pendant ces trois années: Afef SLAMA, Alita DEWI, Singva MA, Olivier DEFOUR, Olivier CHADEBEC, Ali ABAKAR, Kerim MEKKI, Gérald CLAEYS, Michael JOAN, Vincent LECONTE, Stéfan GIURGEA, Iszabela KLUTSCH, Fleur JANET, Jean-Daniel ARNOULD, Aktham ASFOUR, et bien d'autres encore...

Au gouvernement brésilien, qui a soutenu financièrement cette thèse avec une bourse de la CAPES - Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior.

À José Roberto CARDOSO, Professeur à l'Université de São Paulo, qui a donné le "coup d'envoie" de ma carrière scientifique et qui je considère avant tout un grand ami.

À toute ma famille et mes amis qui sont passés par Grenoble pendant ces trois années et qui même à distance m'ont donné la motivation pour réussir cette tâche, parfois très difficile.

À mes nouveaux amis que j'ai eu le plaisir de connaître ici en France et desquels je garderai un très bon souvenir.

Enfin, je voudrais remercier très chaleureusement mes parents pour leur confiance, soutien et encouragement et ma femme Simone, qui est restée à mon côté pendant ces trois années. Je suis sûr que sans sa présence je ne serais pas arrivé à réaliser cette thèse.

Table des Matières

Table des Matières

Introduction Générale

Introduction Générale

Dans les années 70, l'industrie électromécanique a connu une grande révolution concernant l'analyse et la conception de nouveaux produits. L'utilisation de méthodes numériques implémentées par des programmes informatiques dans le but de modéliser un dispositif avant sa production a permis l'apparition d'une nouvelle famille d'outils industriels dénommés outils de Conception Assistée par Ordinateur (CAO).

Dans le domaine de la modélisation de dispositifs électromagnétiques, la Méthode des Éléments Finis (MEF) [Coulomb 1981] occupe depuis plusieurs années une place de première importance parmi les outils de CAO, car elle a souvent été capable de traiter des géométries et des phénomènes physiques assez complexes, en fournissant des résultats d'analyse avec une grande fiabilité.

La modélisation par éléments finis a beaucoup évolué ces dernières années, surtout grâce à la grande capacité de calcul des ordinateurs actuels et à l'apparition de nouvelles technologies de développement de logiciels. Le résultat de cette évolution nous permet d'avoir aujourd'hui une conception de plus en plus rapide, précise et performante du dispositif modélisé.

Dans le cadre de conceptions performantes, nous sommes souvent intéressés à trouver une configuration optimale de façon à satisfaire les besoins des utilisateurs et d'avoir, en même temps, un produit viable d'un point de vue économique. Cela peut se traduire, par exemple, par la maximisation du couple magnétique d'une machine tournante par rapport aux paramètres structurels (nombre de dents, type des encoches, ...), dimensionnels (dimension de

l'entrefer, largeur des encoches, ...) et physiques (type de matériaux, densité de courant, ...) qui interviennent dans la description du dispositif.

Les phénomènes physiques (les saturations magnétiques, les couplages électriques, les courants de Foucault, ...) issus des caractéristiques du dispositif représentent une complexité pour son optimisation, à laquelle nous devons ajouter des contraintes relatives à ses caractéristiques physiques (une valeur de saturation inférieure à un certain taux) ou encore à sa faisabilité géométrique (une valeur minimale pour l'entrefer, par exemple).

La présence de telle complexité demande alors une procédure plus puissante qu'une simple étude paramétrique du dispositif. Cette procédure, qui a déjà été objet de plusieurs applications en électromagnétisme [Saldanha 1992] [Vasconcelos 1994] [Üler 1995] [Saludjian 1997] [Alotto 1998] [Sareni 1999], consiste à combiner la modélisation numérique avec des méthodes d'optimisation classiques.

Malgré la robustesse de cette approche, elle présente plusieurs difficultés liées aux besoins de l'utilisateur (recherche d'une solution globale, fiabilité et précision de la solution, diversité des problèmes traités, temps de calcul raisonnable, ...), aux caractéristiques du problème d'optimisation (non-linéarité du critère d'optimisation par rapport aux paramètres de conception, gradient du critère d'optimisation bien souvent inaccessible, ...) et aux contraintes de l'outil de simulation (erreurs de la méthode dues aux petits intervalles de discrétisation dans l'espace et le temps, temps de calcul important, échange de données avec l'outil d'optimisation, ...).

L'apport de solutions à la problématique constituée par l'ensemble de ces difficultés représente l'objectif de cette thèse.

Le premier chapitre est surtout bibliographique et propose un état de l'art des méthodes d'optimisation. Deux grandes familles de méthodes seront présentées: les déterministes et les stochastiques. Nous allons montrer les caractéristiques principales de chaque famille, ses points forts, ses points faibles et aussi ses méthodes les plus connues et les plus utilisées.

Le deuxième chapitre est consacré à la description d'une architecture logiciel développée de façon à permettre l'implémentation d'un outil qui soit capable de traiter différents problèmes d'optimisation, en utilisant les différentes méthodes de résolution présentées dans le chapitre I. Les solutions aux problèmes concernant l'échange d'informations entre l'outil d'optimisation et l'outil de simulation numérique seront aussi présentées dans ce chapitre.

Dans le troisième chapitre, nous proposons une nouvelle approche d'optimisation basée sur l'utilisation d'une surface de réponse, selon laquelle la fonction qui décrit le comportement du critère à optimiser par rapport aux paramètres de conception est remplacée par une approximation créée en utilisant un algorithme d'interpolation basé sur la Méthode des Éléments Diffus [Nayroles 1991] [Marih 1994] [Duarte 1995] [Hérault 2000]. Nous verrons que cette démarche peut apporter des solutions à plusieurs des difficultés de notre problématique, surtout en ce qui concerne le temps de calcul onéreux. Toujours dans cette philosophie d'optimisation à partir d'une surface de réponse, nous allons présenter dans ce chapitre les aspects les plus importants de la Méthode des Plans d'Expériences [Sado 1991] [Demonsant 1996] [Pillet 1997] [Schimmerling 1998], en mettant en évidence ses contributions dans cette approche.

Enfin, le quatrième chapitre nous permettra de valider cette nouvelle stratégie d'optimisation sur différentes applications en électrotechnique, parmi lesquelles nous trouverons l'optimisation de la géométrie d'un moteur à réluctance variable, l'optimisation d'un contacteur électromagnétique et la résolution du problème 25 du TEAM Workshop.

Chapitre I

Etat de l'Art des Méthodes d'Optimisation

Chapitre I

État de l'Art des Méthodes d'Optimisation

I.1 - Introduction

Dans ce premier chapitre, nous présenterons l'état de l'art des méthodes mathématiques utilisées dans la résolution d'un problème d'optimisation. Nous allons commencer par la présentation de quelques définitions nécessaires à l'application de ces méthodes, ainsi que par l'exposition de concepts de base importants, tels que la formulation mathématique d'un problème d'optimisation.

Ensuite, nous allons consacrer une section aux problèmes d'optimisation non contraints, dans laquelle seront présentées les méthodes d'optimisation les plus connues et les plus utilisées, comme par exemple la méthode du *Simplex*, le *Gradient Conjugué*, les méthodes *Quasi-Newton*, le *Recuit Simulé* et les *Algorithmes Génétiques*.

L'objectif de cette section ne sera ni de comparer la performance de chacune de ces méthodes, ni de montrer tous les détails concernant leur implémentation, mais plutôt de mettre en évidence les caractéristiques qu'elles ont en commun et qui nous permettent de les réunir en différents groupes. Dans ce contexte, nous allons trouver un bref exposé sur le principe de résolution que chacune d'entre elles utilise pour atteindre la solution du problème.

Finalement, nous allons présenter une section consacrée aux problèmes d'optimisation contraints et aux méthodes utilisées pour leur résolution. Encore une fois, nous n'allons pas nous attacher aux détails d'implémentation et de performance des ces méthodes, mais démontrer qu'elles ont aussi des caractéristiques en commun qui nous permettent de les réunir en différents groupes. Parmi ces méthodes, nous trouverons la *Programmation Quadratique Récursive*, le *Lagrangien Augmenté* et les méthodes de *Pénalités*.

I.2 - Définitions

I.2.1 - Formulation Mathématique d'un Problème d'Optimisation

Un problème d'optimisation de dimension n peut être écrit de façon générale sous la forme:

$$(P)\begin{cases} \text{Min } f(x) \in \Re^n \\ g_i(x) \le 0 \quad i = 1,...,p \\ h_j(x) = 0 \quad j = 1,...,q \\ x_{k\min} \le x_k \le x_{k\max} \quad k = 1,...,n \end{cases} \tag{1}$$

où

$f(x)$ est le critère à minimiser appelé aussi *fonction objectif*

x est un vecteur à n variables x_k qui représentent les *paramètres* du problème à optimiser

$g_i(x)$ et $h_j(x)$ représentent respectivement les *contraintes d'inégalité* et *d'égalité*

$x_{k\,min}$ et $x_{k\,max}$ désignent les *contraintes de domaine*

$\Re^n$ est *l'espace de recherche* borné par les contraintes de domaine.

La solution d'un problème d'optimisation est alors donnée par un ensemble de paramètres x^* pour lesquels la fonction objectif présente une valeur minimale, en respectant les contraintes d'égalité, d'inégalité et de domaine.

I.2.2 - Minimum Local et Minimum Global

Un point x^* de l'espace de recherche $\Re^n$ représente un *minimum local* ou *optimum local*, s'il existe un voisinage de x^* noté $V(x^*)$, tel que:

$$\forall x \in V(x^*)\ f(x) \geq f(x^*) \tag{2}$$

Cette relation signifie que dans le voisinage de x^*, défini par un ε, il n'existe aucun point pour lequel $f(x)$ est inférieur à $f(x^*)$.

Un point x^* de l'espace de recherche $\Re^n$ est un *minimum global* ou *optimum global* si:

$$\forall x \in \Re^n\ f(x) \geq f(x^*) \tag{3}$$

Nous pouvons dire aussi que le minimum global est le plus petit minimum local de l'espace de recherche, comme nous montre la Figure 1.

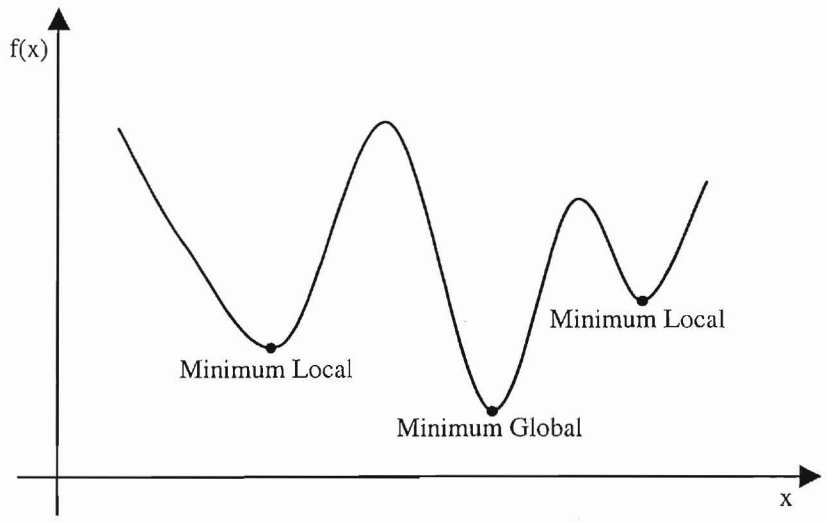


Figure 1 - Représentation du minimum local et global d'une fonction

Lorsqu'une fonction ne contient qu'un minimum local, elle est dite *unimodale*. Dans le cas contraire, elle est dénommée *multimodale*.

I.2.3 - Minimisation et Maximisation

Dans l'équation (1), nous avons défini le problème d'optimisation comme étant la *minimisation* d'une fonction, parfois soumise à des contraintes. Cependant, il existe des situations où nous sommes plutôt intéressés à trouver un point de *maximisation*, c'est-à-dire, à maximiser la fonction objectif.

Dans ce cas, il devient nécessaire de transformer le problème de maximisation en minimisation, car les méthodes d'optimisation qui seront présentées dans les sections I.3 et I.4 sont souvent implémentées en s'appuyant sur des critères de minimisation de la fonction objectif.

Cette transformation du problème d'optimisation peut être facilement obtenue à l'aide d'une simple fonction de transformation appliquée directement sur la fonction objectif originale [Press 1992], comme l'indique l'équation (4).

$$\Phi(f(x)) = -f(x) \tag{4}$$

I.3 - Problèmes d'Optimisation Non Contraints

Un problème d'optimisation est dit *non contraint* s'il ne contient pas de fonction contrainte, c'est-à-dire, si les fonctions $g_i(x)$ et $h_j(x)$ du problème (1) ne sont pas définies, comme dans le cas du problème (5):

$$(P)\begin{cases} \text{Min } f(x) \in \Re^n \\ x_{k\min} \le x_k \le x_{k\max} \qquad k = 1,\ldots,n \end{cases} \tag{5}$$

Une *condition nécessaire* pour que x^* soit minimum local d'un problème non contraint est donnée par (6):

$$\begin{cases} \nabla f(x^*) = 0 \\ H(x^*) \text{ non négative} \end{cases} \tag{6}$$

où

∇f est le gradient de la fonction objectif

$H = \nabla^2 f$ est la matrice de dérivées secondes partielles de f, qualifiée de *Hessien.*

Une *condition suffisante* pour que x^* soit minimum local d'un problème non contraint est donnée par (7):

$$\begin{cases} \nabla f(x^*) = 0 \\ H(x^*) \text{ positive} \end{cases} \tag{7}$$

Les conditions (6) et (7) sont uniquement valables pour des fonctions différentiables et ne s'appliquent pas à des points situés sur les frontières de l'espace de recherche [Sareni 1999].

Dans la pratique, nous classifions les problèmes d'optimisation non contraints selon la nature mathématique de la fonction objectif. Celle-ci peut être unidimensionnelle ou multidimensionnelle, continue ou discontinue, linéaire ou non linéaire, convexe ou non convexe, différentiable ou non différentiable.

Selon les caractéristiques du problème d'optimisation non contraint, nous pouvons appliquer différentes méthodes de résolution pour identifier sa solution. Ces méthodes sont séparées en deux grands groupes: les *méthodes déterministes* et les *méthodes stochastiques.*

I.3.1 - Méthodes d'Optimisation Déterministes

Une méthode d'optimisation est dite déterministe lorsque son évolution vers la solution du problème est toujours la même pour un même contexte initial donné, ne laissant aucune place au hasard. Ce sont en général des méthodes efficaces, peu coûteuses, mais qui nécessitent une configuration initiale (point de départ) pour résoudre le problème. Ce sont souvent des méthodes locales, c'est-à-dire qu'elles convergent vers l'optimum le plus proche du point de départ, qu'il soit local ou global.

Selon la dimension de la fonction objectif à optimiser, les méthodes déterministes peuvent être classifiées en *unidimensionnelles* ou *multidimensionnelles.*

I.3.1.1 - Méthodes Déterministes Unidimensionnelles

Les méthodes déterministes unidimensionnelles sont utilisées dans l'optimisation de fonctions à un seul paramètre. Ces méthodes, aussi appelées méthodes de *Recherche Linéaire* (*Line Search Methods*), sont normalement basées sur des techniques qui permettent de localiser le point minimal de la fonction à partir de réductions successives de l'intervalle de recherche.

Dans la littérature, nous trouvons différentes méthodes unidimensionnelles, parmi lesquelles nous allons présenter la méthode de *Dichotomie* [Culioli 1994], la méthode de la *Section Dorée* [Culioli 1994] [Press 1992] et la méthode de *Brent* [Brent 1973] [Press 1992]. La plupart de ces méthodes ne supposent pas que la fonction à minimiser soit différentiable, ni même continue, mais seulement unimodale [Culioli 1994].

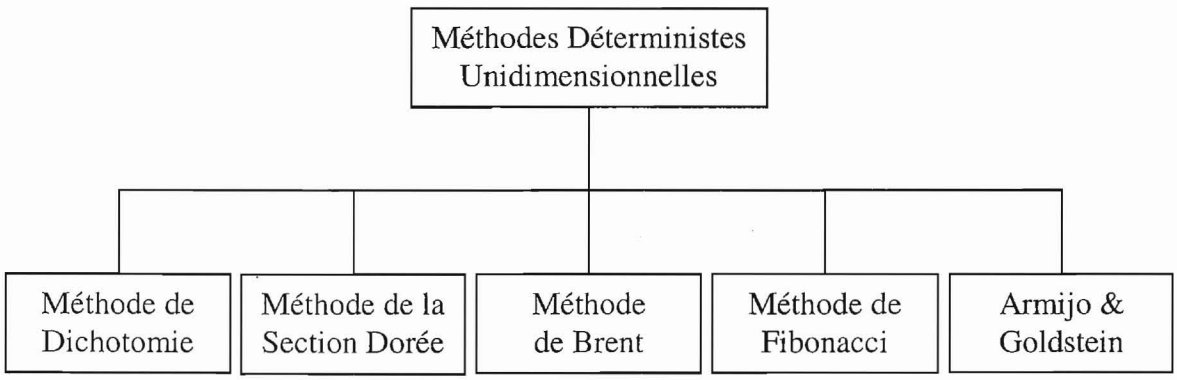


Figure 2 - Principales méthodes déterministes unidimensionnelles

- *Méthode de Dichotomie*

La méthode de *Dichotomie* classique [Culioli 1994] est une méthode unidimensionnelle de subdivision d'intervalles qui s'appuie sur l'existence d'un triplet (x_1, x_2, x_3) définit dans l'intervalle de recherche, tel que $f(x_1) > f(x_2) < f(x_3)$. Le principe de cette méthode consiste à découper en deux les intervalles $[x_1,x_2]$ et $[x_2,x_3]$ de façon à obtenir deux nouveaux points (x_4, x_5) sur lesquels la fonction sera évaluée.

À partir des valeurs de $f(x_1)$, $f(x_2)$, $f(x_3)$, $f(x_4)$ et $f(x_5)$, on choisit parmi x_1, x_2, x_3, x_4 et x_5, le nouveau triplet qui sera utilisé pour faire le prochain découpage. Ainsi, à chaque itération du processus, nous avons une réduction de l'intervalle de recherche en utilisant deux nouvelles évaluations de la fonction, comme nous montrent les différentes images de la

Figure 3. Le processus s'arrête lorsque l'intervalle de découpage devient plus petit qu'un ε prédéterminé.

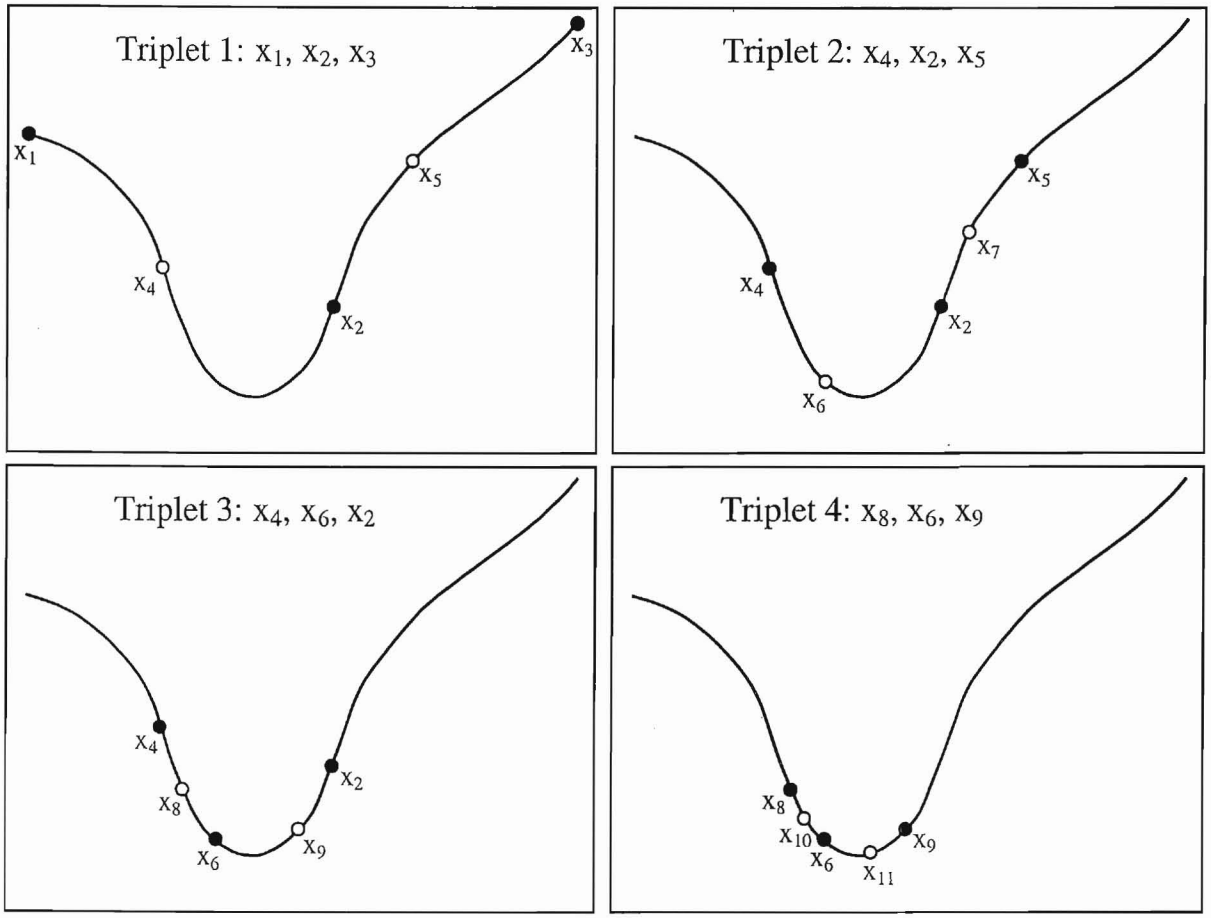


Figure 3 - Méthode de Dichotomie

- *Méthode de la Section Dorée*

La méthode de la *Section Dorée* (*Golden Section Search*) [Culioli 1994] [Press 1992] utilise le même principe de découpage de la méthode précédente, à la différence qu'au lieu de découper l'intervalle de recherche $[x_1,x_3]$ en quatre, elle le découpe en trois, ce qui ne coûte, à chaque itération, qu'une seule évaluation supplémentaire de la fonction, comme nous pouvons le vérifier sur la Figure 4.

Le point de découpage utilisé à chaque itération est donné par une distance égale à $(\sqrt{5}-1)/2 \approx 0{,}6180$ du point initial de l'intervalle de recherche. Cette valeur est égale à

l'inverse du nombre d'or $(\sqrt{5}+1)/2 \approx 1{,}6180$ et son utilisation permet d'obtenir une série de triplets optimale qui accélère la convergence de la méthode.

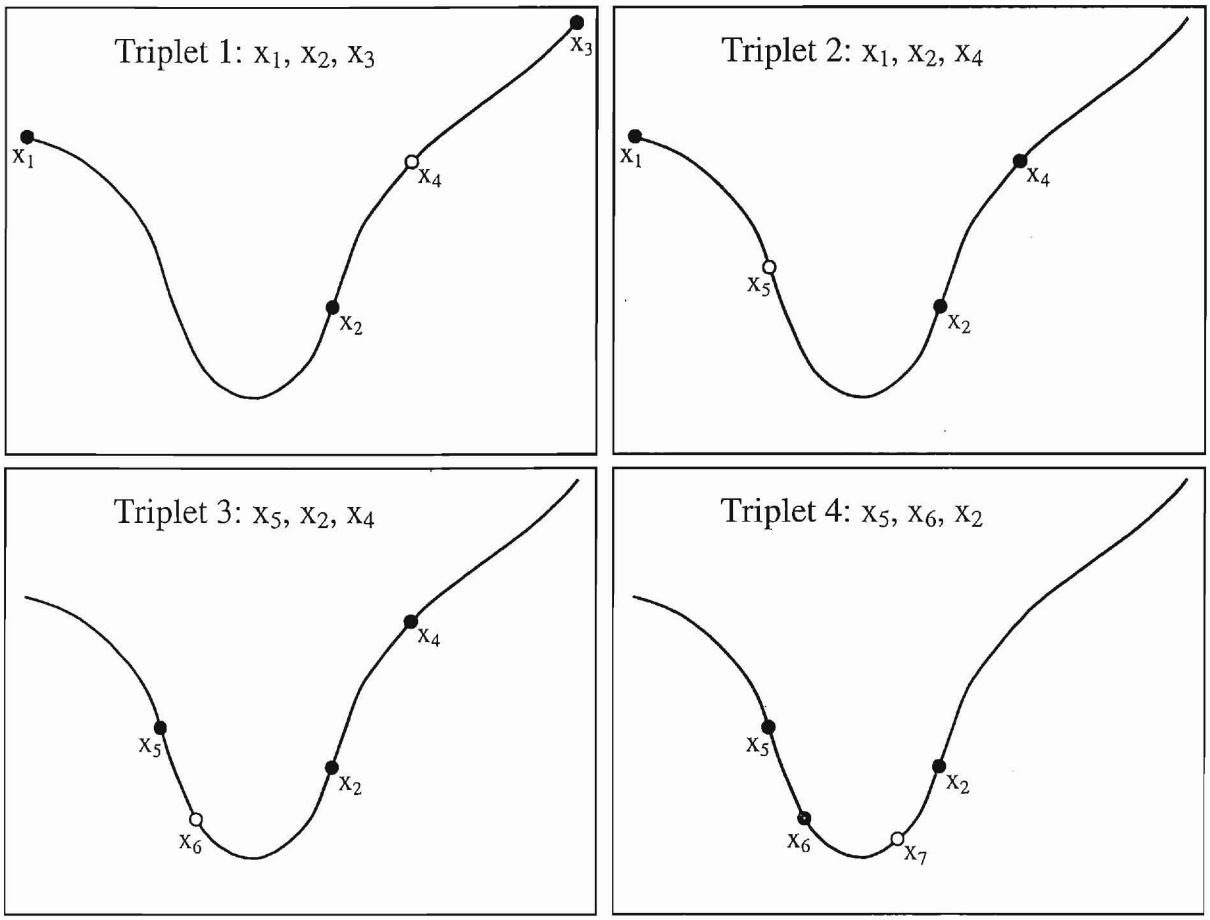


Figure 4 - Méthode de la Section Dorée

- *Méthode de Brent*

La méthode de *Brent* [Brent 1973] [Press 1992] effectue la réduction de l'intervalle de recherche en utilisant une interpolation polynomiale de la fonction, calculée aussi à partir d'un triplet (x_1, x_2, x_3). Dans cette méthode, le point de découpage est donné par l'abscisse de la parabole définie par le triplet, comme nous montre la Figure 5.

Ainsi comme la méthode de la *Section Dorée*, la méthode de *Brent* n'utilise qu'une seule évaluation supplémentaire de la fonction à chaque itération. Dans le cas de fonctions

différentiables, nous pouvons utiliser une variante de cette méthode qui s'appuie sur le gradient de la fonction pour accélérer la convergence du processus de recherche [Press 1992].

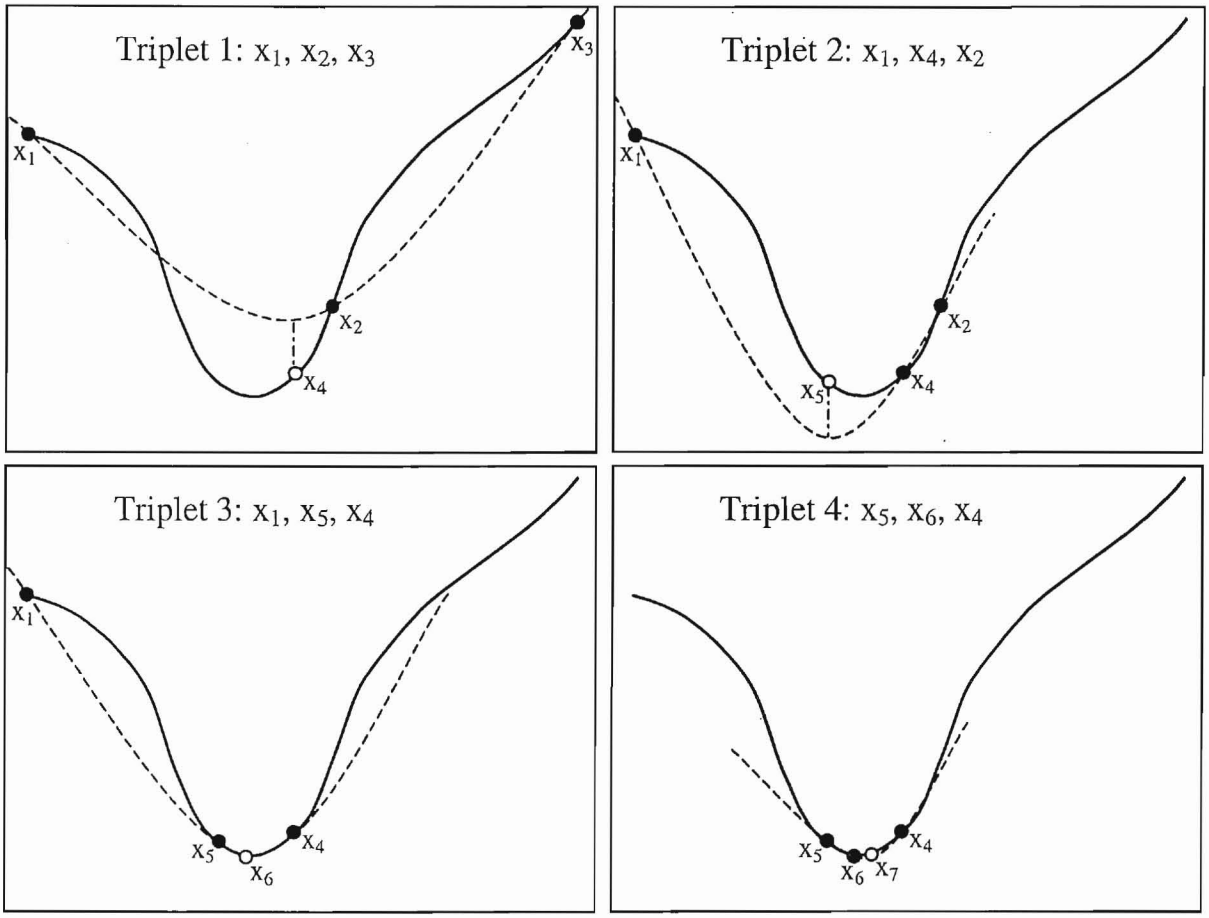


Figure 5 - Méthode de Brent

Il existe encore d'autres méthodes d'optimisation unidimensionnelles, comme par exemple la méthode de *Fibonacci* [Schwefel 1995] et la méthode de recherche linéaire *d'Armijo* et *Goldstein* [Dennis 1987]. Malgré les différences concernant la vitesse de convergence, les critères d'arrêt et le principe de découpage utilisé par ces méthodes, l'utilisation d'un triplet pour orienter la réduction de l'intervalle de recherche reste toujours une caractéristique qu'elles ont en commun.

I.3.1.2 - Méthodes Déterministes Multidimensionnelles

Les méthodes déterministes multidimensionnelles sont consacrées à l'optimisation de fonctions à un paramètre ou plus. Elles peuvent être classées selon l'utilisation de

l'information des dérivées de la fonction objectif par rapport aux paramètres x_k. Elles sont dites *directes* ou *d'ordre 0* si elles n'utilisent que l'information de la valeur de la fonction elle-même. Dans le cas où elles nécessitent aussi le calcul du gradient de la fonction, elles sont dites *indirectes* ou *d'ordre 1*.

Les méthodes d'ordre 0 sont en général peu précises et convergent très lentement vers l'optimum [Kowalik 1968]. En revanche, elles offrent l'avantage de se passer du calcul du gradient, ce qui peut être intéressant lorsque la fonction n'est pas différentiable ou lorsque le calcul de son gradient représente un coût important.

Les méthodes d'ordre 1 permettent d'accélérer la localisation du point d'optimisation, une fois que le gradient donne l'information sur la direction de recherche de la solution. Par contre, elles sont applicables uniquement aux problèmes où la fonction est continûment différentiable.

Nous pouvons diviser les méthodes multidimensionnelles, qu'elles soient directes ou indirectes, en deux différents groupes: les méthodes *analytiques* ou *de descente* et les méthodes *heuristiques* ou *géométriques*.

Les méthodes analytiques se basent sur la connaissance d'une direction de recherche souvent donnée par le gradient de la fonction. La plupart de ces méthodes sont d'ordre 1 et exécutent des minimisations linéaires successives en faisant appel à des méthodes unidimensionnelles [Press 1992]. Les exemples les plus significatifs de méthodes analytiques sont la méthode de la *Plus Grande Pente* [Culioli 1994], le *Gradient Conjugué* [Culioli 1994] [Fletcher 1987] [Press 1992], la méthode de *Powell* [Powell 1965] et les méthodes *Quasi-Newton* [Culioli 1994] [Fletcher 1987] [Press 1992].

Les méthodes heuristiques explorent l'espace par essais successifs en recherchant les directions les plus favorables. À l'opposé des méthodes analytiques, la plupart de ces méthodes sont d'ordre 0. Les implémentations de méthodes géométriques les plus souvent utilisées sont celles de la méthode du *Simplex* [Nelder 1965], la méthode de *Rosenbrock* [Rao 1996] et la méthode de variations locales de *Hooke et Jeeves* [Cherruault 1999].

La Figure 6 montre les méthodes multidimensionnelles les plus importantes avec leur ordre respectif de résolution.

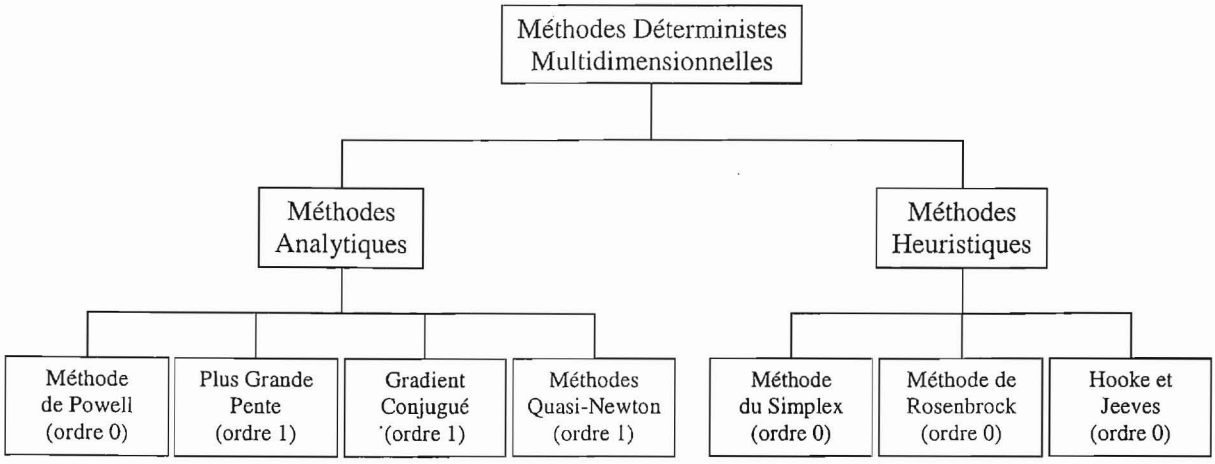


Figure 6 - Principales méthodes déterministes multidimensionnelles

- *Méthode de la Plus Grande Pente*

La méthode de la *Plus Grande Pente* (*Steepest Descent*) [Culioli 1994] ou méthode du *Gradient à Pas Optimal* est une des méthodes les plus classiques utilisées pour minimiser une fonction à plusieurs variables. Cette méthode d'ordre 1 est basée sur la constatation que la direction opposée à celle du gradient de la fonction représente une direction de descente [Press 1992].

Nous pouvons donc, à partir d'un point initial x_0, calculer la valeur du gradient et utiliser une méthode de recherche linéaire pour minimiser la fonction dans la direction de descente opposée. Cette minimisation permet de calculer la valeur du pas optimal α_k qui nous emmène à un nouveau point de recherche à chaque itération du processus, en utilisant l'équation (8).

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k), \tag{8}$$

où

x_k est le point de recherche à l'itération k

x_{k+1} est le nouveau point de recherche calculé à partir de la minimisation de f dans la direction opposée à son gradient.

Le processus s'arrête lorsque $\|x_{k+1} - x_k\| < \varepsilon$, ε étant une tolérance prédéterminée.

- *Gradient Conjugué*

Le *Gradient Conjugué* [Culioli 1994] [Fletcher 1987] [Press 1992] utilise le même principe que la méthode précédente, à la différence que la direction de descente n'est plus donnée par le gradient de la fonction, mais par des directions conjuguées successives. Deux directions h_0 et h_1 sont dites conjuguées par rapport à la matrice *Hessienne H* d'une fonction, si $h_0^T H h_1 = 0$.

Les directions conjuguées peuvent être obtenues directement à partir du *Hessien* de la fonction objectif. Cependant, ce calcul peut représenter un coût important pour la méthode d'optimisation. Pour éviter ce problème, la méthode du *Gradient Conjugué* effectue le calcul des directions conjuguées à partir des équations (9) et (10).

$$h_{k+1} = g_{k+1} + \beta_k h_k \tag{9}$$

$$\beta_k = \frac{g_{k+1} \cdot g_{k+1}}{g_k \cdot g_k} \tag{10}$$

où

h_k et h_{k+1} sont des directions conjuguées

g_k et g_{k+1} sont les directions opposées aux gradients calculés sur les points x_k et x_{k+1}, respectivement.

Ce calcul est uniquement valable si le point x_{k+1} est obtenu à partir d'une minimisation linéaire où h_k représente la direction de recherche [Press 1992], comme nous montre l'équation (11).

$$x_{k+1} = x_k + \alpha_k h_k , \tag{11}$$

où

α_k est le pas optimal donné par une minimisation linéaire.

Le terme β_k donné par (10) a été proposé par Fletcher et Reeves en 1964 [Fletcher 1964]. Quelques années plus tard, en 1971, Polak et Ribière [Polak 1971] ont proposé une variante au calcul de β_k (12) qui semble donner des meilleurs résultats.

$$\beta_k = \frac{(g_{k+1} - g_k) \cdot g_{k+1}}{g_k \cdot g_k} \tag{12}$$

- *Méthodes Quasi-Newton*

Les méthodes *Quasi-Newton* [Culioli 1994] [Fletcher 1987] [Press 1992] consistent à imiter la méthode de *Newton* où l'optimisation d'une fonction est obtenue à partir de minimisations successives de son approximation au second ordre.

À la différence de l'algorithme de *Newton*, les méthodes *Quasi-Newton* ne calculent pas le *Hessien H* de la fonction pour atteindre la solution du problème. Au lieu d'utiliser le *Hessien*, elles utilisent une approximation définie positive de *H* qui peut être obtenue soit à partir de l'expression proposée par *Davidon-Fletcher-Powell (DFP)* [Press 1992] (13), soit par celle proposée par *Broyden-Fletcher-Goldfarb-Shanno (BFGS)* [Press 1992] (14).

$$S_{k+1} = S_k + \frac{\delta_k (\delta_k)^T}{(\delta_k)^T \gamma_k} - \frac{S_k \gamma_k (\gamma_k)^T S_k}{(\gamma_k)^T S_k \gamma_k} \tag{13}$$

$$S_{k+1} = S_k + \left(1 + \frac{(\gamma_k)^T S_k \gamma_k}{(\delta_k)^T \gamma_k} \right) \frac{\delta_k (\delta_k)^T}{(\delta_k)^T \gamma_k} - \frac{\delta_k (\gamma_k)^T S_k + S_k \gamma_k (\delta_k)^T}{(\delta_k)^T \gamma_k} \tag{14}$$

où

$$\delta_k = x_{k+1} - x_k$$

$$\gamma_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

S_k est l'approximation du *Hessien* qui, à l'itération $k = 0$, est égale à une matrice identité.

À chaque itération, le point x_{k+1} est obtenu à partir d'une recherche linéaire que se fait dans la direction donnée par $S_k \nabla f(x_k)$, ce qui nous emmène à l'équation (15).

$$x_{k+1} = x_k - \alpha_k S_k \nabla f(x_k) \tag{15}$$

où

α_k est le pas optimal donné par une minimisation linéaire.

- *Méthode du Simplex*

La méthode du *Simplex* est une méthode directe développée par Nelder et Mead [Nelder 1965], dont l'idée est de modifier un *simplex* de façon à ce qu'il atteigne le point d'optimisation. Le *simplex* est une figure géométrique de dimension *n*, créée à partir de *n+1* points, où chaque dimension correspond à un paramètre du problème à optimiser. Un *simplex*

de deux dimensions est donc représenté par un triangle, un *simplex* de trois dimensions par un tétraèdre, etc.

Pour déplacer le *simplex* vers la région optimale, la méthode vérifie la valeur de la fonction sur chacun des sommets du *simplex* original et déplace le point où la fonction présente sa plus grande valeur vers la direction opposée. Cette transformation s'appelle réflexion et elle est appliquée de façon à conserver le volume original du simplex. Il existe aussi des situations spécifiées par la méthode où le point est déplacé soit par une expansion, soit par une contraction du *simplex*, comme le montre la Figure 7.

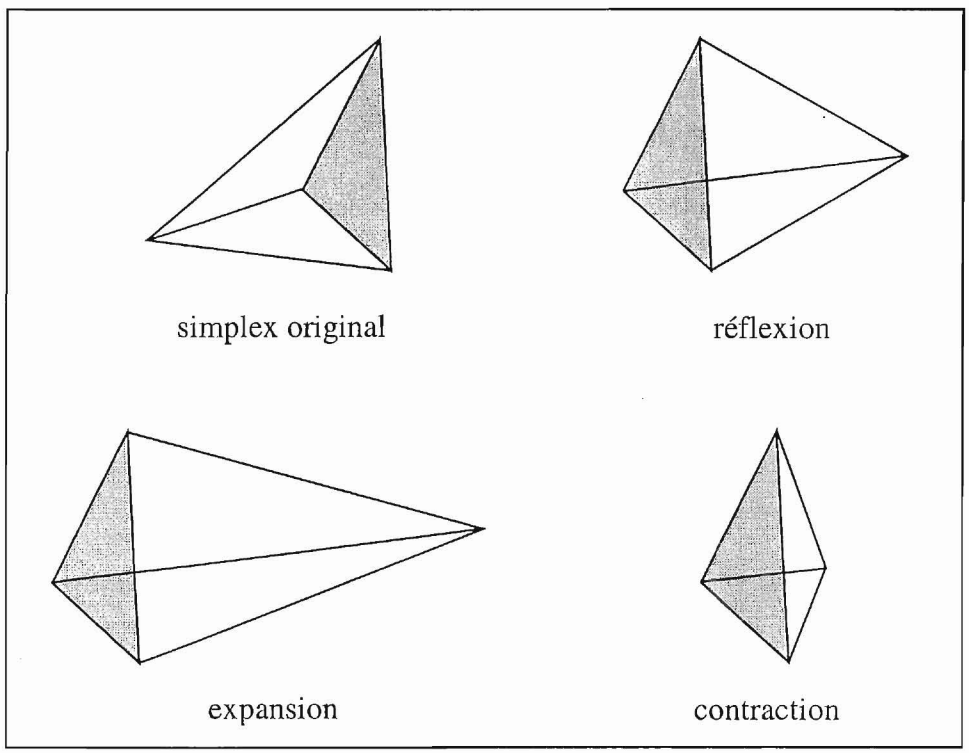


Figure 7 - Méthode du Simplex

Le processus s'arrête au moment où le déplacement donné par une des transformations devient plus petit qu'une tolérance ε prédéterminée. Ainsi comme d'autres méthodes géométriques, la méthode du Simplex n'est pas assez performante, car elle utilise un nombre important d'évaluations de la fonction avant d'atteindre le point minimal. Par contre, elle présente l'avantage de ne pas utiliser la valeur du gradient de la fonction.

I.3.2 - Méthodes d'Optimisation Stochastiques

Les méthodes d'optimisation stochastiques s'appuient sur des mécanismes de transition probabilistes et aléatoires. Cette caractéristique indique que plusieurs exécutions successives de ces méthodes peuvent conduire à des résultats différents pour une même configuration initiale d'un problème d'optimisation.

Ces méthodes ont une grande capacité de trouver l'optimum global du problème. Contrairement à la plupart des méthodes déterministes, elles ne nécessitent ni de point de départ, ni la connaissance du gradient de la fonction objectif pour atteindre la solution optimale. Cependant, elles demandent un nombre important d'évaluations avant d'arriver à la solution du problème.

Parmi les méthodes stochastiques les plus employées, nous distinguons le *Recuit Simulé* [Kirkpatrick 1983], la *Recherche Tabu* [Glover 1989] [Glover 1990] [Hu 1992] et les *Méthodes Évolutionnistes*. Ces dernières regroupent différents algorithmes basés sur le même principe d'exploration de l'espace de recherche en utilisant un ensemble de solutions et pas seulement une solution unique. Comme représentantes des méthodes évolutionnistes, nous avons les *Algorithmes Génétiques* [Holland 1975] [DeJong 1975] [Goldberg 1989] [Michalewicz 1994], les *Stratégies d'Évolution* [Rechenberg 1994], la *Programmation Évolutionniste* [Fogel 1994] et la *Programmation Génétique* [Koza 1992].

La Figure 8 présente les méthodes stochastiques les plus utilisées.

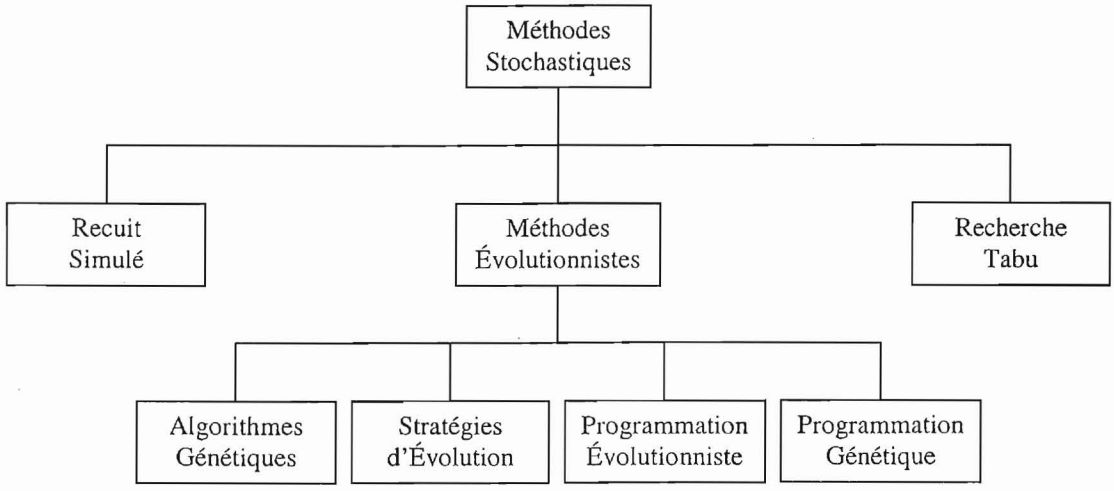


Figure 8 - Principales méthodes stochastiques

- *Recuit Simulé*

Le *Recuit Simulé* (*Simulated Annealing*) [Kirkpatrick 1983] est basé sur le processus de recuit utilisé en métallurgie, dans lequel on cherche à obtenir un matériau sans impureté, représenté par son état d'énergie minimale.

Dans le processus de recuit réel, nous élevons la température du matériau jusqu'à ce qu'il se trouve dans un état d'énergie élevée. Ensuite, nous le refroidissons très lentement de façon à obtenir, à la fin du processus, un matériau constitué par des atomes bien ordonnés, correspondant à une valeur d'énergie stable et minimale.

En 1953, Metropolis [Meropolis 1953] a proposé un modèle qui simulait l'évolution d'une configuration d'atomes vers l'équilibre thermique. Dans ce modèle, une nouvelle configuration est obtenue à partir d'une petite perturbation sur la configuration courante. Cette nouvelle configuration est acceptée avec une probabilité $p = 1$ lorsque la différence d'énergie ΔE entre elle et la configuration courante est inférieure à 0. Dans le cas où $\Delta E > 0$, la probabilité d'acceptation p est donnée par une équation basée sur la *Loi de Boltzmann* (16).

$$p = e^{\frac{-\Delta E}{T}}, \tag{16}$$

où

T est la température du système.

Kirkpatrick [Kirkpatrick 1983] a transposé ce modèle proposé par Metropolis à la méthode de *Recuit Simulé* à partir d'une correspondance entre l'énergie du système et la fonction objectif à minimiser. Dans cette transposition, les différentes configurations d'atomes sont représentées par les paramètres du problème d'optimisation, tandis que la température du système est représentée par une variable de contrôle T.

L'algorithme proposé par Kirkpatrick commence avec une configuration de paramètres choisie au hasard et une température initiale T_0 élevée. Ensuite, à l'aide d'une transformation de voisinage faite à partir d'une petite perturbation aléatoire des paramètres, une nouvelle configuration est générée. La fonction est donc évaluée sur ces deux configurations, ce qui nous permet de calculer l'écart ΔE entre les deux évaluations. Si ΔE est inférieur à zéro, nous remplaçons la configuration originale par la nouvelle configuration obtenue. Dans le cas contraire, nous considérons la probabilité donnée par (16) pour décider si la configuration initiale doit être remplacée ou pas.

À chaque itération de la méthode, ce processus est répété jusqu'à ce que nous obtenions l'équilibre thermique. Au cours de cette transition vers l'équilibre, l'énergie peut localement augmenter, ce qui nous permet de sortir d'un minimum local. Ensuite, nous baissons la température du système [Saludjian 1997] et nous recommençons le processus. L'algorithme s'arrête lorsque nous n'avons plus d'amélioration sensible de la solution ou lorsqu'une certaine valeur de température est atteinte, comme nous le vérifions dans la Figure 9.

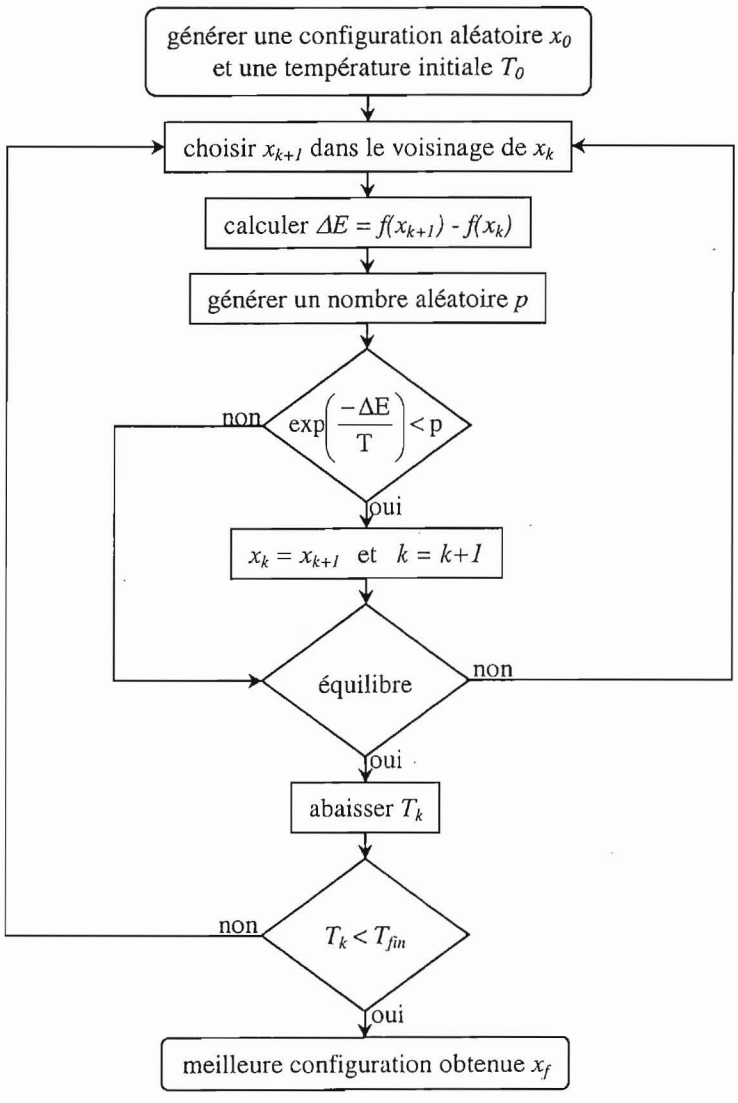


Figure 9 - Recuit Simulé

- *Recherche Tabu*

La *Recherche Tabu* [Glover 1989] [Glover 1990] [Hu 1992] permet d'atteindre le minimum global d'un problème d'optimisation à partir d'une analogie avec la mémoire du cerveau humain.

Le processus de résolution développé par cette méthode commence avec la génération d'une configuration de paramètres au hasard. Ensuite, à chaque itération, le voisinage de la configuration courante est parcouru par une série de mouvements aléatoires de façon à trouver une meilleure solution. Après son exécution, chacun de ces mouvements est ajouté à une liste qui représente la "mémoire" de la méthode. Cette liste, de taille limitée, est dénommée liste *Tabu*.

Les mouvements qui font partie de la liste *Tabu* sont considérés interdits. C'est-à-dire qu'ils ne peuvent pas être exécutés une autre fois tant qu'ils sont dans liste. Par contre, s'il existe un mouvement qui appartient à la liste *Tabu* mais qui en même temps nous emmène à une meilleure solution du problème, une nouvelle exécution de ce mouvement sera alors acceptée. Dans ce cas-là, la nouvelle solution obtenue remplace la solution courante et le processus recommence.

Si pendant le processus d'optimisation la liste *Tabu* devient pleine, nous retirons le plus ancien mouvement de la liste avant d'y ajouter un nouveau. L'algorithme s'arrête lorsque nous n'avons plus d'amélioration sensible de la valeur de la solution. La Figure 10 illustre le processus développé par la méthode.

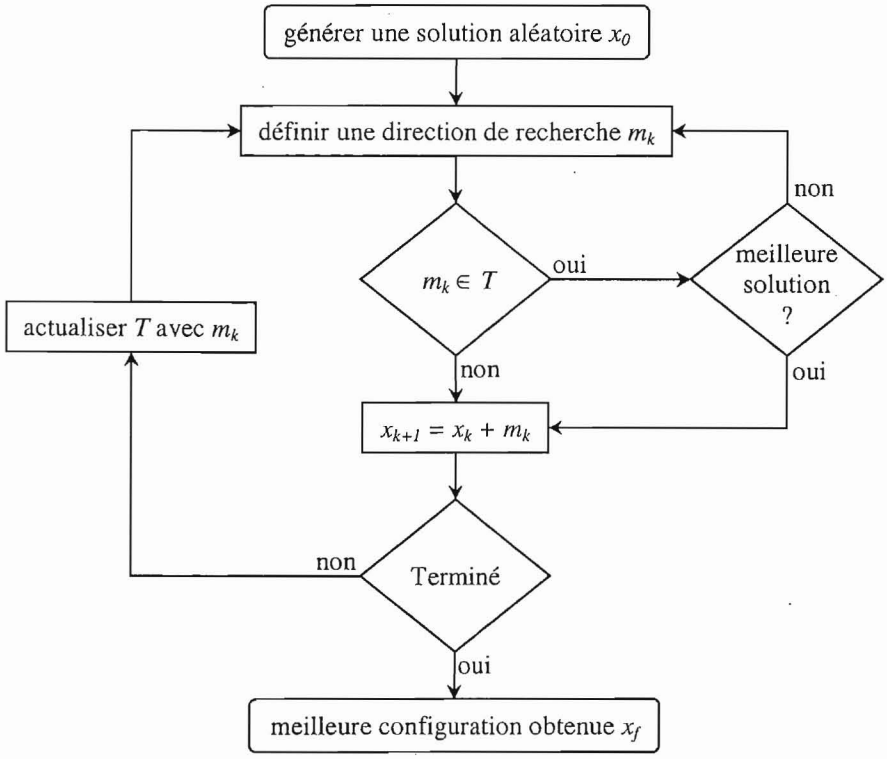


Figure 10 - Recherche Tabu

- *Algorithmes Génétiques*

Les Algorithmes Génétiques (AG) font partie d'une famille de méthodes stochastiques appelée *Méthodes Évolutionnistes* qui reposent sur une analogie avec la *Théorie de l'Évolution Naturelle de Darwin*, selon laquelle les individus d'une population les mieux adaptés à leur environnement ont une plus grande probabilité de survivre et de se reproduire de génération en génération, en donnant des descendants encore mieux adaptés.

Les Algorithmes Génétiques ont été proposés par Holland en 1975 [Holland 1975], puis développées par d'autres chercheurs tels que De Jong [DeJong 1975], Goldberg [Goldberg 1989] et Michalewicz [Michalewicz 1994]. Ils sont actuellement une des méthodes les plus diffusées et les plus utilisées dans la résolution de problèmes d'optimisation dans de nombreux domaines d'application.

Depuis le modèle proposé par Holland en 1975, il y a eu différentes implémentations concernant les AG. Néanmoins, le principe sur lequel la méthode est basée reste toujours le même. Nous allons présenter ici les notions de base des AG ainsi que ses aspects les plus importants. Avant d'expliquer son principe de résolution, nous allons présenter quelques

définitions importantes concernant la terminologie utilisée par cette méthode. Le lecteur intéressé pourra trouver une description plus détaillée des AG et de leurs variantes dans les thèses de [Saludjian 1997], de [Sareni 1999] et dans l'ouvrage de [Gen 1997].

- Terminologie

Comme nous avons déjà mentionné, les AG font partie d'une famille de méthodes basées sur la théorie de l'évolution naturelle de Darwin. Visant à reproduire le même principe de cette théorie dans une méthode d'optimisation, les AG définissent l'ensemble des paramètres du problème à optimiser comme étant l'*individu* de la théorie de Darwin. L'*environnement* de cet individu est représenté par l'espace de recherche du problème d'optimisation, tandis que son *adaptation* à l'environnement est donnée par la valeur de la fonction objectif évaluée sur lui. Finalement, la *population* auquel il appartient est donnée par un ensemble de différents configurations de paramètres, alors que les *générations* sont représentées par les itérations du processus d'optimisation. Ces correspondances sont présentées dans le Tableau I.

TABLEAU I - ANALOGIE ENTRE LES AG ET LA THÉORIE D'ÉVOLUTION NATURELLE

Théorie d'Évolution Naturelle	**Algorithmes Génétiques**
Individu	Ensemble de paramètres
Population	Ensemble d'individus
Environnement	Espace de recherche
Adaptation de l'individu	Évaluation de la fonction objectif
Générations	Itérations de la méthode

- Description de la Méthode

L'algorithme de résolution commence avec la création d'une population P de taille $N > 0$ constituée par des individus générés aléatoirement. Ensuite, on mesure l'adaptation de chacun des individus de P à partir de la valeur de la fonction objectif évaluée sur eux. La prochaine étape du processus consiste à faire évoluer cette population vers une population plus adaptée à chaque génération, en utilisant trois différents opérateurs: la *sélection*, le *croisement* et la *mutation*. Lorsque nous n'avons plus d'amélioration dans l'adaptation des individus de la population, l'algorithme s'arrête. La Figure 11 illustre le processus d'optimisation développé par les AG.

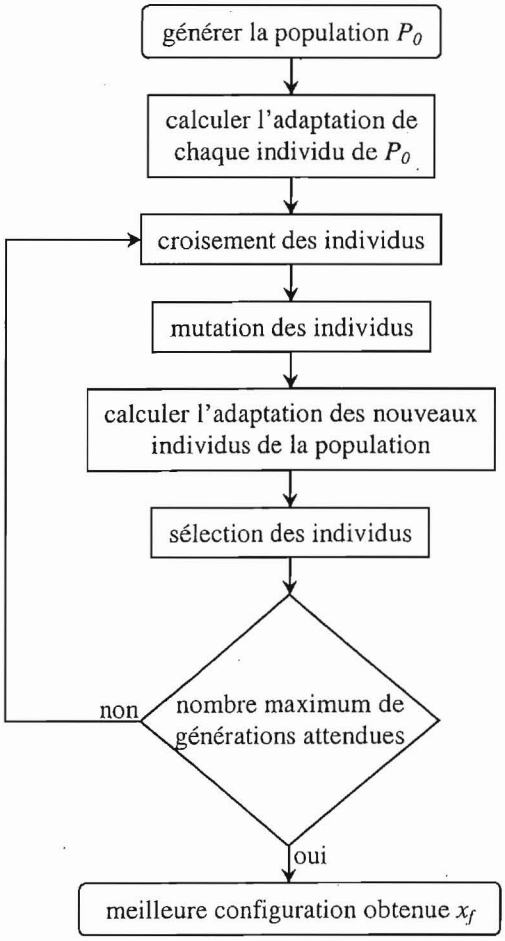


Figure 11 - Algorithme Génétique standard

La procédure effectuée par chacun des opérateurs utilisés par les AG sera décrite dans les paragraphes suivants.

- Opérateur de Sélection

La *sélection* est un opérateur "génétique" appliqué sur la population courante de façon à sélectionner les individus qui iront former la population de la prochaine génération. La sélection de ces individus est basée sur leur valeur d'adaptation. Ainsi, les individus les plus adaptés sont généralement sélectionnés pour constituer la génération suivante, alors que les plus faibles sont exclus sans avoir la possibilité d'avoir des descendants, comme nous montre la Figure 12.

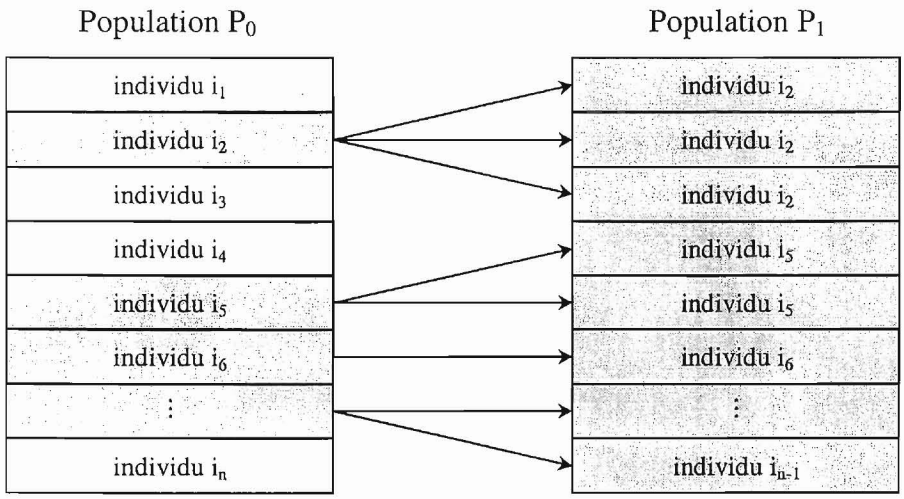


Figure 12 - Sélection des individus d'une population

Il existe différentes façons d'implémenter un opérateur de sélection, parmi lesquelles nous trouvons: la *sélection proportionnelle* et la *sélection par rang*. Néanmoins, le processus de sélection développé par ces différents mécanismes est toujours divisé en deux étapes [Gen 1997] [Saludjian 1997].

La première étape consiste en attribuer à chaque individu i_j un nombre réel p_j qui représente le nombre de descendants attendus pour lui dans la génération suivante. Selon le mécanisme de sélection utilisé, la valeur de p_j est calculée directement ou indirectement à partir de la valeur d'adaptation de l'individu.

Dans le cas d'une *sélection proportionnelle* [Holland 1975], p_j est directement calculé par:

$$p_j = \frac{f(i_j)}{\sum_{k=1}^{N} f(i_k)} \cdot N, \tag{17}$$

où

N est la taille de la population.

Dans la *sélection par rang* proposée par Baker [Baker 1985], on calcule la valeur de p_j en fonction du rang k_j que l'individu occupe dans la population. Cette valeur de k est obtenue à partir d'une liste où les meilleurs individus sont dans les premières positions ($k = 1$), tandis

que les moins performants y occupent les dernières ($k = N$). La valeur de p_j est alors donnée par:

$$p_j = 1 + p_{sel} - \frac{2 \cdot k_j \cdot p_{sel}}{N-1}, \tag{18}$$

où

p_{sel} est la *pression de sélection* définie entre *0* et *1* qui permet de spécifier si tous les individus auront une chance d'être sélectionnés ($p_{sel} = 0$) ou seulement les individus les plus performants ($p_{sel} = 1$).

La deuxième étape du processus de sélection consiste à convertir la valeur du p_j de chaque individu en un nombre de descendants que chacun entre eux aura effectivement dans la prochaine génération. Cette conversion est obtenue à l'aide d'un algorithme d'échantillonnage qui transforme les valeurs réelles des p_j en valeurs entières. Les deux algorithmes d'échantillonnage les plus utilisés sont la *Roue de Loterie* (*Roulette Wheel Sampling*) [Holland 1975] et la *Roue de Loterie Généralisée* (*Stochastic Universal Sampling*) [Baker 1987].

Dans l'algorithme d'échantillonnage proposé par Holland, on crée une roue de loterie divisée en secteurs proportionnels au p_j de chaque individu. Ensuite, on fait tourner la roue un nombre de fois égal à la taille de la population. À chaque coup, on prend une copie (descendant) de l'individu désigné par l'aiguille de la roue pour faire partie de la nouvelle population. Ainsi, les individus avec un plus grand p_j auront un plus grand secteur dans la roue et en conséquence une plus grande probabilité de participer de la génération suivante. La Figure 13 montre une représentation graphique de l'algorithme.

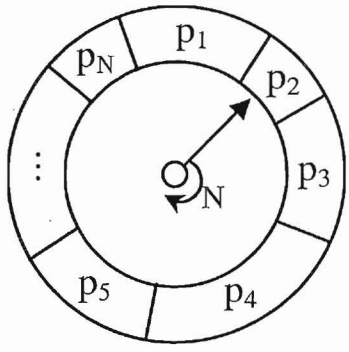


Figure 13 - Echantillonnage par Roue de Loterie

L'algorithme d'échantillonnage proposé par Baker est basé sur le même principe que celui proposé par Holland, à la différence que la roue de loterie contient un nombre d'aiguilles égal à la taille de la population et on ne la fait tourner qu'une seule fois. Le nombre de copies de chaque individu sera alors donné par la position indiqué par chaque aiguille, comme nous montre la Figure 14.

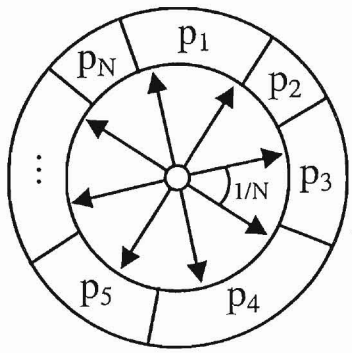


Figure 14 - Echantillonnage par Roue de Loterie Généralisée

À la fin de l'application de l'opérateur de sélection, la nouvelle population contiendra une plus grande proportion des meilleurs individus de la génération précédente. On passe alors à l'étape de reproduction, dans laquelle seront utilisés les opérateurs de *croisement* et de *mutation*.

- Opérateur de Croisement

L'opérateur de *croisement* est utilisé pour échanger les caractéristiques "génétiques" entre les différents individus d'une génération quelconque. Cet échange s'effectue en choisissant deux individus au hasard qui seront "croisés" avec une certaine probabilité de croisement p_c de façon à générer deux nouveaux individus. Dans le cas où nous utilisons le codage réel pour représenter les individus [Saludjian 1997], ce "croisement" peut être obtenu à partir d'un simple échange de paramètres entre les deux parents, comme nous montre la Figure 15.

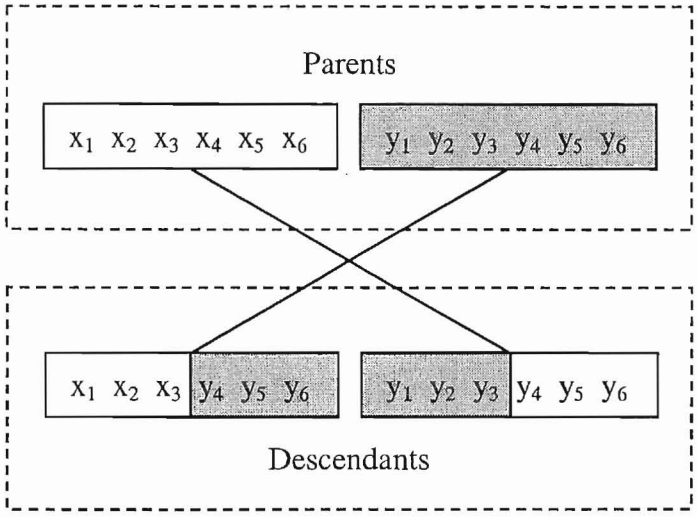


Figure 15 - Croisement entre deux individus

Le croisement représenté sur la Figure 15 est du type 1-point [Holland 1975]. Nous avons encore d'autres implémentations de croisement, tels que le type 2-points [DeJong 1975], le croisement uniforme [Syswerda 1989] et le croisement arithmétique [Janikow 1991]. Malgré ces différentes façons de "croiser" les individus, le but de ces opérateurs reste toujours la conquête de nouvelles régions de l'espace de recherche à partir de l'échange de caractéristiques entre les individus de la population.

- Opérateur de Mutation

L'opérateur de *mutation* est appliqué sur les individus d'une population de façon à obtenir d'autres individus avec des nouvelles caractéristiques "génétiques". Dans le cas d'un codage réel [Saludjian 1997], le mécanisme de mutation peut être implémenté en choisissant un individu de la génération courante au hasard et en modifiant un de ses paramètres aléatoirement avec une probabilité de mutation p_m. Ce mécanisme est dénommé mutation *uniforme* [Davis 1989] et il est illustré dans la Figure 16.

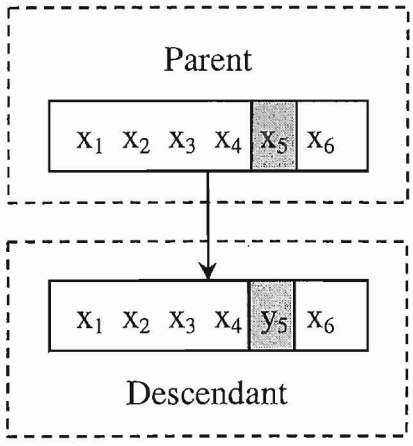


Figure 16 - Mutation d'un individu

Il existe encore d'autres manières d'implémenter une mutation, telles que la mutation *non uniforme* [Davis 1989] et la mutation *aux bornes* [Janikow 1991]. Ainsi comme les opérateurs de croisement, le but de tous ces opérateurs de mutation est d'atteindre des nouvelles régions de l'espace de recherche.

En utilisant les trois opérateurs que nous venons de décrire, les meilleurs individus se propagent de génération en génération, en se combinant ou en échangeant leurs meilleures caractéristiques. En favorisant les meilleurs individus, les régions les plus prometteuses de l'espace de recherche sont explorées, ce qui permet d'atteindre un optimum global.

Cette même approche d'optimisation est employée par les autres méthodes évolutionnistes que nous avons citées précédemment (Figure 8). Les différences que nous trouvons entre ces méthodes sont plutôt liées au codage qu'elles utilisent pour représenter les individus et à la façon dont elles emploient les opérateurs génétiques.

Pourtant, les différentes implémentations des Algorithmes Génétiques qui sont apparues ces dernières années regroupent presque toutes ces variantes dans une même méthode. Le lecteur peut se rapporter aux ouvrages de [Holland 1975] [DeJong 1975] [Goldberg 1989] [Michalewicz 1994] [Rechenberg 1994] [Fogel 1994] et [Koza 1992] pour avoir plus de détails concernant les différentes méthodes évolutionnistes.

I.4 - Problèmes d'Optimisation Contraints

Un problème d'optimisation est dit *problème contraint* s'il contient au moins une fonction contrainte $g_i(x)$ ou $h_j(x)$ dans sa description, comme dans le cas du problème de l'équation (19).

$$(P)\begin{cases} \text{Min } f(x) \in \Re^n \\ g_i(x) \le 0 \quad i = 1,...,p \\ h_j(x) = 0 \quad j = 1,...,q \\ x_{k\min} \le x_k \le x_{k\max} \quad k = 1,...,n \end{cases} \tag{19}$$

Si nous considérons qu'une contrainte d'égalité $h_j(x) = 0$ peut être décrite par deux contraintes d'inégalité $h_j(x) \le 0$ et $-h_j(x) \le 0$, le problème (19) devient alors égal à celui donné par l'équation (20).

$$(P)\begin{cases} \text{Min } f(x) \in \Re^n \\ g_i(x) \le 0 \quad i = 1,...,m = p + 2q \\ x_{k\min} \le x_k \le x_{k\max} \quad k = 1,...,n \end{cases} \tag{20}$$

On appelle *Fonction de Lagrange* associée au problème (20), la fonction:

$$L(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x), \tag{21}$$

où

$\lambda_i \ge 0$ sont appelés *Multiplicateurs de Lagrange*.

Une *condition nécessaire* pour que x^* soit minimum local d'un problème contraint est donnée par les équations de Kuhn-Tucker [Saldanha 1992]:

$$\begin{cases} \nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(x^*) = 0 \\ \lambda_i^* g_i(x^*) = 0 \quad \forall i = 1,...,m \end{cases}, \tag{22}$$

qui à l'aide de l'équation (21) s'écrit:

$$\begin{cases} \nabla_x L(x^*,\lambda^*) = 0 \\ \lambda_i^* g_i(x^*) = 0 \quad \forall i = 1,...,m \end{cases} \tag{23}$$

L'existence de fonctions contraintes dans un problème d'optimisation demande une attention spéciale à la résolution du problème, car une solution qui minimise la fonction objectif ne sera valable que dans le cas où elle respecte aussi les contraintes existantes.

L'ensemble de régions de l'espace de recherche où les contraintes de conception sont vérifiées est dénommé espace *réalisable* ou domaine *admissible*. Inversement, l'espace *irréalisable* ou domaine *interdit* désigne l'ensemble de régions de l'espace où les contraintes sont violées.

La solution d'un problème contraint peut être obtenue à partir de l'application de méthodes qui nous classifions en deux grands groupes: les *Méthodes de Transformation* et les *Méthodes Directes*.

I.4.1 - Méthodes de Transformation

Les *Méthodes de Transformation* ou *Indirectes* [Saldanha 1992] [Vasconcelos 1994] représentent une famille de méthodes qui transforment le problème original avec contraintes en un sous-problème équivalent sans contraintes, en introduisant les contraintes de conception dans la fonction objectif que nous cherchons optimiser.

Une fois que le problème équivalent est créé, un algorithme classique d'optimisation sans contraintes (Gradient Conjugué, Quasi-Newton, ...) est appliqué sur lui de façon à trouver une solution qui sera utilisée pour l'actualiser. Ce processus se répète de façon itérative jusqu'au moment où le critère de convergence est vérifié.

Parmi les méthodes de transformation les plus utilisées, nous avons les *Méthodes de Pénalités* [Caroll 1961] [Fiacco 1968], la *Méthode du Lagrangien Augmenté* [Hestenes 1969] [Rockaffelar 1973], la *Méthode de Variables Mixtes* [Fleury 1986] et la *Méthode des Asymptotes Mobiles* [Svanberg 1987] - Figure 17.

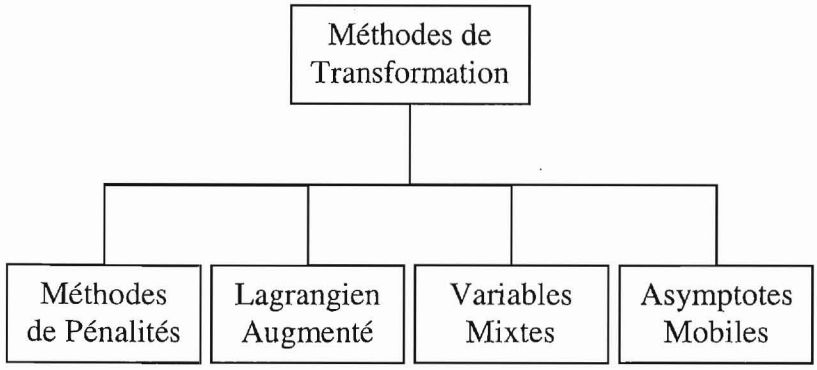


Figure 17 - Principales méthodes de transformation

- *Méthodes de Pénalités*

Les *Méthodes de Pénalités* [Caroll 1961] [Fiacco 1968] [Ryan 1974] sont souvent utilisées dans l'optimisation de problèmes contraints, car elles sont assez simples d'un point de vue théorique et d'une grande efficacité d'un point de vue pratique.

L'idée de ces méthodes est de remplacer la résolution du problème avec contraintes (20) par une suite de résolutions de problèmes sans contraintes, en introduisant dans la fonction objectif une pénalisation concernant chacune des fonctions contraintes violées, comme nous le montre l'équation (24).

$$(P_k)\left\{\text{Min } \Phi(x,r) = f(x) + r_k \sum_{i=1}^{m} W(g_i(x))\right. \tag{24}$$

où

$r_k > 0$ est un coefficient de pénalité actualisé à chaque itération du processus d'optimisation

W est une fonction de pénalisation définie en $\Re \rightarrow \Re$ telle que:

$$\begin{aligned} &W(y) = 0 \quad \text{si } y \geq 0 \\ &W(y) = +\infty \quad \text{si } y < 0 \end{aligned} \tag{25}$$

Selon la nature de la fonction de pénalité W utilisée, les méthodes de pénalités peuvent être divisées en deux classes: les *méthodes de pénalités intérieures* et les *méthodes de pénalités extérieures*.

- Méthodes de Pénalités Intérieures

Les *Méthodes de Pénalités Intérieures* [Caroll 1961] sont aussi appelées méthodes à barrière, car la fonction de pénalité forme une barrière infinie tout au long de la frontière du domaine réalisable Ψ. Les fonctions de pénalités les plus utilisées par ces méthodes sont la fonction inverse (26) et la fonction logarithmique (27).

$$W(g_i(x)) = -\frac{1}{g_i(x)} \tag{26}$$

$$W(g_i(x)) = -\log(-g_i(x)) \tag{27}$$

En utilisant ces fonctions, lorsque x appartient à Ψ, $W(x) > 0$ et lorsque x tend vers sa frontière, $W(x) \rightarrow +\infty$. Par conséquent, nous ne pouvons jamais franchir la frontière de Ψ, et les solutions générées par l'algorithme seront donc admissibles pendant tout le processus d'optimisation. Cependant, ces méthodes présentent l'inconvénient d'avoir besoin d'un point initial qui soit à l'intérieur du domaine réalisable, ce qui n'est pas toujours facile à obtenir.

- Méthodes de Pénalités Extérieures

Les *Méthodes de Pénalités Extérieures* [Fiacco 1968] ne présentent pas le même inconvénient que les méthodes de pénalités intérieures, car l'approximation de la solution est faite par l'extérieur du domaine réalisable Ψ, ce qui nous permet d'avoir un point initial dans cette région de l'espace.

La fonction de pénalité utilisée par ces méthodes est donnée par (28). Cette fonction nous donne une augmentation de la pénalisation à mesure que nous nous éloignons de Ψ.

$$W(g_i(x)) = \max[0, g_i(x)]^2 \tag{28}$$

Contrairement aux méthodes de pénalités intérieures, les solutions générées par ces méthodes ne sont pas toujours admissibles pendant tout le processus d'optimisation. Ceci peut représenter un inconvénient, surtout lorsque l'algorithme ne converge pas et nous nous retrouvons alors avec une solution irréalisable.

- *Lagrangien Augmenté*

Le *Lagrangien Augmenté* [Hestenes 1969] [Powell 1969] [Rockaffelar 1973] est une méthode de transformation basée sur la minimisation d'une fonction L appelée fonction Lagrangienne Augmentée. Cette fonction est créée à partir de l'addition d'une pénalisation à la fonction Lagrangienne classique (21) associée au problème d'optimisation, comme nous pouvons le vérifier dans l'équation (29).

$$L(x,\lambda,r) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + r\sum_{i=1}^{m} g_i^2(x) \qquad (29)$$

où

$r\sum_{i=1}^{m} g_i^2(x)$ représente la fonction de pénalité.

I.4.2 - Méthodes Directes

Les *Méthodes Directes* ou *Primales* [Culioli 1994] [Saldanha 1992] sont des méthodes qui travaillent directement avec le problème contraint original. Ces méthodes sont capables de trouver une solution soit à partir d'une suite de minimisations unidirectionnelles, soit en remplaçant le problème original par une suite de sous-problèmes approchés.

Il existe plusieurs méthodes directes, parmi lesquelles nous remarquons la *Programmation Quadratique Récursive* [Han 1977], la *Méthode de l'Ellipsoïde* [Shor 1977], la *Méthode des Directions Admissibles* [Culioli 1994], la *Méthode du Gradient Réduit* [Culioli 1994] et la *Méthode du Gradient Projeté* [Culioli 1994] - Figure 18.

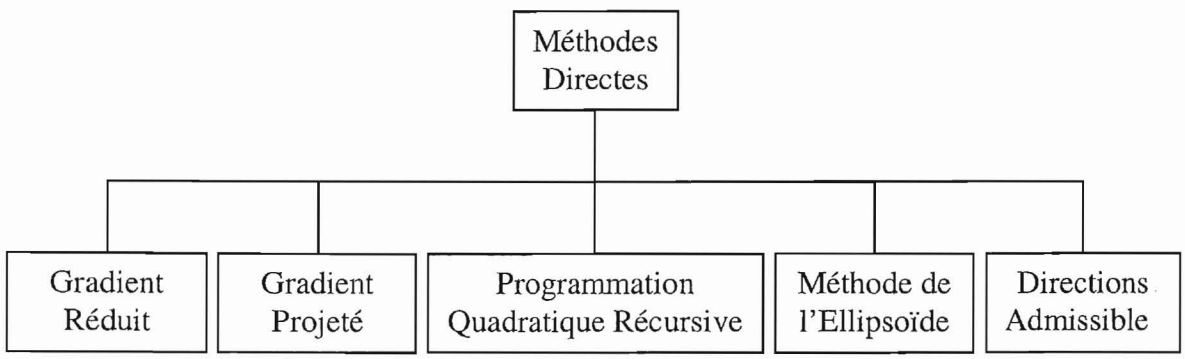


Figure 18 - Principales méthodes directes

I.4.3 - Optimisation Stochastique

La prise en compte des contraintes dans une méthode d'optimisation stochastique est souvent obtenue en utilisant une fonction de pénalité associée à la fonction objectif [Saludjian 1997]. Classiquement, on utilise une fonction de pénalité extérieure [Michalewicz 1994], selon laquelle la fonction à minimiser devient égale à:

$$\Phi(x) = f(x) + r\sum_{i=1}^{m}[\max[0,g_i(x)]]^2 \tag{30}$$

où

r est le coefficient de pénalité.

Contrairement aux méthodes de transformation déterministes, la valeur du coefficient de pénalité *r* reste constante pendant le processus d'optimisation stochastique.

I.5 - Problèmes d'Optimisation à Objectifs Multiples

Lorsque nous cherchons la minimisation ou la maximisation simultanée de plusieurs fonctions décrites par les mêmes variables objets, nous disons que le problème d'optimisation représente un problème multi-objectifs ou multi-critères. La description mathématique de ces problèmes peut être exprimée par:

$$(P)\begin{cases} \text{Min } F = \begin{bmatrix} f_1(x) \\ \vdots \\ f_s(x) \end{bmatrix} \\ g_i(x) \le 0 \quad i = 1,\ldots,p \\ x_{k\min} \le x_k \le x_{k\max} \quad k = 1,\ldots,n \end{cases} \tag{31}$$

où

$f_j(x)$ sont les fonctions objectifs que nous cherchons à minimiser simultanément.

I.5.1 - Optimum de Pareto

La difficulté principale d'un problème d'optimisation à objectifs multiples est liée à la présence de conflits entre les diverses fonctions, puisque les solutions optimales pour un certain objectif donné ne correspondent généralement pas à celles des autres fonctions. Ainsi, la plupart du temps, il n'existe aucun point de l'espace de recherche pour lequel toutes les fonctions objectifs sont optimales simultanément.

Lorsqu'il n'est plus possible d'améliorer la solution d'un des critères sans détériorer les autres, nous sommes en présence d'une solution dite *Pareto-optimale*. Formellement, x^* est un *optimum de Pareto* s'il n'y a aucune solution x pour laquelle:

$$f_j(x) < f_j(x^*) \quad \forall j \in [1,s] \tag{32}$$

La solution d'un problème multi-objectifs est alors donnée par un ensemble d'optima de *Pareto* qui constituent la *frontière de Pareto*. L'identification de cette frontière demande l'application de méthodes d'optimisation multimodales [Sareni 1999]. L'utilisation de ces méthodes, ainsi que le traitement de problèmes à objectifs multiples ne seront pas présentés dans ce travail.

I.6 - Conclusion

Dans ce premier chapitre, nous avons présenté les méthodes les plus importantes utilisées dans la résolution d'un problème d'optimisation. Nous avons remarqué que selon leurs caractéristiques, ces méthodes peuvent être réunies en deux différents groupes: les méthodes déterministes et les méthodes stochastiques.

Nous avons pu observer que les méthodes déterministes sont en général peu coûteuses, mais elles nécessitent souvent d'un point de départ et du gradient de la fonction objectif pour résoudre le problème. À l'opposé, les méthodes stochastiques ne nécessitent ni de point de départ, ni la connaissance du gradient de la fonction objectif pour atteindre la solution optimale. Cependant, elles demandent un nombre important d'évaluations avant d'arriver à la solution du problème.

Malgré le nombre important d'évaluations de la fonction objectif demandé par les méthodes stochastiques, ces méthodes présentent un grand avantage par rapport aux méthodes déterministes: la capacité de trouver l'optimum global du problème.

Le choix de la méthode à utiliser reste toutefois lié au problème à optimiser. Ainsi, le nombre de paramètres existants, la présence ou non de fonctions contraintes, la connaissance du gradient de la fonction et surtout le temps de calcul nécessaire pour faire une évaluation de la fonction objectif sont des facteurs importants à considérer avant de choisir une méthode d'optimisation.

Chapitre II

Architecture d'un Outil d'Optimisation

Chapitre II

Architecture d'un Outil d'Optimisation

II.1 - Introduction

Après avoir montré dans le chapitre précédent les principales définitions concernant les problèmes d'optimisation et les caractéristiques les plus importantes des méthodes utilisées dans sa résolution, nous allons présenter dans ce deuxième chapitre les détails d'une architecture logiciel développée dans le but de permettre l'implémentation d'un outil d'optimisation qui soit générique et performant.

Nous commencerons ce chapitre par une section consacrée à la Programmation Orientée Objet, dans laquelle nous trouverons une présentation des concepts les plus importants concernant un langage orienté objet. L'objectif de cette section ne sera pas de fournir au lecteur un manuel de programmation avec toutes les informations de syntaxe d'un langage orienté objet, mais de lui présenter les principaux aspects d'une architecture basée sur ce type de programmation.

Dans une deuxième section, nous allons montrer l'application de ces concepts dans l'implémentation d'un outil d'optimisation, en s'appuyant sur les définitions et les propriétés des méthodes d'optimisation qui ont été présentées dans le premier chapitre. La préoccupation ici ne sera pas seulement d'avoir un outil robuste du côté de l'utilisateur, permettant la

résolution de différents types de problèmes d'optimisation, mais aussi du côté du développeur, en lui proposant une manipulation simple et une extension assez facile du programme développé.

Enfin, nous allons présenter quelques aspects complémentaires à cette architecture, en considérant l'optimisation de problèmes liés à la simulation numérique, dans laquelle l'évaluation de la fonction objectif dépend d'au moins un résultat calculé par un outil de simulation externe à l'outil d'optimisation.

II.2 - Programmation Orientée Objet

Depuis les années cinquante, il y a eu plusieurs évolutions successives concernant le développement de logiciels. Une des principales a été l'apparition de la Programmation Orientée Objet (POO) [Buzzi-Ferraris 1993] [Stroustrup 1997], créée à la fin des années 70, et qui se présentait comme une nouvelle méthode de programmation qui tendait à se rapprocher de notre manière naturelle d'appréhender le monde.

À la différence de la programmation structurée [Holzner 1993], dont le principe est de diviser un programme en sous-programmes, afin de pouvoir gérer sa complexité en tenant compte avant tout du traitement de données, la conception objet s'intéresse d'abord aux données elles-mêmes, auxquelles elle associe ensuite les traitements.

L'expérience a montré que les données sont ce qu'il y a de plus stable dans la vie d'un programme et il est donc intéressant d'architecturer le programme autour d'elles. La POO partage cette même philosophie et se présente aujourd'hui comme une des avancées les plus significatives dans la technologie de conception de logiciels, ce qui nous a emmené à l'utiliser pour développer l'architecture proposée dans ce travail.

Nous allons présenter maintenant les concepts les plus importants de la Programmation Orientée Objet, parmi lesquels nous trouverons les *classes*, les *objets*, l'*encapsulation*, le *polymorphisme* et l'*héritage*.

II.2.1 - Objets

Les *objets* de la POO sont des entités informatiques qui ont été conçues de façon à reproduire dans un langage de programmation les mêmes propriétés que nous trouvons dans un objet du monde réel. Ces propriétés peuvent être résumées en deux caractéristiques principales que les objets du monde réel présentent: *ils sont dans un certain état* et *ils possèdent un certain comportement.*

Dans la POO, l'état d'un *objet* est représenté par un ensemble de données associées qu'on appelle *variables* ou *attributs* de l'objet. En même temps, son comportement est caractérisé par l'ensemble d'actions qu'il est capable de réaliser. Ces actions sont exécutées par des *méthodes* ou *fonctions membres* qu'on associe à l'objet et qui lui permettent de réagir aux sollicitations extérieures en utilisant les valeurs de ses attributs. Bref, un *objet* représente une collection d'*attributs* munis de *méthodes*, comme dans le cas de la ligne géométrique représentée dans la Figure 19.

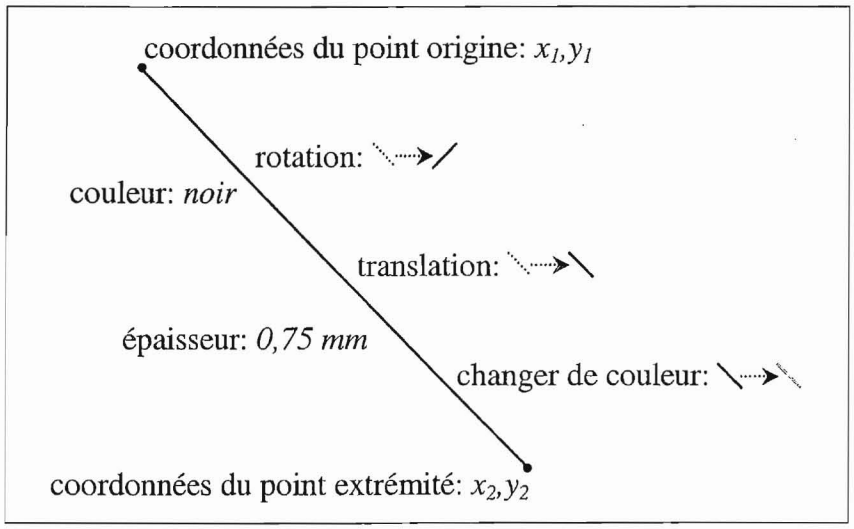


Figure 19 - Représentation d'un objet

Dans cet exemple, l'état de la ligne est décrit par les attributs *coordonnées du point origine*, *coordonnées du point extrémité*, *couleur* et *épaisseur*, tandis que les actions qu'elle est capable de réaliser sont données par les méthodes *rotation*, *translation* et *changer de couleur*. Le rôle de chacune de ces méthodes est alors de modifier les valeurs des attributs de façon à changer l'état de la ligne géométrique.

II.2.2 - Classes

Une *classe* est une structure de données qui contient les attributs et les méthodes communes à tous les objets d'une même nature. Elle représente un "moule" utilisé pour créer des objets qui ont des caractéristiques en commun mais qui peuvent être dans des états différents. On dit qu'un objet est une *instance* d'une classe.

La Figure 20 illustre l'utilisation d'une classe pour créer des objets "ligne géométrique" de l'exemple précédent. Nous vérifions que toutes les lignes ont les mêmes attributs et méthodes, mais chacune se trouve dans un état différent.

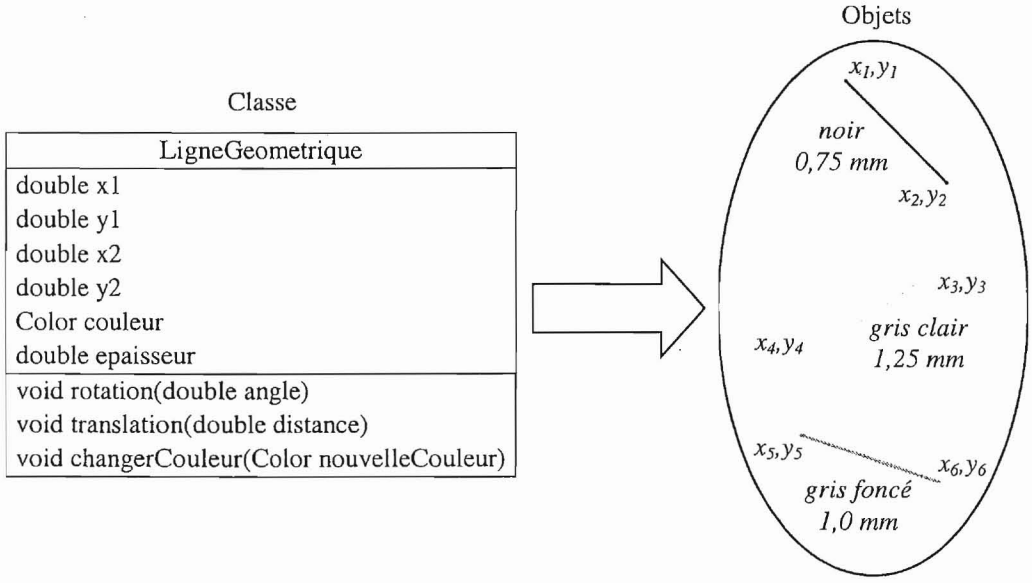


Figure 20 - Représentation d'une classe

II.2.3 - Encapsulation

L'*encapsulation* [Buzzi-Ferraris 1993] est un mécanisme qui classifie les attributs et les méthodes d'une classe (objet) en définissant leurs niveaux de visibilité, de façon à empêcher l'accès à ses données par un autre moyen que les services proposés. L'encapsulation est surtout utilisée pour garantir l'intégrité des données contenues dans un objet.

Il existe trois différents niveaux de visibilité que le mécanisme d'encapsulation nous permet de définir. Le premier est le niveau *public*, par lequel les fonctions de tous les objets

peuvent accéder aux attributs et aux méthodes d'un objet définis avec ce niveau de visibilité. Il s'agit du plus bas niveau de protection des données existant.

Le deuxième est le niveau de visibilité *protégé*, dans lequel l'accès aux données est réservé aux fonctions des objets qui font partie de la même hiérarchie (II.2.5), c'est-à-dire par les méthodes de l'objet lui-même et par ses dérivés. Enfin, la visibilité *privée* limite exclusivement l'accès aux données aux méthodes de l'objet lui-même. Il s'agit du niveau de protection des données le plus élevé.

La Figure 21 présente l'encapsulation des données de l'objet "ligne géométrique" de l'exemple précédent. Nous pouvons observer que les attributs de cet objet sont exclusivement accessibles par ses méthodes, car ils ont étés définis avec une visibilité privée (*private*). Par contre, les méthodes de rotation et de translation sont accessibles par tous les autres objets, car son niveau de visibilité est *public*. Finalement, la méthode de changement de couleur est accessible par un groupe d'objets restreint, car elle est une méthode protégée (*protected*).

LigneGeometrique
private double x1 private double y1 private double x2 private double y2 private Color couleur private double epaisseur
public void rotation(double angle) public void translation(double distance) protected void changerCouleur(Color nouvelleCouleur)

Figure 21 - Encapsulation

II.2.4 - Polymorphisme

Le *polymorphisme* est une propriété essentielle de la POO caractérisée par la possibilité d'avoir, dans un même programme, plusieurs fonctions de même nom mais possédant des comportements différents.

En utilisant le mécanisme de polymorphisme, nous pouvons attribuer le même nom à différentes méthodes d'une classe ou encore à différentes méthodes de différentes classes. C'est lui qui se charge de choisir automatiquement la bonne méthode à utiliser en fonction du type ou du nombre d'arguments spécifiés lors de son appel.

En revenant à l'exemple précédent, nous pouvons observer dans la Figure 22 l'existence de deux méthodes de rotation associées à l'objet ligne géométrique. La première méthode réalise une rotation toujours dans la même direction, tandis que la deuxième nous permet de choisir la direction de rotation. Les deux méthodes possèdent le même nom et elles ne sont distinguées que par le nombre d'arguments.

LigneGeometrique
protected double x1 protected double y1 protected double x2 protected double y2 protected Color couleur protected double epaisseur
public void rotation(double angle) public void rotation(double angle, boolean direction) public void translation(double distance) public void changerCouleur(Color nouvelleCouleur)

Figure 22 - Polymorphisme

II.2.5 - Héritage

L'*héritage* est l'un des plus importants concepts de la POO, permettant de créer une nouvelle classe à partir d'une autre existante. Ce mécanisme, parfois appelé *dérivation de classes*, représente la notion de partage de données, dans laquelle une classe utilise les attributs et les méthodes de sa super classe (la classe dont elle dérive) comme s'ils étaient définis directement sur elle-même.

L'intérêt majeur de l'héritage est de pouvoir définir de nouveaux attributs et de nouvelles méthodes pour la classe dérivée, qui viennent s'ajouter à ceux et celles déjà hérités. Ainsi, nous pouvons créer une hiérarchie de classes de plus en plus spécialisées, sans avoir besoin de repartir de zéro lorsque nous voulons spécialiser une classe existante. Bref, le mécanisme d'héritage nous donne la possibilité de réutiliser les classes déjà implémentées.

La capacité de dériver une classe à partir d'une autre nous permet aussi de définir des *classes abstraites* qui contiennent les caractéristiques communes à toutes leurs descendantes, mais qui ne peuvent pas être utilisées directement pour créer des objets. Ces classes sont normalement utilisées pour définir la structure de données que leurs descendants doivent posséder.

La Figure 23 nous montre un exemple d'héritage appliqué à la classe "ligne géométrique", dans lequel nous pouvons vérifier que les classes dérivées partagent quelques attributs et quelques méthodes, mais qu'elles ont aussi leurs propres données.

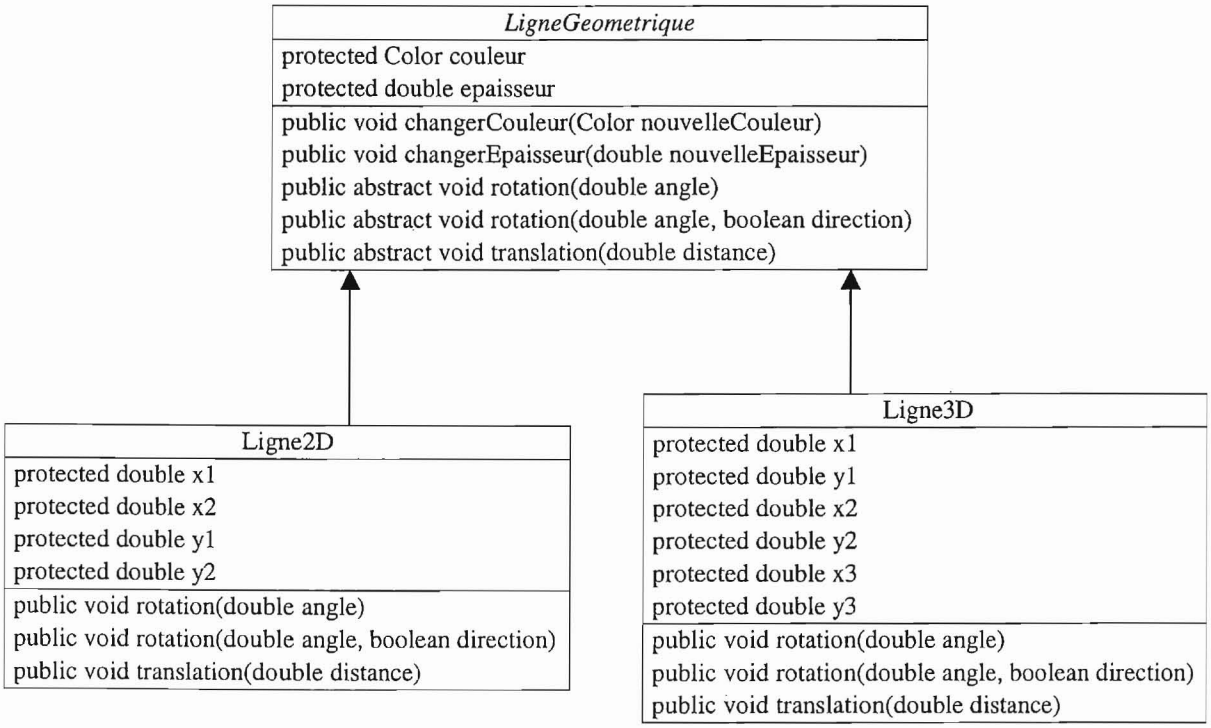


Figure 23 - Héritage

II.2.6 - Comparaison entre JAVA™ et C++

Parmi les langages de programmation orientés objet les plus diffusés actuellement, nous avons le C++ [Stroustrup 1997] qui date de 1982 et le JAVA™ [Horstmann 1999], dont les premières applications datent de 1995. Malgré son temps d'existence relativement court, nous avons choisi le JAVA™ pour développer notre outil d'optimisation, car il présente plusieurs avantages par rapport au C++, à savoir:

- La *portabilité*: JAVA™ a été conçu pour créer des programmes qui tournent sur n'importe quelle plateforme existante. Cette caractéristique nous permet d'utiliser notre outil sous différents systèmes d'exploitation sans avoir besoin de modifier le code source.

- La *gestion de mémoire automatique*: l'environnement d'exécution de JAVA™ possède un processus de "ramasse-miettes" (*garbage collector*) qui s'occupe de la gestion

automatique de la mémoire utilisée par le programme. Cette gestion automatique évite des erreurs de programmation souvent rencontrées dans les programmes écrits en C++.

- Concepts de *parallélisme*: les applications en JAVA™ peuvent posséder plusieurs flux de contrôle ou *threads*, ainsi que des méthodes pour gérer ces *threads*, en simplifiant le développement et la gestion de programmes destinés à travailler avec du calcul distribué, ce qui peut être très utile dans un processus d'optimisation.

- La notion de *packages*: JAVA™ permet de regrouper l'ensemble de classes qui font partie d'une application en *packages*, ce qui élimine la redondance occasionnée par l'utilisation de fichiers d'en-tête (*header files*) et facilite la compréhension du code développé.

Nous avons encore d'autres différences concernant les deux langages [Lemay 2000], mais qui n'ont pas eu d'influence directe sur notre choix.

II.3 - Un Outil d'Optimisation Orienté Objet

Après l'exposition des principaux concepts de la Programmation Orientée Objet, nous allons présenter dans cette section l'application de ces concepts dans l'implémentation d'un outil d'optimisation. Dans ce contexte, nous décrirons un ensemble de classes idéalisées de façon à permettre la création de cet outil.

Ces classes seront présentées en deux étapes. Tout d'abord, nous allons concentrer notre attention vers les classes utilisées dans *l'étape de description* du problème d'optimisation. Pour cela, nous allons reprendre les définitions de problèmes contraints et non contraints du chapitre précédent.

Ensuite, nous allons présenter les classes consacrées à *l'étape de résolution* du problème, en revenant aux méthodes d'optimisation présentées dans le premier chapitre. Le but de cette étape sera de réunir ces classes en différents groupes, de façon à partager les caractéristiques qu'elles ont en commun.

II.3.1 - Description d'un Problème d'Optimisation

Les classes qui seront présentées ici ont été idéalisées de façon à permettre la conception d'un problème d'optimisation quelconque. Elles représentent les classes les plus importantes de l'architecture proposée, car les objets créés à partir d'elles sont utilisés par toutes les autres classes qui font partie de cette architecture, surtout celles concernant les méthodes d'optimisation.

II.3.1.1 - Classe *OptimizationProblem*

Comme nous avons pu vérifier dans le premier chapitre, un problème d'optimisation non contraint (5) est décrit par une fonction objectif ($f(x)$) et un ensemble de paramètres avec ses bornes de variation (x_{kmin} et x_{kmax}) qui représentent les contraintes de domaine. Lorsqu'il s'agit d'un problème contraint (20), nous avons encore une liste de fonctions contraintes ($g_i(x)$) qui font partie de sa description.

Un objet qui représente un problème d'optimisation doit alors contenir au moins deux attributs: une fonction objectif et un ensemble de paramètres. En plus, il doit savoir s'il représente un problème contraint et, dans ce cas-là, avoir une liste de fonctions contraintes. Il doit savoir aussi s'il s'agit d'un problème de minimisation ou de maximisation et, dans le deuxième cas, se charger de transformer la valeur de l'évaluation de la fonction objectif comme indiqué en (4).

Enfin, cet objet doit posséder des méthodes qui permettent d'accéder à ses attributs et d'autres qui soient capables de retourner la valeur de l'évaluation de la fonction objectif, ainsi que de son gradient, pour une configuration de paramètres donnée.

La Figure 24 présente les attributs et les méthodes les plus importants de la classe *OptimizationProblem* avec ses dérivées *ConstrainedProblem* et *UnconstrainedProblem* qui ensemble, permettent la création d'objets avec toutes ces caractéristiques.

Les attributs *Function* et *ParametersSet* correspondent respectivement à la fonction objectif et à l'ensemble de paramètres du problème à optimiser. Les classes utilisées pour créer ces objets seront détaillées dans la suite.

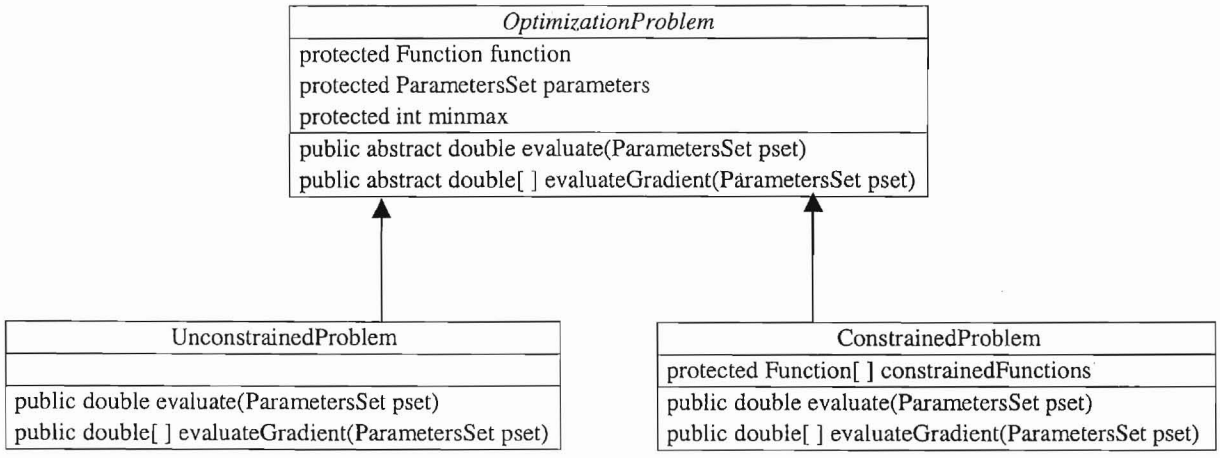


Figure 24 - Classe *OptimizationProblem*

Les méthodes *evaluate* et *evaluateGradient* retournent respectivement la valeur de la fonction objectif et de son gradient. Lorsqu'il s'agit d'un problème de maximisation, indiqué par l'attribut *minmax*, ces méthodes se chargent de transformer ces valeurs (4). Dans le cas d'un problème contraint dont la solution est obtenue par l'application d'un algorithme de pénalité, les valeurs retournées par ces méthodes correspondent à l'évaluation de la fonction objectif augmentée d'une pénalité donnée par les fonctions contraintes non vérifiées.

II.3.1.2 - Classe *Parameter*

Dans le premier chapitre, nous avons vu qu'un paramètre représente une des variables du problème à optimiser (1). Dans l'optimisation de dispositifs électromagnétiques, il peut représenter soit une grandeur géométrique, comme par exemple la dimension d'un entrefer, soit une grandeur physique, comme le courant dans une bobine. Pourtant, dans tous les cas il peut être vu comme une simple variable analytique décrite par son *nom*, sa *valeur* et ses *limites de variations* (contraintes de domaine).

Dans ce contexte, un objet qui correspond à un paramètre d'un problème d'optimisation quelconque doit toujours contenir ces quatre attributs, ainsi que des méthodes pour y accéder, comme dans le cas de la classe *Parameter*, présentée dans la Figure 25.

Parmi les attributs de cette classe, le nom du paramètre est le plus important, car il représente l'identité de chaque paramètre existant, ce qui nous permet de les identifier avant de les utiliser.

Parameter
protected String name protected double value protected double minValue protected double maxValue
public String getName() public void setName(String newName) public double getValue() public void setValue(double newValue) public double getMinValue() public double getMaxValue() public void setLimits(double min, double max) public void generateValue()

Figure 25 - Classe *Parameter*

La méthode *setLimits* s'occupe de vérifier si les limites de variation définies sont valables (*minValue* < *maxValue*), tandis que la méthode *setValue* se charge de vérifier si la valeur du paramètre est comprise entre les limites de variation spécifiées. Enfin, la méthode *generateValue* permet de définir la valeur du paramètre à partir de la génération aléatoire d'un nombre réel compris entre les limites de variation, ce qui devient très utile lorsque nous utilisons une méthode d'optimisation stochastique.

II.3.1.3 - Classe *ParametersSet*

La plupart des problèmes d'optimisation sont décrits par des fonctions objectifs multidimensionnelles, c'est-à-dire, à plus d'un paramètre. Chacun de ces paramètres doit avoir une identité unique et, comme nous venons de voir, cette identité est donnée par son nom.

Pour éviter l'existence de deux paramètres avec le même nom et pouvoir les utiliser d'une façon assez simple, il devient intéressant d'avoir un objet qui contienne tous les paramètres du problème à optimiser. La classe *ParametersSet*, illustrée dans la Figure 26, peut être utilisée dans ce but.

Cette classe contient un tableau d'objets du type *Parameter* qui sont accessibles à partir des méthodes où nous spécifions soit le nom du paramètre, soit la position qu'il occupe dans le tableau. Cela rend possible la récupération et la modification facile et fiable de n'importe quelle donnée de n'importe quel paramètre, sans avoir de confusions entre eux.

ParametersSet
protected Parameter[ ] parameters
public void add (Parameter param) public void remove(Parameter param) public int getNumberOfParameters() public String getName(int pos) public double getValue(String name) public double getValue(int pos) public void setValue(String name, double value) public void setValue(int pos, double value)

Figure 26 - Classe *ParametersSet*

La méthode *getNumberOfParameters* nous permet de récupérer le nombre de paramètres existants dans un objet *ParametersSet*. La méthode *add* est utilisée pour ajouter un nouveau paramètre au tableau existant, en vérifiant d'abord son nom, de façon à éviter l'existence de deux paramètres de même nom.

II.3.1.4 - Classe *Function*

La classe *Function* est la super classe de toutes les classes utilisées pour créer une fonction, soit objectif soit contrainte. Elle est une classe abstraite, ce qui veut dire qu'elle ne peut pas être directement utilisée pour créer des objets.

La Figure 27 présente les attributs et les méthodes les plus importants de cette classe, parmi lesquels nous trouvons l'ensemble de paramètres qui la décrit, le nom de la fonction et un compteur pour indiquer le nombre d'évaluations de la fonction pendant le processus d'optimisation.

Function
protected String name protected Formula formula protected ParametersSet parameters protected int evaluationsCount protected GradientEstimator gradEstimator
public abstract double evaluate(ParametersSet pset) public abstract double[ ] evaluateGradient(ParametersSet pset)

Figure 27 - Classe *Function*

La méthode *evaluate* doit être implémentée par toutes les classes dérivées de *Function*. C'est la plus importante méthode de la classe, car elle est utilisée pour évaluer la fonction pour un ensemble de paramètres donné.

La méthode *evaluateGradient*, à son tour, doit être surchargée de façon à calculer le gradient de la fonction, souvent utilisé par les méthodes déterministes. Dans le cas où ce calcul ne peut pas être effectué de façon explicite, cette méthode se charge de calculer le gradient par Différences Finies [Kadded 1993] ou par une méthode similaire définie par l'attribut *GradientEstimator*.

Enfin, l'attribut *Formula* représente l'expression analytique qui décrit la fonction. En ce qui concerne la description d'une fonction, nous pouvons avoir deux différentes situations: soit la fonction est exprimée par ses paramètres de façon *explicite*, soit la relation entre elle et ses paramètres est donnée de façon *implicite*.

- Fonctions Explicites

Dans ce premier cas, nous pouvons évaluer la fonction de façon directe, en remplaçant tout simplement les valeurs des paramètres dans l'expression qui la décrit. C'est le cas de *fonctions analytiques*, comme par exemple la fonction de *Rastrigin* à *n* paramètres, décrite par l'équation (33) et représentée dans la Figure 28.

$$\begin{cases} f(x) = 10 \cdot n + \sum_{i=1}^{n} (x_i - 2)^2 - 10\cos(2\pi(x_i - 2)) \\ -5.0 < x_i < 5.0 \end{cases} \tag{33}$$

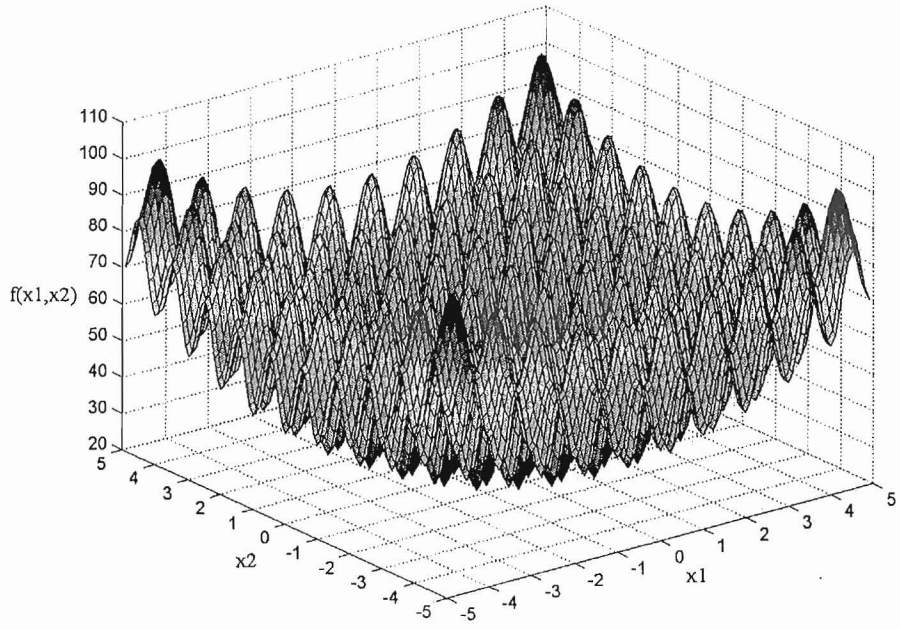


Figure 28 - Fonction de Rastrigin

- Fonctions Implicites

Dans cette deuxième situation, nous ne pouvons pas évaluer la fonction en utilisant directement l'expression donnée, car elle est composée d'au moins un résultat provenant d'un outil de calcul numérique externe à elle. Nous trouvons ce type de situation dans la résolution de problèmes décrits par des *fonctions numériques*, comme par exemple l'optimisation de dispositifs électromagnétiques analysés par la Méthode des Éléments Finis, où l'évaluation de la fonction demande au moins une résolution par éléments finis, comme le montre la Figure 29.

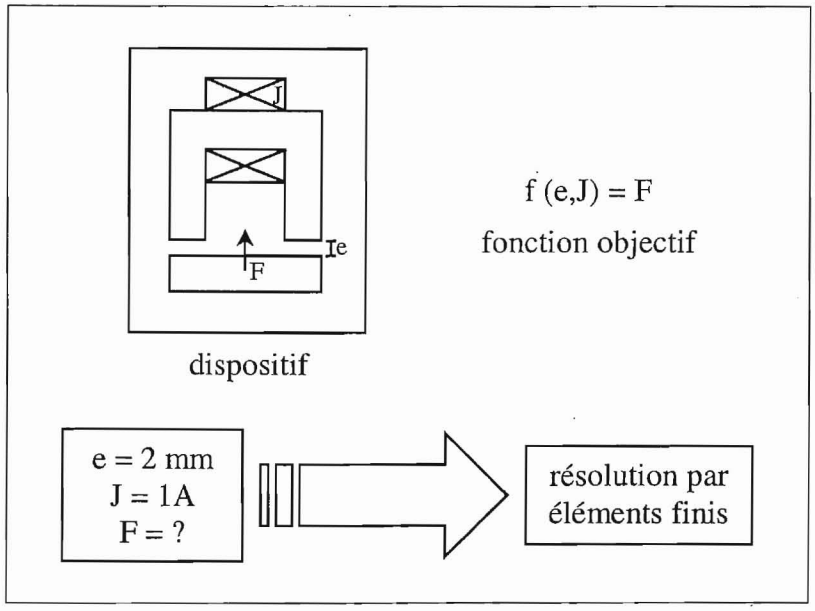


Figure 29 - Fonction implicite

L'optimisation de fonctions implicites représente un problème majeur pour la description et la résolution du problème d'optimisation, car elle va exiger l'existence d'un module de pilotage des outils numériques impliqués dans le calcul des résultats qui composent la fonction. Ce module de pilotage doit effectuer l'échange d'informations entre les différents modules du système. Les détails de son implémentation seront présentés dans la section II.4.

II.3.2 - Résolution d'un Problème d'Optimisation

Une fois que nous avons spécifié les classes nécessaires à la description d'un problème d'optimisation, il nous reste à définir celles qui seront utilisées dans sa résolution. Pour implémenter ces classes, nous allons revenir aux méthodes d'optimisation présentées dans le chapitre I, en prenant pour base les caractéristiques qu'elles ont en commun.

Nous retrouvons ces méthodes dans le Tableau II, dans lequel nous avons aussi une reproduction de la classification qui a été donnée dans le premier chapitre.

TABLEAU II - HIÉRARCHIE DE MÉTHODES D'OPTIMISATION

Méthode d'Optimisation	Méthodes Déterministes	Méthodes d'Optimisation sans Contraintes	Méthodes Unidimensionnelles	Dichotomie	
				Section Dorée	
				Brent	
				Fibonacci	
				Armijo et Goldstein	
			Méthodes Multidimensionnelles	Méthodes Heuristiques	Simplex
					Rosenbrock
					Hooke et Jeeves
				Méthodes Analytiques	Plus Grande Pente
					Gradient Conjugué
					QuasiNewton
					Powell
		Méthodes d'Optimisation avec Contraintes	Méthodes de Transformation	Pénalités	
				Lagrangien Augmenté	
				Variables Mixtes	
				Asymptotes Mobiles	
			Méthodes Directes	Ellipsoïde	
				Programmation Quadratique Recursive	
				Directions Admissibles	
				Gradient Réduit	
				Gradient Projeté	
	Méthodes Stochastiques	Méthodes Évolutionnistes	Algorithmes Génétiques		
			Stratégies d'Évolution		
			Programmation Évolutionniste		
			Programmation Génétique		
		Recuit Simulé			
		Recherche Tabu			

Cette classification naturelle de méthodes d'optimisation servira de base pour la création de la hiérarchie de classes de l'architecture proposée, représentée dans le Tableau III, dont les classes détachées en gras ont été effectivement implémentées dans l'outil. Dans cette hiérarchie, toutes les classes sont abstraites, à l'exception des celles qui sont dans le niveau le plus bas et qui contiennent alors l'algorithme de résolution de la méthode.

TABLEAU III - HIÉRARCHIE DE CLASSES CONCERNANT LES MÉTHODES D'OPTIMISATION

Optimizer	**Deterministic Optimizer**	**Unconstrained Optimizer**	**OneDimension Optimizer**	Bisection	
				GoldenSection	
				Brent	
				Fibonacci	
				ArmijoGoldstein	
			MultiDimension Optimizer	**Heuristic Optimizer**	Simplex
					Rosenbrock
					HookeJeeves
				Analytical Optimizer	Steepest Descent
					Conjugate Gradient
					QuasiNewton
					Powell
		Constrained Optimizer	**Transformer Optimizer**	**Penalty**	
				AugmentedLagrangian	
				VariablesMixtes	
				AsymptotesMobiles	
			DirectOptimizer	Ellipsoïde	
				SequentialQuadratic Programming	
				DirectionsAdmissibles	
				ReducedGradient	
				ProjectedGradient	
	Stochastic Optimizer	**Evolutionary Algorithm**	**GeneticAlgorithms**		
			EvolutionStrategy		
			EvolutionaryProgramming		
			GeneticProgrammimg		
		SimulatedAnnealing			
		TabuSearch			

Les classes qui sont en gris foncé jouent un rôle très important dans cette hiérarchie, car elles ont au moins un attribut ou une fonction utilisés par ses descendantes. Les détails à propos de ces classes seront donnés dans les sections qui suivent.

Les classes qui sont en gris clair composent le niveau le plus bas de la hiérarchie et contiennent une fonction dans laquelle l'algorithme de résolution de la méthode est implémenté. À part les algorithmes de résolution qui ont été déjà décrits dans le premier chapitre, ces méthodes n'ont rien de spécial et pour cette raison nous n'allons pas les détailler.

Finalement, nous avons les classes en blanc qui n'ont été créées que pour favoriser la classification des méthodes dans la hiérarchie. Ces classes n'ont ni des méthodes, ni des attributs et pour cette raison elles ne seront pas décrites.

II.3.2.1 - Classe *Optimizer*

Dans le niveau le plus élevé de la hiérarchie, nous avons la classe abstraite *Optimizer* qui possède les caractéristiques communes à toutes les méthodes d'optimisation. Cette classe contient le nom de la méthode, une référence au problème à optimiser, une liste de configurations de paramètres représentant l'évolution de la solution du problème et une référence à la solution obtenue à la fin du processus d'optimisation.

Elle contient aussi une fonction abstraite qui doit être implémentée par les classes qui sont dans le niveau le plus bas de la hiérarchie et dans laquelle l'algorithme d'optimisation réalisé par la méthode doit apparaître. Toutes ces données sont reproduites dans la Figure 30.

Optimizer
protected String name protected OptimizationProblem problem protected LinkedList evolution protected ParametersSet solution
public abstract optimize(OptimizationProblem problem)

Figure 30 - Classe *Optimizer*

II.3.2.2 - Classe *DeterministicOptimizer*

Comme nous avons remarqué dans la section I.3.1, la plupart des méthodes déterministes ont deux caractéristiques particulières: elles ont besoin d'un point de départ pour démarrer le processus d'optimisation et leur critère d'arrêt est normalement donné soit par le nombre d'itérations effectuées, soit par une tolérance de convergence spécifiée.

Nous pouvons trouver ces propriétés dans la classe *DeterministicOptimizer*, présentée dans la Figure 31, qui représente la super classe de toutes les méthodes déterministes.

DeterministicOptimizer
protected int maxIterations protected double tol protected ParametersSet initialPoint

Figure 31 - Classe *DeterministicOptimizer*

II.3.2.3 - Classe *OneDimensionOptimizer*

Les méthodes déterministes unidimensionnelles sont presque toutes basées sur un algorithme de réduction de l'intervalle de recherche dans lequel on fait appel à l'identification d'un triplet (x_1, x_2, x_3), tel que $f(x_1) > f(x_2) < f(x_3)$.

La classe *OneDimensionOptimizer* contient alors une fonction capable d'identifier ce triplet à partir d'une direction de recherche donnée. Elle contient aussi une fonction abstraite représentant l'algorithme de recherche linéaire qui doit être implémenté par toutes ses classes dérivées - Figure 32.

OneDimensionOptimizer
public abstract Vector bracket(double[ ] direction) public abstract ParametersSet search(ParametersSet pset, double[ ] direction)

Figure 32 - Classe *OneDimensionOptimizer*

II.3.2.4 - Classe *AnalyticalOptimizer*

La section I.3.1.2 a montré que les méthodes déterministes multidimensionnelles du type analytique font appel à des méthodes de minimisation linéaire pour minimiser la fonction en différentes directions de recherche à chaque itération de la méthode. La classe *AnalyticalOptimizer* contient alors un attribut de référence à une méthode de recherche linéaire quelconque, comme nous pouvons le vérifier dans la Figure 33.

AnalyticalOptimizer
protected OneDimensionOptimizer oneDimOpt

Figure 33 - Classe *AnalyticalOptimizer*

II.3.2.5 - Classe *TransformerOptimizer*

Ainsi comme les méthodes déterministes multidimensionnelles font appel à des méthodes de recherche linéaire, nous avons vu dans la section I.4.1 que les méthodes déterministes de transformation font appel à des méthodes de résolution sans contraintes après avoir transformé le problème contraint en non contraint. Dans ce cas-là, il devient nécessaire d'ajouter à la classe *TransformerOptimizer* un attribut de référence à une méthode multidimensionnelle de résolution sans contraintes quelconque - Figure 34.

TransformerOptimizer
protected MultiDimensionOptimizer multiDimOpt

Figure 34 - Classe *TransformerOptimizer*

II.3.2.6 - Classe *StochasticOptimizer*

La classe *SthochasticOptimizer* représente la super classe de toutes les méthodes stochastiques implémentées dans cette architecture. Le principal attribut qu'elle contient, est un générateur de nombres aléatoires représenté par l'objet *RandomGenerator* (Figure 35), car comme nous savons, les méthodes stochastiques sont basées sur des techniques de recherche aléatoire.

StochasticOptimizer
protected RandomGenerator random

Figure 35 - Classe *StochasticOptimizer*

II.3.2.7 - Classe *EvolutionaryAlgorithm*

La classe *EvolutionaryAlgorithm* contient les propriétés communes à toutes les méthodes évolutionnistes, telles que la population d'individus, le critère d'arrêt défini par le nombre de générations et la liste d'opérateurs génétiques utilisés par l'algorithme de résolution - Figure 36.

EvolutionaryAlgorithm
protected int generations protected ParametersSet[ ] population protected GeneticOperators[ ] operators

Figure 36 - Classe *EvolutionaryAlgorithm*

II.4 - Pilotage des Outils de Simulation Numérique

La résolution d'un problème d'optimisation lié à la simulation par calcul numérique demande un traitement particulier, car la valeur d'évaluation de sa fonction objectif est calculée en utilisant au moins un résultat provenant d'un outil de simulation externe à l'outil d'optimisation.

En plus, dans le cas de simulations en électrotechnique, ces résultats viennent souvent de l'analyse de dispositifs constitués par des géométries et des physiques complexes, ce qui représente une autre contrainte à leur obtention, car l'outil de simulation doit effectuer différentes tâches (modifier la géométrie, générer le maillage, résoudre le système d'équations, ...) avant de les calculer.

Pour avoir accès à ces résultats, un module de pilotage qui effectue l'échange d'informations entre l'outil d'optimisation et l'outil de simulation devient alors nécessaire. Ce module de pilotage doit aussi coordonner les tâches réalisées par l'outil de simulation, en effectuant les étapes suivantes:

- Tout d'abord, il doit démarrer l'outil de simulation numérique et lui demander le chargement d'un projet dans lequel nous trouvons les paramètres et les résultats qui seront utilisés lors de l'évaluation de la fonction objectif. Une fois le projet chargé, le module de pilotage doit récupérer ces informations, les stocker dans sa base de données et les envoyer à l'outil d'optimisation.

- Dans l'étape d'évaluation de la fonction objectif, le module de pilotage doit actualiser sa base de données en utilisant les valeurs des paramètres spécifiées par l'outil d'optimisation. Il doit ensuite envoyer ces valeurs à l'outil de simulation, en lui demandant la modification des paramètres du projet chargé et un nouveau calcul

des résultats utilisés dans l'évaluation de la fonction objectif. À la fin de ce calcul, le module de pilotage doit récupérer les résultats et les envoyer à l'outil d'optimisation.

- Pendant tout le processus d'optimisation, le module de pilotage doit se charger de la synchronisation entre les deux outils, de façon à les mettre en veille ou les relancer quand c'est nécessaire.

- Le module de pilotage ne doit pas rester attaché à un seul outil de simulation, mais au contraire, il doit permettre l'échange d'informations entre l'outil d'optimisation et différents outils de simulation numériques, ayant chacun ses propres caractéristiques.

La Figure 37 illustre les différentes tâches qui doivent être exécutées par le module de pilotage.

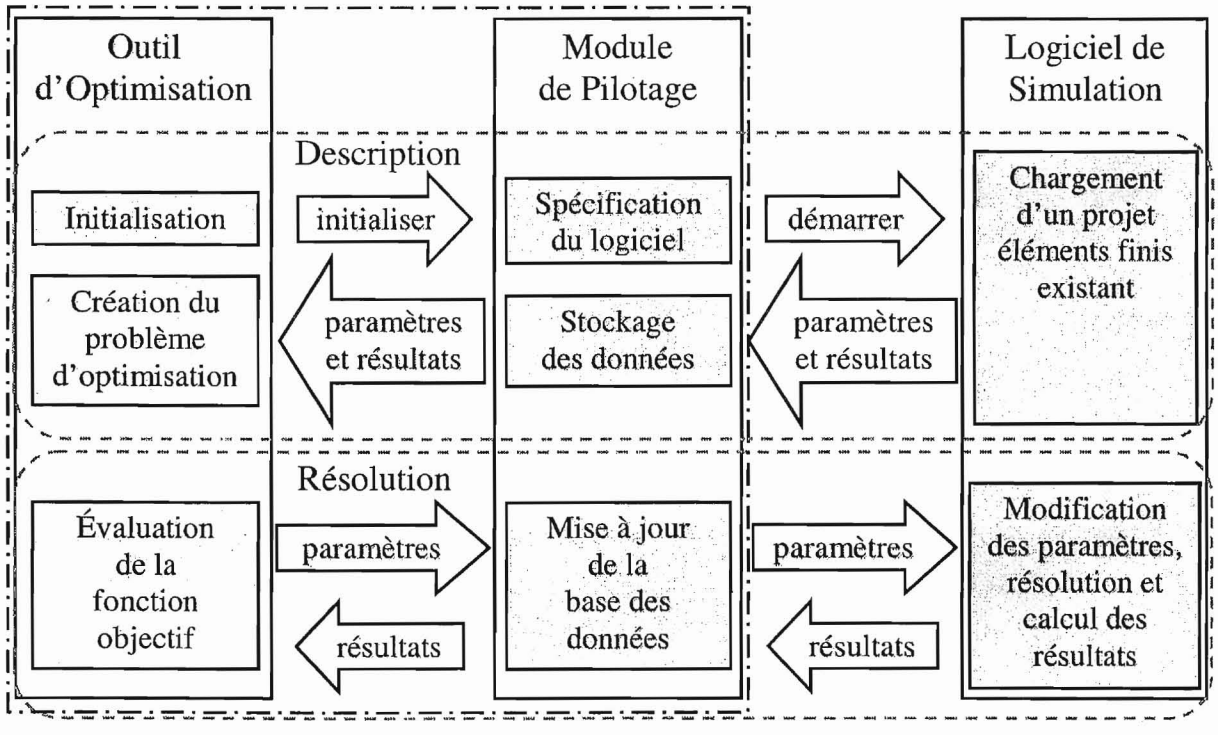


Figure 37 - Tâches effectuées par le module de pilotage

Nous avons ajouté à l'architecture de l'outil d'optimisation proposée un ensemble de classes dont les attributs et les méthodes permettent la réalisation de ces tâches.

Au niveau le plus élevé de la hiérarchie créée à partir de ces classes, nous trouvons les attributs destinés au stockage des informations récupérées de l'outil de simulation. Nous avons aussi les méthodes utilisées pour manipuler ces attributs et celles utilisées pour effectuer l'échange d'informations du coté de l'outil d'optimisation.

Le transfert de données du coté de l'outil de simulation est réalisé par des méthodes situées dans des classes plus spécialisées qui se trouvent au niveau le plus bas de la hiérarchie. Chacune de ces classes contient aussi les méthodes utilisées dans la synchronisation de l'outil de simulation auquel elle se réfère.

La Figure 38 présente la hiérarchie de classes de pilotage concernant les différents outils de simulation numérique.

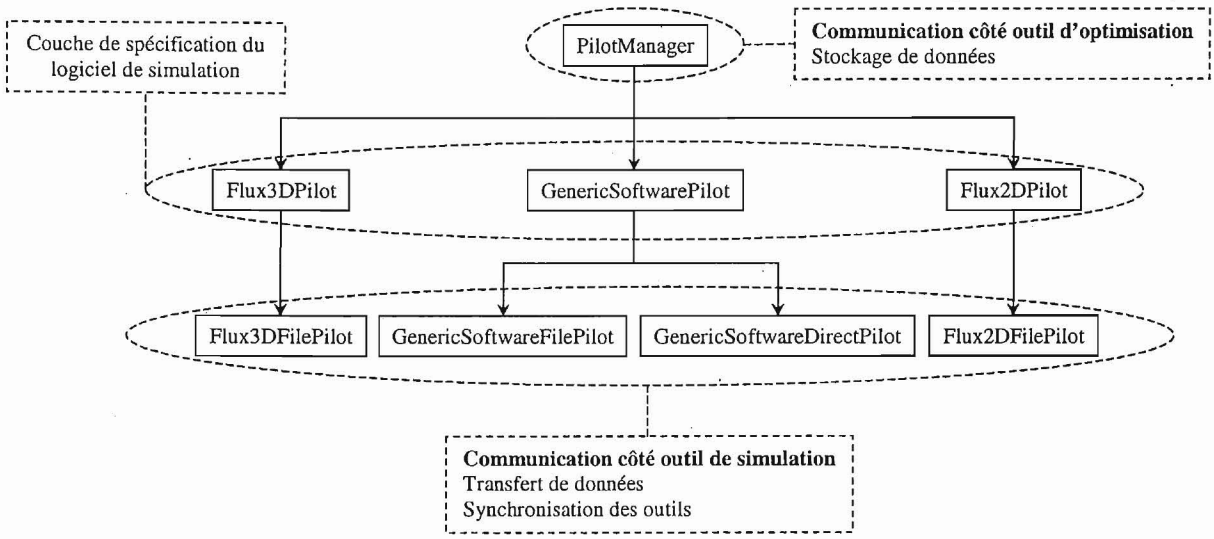


Figure 38 - Hiérarchie de classes du module de pilotage

II.4.1 - Stockage de Données

Le module de pilotage possède une base de données complètement indépendante du reste de l'outil d'optimisation et de l'outil de simulation numérique. C'est-à-dire que les données concernant les paramètres et les résultats utilisés par ces trois composants ne partagent pas la même adresse en mémoire, ce qui nous permet d'avoir un plus grand contrôle sur les données qui composent le système.

II.4.1.1 - Stockage de Paramètres

Le stockage de paramètres est fait en utilisant une liste d'objets du type *Parameter* que nous avons présenté dans la section II.3.1.2. L'accès à ces objets est réalisé par des fonctions où nous spécifions le nom du paramètre que nous voulons manipuler.

Lorsque nous modifions la valeur d'un paramètre, il est ajouté à une deuxième liste de paramètres modifiés. À partir de cette liste, nous pouvons identifier les paramètres qui doivent être modifiés dans le projet chargé par l'outil de simulation lors d'une prochaine évaluation de la fonction objectif.

II.4.1.2 - Stockage de Résultats

Comme les paramètres, les résultats sont aussi stockés dans une liste d'objets. Un objet qui décrit un résultat doit posséder au moins un nom et une valeur comme attributs.

Pour éviter une nouvelle évaluation de résultats qui ont déjà été calculés par l'outil de simulation, le module de pilotage contient un tableau qui nous permet de récupérer le résultat associé à une configuration de paramètres qui a déjà été évaluée.

II.4.2 - Transfert de Données

Nous pouvons remarquer dans la Figure 37 que l'échange d'informations entre l'outil d'optimisation et l'outil numérique passe toujours par une couche représentée par le module de pilotage. De cette façon, nous avons deux étapes de transfert de données à analyser: le transfert entre l'outil d'optimisation et le module de pilotage et celui entre le module de pilotage et l'outil de simulation.

II.4.2.1 - Transfert entre le Module de Pilotage et l'Outil d'Optimisation

Dans la mesure où le module de pilotage partage le même code que l'outil d'optimisation (nous pouvons même dire qu'il appartient à l'outil d'optimisation), l'échange d'informations s'effectue de façon directe, par simple appel à des méthodes liées aux objets qui contiennent les données.

II.4.2.2 - Transfert entre le Module de Pilotage et l'Outil de Simulation

L'échange d'informations entre le module de pilotage et l'outil de simulation numérique ne peut pas s'effectuer de façon directe, car le module de pilotage n'a pas d'accès ni au code ni à la base de données de l'outil de simulation. Cependant, nous avons d'autres façons d'effectuer l'échange d'informations entre le module de pilotage et l'outil de simulation numérique [Merrouche 1997].

La première, appelée *communication active* [Merrouche 1997], consiste en créer une bibliothèque dynamique [Horstmann 1999] qui soit accessible par les deux parties du système, dans laquelle nous avons des fonctions qui ont accès aux bases de données des deux cotés. Nous revenons alors à la situation de partage de code qui permet une communication directe entre les outils.

Une deuxième façon de réaliser l'échange d'informations entre les deux outils peut être obtenue en utilisant des fichiers de transfert dans lesquels nous écrivons les informations que nous voulons échanger. Cette solution, connue par le nom de *communication passive* [Merrouche 1997], est plus facile à mettre en œuvre, mais moins efficace. En plus, elle exige que l'outil de simulation soit capable d'interpréter les fichiers de transfert, ce qui représente une condition qui n'est pas toujours facile à satisfaire.

II.4.3 - Synchronisation de Tâches

Normalement, le processus d'un programme informatique possède un seul flux de contrôle (*thread*) dans lequel les étapes réalisées se déroulent séquentiellement. Ceci indique que pendant toute l'exécution du programme, nous n'avons jamais deux fonctions exécutées en même temps.

Cependant, il existe des situations où nous nous sommes en présence de processus à plusieurs flux de contrôle (*multithreads*), dans lesquels nous avons deux fonctions ou plus qui tournent simultanément. Dans cette situation, les étapes réalisées ne se déroulent plus de façon séquentielle, ce qui demande une synchronisation des différents *threads* dans le but de garantir que le déroulement des tâches devienne séquentiel.

Une troisième situation qui peut se présenter est celle que nous avons lorsque nous utilisons un outil de simulation numérique qui ne fait pas partie du même code de l'outil

d'optimisation. Dans ce cas, nous avons deux processus qui se déroulent de façon concomitante et qui doivent alors être aussi synchronisés.

La synchronisation de deux processus se présente comme une démarche dont le but est d'activer un processus alors que l'autre reste en veille. La partie la plus délicate de cette démarche consiste à bien choisir la façon et surtout le moment de mettre chacun des processus en sommeil, ainsi que de les réveiller. Dans le cas où nous utilisons des fichiers pour faire le transfert de données, ce moment est représenté par l'instant de finalisation d'écriture du fichier.

II.5 - Conclusion

Nous avons présenté dans ce deuxième chapitre une architecture logiciel consacrée au développement d'un outil d'optimisation. Cette architecture a été conçue de façon à permettre l'optimisation de différents problèmes, en utilisant différentes méthodes de résolution, dont la grande diversité nous a amenés à appliquer les concepts de la Programmation Orientée Objet dans son implémentation.

Les solutions répondant aux quelques difficultés posées par les problèmes d'optimisation liés à la simulation numérique, telles que l'échange de données entre les composants du système et le contrôle de tâches de l'outil de simulation, ont aussi été considérées lors de l'implémentation de cette architecture.

Dans le chapitre suivant, nous allons présenter les solutions concernant d'autres difficultés issues de la simulation numérique, notamment le temps de calcul onéreux et les erreurs dues aux petits intervalles de discrétisation dans l'espace.

Chapitre III

Optimisation Liée à la Simulation Numérique

Chapitre III

Optimisation Liée à la Simulation Numérique

III.1 - Introduction

Dans le cadre de méthodes de simulation numérique utilisées dans l'analyse de dispositifs électromagnétiques, la Méthode des Éléments Finis (MEF) [Coulomb 1981] se présente aujourd'hui comme une des méthodes les plus robustes et les plus généralistes que nous ayons. La simulation par éléments finis nous permet d'obtenir des résultats assez fiables, même en situation où le dispositif analysé présente une configuration complexe.

Cependant, le temps de calcul demandé par la MEF pour atteindre une solution peut devenir assez important, surtout lorsque l'analyse du dispositif est réalisée en trois dimensions.

Considérant que la résolution d'un problème d'optimisation peut demander quelques centaines, à voir quelques milliers d'évaluations de la fonction objectif et que dans le cas de problèmes liés à la MEF, chaque évaluation représente une résolution par élément finis, nous pourrions arriver à des situations où le temps de calcul serait égal à plusieurs jours de simulation.

D'ailleurs, la difficulté voire l'impossibilité d'évaluer le gradient de la fonction objectif directement à partir d'une simulation par éléments finis peut limiter l'utilisation des méthodes d'optimisation déterministes les plus performantes dans la résolution de problèmes d'optimisation liés à la simulation numérique.

Enfin, les méthodes d'optimisation classiques ont la caractéristique d'avancer à petits pas lorsqu'elles se rapprochent de la solution du problème, ce qui peut entraîner des "bruits numériques" de remmaillage, issus des petits intervalles de variation de la géométrie du dispositif analysé par éléments finis [Weeber 1992].

Pour éviter ces inconvénients posés par la simulation numérique, nous allons présenter dans ce chapitre une approche d'optimisation dans laquelle la fonction objectif est remplacée par une approximation dénommée *Surface de Réponse*, dont la construction est réalisée à l'aide de la *Méthode des Éléments Diffus* et de la *Méthode des Plans d'Expériences*.

III.2 - Surface de Réponse

Une *Surface de Réponse* est une approximation de la fonction objectif qui apporte une connaissance globale du comportement du problème d'optimisation. Elle est obtenue à partir des résultats calculés pour différentes configurations de paramètres bien choisies, puis par l'application d'un algorithme d'interpolation.

Le grand avantage d'utiliser une surface de réponse pour remplacer la fonction objectif réelle est que le coût de sa construction peut être élevé, mais celui de son évaluation est presque nul, ce qui peut réduire énormément le temps de calcul lors de l'optimisation. La Figure 39 présente la comparaison entre l'approche standard d'optimisation et celle basée sur l'utilisation d'une surface de réponse.

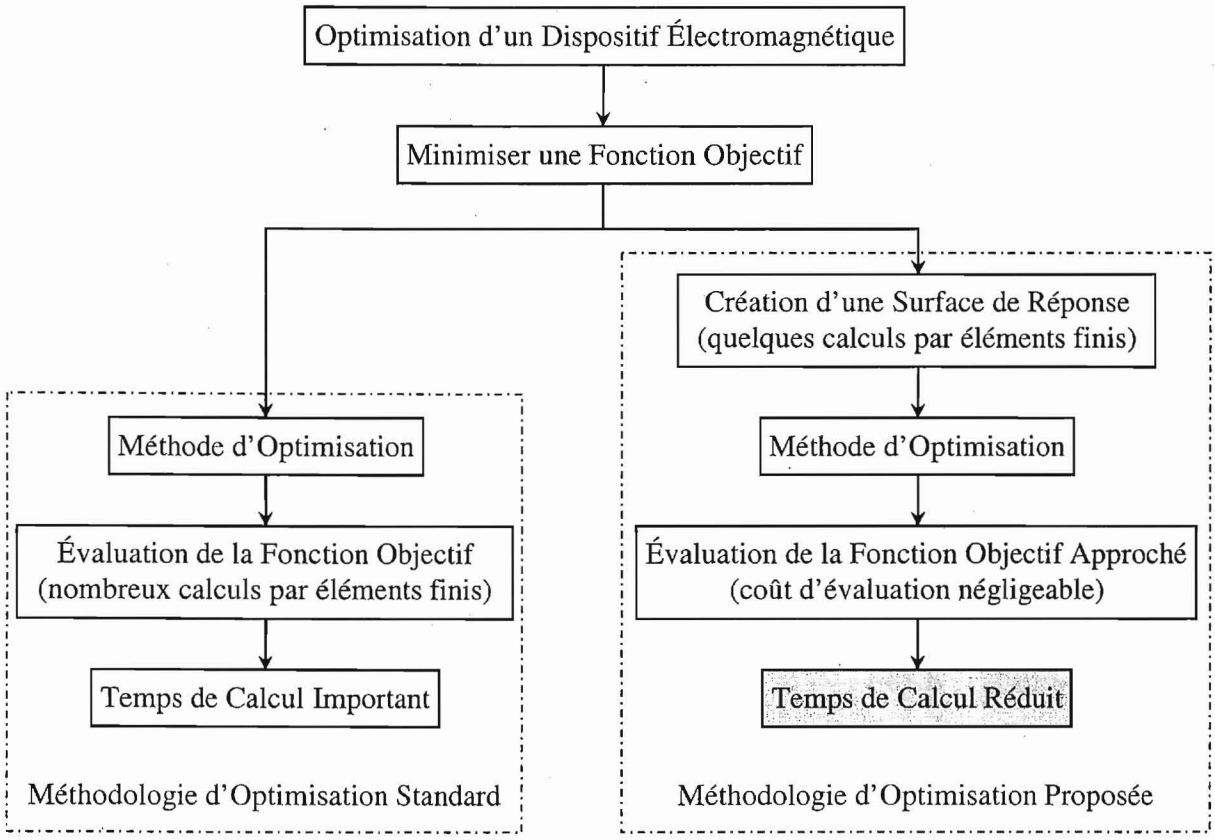


Figure 39 - Optimisation à partir d'une surface de réponse

Plusieurs méthodes d'approximation de fonctions objectifs liées à la simulation numérique ont été proposées ces dernières années, parmi lesquelles nous pouvons citer celles basées sur des *Fonctions à Base Radiale* (*Radial Basis Function*) [Abakar 1998] et sur des *Fonctions à Base Multiquadrique* (*Multiquadric Basis Function*) [Alotto 1996], parfois couplées à des réseaux de neurones.

Malgré la popularité de ces méthodes, elles possèdent quelques points faibles qui peuvent limiter leur utilisation, comme par exemple la difficulté de réglage de certains paramètres liés à la méthode et la nécessité de résoudre des systèmes matriciels pleins pour identifier les coefficients d'approximation.

Récemment, les travaux de Hérault [Hérault 1999] [Hérault 2000] ont montré que l'application de la *Méthode des Éléments Diffus* [Nayroles 1991] peut nous apporter des résultats intéressants dans le contexte d'approximation de fonctions objectifs. En plus, l'approximation par éléments diffus présente plusieurs avantages par rapport à d'autres méthodes d'approximation, parmi lesquelles nous avons le petit nombre de paramètres à

régler, la construction de matrices creuses pour déterminer les coefficients de l'approximation et sa facilité de mise en œuvre.

III.2.1 - Surface de Réponse par la Méthode des Éléments Diffus

La *Méthode des Éléments Diffus* (MED) [Nayroles 1991] [Marih 1994] fait partie d'une famille de méthodes numériques appelée *Méthodes Sans Maillage* (*Meshless Methods*) [Duarte 1995] [Hérault 2000], dont le but est de créer une approximation discrète de grandeurs continues dans un domaine d'étude.

Cette approximation est obtenue à partir d'une discrétisation du domaine, reposant sur un nuage de points appelés *nœuds*, sur lesquels on calcule les valeurs des variables d'état dénommées *valeurs nodales*. Chaque nœud représente le centre d'un "élément" dont la forme est normalement donnée par une "boule" de rayon *r* représentant sa zone d'influence, comme nous montre la Figure 40.

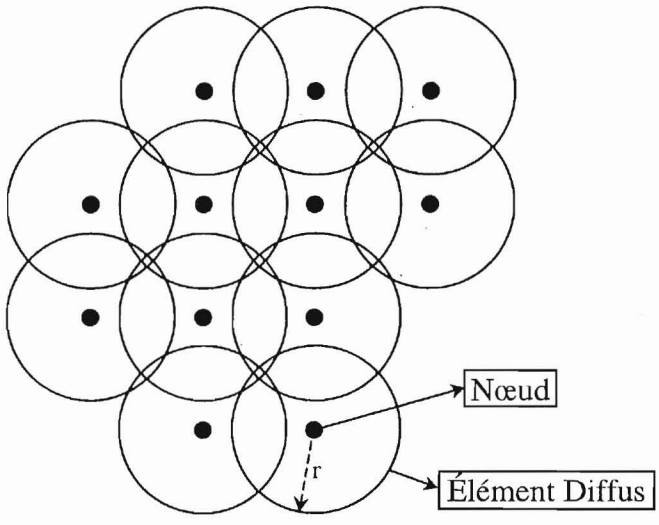


Figure 40 - Représentation des éléments diffus

À la différence de la Méthode des Éléments Finis, les nœuds des Méthodes Sans Maillage ne sont pas connectés entre eux par un maillage, d'où le nom de ces méthodes.

III.2.1.1 - Formulation

- *Calcul des Fonctions de Forme*

Soit $f(x)$ une fonction continue définie sur un domaine d'étude Ω et n le nombre de nœuds $x_{i\ (i\ =\ 1,\ldots,n)}$ sur lesquels nous connaissons la valeur de $f(x_i)$. Nous cherchons à obtenir une approximation $\tilde{f}(x)$ de $f(x)$ sur le point d'approximation $\bar{x}$, telle que:

$$f^{\bar{x}}(x) \approx \tilde{f}^{\bar{x}}(x) = \sum_{k=1}^{m} P_k(x-\bar{x})\alpha_k^{\bar{x}} = \left[P(x-\bar{x})\right]^T\left[\alpha^{\bar{x}}\right], \tag{34}$$

où

P est une base polynomiale avec m termes sur laquelle l'approximation sera projetée

α_k sont les variables généralisées de l'approximation.

Les coefficients α_k sont calculés par la minimisation de l'erreur quadratique pondérée commise sur l'approximation, indiquée dans l'équation (35).

$$\min \mathfrak{I}^{\bar{x}}(\alpha^{\bar{x}}) = \sum_{i=1}^{n} W_i(x_i-\bar{x})(f_i-\tilde{f}^{\bar{x}}(x_i))^2\,, \tag{35}$$

où

W_i est une fonction de pondération de support limité dont le centre est le nœud x_i

f_i est la valeur nodale associée au nœud x_i.

La minimisation de $\mathfrak{I}(\alpha)$ est obtenue en considérant

$$\frac{\partial \mathfrak{I}^{\bar{x}}(\alpha^{\bar{x}})}{\partial \alpha_j^{\bar{x}}} = 0,\ \forall\ j \in [1,m]\,, \tag{36}$$

ce qui nous donne

$$\frac{\partial \mathfrak{I}^{\bar{x}}(\alpha^{\bar{x}})}{\partial \alpha_j^{\bar{x}}} = -2\sum_{i=1}^{n} W_i(x_i-\bar{x})(f_i-\tilde{f}^{\bar{x}}(x_i))\left(\frac{\partial \tilde{f}^{\bar{x}}(x_i)}{\partial \alpha_j^{\bar{x}}}\right) = 0\,, \tag{37}$$

ou alors

$$-2\sum_{i=1}^{n} W_i(x_i-\bar{x})\left(f_i-\sum_{k=1}^{m} P_k(x_i-\bar{x})\alpha_k^{\bar{x}}\right)P_j(x_i-\bar{x})=0 \tag{38}$$

Après expansion de (38), nous avons

$$\sum_{i=1}^{n} W_i(x_i-\bar{x})P_j(x_i-\bar{x})\left(\sum_{k=1}^{m} P_k(x_i-\bar{x})\alpha_k^{\bar{x}}\right)=\sum_{i=1}^{n} W_i(x_i-\bar{x})P_j(x_i-\bar{x})f_i\,, \tag{39}$$

ce qui nous permet d'écrire

$$\left[\alpha^{\bar{x}}\right]=\left[A^{\bar{x}}\right]^{-1}\sum_{i=1}^{n}\left[B^{\bar{x}}\right]f_i\,, \tag{40}$$

avec

$$\begin{cases}\left[A^{\bar{x}}\right]=\sum_{j=1}^{m}\sum_{k=1}^{m}\sum_{i=1}^{n} W_i(x_i-\bar{x})P_j(x_i-\bar{x})P_k(x_i-\bar{x})\\ \left[B^{\bar{x}}\right]=\sum_{j=1}^{m} W_i(x_i-\bar{x})P_j(x_i-\bar{x})\end{cases} \tag{41}$$

L'approximation donnée par l'équation (34) devient alors égale à

$$\begin{cases}\tilde{f}^{\bar{x}}(x)=\sum_{i=1}^{n}\phi_i^{\bar{x}}(x)f_i\\ \phi_i^{\bar{x}}(x)=\left[P(x-\bar{x})\right]^T\left[A^{\bar{x}}\right]^{-1}\left[B^{\bar{x}}\right]\end{cases}, \tag{42}$$

où

ϕ_i est la *fonction de forme* associée au nœud *i*.

- *Identification des Valeurs Nodales*

Pour déterminer les valeurs nodales f_i de l'équation (42), il faut disposer d'un nombre de nœuds au moins égal au nombre de coefficients α_k. Nous avons deux différentes manières de

calculer les valeurs nodales: *l'identification par collocation* et *l'identification par moindres carrés*.

- Identification par Collocation

L'identification par collocation consiste tout simplement à poser pour chaque nœud x_i existant:

$$\tilde{f}(x_i) = \sum_{i=1}^{n} \phi_i(x_i) f_i = f(x_i), \tag{43}$$

L'équation (43) nous permet de créer un système matriciel carré du type $A \cdot X = B$ de dimension $n \times n$ et des coefficients a_{ij} et b_i donnés par (44), dont la solution représente les valeurs de f_i.

$$\begin{cases} a_{ij} = \phi_i(x_i) \\ b_i = f(x_i) \end{cases} \tag{44}$$

Le nombre de coefficients a_{ij} non nuls d'une ligne i quelconque est égal au nombre de nœuds que l'élément i englobe. Ceci nous permet d'avoir une matrice creuse, lorsque ce nombre est restreint. Cependant, comme aij ≠ aji, cette matrice n'est pas symétrique, ce qui peut poser des problèmes liés à la performance de la méthode lors de sa résolution.

- Identification par Moindres Carrés

Dans l'identification par moindres carrés, nous créons un système matriciel à partir de la minimisation de la fonctionnelle donnée par (45).

$$\Im = \left(\sum_{i=1}^{n} \phi_i(x_i) f_i - f(x_i) \right)^2 \tag{45}$$

Ce système matriciel est toujours creux et carré de dimension $n \times n$, avec des coefficients a_{ij} et b_i calculés par (46), dont la solution nous donne les valeurs des coefficients f_i.

$$\begin{cases} a_{ij} = \phi_i(x_i)\phi_j(x_i) \\ b_i = \phi_i(x_i)f(x_i) \end{cases} \tag{46}$$

À la différence de l'identification par collocation, l'identification par moindres carrés nous donne une matrice symétrique, ce qui permet l'utilisation de méthodes de résolution de système plus performantes, comme par exemple l'ICCG [Ciarlet 1982]. De cette façon, nous considérons les moindres carrés comme étant la meilleure approche pour identifier les valeurs nodales.

- *Calcul du Gradient*

Lorsque nous avons une base polynomiale P d'ordre supérieur à 0, nous pouvons calculer le gradient de la fonction d'approximation $\tilde{f}(x)$ donnée par l'équation (42) à partir du gradient des fonctions de forme ϕ_i représenté dans l'équation (47). Cela nous permet d'obtenir une approximation du gradient de la fonction objectif et par conséquent d'utiliser des méthodes déterministes dans la résolution d'un problème d'optimisation lié à la simulation par éléments finis.

$$\begin{cases} \nabla \tilde{f}^{\bar{x}}(x) = \sum_{i=1}^{n} \nabla \phi_i^{\bar{x}}(x) f_i \\ \nabla \phi_i^{\bar{x}}(x) = \left[\nabla P(x - \bar{x})\right]^T \left[A^{\bar{x}}\right]^{-1} \left[B^{\bar{x}}\right] \end{cases} \tag{47}$$

- *Zone d'Influence des Nœuds*

La *zone d'influence* d'un nœud i correspond à la région du domaine où le nœud intervient dans l'approximation de la fonction. Cette zone d'influence est définie par une fonction de pondération W_i associée au nœud i, possédant un support limité de façon à donner un caractère local à l'approximation.

Parmi les différentes possibilités que nous avons pour définir la fonction de pondération associée au nœud [Hérault 2000], nous avons choisi celle donnée par l'équation (48), dont l'allure est représentée dans la Figure 41.

$$W_i(x) = \begin{cases} \left(1 - \left(\dfrac{\|x - x_i\|_2}{r_i}\right)^2\right)^2 & \text{pour } \|x - x_i\|_2 \le r_i \\ 0 & \text{sinon} \end{cases}, \tag{48}$$

où

x_i sont les coordonnées du nœud i

r_i est l'envergure de la zone d'influence du nœud i

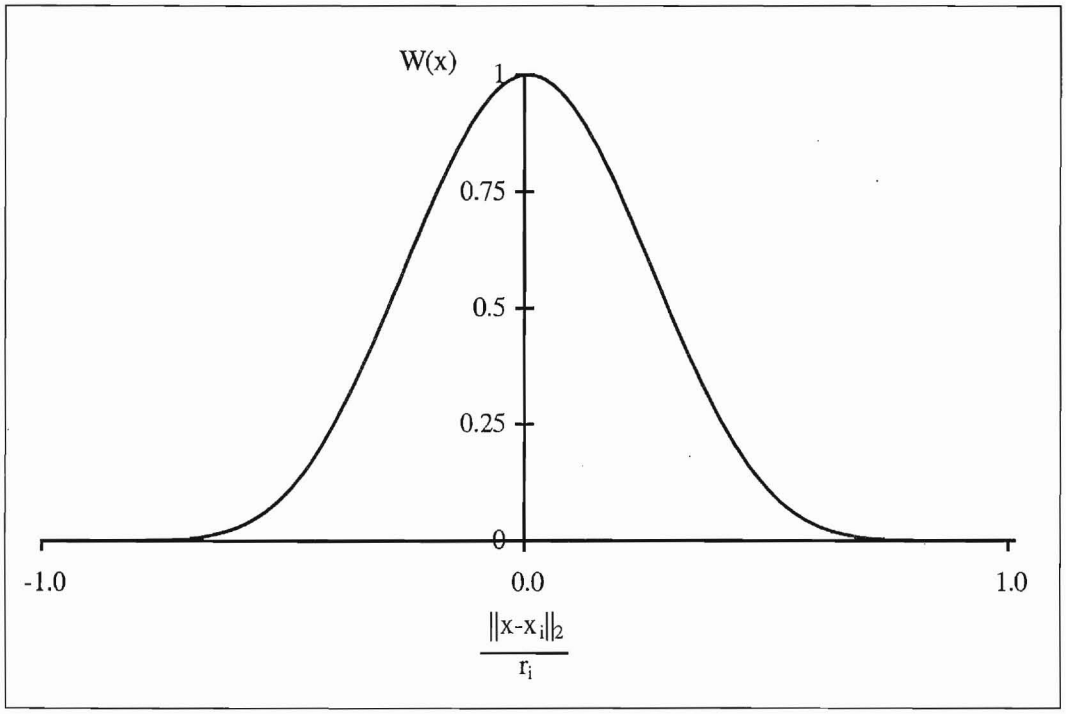


Figure 41 - Allure de la fonction de pondération

Nous pouvons vérifier que l'envergure de la fonction de pondération est réellement limitée, ce qui donne l'aspect local à l'approximation. En ce qui concerne la forme de l'élément associé à cette fonction, elle peut être représentée, dans le cas d'une fonction à deux paramètres, par une "boule" de rayon r_i dont le centre est le nœud i - Figure 40.

III.2.1.2 - Processus Standard d'Approximation

Soit $f(x)$ une fonction objectif à n paramètres $x_{k\ (k\ =\ 1,\ \ldots,\ n)}$ qui doit être approchée par la Méthode des Éléments Diffus. Dans un processus standard d'approximation par cette méthode, nous déroulons les étapes présentées dans la Figure 42.

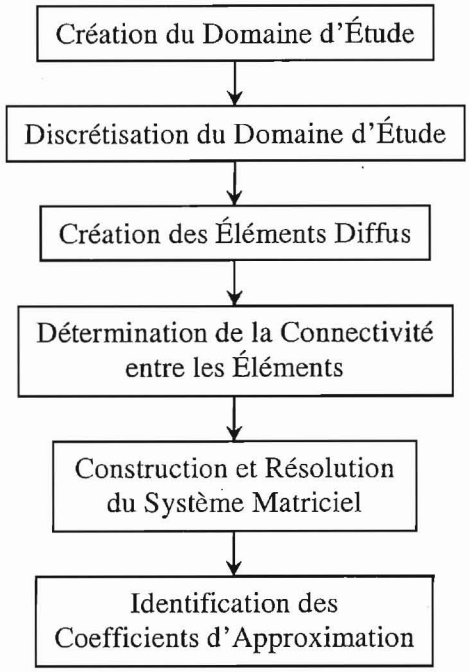


Figure 42 - Étapes du processus standard d'approximation par la MED

- *Création du Domaine Discrétisé*

Dans cette première étape du processus d'approximation, on crée un domaine représenté par un hyper cube de dimension égale au nombre de paramètres n de la fonction à approcher et limité par les bornes de variations de chaque paramètre. Ensuite, on effectue une transformation de variables par rapport à *0.0* et *1.0*, de façon à obtenir un domaine d'étude normalisé, comme l'indique la Figure 43 où nous avons la représentation d'un domaine créé pour l'approximation d'une fonction à deux paramètres $f(x_1,x_2)$.

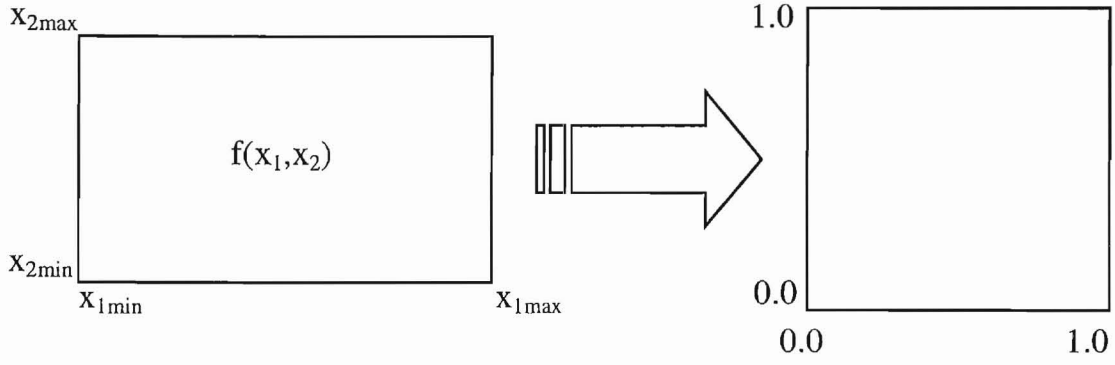


Figure 43 - Normalisation du domaine d'approximation de la fonction

Après la construction du domaine, nous effectuons sa discrétisation en prenant des intervalles réguliers définis par le nombre de points d'approximation souhaités pour chacune de ses directions. Les intersections entre ces intervalles iront alors représenter les points sur lesquels les nœuds utilisés dans l'approximation seront créés, comme nous montre la Figure 44.

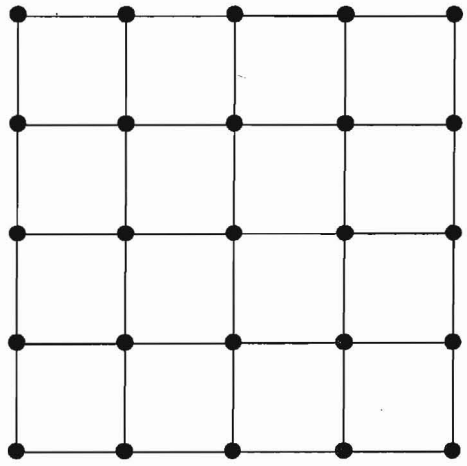


Figure 44 - Domaine discrétisé

Le fait d'avoir des nœuds suffisamment éloignés dans le domaine de discrétisation évite l'évaluation de la fonction réelle pour des configurations de paramètres qui soient trop proches entre elles et par conséquent la présence des "bruits numériques" de remaillage, liés aux petits intervalles de variation de la géométrie du dispositif lors d'une analyse par éléments finis [Weeber 1992].

- *Création des Éléments Diffus*

Une fois que les nœuds de l'approximation ont été définis, nous effectuons la mise en place des éléments associés à eux en choisissant les valeurs de l'envergure de la zone d'influence r_i de chaque nœud pour lesquelles le recouvrement du domaine d'étude est garanti. Les détails concernant les valeurs des r_i, ainsi que l'ordre des éléments seront présentés dans la section III.2.1.3.

La Figure 45 montre une configuration d'éléments dans un domaine d'étude discrétisé en 5 points par direction.

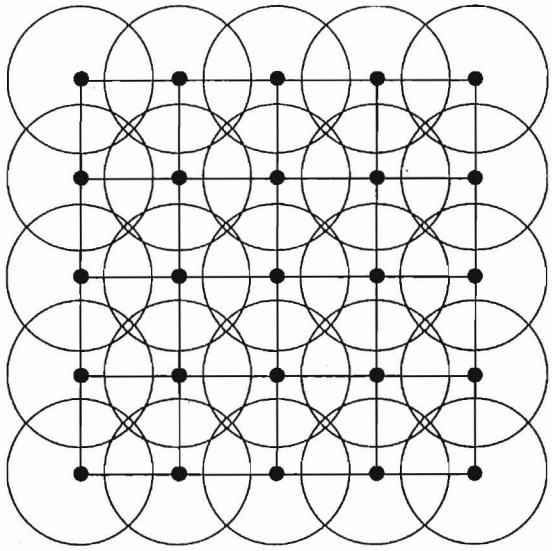


Figure 45 - Éléments utilisés pour obtenir la surface de réponse

- *Détermination des Coefficients d'Approximation*

La dernière étape du processus est la détermination des coefficients d'approximation f_i. Pour cela, il faut tout d'abord évaluer la fonction objectif sur les nœuds obtenus dans l'étape de discrétisation. Il faut aussi déterminer, en considérant les valeurs des rayons r_i, la connectivité existante entre chaque élément du domaine et ses voisins, ce qui nous permettra de calculer, à partir de l'équation (42), les valeurs de fonction de forme ϕ_i.

Avec les valeurs des fonctions de forme ϕ_i et des évaluations de la fonction objectif, il devient alors possible de calculer les coefficients a_{ij} et b_i de l'équation (46) qui seront utilisés pour construire le système matriciel dont la solution nous fournira les valeurs nodales f_i utilisées pour approcher la fonction.

- *Application sur une Fonction à Deux Paramètres*

Pour illustrer le résultat de l'approximation obtenue à partir des éléments diffus, nous avons appliqué cette méthode sur la fonction polynomiale à deux paramètres donnée par l'équation (49) dont la représentation se trouve dans la Figure 46.

$$\begin{cases} f(x,y) = 0{,}01 \cdot ((x+0{,}5)^4 - 30x^2 - 20x + (y+0{,}5)^4 - 30y^2 - 20y) \\ -7{,}0 < x < 6{,}0 \\ -7{,}0 < y < 6{,}0 \end{cases} \tag{49}$$

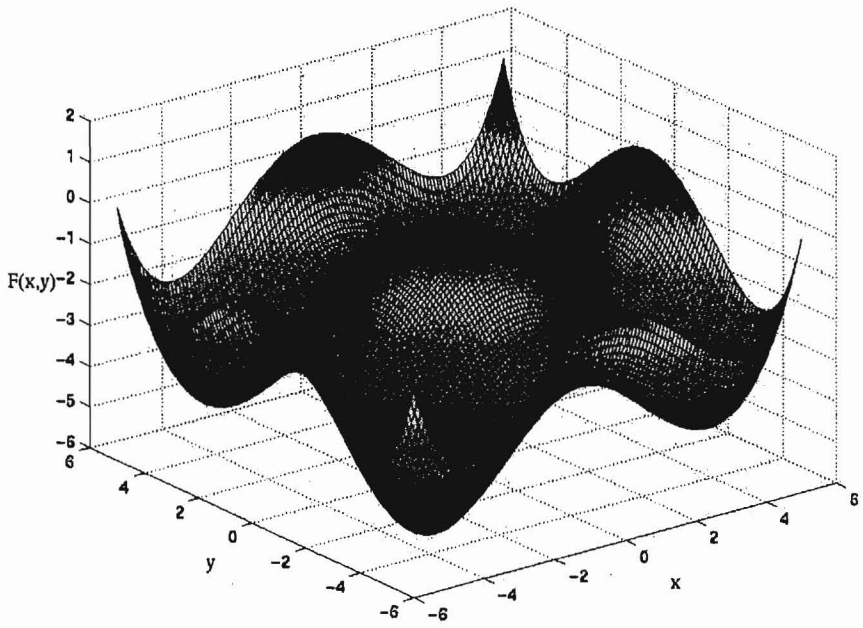


Figure 46 - Fonction polynomiale à 2 paramètres

Nous avons utilisé une approximation constituée par des éléments d'ordre 2 avec un rayon égale à 3 fois le pas inter nodal, distribués sur une grille régulière de 5 points par direction, totalisant 25 éléments. Le résultat obtenu est présenté dans la Figure 47.

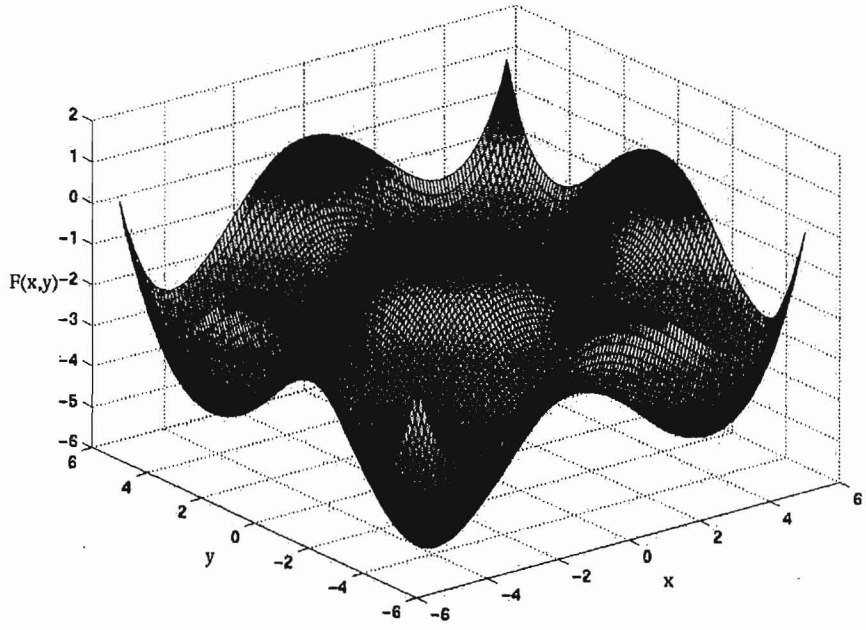


Figure 47 - Surface de réponse de la fonction polynomiale à 2 paramètres

À priori, les deux fonctions semblent proches, en fait, la qualité de l'approximation dépend de quelques facteurs [Hérault 2000] [Costa 2001c] que nous allons étudier maintenant.

III.2.1.3 - Influence des Facteurs Associés à la MED

La qualité de l'approximation obtenue à partir de l'application de la Méthode des Éléments Diffus dépend de différents facteurs, parmi lesquels nous avons l'ordre de l'approximation, le rayon des éléments et surtout le nombre d'éléments utilisés.

Nous avons fait une analyse de l'influence de chacun de ces facteurs en utilisant la fonction polynomiale *f(x)* de l'équation (50), dont l'allure est présentée dans la Figure 48.

$$\begin{cases} f(x) = 0{,}01 \cdot ((x+0{,}5)^4 - 30x^2 - 20x) \\ -7{,}0 < x < 6{,}0 \end{cases} \tag{50}$$

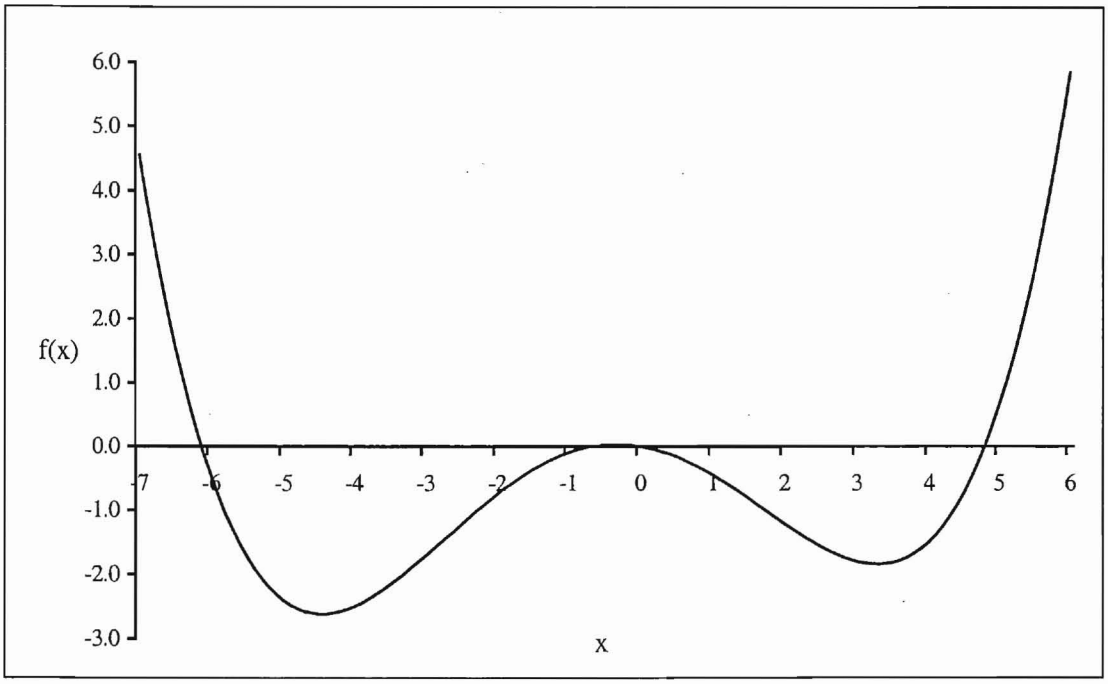


Figure 48 - Allure de la fonction polynomiale f(x)

- *Influence de l'Ordre de l'Approximation*

L'ordre de l'approximation est défini par la base polynomiale *P* utilisée dans le calcul des fonctions de forme associés aux éléments - équation (42). Le Tableau IV présente trois différents ordres d'approximation et leur base polynomiale respective, ainsi que le nombre minimal de nœuds qu'un élément correspondant à chacun de ces ordres doit englober.

TABLEAU IV - DIFFÉRENTS ORDRES D'APPROXIMATION

Ordre	**Base Polynomiale P(x)**	**Nombre Minimal de Nœuds (n = dimension du domaine)**
0	1	1
1	$1 + x$	$n + 1$
2	$1 + x + x^2$	$\frac{(n+1)\cdot(n+2)}{2}$

La Figure 49 présente l'approximation obtenue pour la fonction de l'équation (50) en utilisant 7 éléments d'ordre 0, 1 et 2.

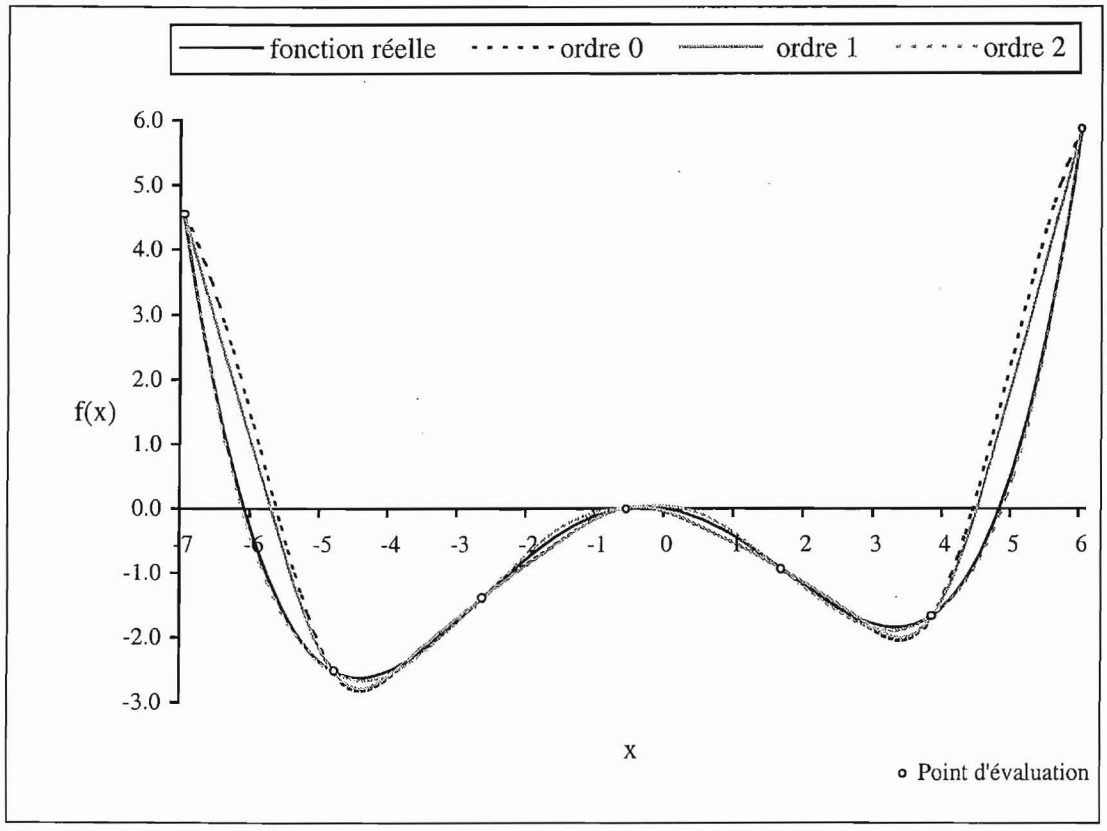


Figure 49 - Comparaison entre différents ordres d'approximation

Nous pouvons vérifier que l'approximation d'ordre 2 nous donne un meilleur résultat par rapport aux autres ordres. Cependant, le nombre minimal de nœuds que les éléments doivent contenir peut devenir assez important lorsque le nombre de paramètres de la fonction augmente.

Pour garantir ce nombre minimal de nœuds englobés, le rayon des éléments doit être suffisamment grand, ce qui peut compromettre l'obtention d'un système matriciel creux, ainsi que le caractère local de l'approximation.

L'approximation d'ordre 0 ne présente pas le même problème que celle d'ordre 2. Par contre, la qualité de l'approximation peut devenir assez faible lorsque la fonction présente plusieurs oscillations. En plus, l'utilisation d'éléments d'ordre 0 nous donne une fonction approchée non différentiable, ce qui empêche son utilisation dans le cas où nous voudrions appliquer une méthode d'optimisation déterministe indirecte - I.3.1.2.

L'approximation en utilisant des éléments d'ordre 1 se présente alors comme la meilleure alternative pour approcher une fonction, car elle nous permet de travailler avec des

systèmes matriciels creux et d'avoir une approximation qui soit meilleure que celle obtenue par des éléments d'ordre 0 et aussi différentiable.

- *Influence du Rayon des Éléments*

La valeur du rayon *r* des éléments doit, avant tout, garantir le recouvrement du domaine d'étude. Dans le cas d'éléments créés dans une grille régulière dont le nombre de nœuds est le même en toutes les directions du domaine, la valeur minimale de *r* est calculée à partir de:

$$r_{min} = \frac{d \cdot \sqrt{n}}{2}, \tag{51}$$

où

n est la dimension du domaine d'étude

d est le pas inter nodal illustré dans la Figure 50.

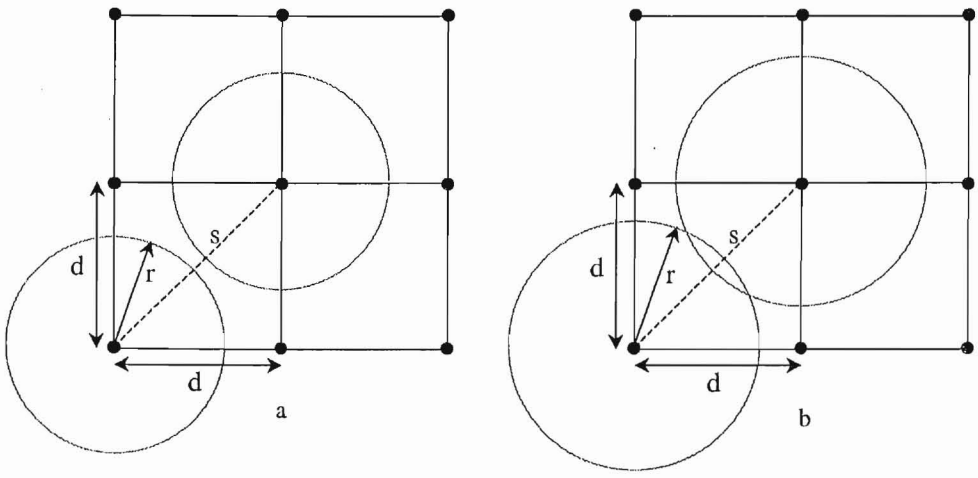


Figure 50 - Rayon minimum de recouvrement

Le rayon *r* joue un rôle important dans le résultat de l'approximation et d'après [Hérault 2000], la valeur pour laquelle nous avons une meilleure approximation est égale à:

$$1{,}51 \cdot d < r_{opt} < 1{,}90 \cdot d, \tag{52}$$

Nous avons utilisé 7 éléments de premier ordre avec 3 différentes valeurs de *r* pour approcher la fonction de l'équation (50), dont les résultats sont présentés sur la Figure 51 et

qui nous permettent de vérifier que la valeur optimale de r est effectivement comprise dans l'intervalle donné par l'équation (52).

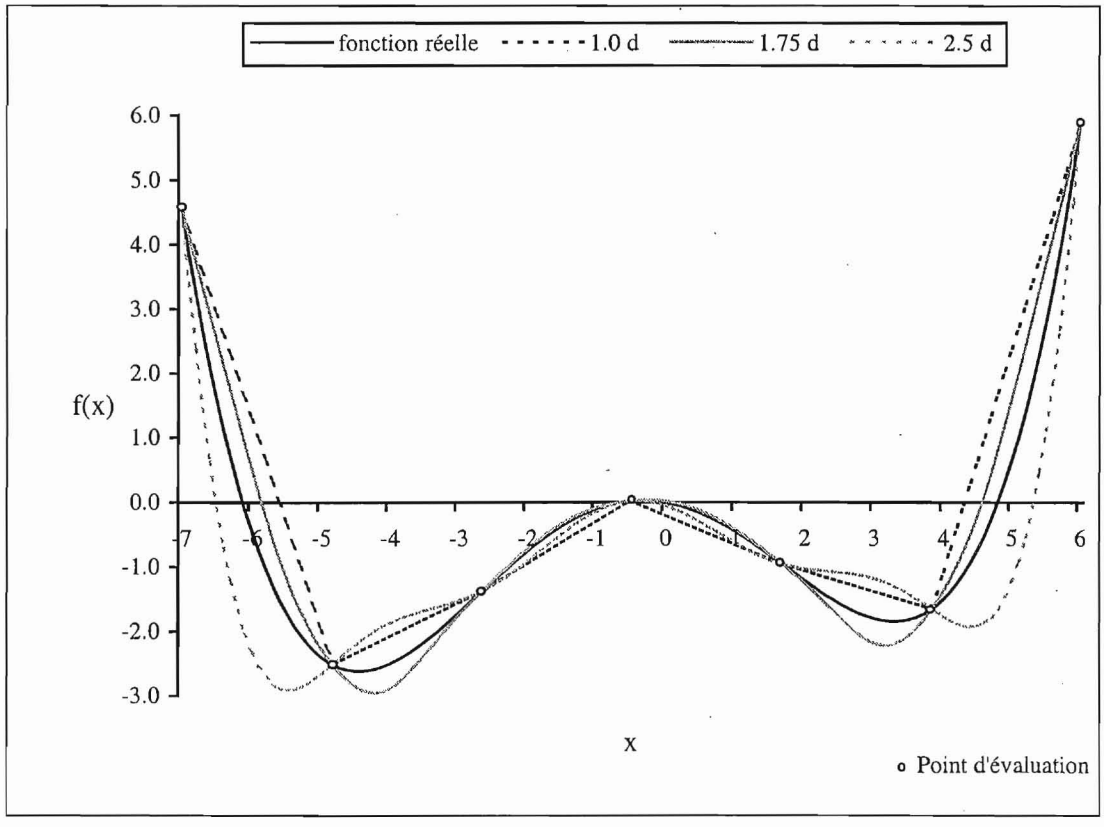


Figure 51 - Comparaison entre différents rayons

- *Influence du Nombre de Nœuds*

Le nombre de nœuds utilisés dans l'approximation est sans aucun doute le facteur le plus influent dans la qualité de l'approximation, comme nous pouvons le vérifier à la Figure 52.

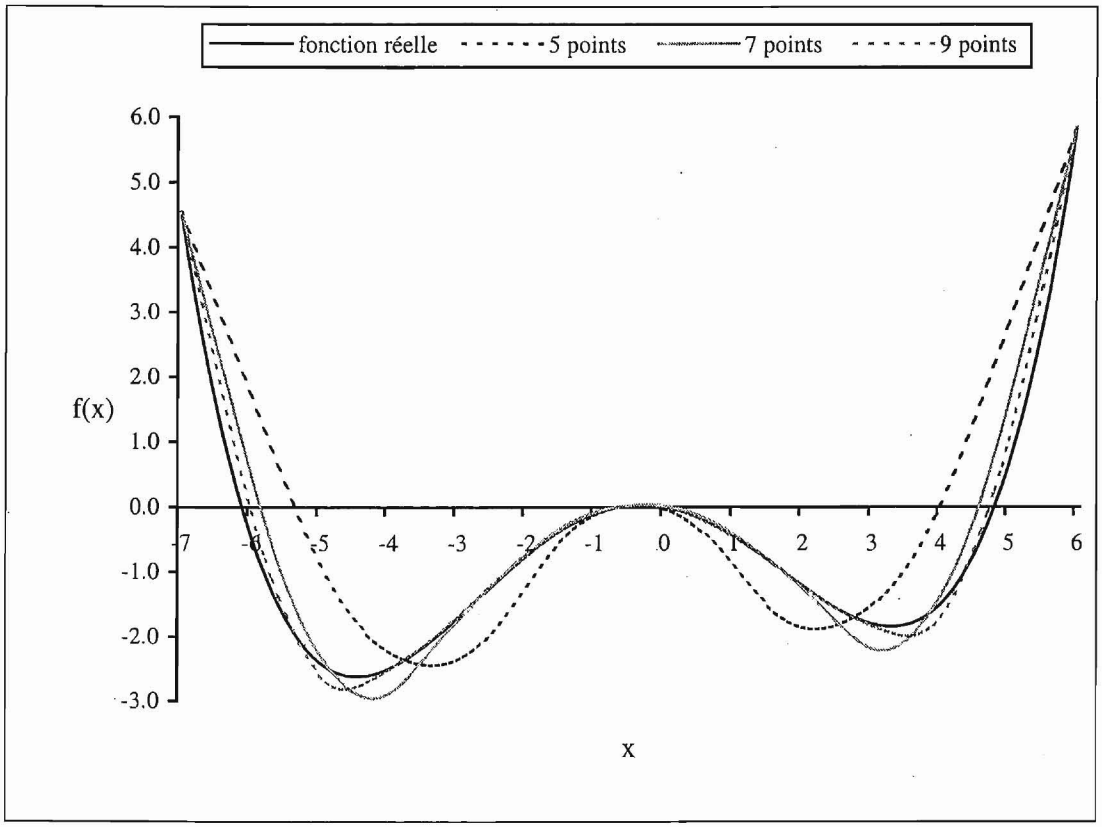


Figure 52 - Comparaison entre différents discrétisations

L'utilisation d'une discrétisation importante du domaine d'étude nous emmène à une approximation de meilleure qualité, car nous avons un plus grand nombre de nœuds utilisés pour approcher la fonction. Pourtant, lorsque le nombre de paramètres de la fonction augmente, le nombre de nœuds atteint rapidement les milliers, comme nous montre le Tableau V.

TABLEAU V - NOMBRE D'ÉVALUATIONS PAR RAPPORT À LA DISCRÉTISATION DU DOMAINE

Points par Direction	**Nombre de Paramètres**			
	2	**3**	**4**	**5**
3	9	27	81	243
4	16	64	256	1.024
5	25	125	625	3.125
6	36	216	1.296	7.776
7	49	343	2.401	16.807
8	64	512	4.096	32.768
9	81	729	6.561	59.049

Cette situation peut compromettre l'intérêt de l'approche d'approximation, car le temps de calcul nécessaire pour atteindre une solution peut dépasser celui demandé par l'approche d'optimisation standard. Pour ne pas arriver à cette situation, nous pouvons limiter le nombre de nœuds créés pendant l'étape de discrétisation et en même temps assurer la qualité de l'approximation en utilisant un algorithme adaptatif qui demande l'ajout de nouveaux points seulement dans certaines régions du domaine. Cet algorithme nous fournira alors ce que nous appelons d'une *Surface de Réponse Adaptative.*

III.2.2 - Surface de Réponse Adaptative

La surface de réponse adaptative nous permet d'avoir une approximation de la fonction à partir d'un nombre réduit de nœuds. Nous avons développé deux différentes approches d'adaptativité: *l'adaptativité par rapport à l'optimum* et *l'adaptativité par rapport à la qualité de l'approximation.*

III.2.2.1 - Adaptativité par Rapport à la Qualité de l'Approximation

Le but de l'algorithme que nous avons développé pour créer une *surface de réponse adaptative par rapport à la qualité de l'approximation* est de comparer deux différentes approximations d'une même fonction et d'identifier les régions dans lesquelles il nous faut une plus grande concentration de nœuds de façon à nous donner une meilleure représentation de la fonction objectif.

- *Étapes du Processus Adaptatif*

Ainsi comme le processus standard d'approximation, le processus adaptatif commence à partir de la définition d'un domaine d'étude de dimension égale au nombre de paramètres de la fonction à approcher. Ensuite, nous effectuons la discrétisation de ce domaine en n'utilisant que 3 points par direction.

Dans le cas d'une fonction à deux paramètres, cette discrétisation initiale génère 9 nœuds et 4 différentes sous-régions *I*, *II*, *III* et *IV*. Chacune de ces régions représente une structure de données dénommée *quadtree* [Marih 1994] dont le "père" est un autre *quadtree* décrit par les points *ABCD*, comme nous montre la Figure 53.

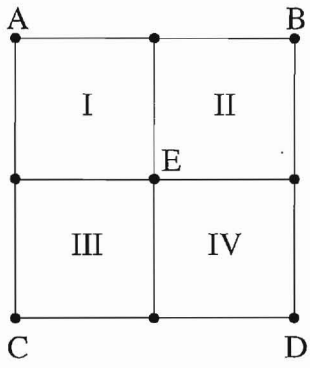


Figure 53 - Première itération de la surface de réponse adaptative

Une fois créés les éléments et identifiées les valeurs nodales de chacun des nœuds de cette configuration initiale, nous identifions les quadtrees dans lesquels la surface de réponse donnée par cette configuration d'éléments présente un écart important par rapport à la vraie fonction objectif.

Pour cela, nous vérifions l'influence du nœud central *E* du quadtree père sur la surface de réponse existante. Cette vérification est faite à partir de la comparaison entre la surface de réponse donnée par la Figure 54-a avec celle donnée par la Figure 54-b sur la région correspondante au quadtree père, en calculant l'erreur d'approximation donnée par:

$$\varepsilon_{\text{sous-région}} = \frac{\dfrac{\sum_{i=1}^{N} |f_a(p_i) - f_b(p_i)|}{N}}{|M|}, \tag{53}$$

où

f_a est l'approximation donnée par la configuration *a*

f_b est l'approximation donnée par la configuration *b*

p_i est le point sur lequel nous évaluons la fonction

N est le nombre de points d'évaluation utilisés pour comparer les deux approximations

M est la moyenne de la valeur de la fonction sur les 9 points initiaux.

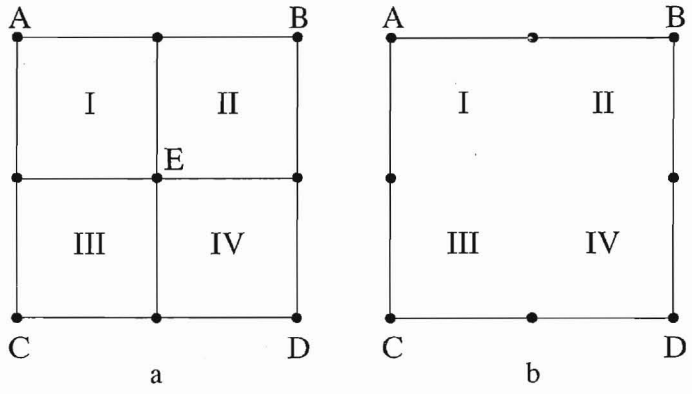


Figure 54 - Comparaison entre les deux surfaces de réponses de la première itération

L'approximation sur les sous-régions *I*, *II*, *III* et *IV* est considérée satisfaisante si l'erreur calculée par (53) est inférieure à une certaine valeur (5%, par exemple). Autrement, nous ajoutons un nouveau point au centre de chaque sous-région, ce qui nous permet d'obtenir une nouvelle configuration représentée sur la Figure 55.

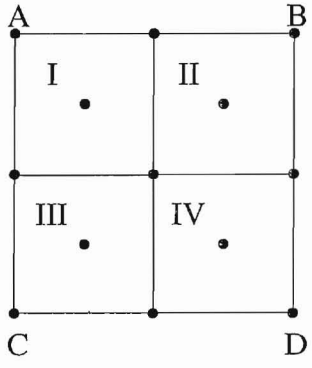


Figure 55 - Deuxième itération de la surface de réponse adaptative

Le but maintenant est de vérifier l'influence des nœuds qui ont été ajoutés sur la valeur de l'approximation, ce qui nous permettra d'identifier les régions dans lesquelles une nouvelle discrétisation est encore nécessaire. Nous procédons alors d'une façon similaire à celle de l'étape précédente, en comparant les approximations données par la Figure 56-a et la Figure 56-b en chaque sous-région du domaine.

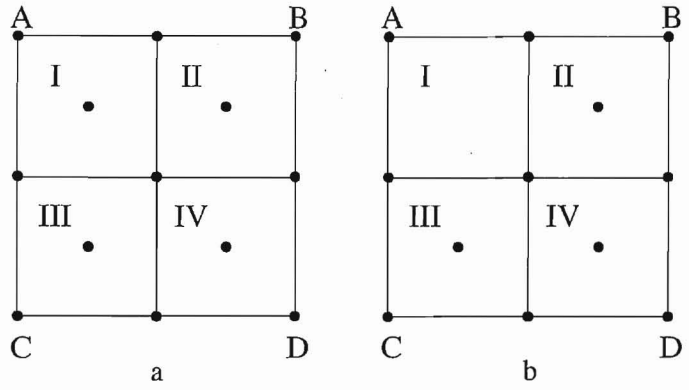


Figure 56 - Comparaison entre les deux surfaces de réponses de la deuxième itération

Les sous-régions dans lesquelles l'erreur calculée par (53) est supérieure à 5% seront subdivisées en nouveaux quadtrees, tandis que les autres ne seront pas modifiées, comme illustre la Figure 57-a.

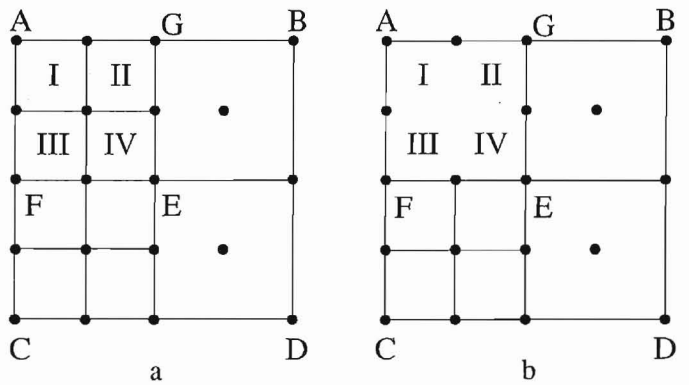


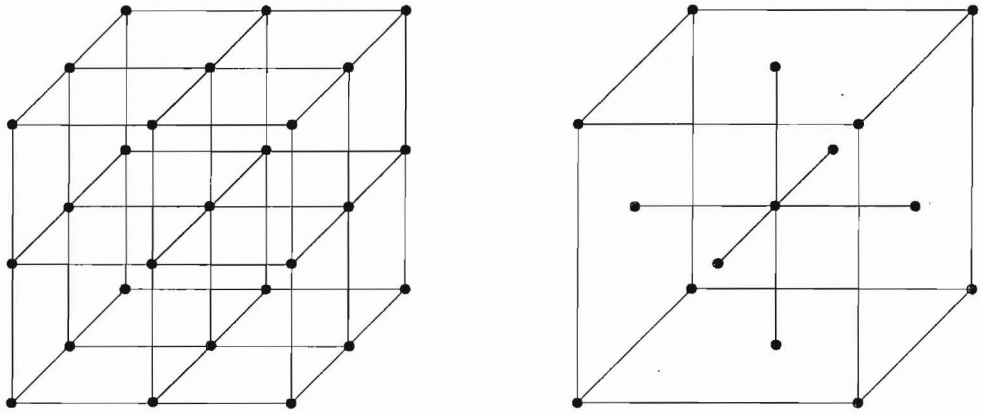
Figure 57 - Comparaison entre les deux surfaces de réponses de la troisième itération

Nous pouvons remarquer que les configurations de quadtrees de la Figure 57-a et de la Figure 57-b sont similaires à celles de la Figure 54-a et de la Figure 54-b, à la différence que nous avons un niveau de quadtrees en plus. De cette façon, l'algorithme retombe dans sa première étape et il se déroulera jusqu'au moment où nous n'aurons plus de région à subdiviser.

- *Généralisation à Trois Paramètres ou Plus*

L'algorithme adaptatif que nous venons de décrire peut être utilisé pour construire la surface de réponse d'une fonction à trois paramètres ou plus en faisant tout simplement la subdivision complète des "n-trees" (quadtrees de *n* dimensions) en toutes leurs directions, comme illustre la Figure 58-a.

Malgré le fait du nombre important de nœuds que cette approche peut entraîner lorsque le nombre de paramètres augmente, nous l'avons utilisé dans les applications qui seront présentées dans le chapitre suivant.



a b

Figure 58 - Subdivision complète et incomplète d'un n-tree

Pour réduire le nombre des nœuds ajoutés à chaque itération de l'algorithme, nous pouvons effectuer une subdivision incomplète des n-trees, dans laquelle nous n'ajoutons des nœuds qu'au centre de chaque facette du n-tree (Figure 58-b). Cette approche, connue par le nom de composite centré [Schimmerling 1998], n'a pas été testée et ne sera donc pas présentée dans ce travail.

- *Applications*

Nous avons choisi deux différentes applications pour montrer les avantages de la surface de réponse adaptative: une fonction polynomiale et une fonction sinusoïdale.

- Fonction Polynomiale

Dans la première application, nous avons utilisé l'algorithme d'adaptativité sur la fonction donnée par l'équation (49). La surface de réponse obtenue à la fin des 4 itérations de la méthode adaptative est celle présentée sur la Figure 59.

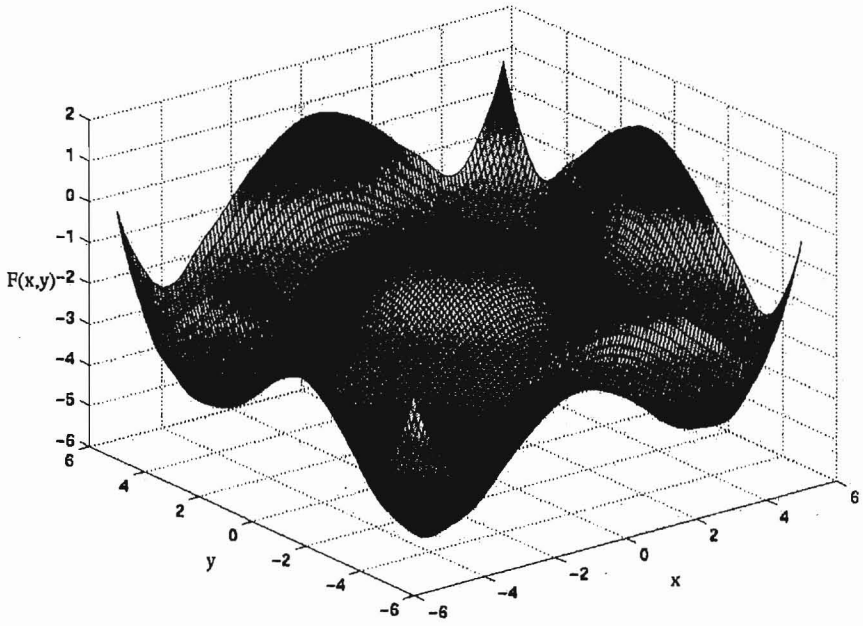


Figure 59 - Surface de réponse adaptative de la fonction polynomiale à 2 paramètres

La Figure 60 présente la configuration finale de nœuds, où nous pouvons remarquer une concentration d'éléments sur quelques régions spécifiques du domaine, notamment celles où la fonction présente une plus grande variation. Le nombre total de nœuds utilisés est égal à 113 et l'erreur commise par l'approximation vis à vis de la fonction réelle, calculée à partir de l'équation (54), est de 8,9%.

$$\varepsilon = \sqrt{\frac{\sum_{i=1}^{N_p} (f(p_i) - f_{app}(p_i))^2}{(f(p_i))^2}}, \tag{54}$$

où

f est la fonction réelle

f_{app} est la fonction approchée obtenue à la fin de l'algorithme

p_i est le point sur lequel nous évaluons la fonction

N_p (=100 x 100) est le nombre de points d'évaluation utilisés pour calculer l'erreur.

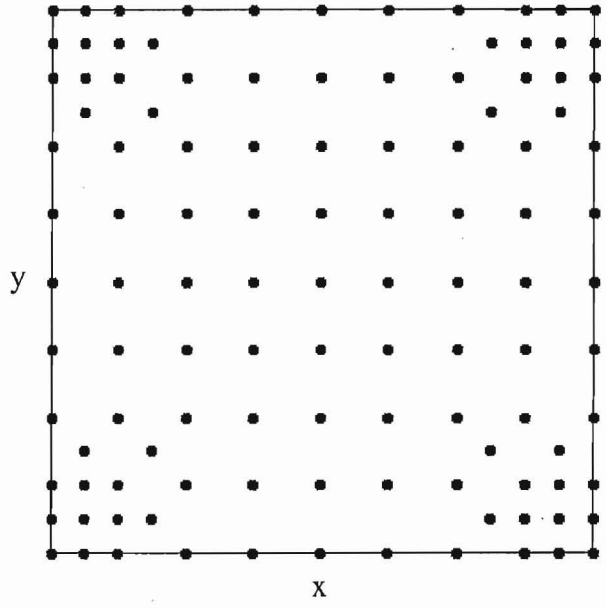


Figure 60 - Configuration finale de nœuds

- Fonction Sinusoïdale

Nous avons aussi appliqué l'algorithme adaptatif sur une fonction sinusoïdale à deux paramètres, donnée par l'équation (55). Nous pouvons remarquer dans la Figure 61 que cette fonction est bien plate dans la direction x sur une région du domaine et en même temps elle présente une forte oscillation sur d'autres régions.

$$\begin{cases} f(x,y) = x \cdot \sin(x^2) + y^2 \\ -3{,}0 < x < 0{,}0 \\ -2{,}0 < y < 2{,}0 \end{cases} \tag{55}$$

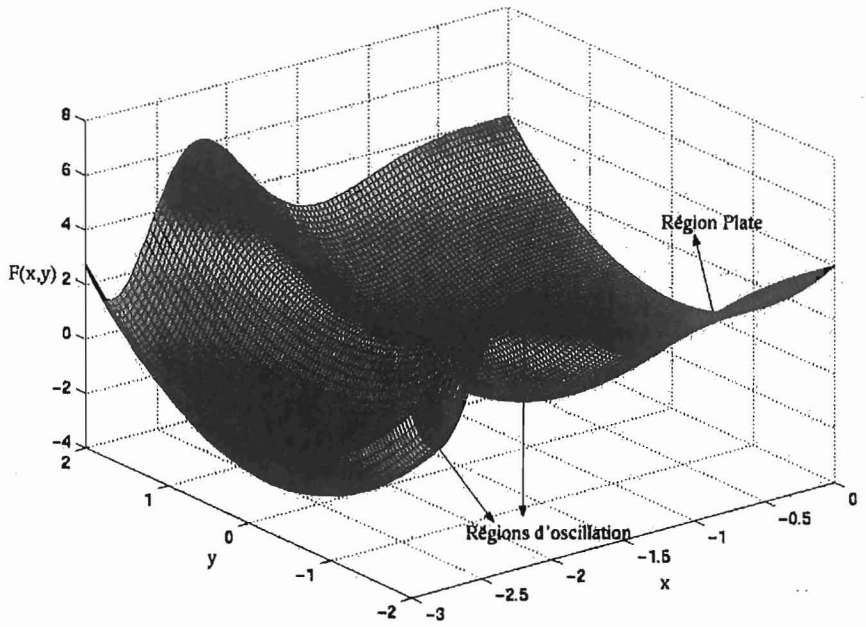


Figure 61 - Allure de la fonction sinusoïdale à 2 paramètres

La Figure 62 présente la surface de réponse obtenue par l'application de la méthode adaptative, tandis que la configuration de nœuds générés après 4 itérations est montrée dans la Figure 63.

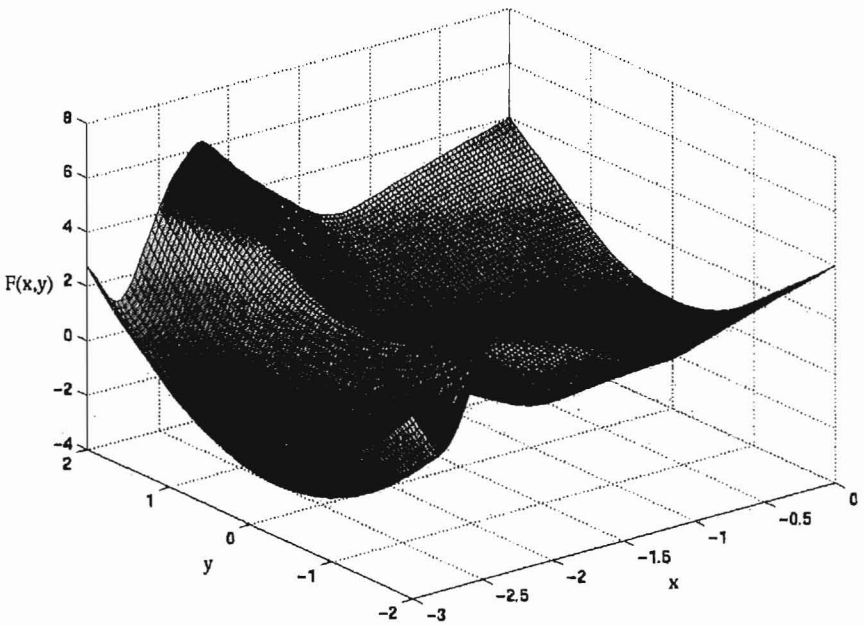


Figure 62 - Surface de réponse adaptative de la fonction sinusoïdale à 2 paramètres

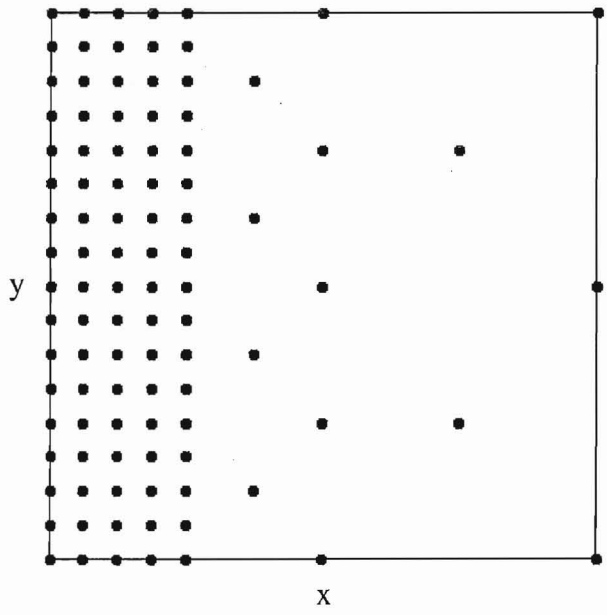


Figure 63 - Configuration finale de nœuds

Dans cet exemple, nous pouvons visualiser de façon claire la concentration de nœuds dans la région d'oscillation de la fonction. Le nombre de nœuds utilisés pour l'approximation est égal à 99, tandis que l'erreur de l'approximation calculée à partir de l'équation (54) est de 10,8%.

III.2.2.2 - Adaptativité par Rapport à l'Optimum

Une autre manière de créer une surface de réponse adaptative consiste à ajouter des nœuds dans les régions les plus favorables pour obtenir l'optimum global de la fonction. Ce type d'approche, qui a déjà été étudiée par d'autres auteurs [Alotto 1996] [Pahner 2000], nous fournit une *surface de réponse adaptative par rapport à l'optimum.* Ici, nous proposons une nouvelle version de cette approche, en utilisant la méthode des éléments diffus.

- *Étapes du Processus Adaptatif*

La démarche initiale utilisée pour obtenir l'approximation adaptative est la même que dans l'approche standard: créer un domaine de dimension égale au nombre des paramètres, discrétiser ce domaine, générer les éléments diffus et résoudre le système matriciel donné par la configuration d'éléments existante.

Pourtant, la discrétisation initiale doit être assez faible (de l'ordre de trois points par direction), de façon à demander un nombre restreint d'évaluations de la fonction objectif - Figure 64.

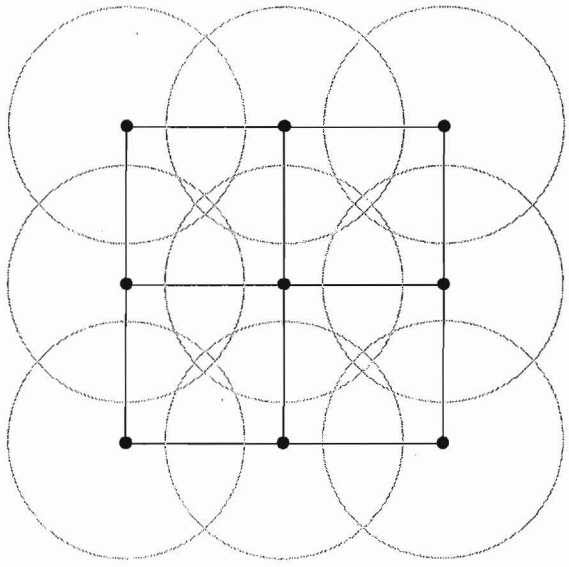


Figure 64 - Configuration d'éléments à la fin de la première itération

Ensuite, nous utilisons une méthode d'optimisation globale, comme par exemple les algorithmes génétiques, pour localiser le point optimal de la fonction approchée par cette configuration initiale. Une fois que nous avons localisé le point optimal, nous cherchons l'élément dont le centre est le plus proche de ce point et qui en même temps le contient - Figure 65.

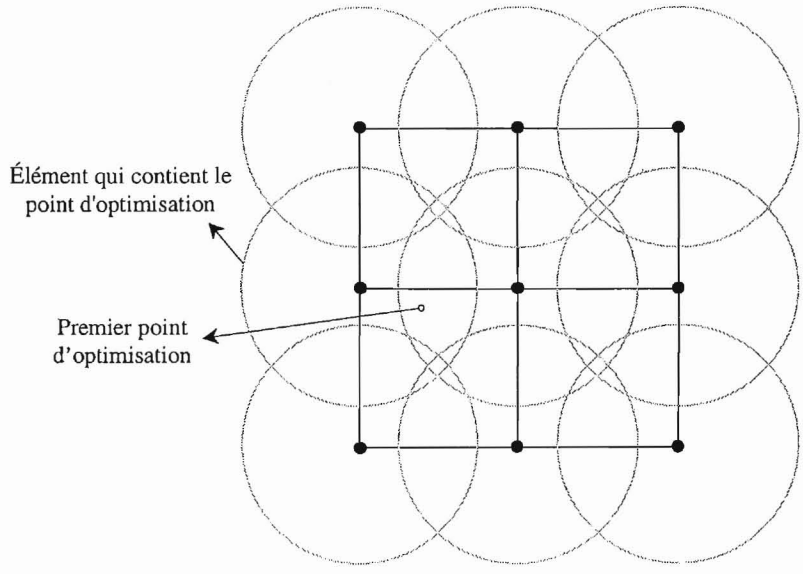


Figure 65 - Point d'optimisation trouvé à la fin de la première itération

À côté du centre de cet élément, nous créons des nouveaux éléments, dont les centres sont à une distance égale à la moitié de la distance entre deux nœuds de la configuration initiale. Le rayon de chaque nouvel élément, ainsi que de celui qui contient le point d'optimisation, sera égal à la moitié du rayon des éléments initiaux, comme nous pouvons vérifier sur la Figure 66.

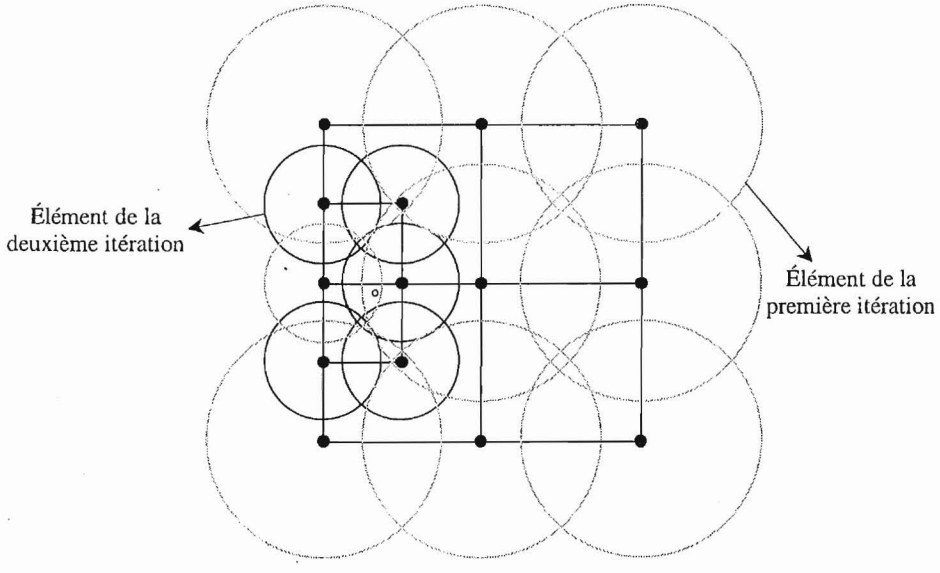


Figure 66 - Configuration d'éléments à la fin de la deuxième itération

Après avoir obtenu cette nouvelle configuration, nous créons un autre système matriciel avec tous les éléments existants, dont la solution nous permet d'obtenir une approximation plus précise de la fonction objectif, surtout dans la région qui contient les nouveaux éléments ajoutés. Ce processus se répète jusqu'au moment où le point optimal obtenu par deux itérations consécutives est le même dans une tolérance prédéterminée.

- *Application*

Nous avons appliqué la méthode de surface de réponse adaptative par rapport à l'optimum sur la fonction $f(x,y)$ de l'équation (49), en utilisant des éléments d'ordre 1. Comme nous pouvons observer dans la Figure 46, cette fonction possède 4 différents points d'optimisation, parmi lesquels nous avons 3 locaux et 1 global, représenté par le point de coordonnées (-4,4537;-4,4537) sur lequel $f(x,y)$ est égale à -5,2327.

Nous avons utilisé un algorithme génétique constitué par une population de 30 individus et 200 générations pour trouver le minimum global à chaque itération du processus adaptatif. Le Tableau VI présente le point d'optimisation obtenu à la fin de chaque itération, où nous pouvons vérifier qu'après 3 itérations de la méthode adaptative, nous nous trouvons déjà à côté du point d'optimisation réel de la fonction.

TABLEAU VI - ÉVOLUTION DE LA MÉTHODE ADAPTATIVE

Itération	Évaluations	x	y	f(x,y)
1	9	-0,7908	-0,7908	0,0210
2	17	-3,5004	-3,5004	-4,8600
3	25	-4,5859	-4,5859	-5,6730
4	33	-4,3886	-4,3886	-5,3563
5	41	-4,3685	-4,3685	-5,2173

Après 5 itérations du processus, nous avons la configuration de nœuds illustrée sur la Figure 67, dans laquelle nous pouvons vérifier la concentration d'éléments autour de l'optimum global de la fonction. L'algorithme génétique a fait environ 6.000 appels à l'approximation de la fonction objectif par itération, tandis qu'elle n'a été évaluée que 41 fois.

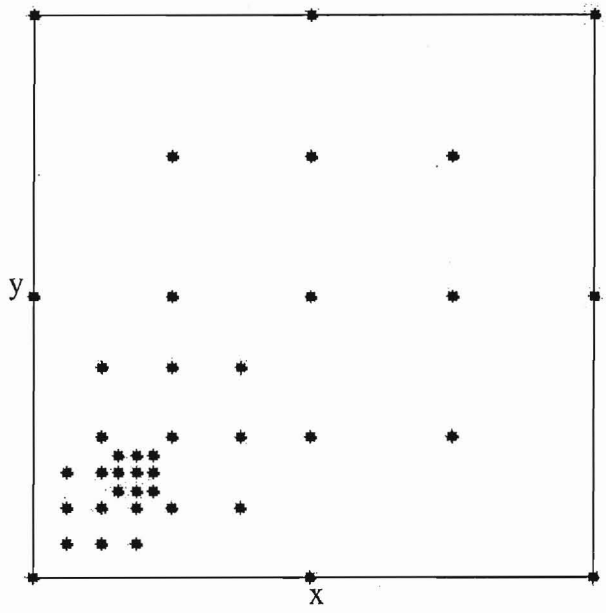


Figure 67 - Configuration de nœuds à la fin de la cinquième itération

III.3 - Identification des Paramètres Influents sur la Valeur de la Fonction Objectif

Nous avons vu que l'utilisation d'une surface de réponse adaptative peut nous amener à une réduction significative du nombre d'évaluations de la fonction objectif, surtout quand nous utilisons l'algorithme d'adaptativité par rapport à l'optimum de la fonction.

Cependant, lorsque le nombre de paramètres de la fonction augmente, la performance de l'approche adaptative peut se trouver en échec, car le nombre d'évaluations de la fonction devient encore prohibitif, même en utilisant une discrétisation initiale du domaine assez faible.

Pour éviter cet inconvénient, nous allons essayer d'identifier les paramètres qui n'ont pas d'influence significative sur la valeur de la fonction, ce qui nous permettra de ne pas les considérer dans le processus de création de la surface de réponse. L'identification de ces paramètres sera obtenue à partir de l'application de la *Méthode des Plans d'Expériences* [Sado 1991] [Demonsant 1996] [Pillet 1997] [Schimmerling 1998].

III.3.1 - Méthode des Plans d'Expériences

La *Méthode des Plans d'Expériences* est une méthode analytique qui présente plusieurs applications dans un processus d'optimisation, parmi lesquelles nous avons l'identification des paramètres significatifs pour la valeur de la fonction objectif, la construction d'une surface de réponse et même l'optimisation de la fonction.

Dans ce travail, nous n'allons nous intéresser qu'à l'identification des paramètres significatifs pour la valeur de la fonction. Les détails concernant les autres applications des Plans d'Expériences peuvent être trouvés dans les ouvrages de [Sado 1991] [Pillet 1997] [Schimmerling 1998] et dans la thèse de Gillon [Gillon 1997].

III.3.1.1 - Principe de la Méthode

Soit y la réponse d'un système constitué par k différents facteurs (paramètres) x_i, ayant chacun 2 niveaux de variation x_{i-} et x_{i+}. Nous cherchons un modèle algébrique linéaire de la forme donnée par l'équation (56) qui nous permette de prévoir la valeur de y pour une combinaison quelconque entre les facteurs x_i.

$$y = a_0 + a_1x_1 + a_2x_2 + ... + a_kx_k + a_{12}x_1x_2 + ... + a_{1k}x_1x_k + ... + a_{1...k}x_1...x_k, \qquad (56)$$

où

a_j sont des coefficients qui représentent l'effet des facteurs et des leurs interactions sur la réponse du système.

Les coefficients a_j sont obtenus en observant la valeur de y pour différentes combinaisons entre les niveaux des facteurs x_i. L'ensemble de combinaisons réalisées de façon optimale dans le but d'obtenir les coefficients a_j constitue la *matrice d'essais* d'un *Plan d'Expériences* - Tableau VII.

TABLEAU VII - MATRICE D'ESSAIS D'UN PLAN D'EXPÉRIENCES POUR 5 FACTEURS

x_1	x_2	x_3	x_4	x_5	y
x_{1-}	x_{2-}	x_{3-}	x_{4+}	x_{5+}	y_1
x_{1-}	x_{2-}	x_{3+}	x_{4+}	x_{5-}	y_2
x_{1-}	x_{2+}	x_{3-}	x_{4-}	x_{5+}	y_3
x_{1-}	x_{2+}	x_{3+}	x_{4+}	x_{5-}	y_4
x_{1+}	x_{2-}	x_{3-}	x_{4-}	x_{5-}	y_5
x_{1+}	x_{2-}	x_{3+}	x_{4+}	x_{5+}	y_6
x_{1+}	x_{2+}	x_{3-}	x_{4-}	x_{5-}	y_7
x_{1+}	x_{2+}	x_{3+}	x_{4+}	x_{5+}	y_8

III.3.1.2 - Plan Factoriel Complet

Le *Plan Factoriel Complet* est le plan d'expériences dans lequel nous avons une matrice d'essais qui représentent toutes les combinaisons possibles entre les niveaux des facteurs x_i, comme celle représentée dans le Tableau VIII.

TABLEAU VIII - PLAN FACTORIEL COMPLET POUR 4 FACTEURS

x1	x_2	x_3	x_4	y
x_{1-}	x_{2-}	x_{3-}	x_{4-}	y_1
x_{1-}	x_{2-}	x_{3-}	x_{4+}	y_2
x_{1-}	x_{2-}	x_{3+}	x_{4-}	y_3
x_{1-}	x_{2-}	x_{3+}	x_{4+}	y_4
x_{1-}	x_{2+}	x_{3-}	x_{4-}	y_5
x_{1-}	x_{2+}	x_{3-}	x_{4+}	y_6
x_{1-}	x_{2+}	x_{3+}	x_{4-}	y_7
x_{1-}	x_{2+}	x_{3+}	x_{4+}	y_8
x_{1+}	x_{2-}	x_{3-}	x_{4-}	y_9
x_{1+}	x_{2-}	x_{3-}	x_{4+}	y_{10}
x_{1+}	x_{2-}	x_{3+}	x_{4-}	y_{11}
x_{1+}	x_{2-}	x_{3+}	x_{4+}	y_{12}
x_{1+}	x_{2+}	x_{3-}	x_{4-}	y_{13}
x_{1+}	x_{2+}	x_{3-}	x_{4+}	y_{14}
x_{1+}	x_{2+}	x_{3+}	x_{4-}	y_{15}
x_{1+}	x_{2+}	x_{3+}	x_{4+}	y_{16}

Le nombre d'expériences réalisées par un plan complet à 2 niveaux est donné par:

$$n = 2^k, \tag{57}$$

où

k est le nombre de facteurs considérés.

Le grand avantage du plan factoriel complet c'est qu'il permet d'estimer non seulement les effets principaux des facteurs, mais également de toutes leurs interactions deux à deux, trois à trois, ..., jusqu'à l'interaction qui fait intervenir les *k* facteurs. Néanmoins, lorsque le nombre de facteurs augmente, son utilisation conduit rapidement à un nombre prohibitif d'expériences à réaliser.

III.3.1.3 - Plan Factoriel Fractionnaire

Normalement, nous ne sommes pas intéressés à identifier l'effet de toutes les interactions existantes dans le modèle, car les interactions d'ordre 2 (du type $x_1x_2x_3$) ou supérieur sont bien souvent considérées négligeables. Nous pouvons alors utiliser un *Plan Factoriel Fractionnaire* ou *Incomplet* qui nous permet d'estimer l'effet des facteurs et des interactions les plus importantes sur la réponse du système avec un nombre d'expériences réduit.

Les plans fractionnaires sont normalement construits en utilisant des matrices d'essais prédéfinies basées sur les *Tableaux de Taguchi* [Pillet 1997] ou sur les *Générateurs de G. Box* [Demonsant 1996]. Ainsi comme les plans complets, les plans fractionnaires sont dits optimaux, car les colonnes de leur matrice d'essais sont orthogonales entre elles [Pillet 1997], comme nous montre le Tableau IX.

TABLEAU IX - PLAN FACTORIEL FRACTIONNAIRE POUR 4 FACTEURS

x_1	x_2	x_3	x_4	y
x_{1-}	x_{2-}	x_{3-}	x_{4-}	y_1
x_{1-}	x_{2-}	x_{3+}	x_{4+}	y_2
x_{1-}	x_{2+}	x_{3-}	x_{4+}	y_3
x_{1-}	x_{2+}	x_{3+}	x_{4-}	y_4
x_{1+}	x_{2-}	x_{3-}	x_{4+}	y_5
x_{1+}	x_{2-}	x_{3+}	x_{4-}	y_6
x_{1+}	x_{2+}	x_{3-}	x_{4-}	y_7
x_{1+}	x_{2+}	x_{3+}	x_{4+}	y_8

- *Identification des Confusions*

Lorsque nous utilisons un plan d'expériences à 2 niveaux, nous pouvons représenter la matrice d'essais du plan en utilisant la notation proposée par l'analyste français Jacques Hadamard [Sado 1991], selon laquelle les niveaux minimal et maximal d'un facteur sont respectivement donnés par les valeurs *–1* et *+1*, comme nous montre le Tableau X.

TABLEAU X - NOTATION D'HADAMARD

x_1	x_2	x_3	x_4	y
−1	−1	−1	−1	y_1
−1	−1	+1	+1	y_2
−1	+1	−1	+1	y_3
−1	+1	+1	−1	y_4
+1	−1	−1	+1	y_5
+1	−1	+1	−1	y_6
+1	+1	−1	−1	y_7
+1	+1	+1	+1	y_8

En utilisant cette notation, nous pouvons facilement constater que dans le plan du Tableau X, la colonne correspondant au facteur x_4 est égale au produit entre les colonnes des facteurs x_1, x_2 et x_3. Nous disons alors qu'existe une *confusion* entre le facteur x_4 et l'interaction $x_1x_2x_3$ ou encore que x_4 et $x_1x_2x_3$ sont "aliasés" [Sado 1991] [Pillet 1997].

La notation d'Hadamard, ainsi que les tableaux de Taguchi [Pillet 1997] et les générateurs de G. Box [Demonsant 1996], nous permettent d'identifier toutes les confusions existantes entre les facteurs et leurs interactions par une simple multiplication entre les colonnes de la matrice d'essais.

- *Contrastes*

L'utilisation d'un plan fractionnaire pour identifier les coefficients a_j du modèle diminue considérablement le nombre d'expériences effectuées. Cependant les coefficients a_j calculés ne représentent plus l'effet des facteurs et de leurs interactions séparément, mais une somme de ces effets, comme nous pouvons vérifier dans la Figure 68. Cette somme, constituée par des *confusions* entre les effets, est dénommée *contraste* du plan fractionnaire [Sado 1991].

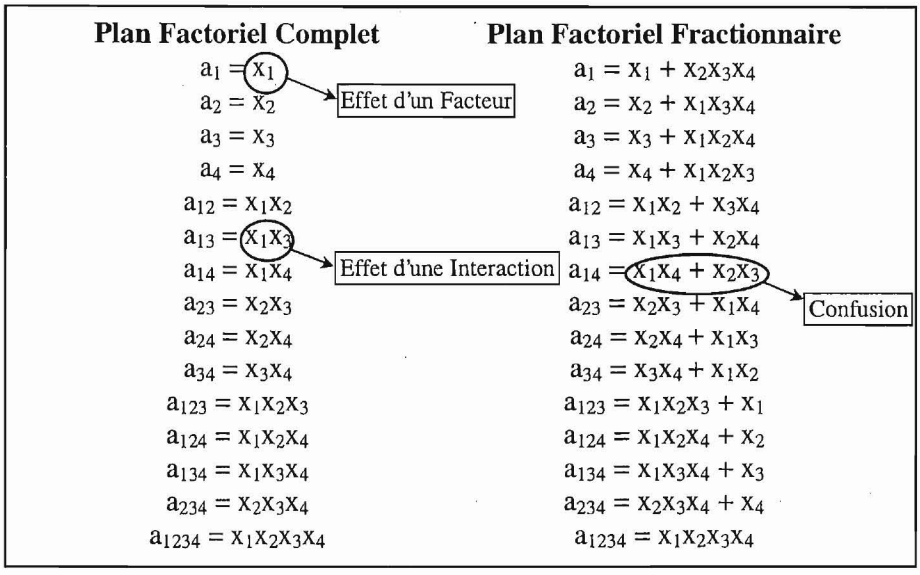


Figure 68 - Confusions et contrastes d'un plan fractionnaire

III.3.1.4 - Estimation des Coefficients du Modèle

- *Estimation du Coefficient* a_0

Nous pouvons estimer la valeur du coefficient a_0 à partir de la moyenne arithmétique de toutes les réponses y_i observées, soit:

$$a_0 = \overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \,, \tag{58}$$

où

y_i est la réponse observée pour l'expérience i

n est le nombre d'expériences réalisées.

- *Estimation des Coefficients Concernant les Effets Principaux*

L'effet principal d'un facteur x quelconque s'obtient en comparant les valeurs prises par y quand x passe du niveau x_- au niveau x_+, comme nous montre la Figure 69.

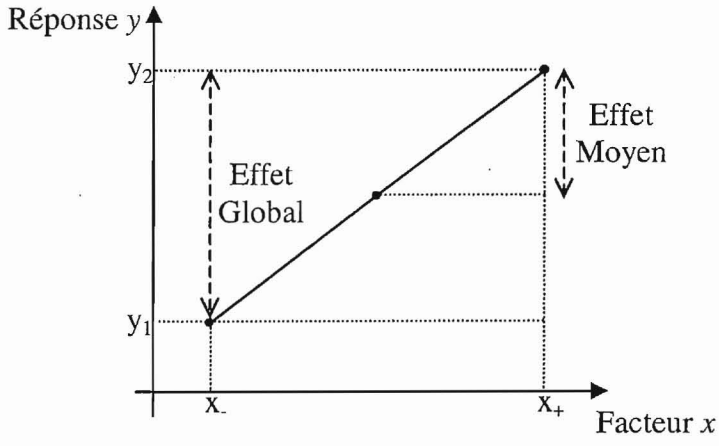


Figure 69 - Effet d'un facteur

Nous pouvons calculer l'effet d'un facteur x_j au niveau x_{j+} et par conséquent le coefficient a_j associé à cet effet en utilisant l'équation suivante:

$$a_j = e_{a_j} = y^+_{x_j} - a_0, \tag{59}$$

dans laquelle

$$y^+_{x_j} = \frac{1}{n^+} \sum_{i=1}^{n^+} y_i^+, \tag{60}$$

où

y_i^+ est la réponse observée pour l'expérience i quand x_j est au niveau x_{j+}

n^+ est le nombre d'expériences dans lesquelles x_j est au niveau x_{j+}

e_{a_j} est l'effet du coefficient a_j

Si nous considérons x_j au niveau x_{j-} au lieu de le considérer au niveau x_{j+} pour calculer le coefficient a_j, la valeur obtenue sera égale à l'opposée de celle donnée par l'équation (59).

III.3.1.5 - Identification des Facteurs Significatifs

Une fois que nous avons montré la façon de calculer les coefficients a_j du modèle de l'équation (56) et d'identifier les confusions existantes entre ces coefficients (dans le cas où nous utilisons un plan factoriel fractionnaire), il nous reste à déterminer les coefficients et par conséquent les facteurs qui ont une influence significative sur la réponse du modèle.

- *Calcul des Contributions des Contrastes*

Une étude de la variance liée au test de Snedecor-Fisher [Gillon 1997] [Pillet 1997] [Gillon 2000] de la réponse y par rapport aux facteurs est souvent utilisée dans le but d'identifier les facteurs significatifs, mais selon Schimmerling [Schimmerling 1998], cette méthodologie perd son sens lorsque nous utilisons la méthode des plans d'expériences pour analyser des résultats sans composantes aléatoires, dans lesquels la réponse est toujours la même pour une même configuration de niveaux de paramètres.

En fait, les résultats présentés en [Giurgea 2000] pour deux différentes applications liées à la simulation par éléments finis, dont les réponses n'ont pas de composantes aléatoires, montrent qu'une étude de la variance liée au test de Snedecor-Fisher ne fournit pas de conclusion fiable à propos de l'influence des facteurs.

Schimmerling nous propose alors d'identifier les facteurs significatifs en évaluant la contribution des coefficients (ou des contrastes, si nous utilisons des plans fractionnaires) sur la réponse du modèle à partir de la normalisation de leur valeur par rapport à la *somme des carrés des écarts* des réponses, comme l'indiquent les équations suivantes:

$$C_{a_j} = \frac{SCE(a_j)}{SCE(y)}[\%], \tag{61}$$

dans laquelle

$$SCE(y) = \sum_{i=1}^{n}(y_i - \bar{y})^2 \tag{62}$$

$$SCE(a_j) = \frac{n}{s}\sum_{j=1}^{s}(e_{a_j})^2, \tag{63}$$

où

s est le nombre de niveaux du plan

e_{a_j} est l'effet du facteur a_j calculé à partir de l'équation (59)

C_{a_j} est la contribution du facteur ou du contraste associé au coefficient a_j

La contribution donnée par (61) sera alors considérée significative à partir d'une valeur C_{lim} prédéterminée. D'après [Giurgea 2000], cette valeur empirique doit être de l'ordre de *5%*.

- *Interprétation des Résultats*

Lorsque nous utilisons des plans fractionnaires, l'interprétation correcte des résultats concernant les contributions des contrastes dépend de quelques considérations importantes que nous devons prendre en compte [Sado 1991] [Costa 2001a] [Costa 2001b]:

- Les interactions d'ordre élevé (supérieur à deux) peuvent être considérées négligeables.
- Lorsqu'un contraste est négligeable, tous les effets qui composent ce contraste peuvent être aussi considérés négligeables.
- Deux facteurs significatifs peuvent générer une interaction aussi significative. Par contre, deux facteurs non significatifs ne génèrent pas d'interactions significatives.

Ces considérations seront confirmées par les applications qui seront présentées dans le chapitre suivant.

III.4 - Conclusion

Nous avons vu que l'utilisation d'une surface de réponse peut nous apporter plusieurs solutions aux difficultés posées par les problèmes d'optimisation liés à la simulation numérique, notamment en ce qui concerne le temps de calcul onéreux.

Nous avons montré les avantages de l'utilisation de la Méthode des Éléments Diffus dans la construction de la surface de réponse, parmi lesquelles nous pouvons citer la facilité de sa mise en œuvre et sa capacité de fournir des surfaces de réponses adaptatives.

Finalement, nous avons vu que la Méthode des Plans d'Expériences peut nous donner des informations précieuses concernant l'influence des paramètres sur la valeur de la fonction objectif, ce qui peut orienter la construction de la surface de réponse et en même temps garantir la qualité de la solution du problème d'optimisation.

Chapitre IV

Applications en Electrotechnique

Chapitre IV

Applications en Électrotechnique

IV.1 - Introduction

Dans ce dernier chapitre, nous allons présenter l'application de la méthodologie d'optimisation basée sur la construction d'une surface de réponse dans la résolution de trois problèmes en électrotechnique différents.

Nous commencerons par la résolution du problème 25 du TEAM Workshop, qui fait partie d'une série de problèmes tests internationaux permettant de valider des nouvelles méthodes de résolution. Ensuite, nous allons optimiser la géométrie d'un moteur à réluctance variable, dans le but de maximiser le couple magnétique exercé par ce moteur.

Nous allons utiliser pour ces deux applications la même démarche de résolution, dans laquelle nous trouverons la description du problème d'optimisation avec la définition de la fonction objectif et des paramètres à optimiser, l'identification des paramètres significatifs en utilisant la méthode des plans d'expériences, la construction d'une surface de réponse de la fonction objectif par rapport aux paramètres significatifs et finalement l'optimisation de ces paramètres.

Dans notre troisième application, nous allons présenter la démarche d'utilisation de la méthodologie de surface de réponse dans l'optimisation d'un contacteur électromagnétique soumis à des contraintes de conception.

IV.2 - Résolution du Problème 25 du TEAM Workshop

IV.2.1 - Description du Problème

Le but du problème 25 du TEAM Workshop [Takahashi 1996] [Alotto 1998] est d'optimiser la forme d'un moule utilisé dans la construction d'aimants permanents anisotropes, afin d'obtenir une distribution radiale de flux dans la cavité indiquée sur la Figure 70.

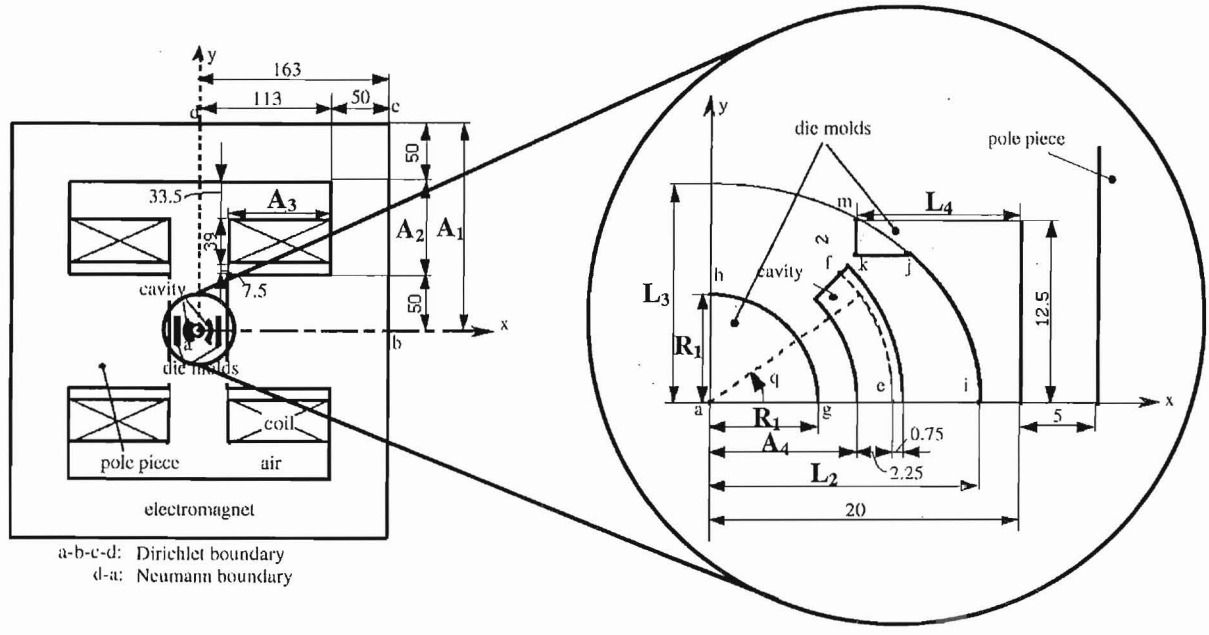


Figure 70 - Problème 25 du TEAM Workshop

Le dispositif est alimenté par 2 bobines inductrices, dont chaque enroulement porte 4253 ampère-tours, tandis que son armature est constituée par un matériau magnétique non-linéaire.

Ce problème peut être résolu par la méthode des éléments finis, en utilisant une formulation magnétostatique bidimensionnelle non-linéaire. Nous avons alors utilisé le module de résolution en deux dimensions du logiciel d'éléments finis Flux3D® pour analyser ce dispositif.

IV.2.1.1 - Paramètres à optimiser

La forme du moule est définie par deux pièces magnétiques dont la géométrie dépend des paramètres R_1, L_2, L_3 et L_4, qui peuvent être visualisés dans la Figure 70. Nous avons ajouté quatre nouveaux paramètres géométriques A_1, A_2, A_3, A_4 à la description du problème original.

À priori, ces nouveaux paramètres, illustrés aussi dans la Figure 70, n'ont pas d'influence significative dans la valeur de la fonction objectif et nous allons vérifier cette considération en utilisant les plans d'expériences.

En ce qui concerne les limites de variations des paramètres du problème original et de ceux qui ont été ajoutés à la description du problème, ils sont donnés par les valeurs montrées dans le Tableau XI.

TABLEAU XI - LIMITES DE VARIATION DES PARAMÈTRES

Paramètre	Valeur Minimale (mm)	Valeur Maximale (mm)
R_1	5,0	9,4
L_2	12,6	18,0
L_3	14,0	45,0
L_4	4,0	19,0
A_1	170,0	190,0
A_2	70,0	90,0
A_3	86,0	88,0
A_4	9,5	11,0

IV.2.1.2 - Fonction Objectif

L'objectif recherché est de disposer le long de la courbe *ef* de la Figure 70, d'une induction $\vec{B}$ dont les composants valent:

$$\begin{cases} B_x = 0{,}35\cos\theta \\ B_y = 0{,}35\sin\theta \end{cases}, \tag{64}$$

où

θ ($0° < \theta < 45°$) est l'angle par rapport à l'axe des abscisses.

Le problème peut être décrit alors par la minimisation de la fonction objectif de l'équation (65) qui représente l'erreur commise par l'induction mesurée vis à vis de la valeur de référence sur la courbe *ef*.

$$f = \sum_{i=1}^{n} \left\{ \left(B_{i_x} - 0{,}35\cos\theta_i \right)^2 + \left(B_{i_y} - 0{,}35\sin\theta_i \right)^2 \right\}, \tag{65}$$

où

n (=10) est le nombre de points utilisés pour mesurer l'induction.

IV.2.2 - Résolution du Problème

Nous avons effectué la résolution de ce problème en trois différentes étapes, à savoir:

- L'identification des paramètres significatifs.
- L'optimisation en utilisant une surface de réponse adaptative et une surface de réponse régulière.
- Comparaison des résultats obtenus avec la méthodologie standard d'optimisation.

IV.2.2.1 - Identification des Paramètres Significatifs

Dans la première étape de notre application nous avons utilisé la méthode des Plans d'Expériences de façon à identifier les paramètres qui ont une influence significative sur la valeur de la fonction objectif *f*.

- *Identification à partir d'un Plan Factoriel Fractionnaire*

L'équation (57) nous montre que l'utilisation d'un plan factoriel complet à 2 niveaux pour le débroussaillage de paramètres de ce problème demanderait 256 (2^8) évaluations de la fonction objectif, ce qui représenterait un temps de résolution assez important.

De façon à réduire le temps de résolution, nous avons décidé d'utiliser un plan factoriel fractionnaire de 16 expériences basé sur le *Tableau L_{16} de Taguchi* [Pillet 1997]. Les valeurs utilisées pour les paramètres en chaque expérience, ainsi que les résultats des évaluations issues du calcul par éléments finis sont montrés dans le Tableau XII.

TABLEAU XII - RÉSULTATS DES EXPÉRIENCES DU PLAN FACTORIEL FRACTIONNAIRE

R_1	L_2	L_3	L_4	A_1	A_2	A_3	A_4	f
5,0	12,6	14,0	4,0	170,0	70,0	86,0	9,5	0,0522
5,0	12,6	14,0	19,0	170,0	90,0	90,0	11,0	0,0710
5,0	12,6	45,0	4,0	190,0	90,0	90,0	9,5	0,2288
5,0	12,6	45,0	19,0	190,0	70,0	86,0	11,0	0,1712
5,0	18,0	14,0	4,0	190,0	90,0	86,0	11,0	0,1273
5,0	18,0	14,0	19,0	190,0	70,0	90,0	9,5	0,1756
5,0	18,0	45,0	4,0	170,0	70,0	90,0	11,0	0,1665
5,0	18,0	45,0	19,0	170,0	90,0	86,0	9,5	0,2646
9,4	12,6	14,0	4,0	190,0	70,0	90,0	11,0	1,2066
9,4	12,6	14,0	19,0	190,0	90,0	86,0	9,5	0,3466
9,4	12,6	45,0	4,0	170,0	90,0	86,0	11,0	1,1023
9,4	12,6	45,0	19,0	170,0	70,0	90,0	9,5	0,5924
9,4	18,0	14,0	4,0	170,0	90,0	90,0	9,5	0,1230
9,4	18,0	14,0	19,0	170,0	70,0	86,0	11,0	0,0040
9,4	18,0	45,0	4,0	190,0	70,0	86,0	9,5	0,0537
9,4	18,0	45,0	19,0	190,0	90,0	90,0	11,0	0,0188

En utilisant les équations présentées dans les sections III.3.1.4 et III.3.1.5, nous avons identifié tous les contrastes dont la contribution sur la valeur de la fonction est supérieure à 5%. Ces contrastes, ainsi que leurs valeurs de contribution, sont présentés dans le Tableau XIII.

TABLEAU XIII - CONTRIBUTIONS OBTENUES PAR L'APPLICATION D'UN PLAN FRACTIONNAIRE

Contraste	Confusions	Contribution (%)
A	$\mathbf{R_1} + L_2L_3A_1 + L_3L_4A_3 + (*)$	14,91
B	$\mathbf{L_2} + R_1L_3A_1 + L_3L_4A_2 + (*)$	25,02
C	$\mathbf{L_4} + L_2L_3A_2 + R_1L_3A_3 + (*)$	6,23
D	$\mathbf{R_1L_2} + L_3A_1 + L_4A_4 + A_2A_3 + (*)$	33,01
E	$\mathbf{R_1L_4} + L_3A_3 + L_2A_4 + A_1A_2 + (*)$	8,27
F	$\mathbf{L_2L_4} + L_3A_2 + R_1A_4 + A_1A_3 + (*)$	6,10

(*) quelques interactions d'ordre supérieur.

D'après les considérations de la section III.3.1.5, les interactions d'ordre supérieur à 2 peuvent être souvent considérées négligeables. Cela nous permet de dire que ce ne sont pas les interactions qui font partie des contrastes *A*, *B* et *C* qui contribuent pour qu'ils soient significatifs, mais les effets principaux des facteurs R_1, L_2 et L_4.

En ce qui concerne les contrastes *D*, *E* et *F* contenant des interactions d'ordre 2, nous pouvons dire que seulement les interactions provenant des facteurs significatifs sont aussi significatives, soit R_1L_2, R_1L_4 et L_2L_4.

Ainsi, nous pouvons ne considérer que R_1, L_2 et L_4 comme étant les paramètres influents à l'optimisation du problème, ce qui représente une réduction de plus de 50% dans le nombre de paramètres à optimiser.

- *Validation à partir d'un Plan Factoriel Complet*

Pour valider les conclusions que nous venons de montrer, nous avons utilisé le plan factoriel complet sur le problème original, dans lequel nous n'avons que 4 paramètres (R_1, L_2, L_3 et L_4) à optimiser. Cela nous a permis d'obtenir des résultats sans avoir des confusions entre les facteurs et leurs interactions en n'effectuant que 16 expériences, dont les valeurs sont montrées dans le Tableau XIV.

TABLEAU XIV - RÉSULTATS DES EXPÉRIENCES DU PLAN FACTORIEL COMPLET

R_1	L_2	L_3	L_4	f
5,0	12,6	14,0	4,0	0,0543
5,0	12,6	14,0	19,0	0,0767
5,0	12,6	45,0	4,0	0,2197
5,0	12,6	45,0	19,0	0,1660
5,0	18,0	14,0	4,0	0,1187
5,0	18,0	14,0	19,0	0,1905
5,0	18,0	45,0	4,0	0,1737
5,0	18,0	45,0	19,0	0,2475
9,4	12,6	14,0	4,0	1,1039
9,4	12,6	14,0	19,0	0,4071
9,4	12,6	45,0	4,0	1,2488
9,4	12,6	45,0	19,0	0,5248
9,4	18,0	14,0	4,0	0,1068
9,4	18,0	14,0	19,0	0,0063
9,4	18,0	45,0	4,0	0,0622
9,4	18,0	45,0	19,0	0,0404

Le Tableau XV présente les contributions des facteurs et de leurs interactions calculées en utilisant les équations des sections III.3.1.4 et III.3.1.5. Les lignes contenant les facteurs et les interactions dont la contribution à la réponse du modèle est au-dessus de 5% sont détachés et nous pouvons vérifier que le paramètre L_3 n'en fait pas partie.

TABLEAU XV - CONTRIBUTIONS OBTENUES PAR L'APPLICATION DU PLAN FACTORIEL COMPLET

Facteur ou Interaction	Contribution (%)
R_1	15,13
L_2	25,01
L_3	1,09
L_4	6,34
R_1L_2	32,99
R_1L_3	0,05
R_1L_4	8,50
L_2L_3	0,57
L_2L_4	6,43
L_3L_4	0,01
$R_1L_2L_3$	0,07
$R_1L_2L_4$	0,08
$R_1L_3L_4$	0,03
$L_2L_3L_4$	3,68
$R_1L_2L_3L_4$	0,01

Nous concluons alors que le paramètre L_3 peut effectivement être considéré négligeable, ce que nous avions déjà vérifié en utilisant un plan fractionnaire. Les considérations présentées dans la section III.3.1.5 sont donc valables.

IV.2.2.2 - Optimisation des Paramètres

Après avoir identifié les paramètres qui ont une influence significative sur la valeur de la fonction objectif f, nous avons optimisé leur valeur en utilisant différentes méthodologies d'optimisation.

- *Optimisation à partir d'une Surface de Réponse Régulière*

Tout d'abord, nous avons construit une surface de réponse de la fonction par rapport aux paramètres significatifs R_1, L_2 et L_4, en utilisant des éléments diffus de premier ordre répartis sur une grille régulière discrétisée en 7 points par direction du domaine, ce qui a demandé 343 (7^3) évaluations par élément finis.

La Figure 71 présente une coupe de la surface de réponse obtenue pour la valeur maximale de L_4 (19,0 mm).

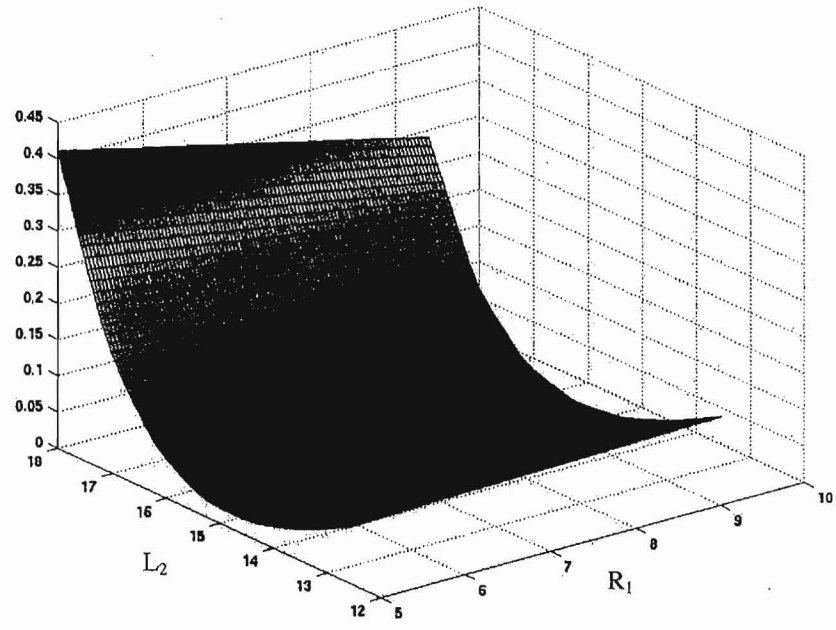


Figure 71 - Surface de réponse régulière

Une fois que la surface de réponse régulière a été générée, nous avons appliqué une méthode de résolution stochastique du type algorithme génétique constitué par une population de 30 individus qui évoluent durant 300 générations. La valeur obtenue pour chaque paramètre à la fin du processus d'optimisation, ainsi que l'évaluation de la fonction objectif évalué à partir de la surface de réponse (f_{sr}) et par calcul éléments finis (f_{mef}) sont présentées dans le Tableau XVI.

TABLEAU XVI - VALEURS DES PARAMÈTRES OPTIMISÉS PAR SURFACE DE RÉPONSE RÉGULIÈRE

Paramètre	**Valeur**
R_1	7,1675
L_2	14,0804
L_4	14,3550
f_{sr}	4,11E-4
f_{mef}	2,15E-4

En ce qui concerne la valeur des 5 autres paramètres du problème, nous avons pris les décisions suivantes pour construire la surface de réponse:

- le paramètre L_3 a été fixé en sa valeur minimale, car les résultats de l'application du plan d'expériences nous ont permis de vérifier que la fonction présente une valeur plus petite lorsque L_3 est dans sa valeur minimale.

- les paramètres A_1, A_2, A_3 et A_4 ont été fixés en leurs valeurs originales, présentés dans le Tableau XVII, ce qui nous permettra de comparer dans la section IV.2.2.3 les résultats obtenus avec d'autres références d'optimisation du problème original [Alotto 1998].

TABLEAU XVII - VALEURS UTILISÉES POUR LES PARAMÈTRES NON OPTIMISÉS

Paramètre	Valeur
L_3	14,0
A_1	180,0
A_2	80,0
A_3	88,0
A_4	9,5

La Figure 72 nous montre les lignes équipotentielles du dispositif optimisé, où nous pouvons visualiser l'induction radiale sur la cavité du moule.

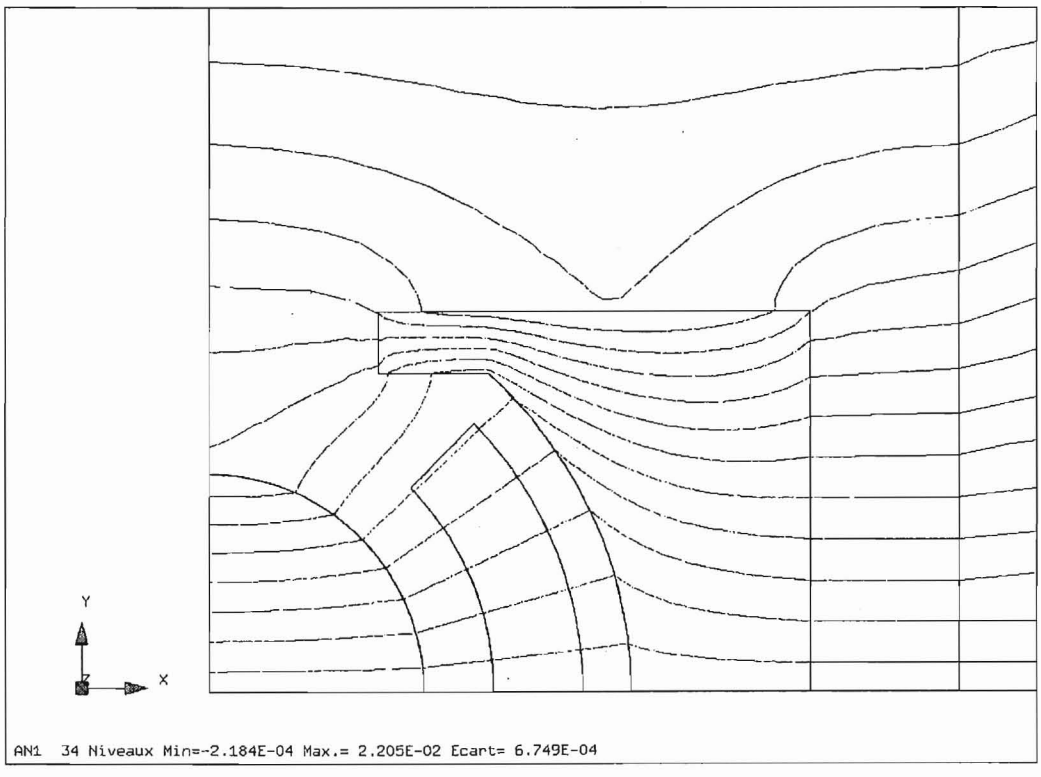


Figure 72 - Dispositif optimisé par surface de réponse régulière

- *Optimisation à partir d'une Surface de Réponse Adaptative*

- Adaptativité par rapport à la qualité de l'approximation

Dans une deuxième étape de notre analyse, nous avons utilisé la méthodologie de surface de réponse adaptative par rapport à la qualité de l'approximation présentée dans la section III.2.2.1 pour optimiser les paramètres R_1, L_2 et L_4.

L'algorithme adaptatif a fait appel à 79 calculs par éléments finis au cours de 2 itérations pour construire la surface de réponse. La Figure 73 présente une coupe de la surface de réponse obtenue pour la valeur maximale de L_4, où nous pouvons percevoir une grande similarité avec celle obtenue par l'approximation régulière (Figure 71).

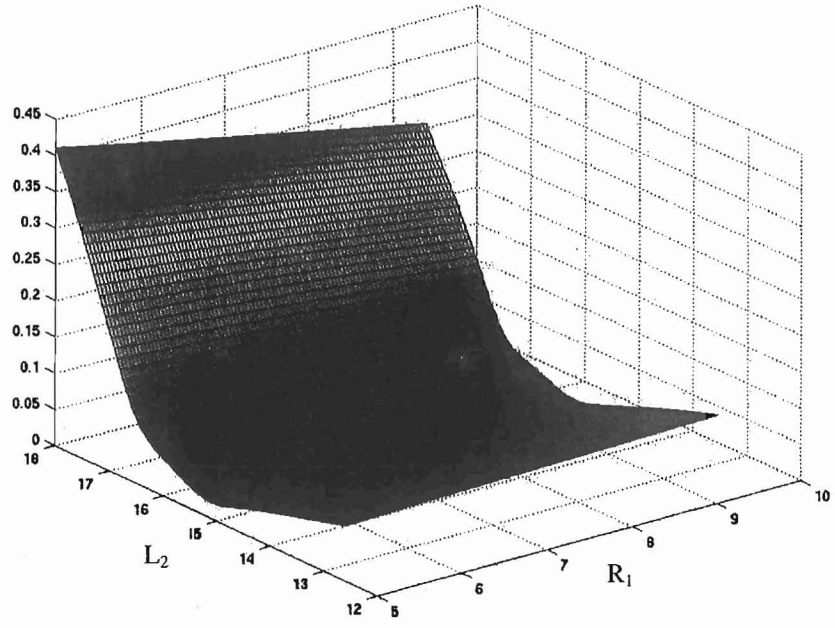


Figure 73 - Surface de réponse adaptative

Après avoir obtenu la surface de réponse, nous avons appliqué sur elle une méthode de résolution stochastique du type algorithme génétique avec une configuration de 30 individus et 300 générations pour identifier le point d'optimisation global.

Les valeurs des paramètres obtenues à la fin de l'optimisation sont présentées dans le Tableau XVIII, tandis que la Figure 74 présente la géométrie du dispositif correspondante à ces valeurs.

TABLEAU XVIII - VALEURS DES PARAMÈTRES OPTIMISÉS PAR SURFACE DE RÉPONSE ADAPTATIVE

Paramètre	Valeur
R_1	8,1039
L_2	16,0764
L_4	15,7306
f_{sr}	2,51E-3
f_{mef}	7,44E-4

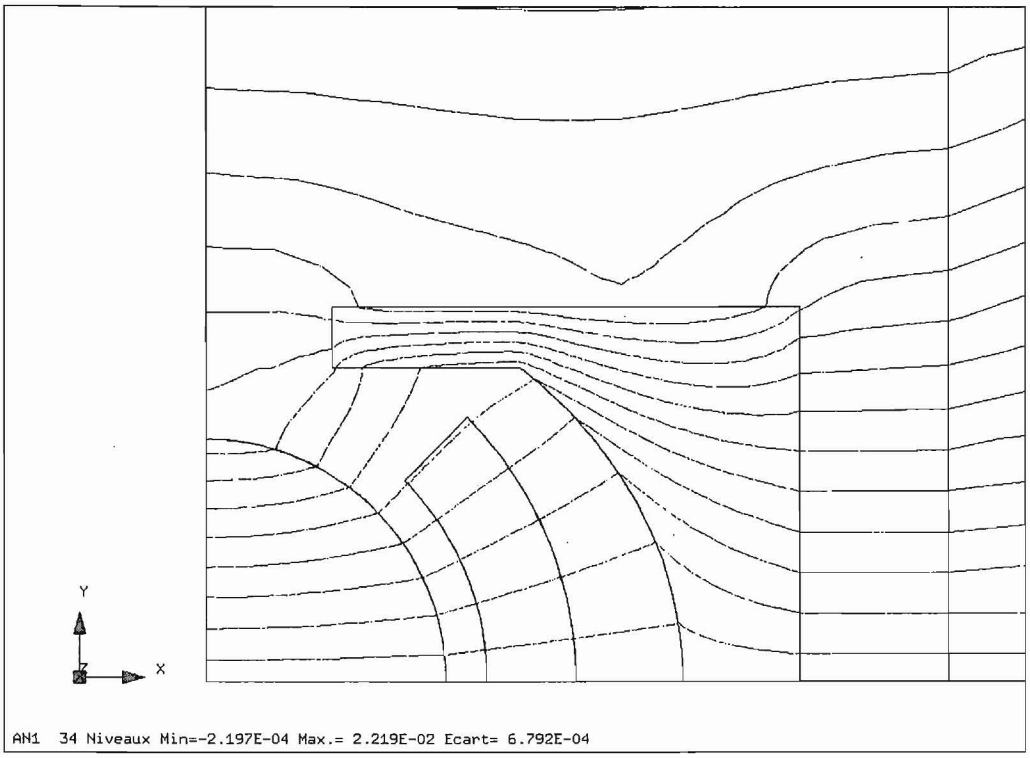


Figure 74 - Dispositif optimisé par surface de réponse adaptative par rapport à la qualité

- Adaptativité par rapport à l'optimum

Pour finaliser notre analyse, nous avons utilisé la méthodologie de surface de réponse adaptative par rapport à l'optimum présentée dans la section III.2.2.2 pour optimiser les paramètres R_1, L_2 et L_4.

Nous avons appliqué une méthode de résolution stochastique du type algorithme génétique avec une configuration de 30 individus et 300 générations pour identifier le point d'optimisation à chaque itération de l'algorithme adaptatif.

Après 3 itérations du processus adaptatif, nous avons trouvé les valeurs optimales de R_1, L_2 et L_4. L'algorithme a fait appel à 79 calculs par éléments finis, tandis que les AG ont demandé environ 27.000 évaluations de la fonction approchée. Le Tableau XIX présente les valeurs des paramètres obtenues à la fin de chaque itération du processus, ainsi que la valeur de la fonction objectif approchée. La géométrie résultante de cette configuration de paramètres est montrée dans la Figure 75.

TABLEAU XIX - ITÉRATIONS DE LA SURFACE DE RÉPONSE ADAPTATIVE PAR RAPPORT À L'OPTIMUM

Itération	Évaluations	R_1	L_2	L_4	f_{sr}
1	27	7,0295	15,9658	11,7223	4,68E-3
2	53	7,8486	14,4157	15,9342	9,02E-3
3	79	7,4785	14,5441	15,8723	7,09E-3

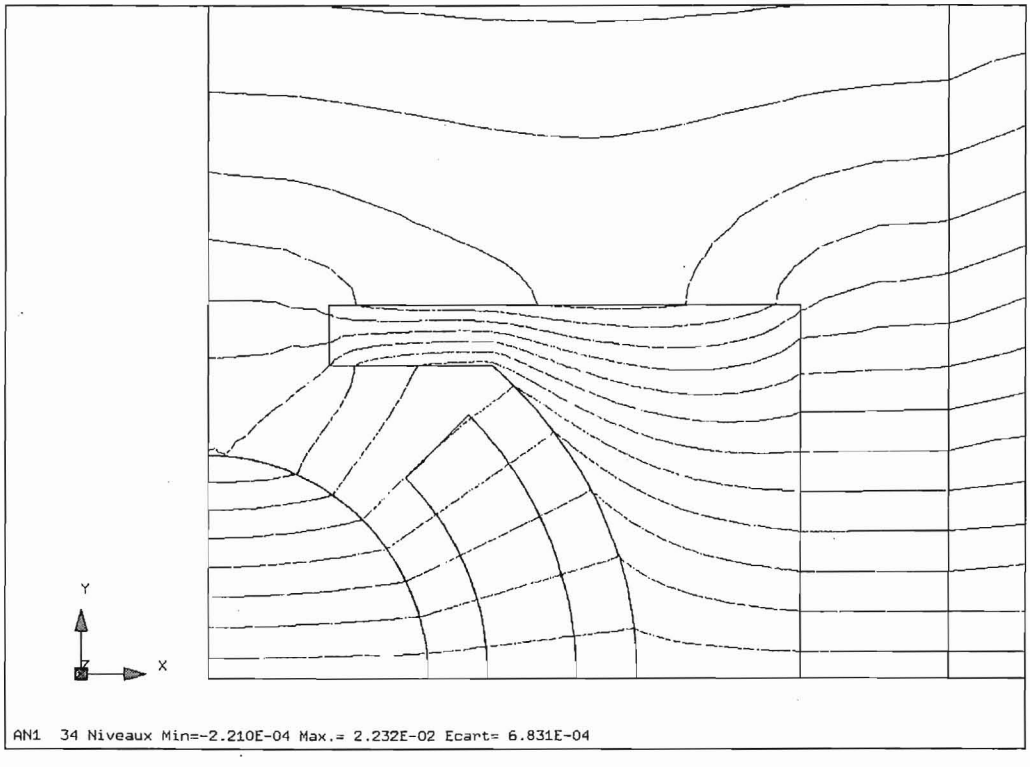


Figure 75 - Dispositif optimisé par surface de réponse adaptative par rapport à l'optimum

Pendant l'application de la méthode adaptative, les valeurs des autres paramètres du problème ont été fixées dans celles présentées sur le Tableau XVII.

IV.2.2.3 - Comparaison des Résultats

Dans cette dernière étape de notre analyse, nous avons effectué la comparaison des résultats obtenus dans les approches d'optimisation par surface de réponse avec ceux obtenus par l'application directe d'un algorithme génétique sur la fonction objectif réelle [Alotto 1998].

Le Tableau XX présente les valeurs des paramètres R_1, L_2, L_3 et L_4 obtenues dans les trois différentes approches, ainsi que la valeur de la fonction objectif et le nombre d'évaluations correspondants.

TABLEAU XX - COMPARAISON DES RÉSULTATS OBTENUS

Approche	**R_1**	**L_2**	**L_3**	**L_4**	**f_{mef}**	**Évaluations**
Directe	7,2877	13,9972	14,0	14,6700	8,92E-4	2360
SR Régulière	7,1675	14,0804	14,0	14,3550	2,15E-4	343
SR Adaptative Qualité	8,1039	16,0764	14,0	15,7306	7,44E-4	79
SR Adaptative Optimum	7,4785	14,5441	14,0	15,8723	1,43E-3	79

Comme nous pouvons observer, la meilleure configuration de paramètres a été obtenue par l'application de la surface de réponse régulière, avec un nombre acceptable d'appels à la fonction objectif.

Les valeurs de la méthode directe sont restées proches de celles de la surface de réponse régulière. Cependant, le nombre d'évaluations de la fonction objectif s'est présenté beaucoup plus important.

Les configurations de paramètres obtenues par l'application de la surface de réponse adaptative ne sont pas resté assez proche des autres et elles semble représenter un minimum local de la fonction objectif, contrairement à ce que nous attendions. Cela peut être justifier par l'allure de la fonction approchée qui semble avoir plusieurs "plateaux", comme nous le vérifions sur la Figure 71.

Malgré ce détail important concernant la différence entre les résultats, nous pouvons dire que les méthodes adaptatives apportent quelques avantages, vu le nombre réduit d'évaluations de la fonction objectif qu'elles ont utilisé.

IV.3 - Optimisation d'un Moteur à Réluctance Variable

IV.3.1 - Description du Problème

Le but de cette application est d'optimiser la géométrie d'un moteur à réluctance variable de façon à maximiser le couple magnétique exercé par la machine lorsque le rotor est dans la position illustrée par la Figure 76.

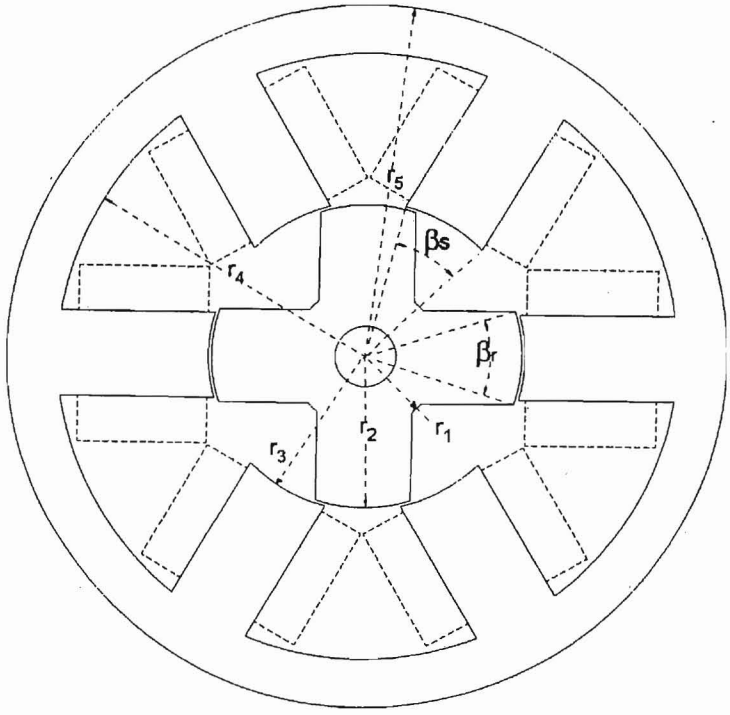


Figure 76 - Géométrie du moteur à réluctance variable

Ce problème peut être résolu par la méthode des éléments finis, en utilisant une formulation magnétostatique bidimensionnelle non-linéaire. Une fois encore, nous avons utilisé le module de résolution en deux dimensions du logiciel d'éléments finis Flux3D® pour analyser ce moteur.

IV.3.1.1 - Paramètres à optimiser

La géométrie de la machine est décrite par 7 différents paramètres géométriques (r_1, r_2, r_3, r_4, r_5, β_r et β_s) indiqués dans la Figure 76, parmi lesquels nous n'allons considérer que 6, car r_3 dépend de r_2. L'entrefer du moteur est constant et de valeur égale à 0,25mm.

Les limites de variation de ces paramètres ont été choisies de façon à éviter une configuration pour laquelle la géométrie de la machine ne soit pas réalisable. Ces limites, ainsi que les valeurs utilisées dans le projet original du moteur [DePaula 2000], sont montrées dans le Tableau XXI.

TABLEAU XXI - LIMITES DE VARIATION DES PARAMÈTRES

Paramètre	Valeur Minimale	Valeur Maximale	Valeur Initiale
r_1	8,0 mm	18,0 mm	12,0 mm
r_2	20,0 mm	35,0 mm	25,0 mm
r_4	46,0 mm	53,0 mm	50,0 mm
r_5	58,0 mm	65,0 mm	58,0 mm
β_r	0,40 rad	0,90 rad	0,6283 rad
βs	0,40 rad	0,72 rad	0,5585 rad

IV.3.1.2 - Fonction Objectif

Dans la mesure où nous cherchons à maximiser le couple magnétique, le problème d'optimisation peut être décrit par la maximisation de la fonction objectif f donnée par l'équation suivante.

$$f = T, \tag{66}$$

où

T est le couple magnétique [N.m] exercé par le moteur lorsque le rotor se trouve dans la position souhaitée.

IV.3.2 - Résolution du Problème

Comme dans l'application précédente, nous avons effectué l'analyse de ce problème en trois différentes étapes.

IV.3.2.1 - Identification des Paramètres Significatifs

Dans cette première étape, nous avons appliqué la méthode des Plans d'Expériences dans le but d'identifier les paramètres qui ont une influence significative sur la valeur de la fonction objectif *f*.

- *Identification à partir d'un Plan Factoriel Fractionnaire*

Tout d'abord, nous avons utilisé un plan fractionnaire constitué par 16 expériences basé sur le *Tableau L_{16} de Taguchi* [Pillet 1997] de façon à limiter le nombre d'évaluations de la fonction objectif.

Le Tableau XXII présente les valeurs des paramètres utilisées dans chaque expérience réalisée, ainsi que la valeur de l'évaluation correspondante issue du calcul par éléments finis.

TABLEAU XXII - RÉSULTATS DES EXPÉRIENCES DU PLAN FACTORIEL FRACTIONNAIRE

r_1	r_2	r_4	r_5	β_r	β_s	**f (N.m)**
8,0	20,0	46,0	58,0	0,40	0,40	0,376
8,0	20,0	46,0	65,0	0,40	0,72	2,148
8,0	20,0	53,0	58,0	0,90	0,72	1,442
8,0	20,0	53,0	65,0	0,90	0,40	2,089
8,0	35,0	46,0	58,0	0,90	0,72	2,838
8,0	35,0	46,0	65,0	0,90	0,40	3,805
8,0	35,0	53,0	58,0	0,40	0,40	0,334
8,0	35,0	53,0	65,0	0,40	0,72	3,890
18,0	20,0	46,0	58,0	0,90	0,40	1,576
18,0	20,0	46,0	65,0	0,90	0,72	1,742
18,0	20,0	53,0	58,0	0,40	0,72	1,687
18,0	20,0	53,0	65,0	0,40	0,40	0,488
18,0	35,0	46,0	58,0	0,40	0,72	3,987
18,0	35,0	46,0	65,0	0,40	0,40	0,392
18,0	35,0	53,0	58,0	0,90	0,40	3,629
18,0	35,0	53,0	65,0	0,90	0,72	3,723

Les contributions des contrastes significatifs (calculées à partir des équations des sections III.3.1.4 et III.3.1.5) ainsi que les confusions qui constituent ces contrastes, sont présentées dans le Tableau XXIII. Les contributions de tous les autres contrastes ont présenté une valeur au-dessous de 5%, ce qui nous a permis de les considérer comme étant négligeables.

TABLEAU XXIII - CONTRIBUTIONS OBTENUES PAR L'APPLICATION D'UN PLAN FRACTIONNAIRE

Contraste	Confusions	Contribution (%)
A	$\mathbf{r_2} + r_1r_4\beta_r + r_4r_5\beta_s + (*)$	27,47
B	$\beta_r + r_1r_2r_4 + r_1r_5\beta_s + (*)$	12,79
C	$\beta_s + r_2r_4r_5 + r_1r_5\beta_r + (*)$	17,29
D	$\beta_r\beta_s + r_1r_5 + r_1r_2r_4\beta_s + (*)$	29,62
E	$\mathbf{r_2\beta_r\beta_s} + r_1r_2r_5 + r_4r_5\beta_r + r_1r_4\beta_s$ (*)	5,12

(*) quelques interactions d'ordre supérieur.

En analysant ces résultats et en prenant en compte les considérations qui ont été évoquées dans la section III.3.1.5, nous pouvons conclure que seulement les paramètres r_2, β_r et β_s sont significatifs pour la valeur de la fonction objectif. En plus, nous pouvons dire qu'il existe une forte interaction entre les paramètres β_r et β_s.

- *Validation à partir d'un Plan Factoriel Complet*

Une fois encore, nous avons utilisé le plan factoriel complet pour valider les résultats d'analyse du plan fractionnaire. Il faut rappeler que cette étape de notre analyse ne fait normalement pas partie de la démarche que nous proposons dans ce travail, et que nous ne l'avons effectué que pour valider les considérations de la section III.3.1.5.

Après avoir réalisé les 64 (2^6) expériences demandées par le plan complet et avoir calculé les contributions des facteurs et de leurs interactions à la réponse du modèle, nous avons obtenu les résultats qui sont montrés dans le Tableau XXIV.

TABLEAU XXIV - CONTRIBUTIONS OBTENUES PAR L'APPLICATION DU PLAN FACTORIEL COMPLET

Facteur ou Interaction	Contribution (%)
r_1	0,01
r_2	21,52
r_4	0,94
r_5	0,80
β_r	9,29
β_s	12,78
$\beta_r\beta_s$	35,79
$r_2\beta_r\beta_s$	7,62
...	< 5,0

Les résultats montrent que seulement les facteurs r_2, β_r et β_s font partie des contributions supérieures à 5%, ce qui confirme les conclusions qui ont été prises par l'application du plan fractionnaire.

IV.3.2.2 - Optimisation des Paramètres

Après avoir identifié les paramètres influents sur la valeur du couple magnétique exercé par le moteur, nous avons effectué l'étape d'optimisation de ces paramètres. Pour cela, nous avons utilisé la méthodologie d'optimisation présentée dans la section III.2, basée sur la création d'une surface de réponse.

- *Optimisation à partir d'une Surface de Réponse Régulière*

Nous avons commencé par l'utilisation d'une surface de réponse régulière construite à partir de la variation des 3 paramètres significatifs r_2, β_r et β_s sur une grille régulière constituée par 9 points par direction auxquels nous avons associé des éléments diffus de premier ordre.

La construction de cette surface de réponse a exigé alors 729 ($=9^3$) calculs par éléments finis. En ce qui concerne les autres paramètres, nous avons décidé d'utiliser leurs valeurs originales présentées dans le Tableau XXI.

La Figure 77 présente une coupe de la surface de réponse obtenue pour la valeur maximale de r_2 (35,0 mm). Dans cette représentation, nous pouvons visualiser la forte interaction existante entre les paramètres β_r et β_s.

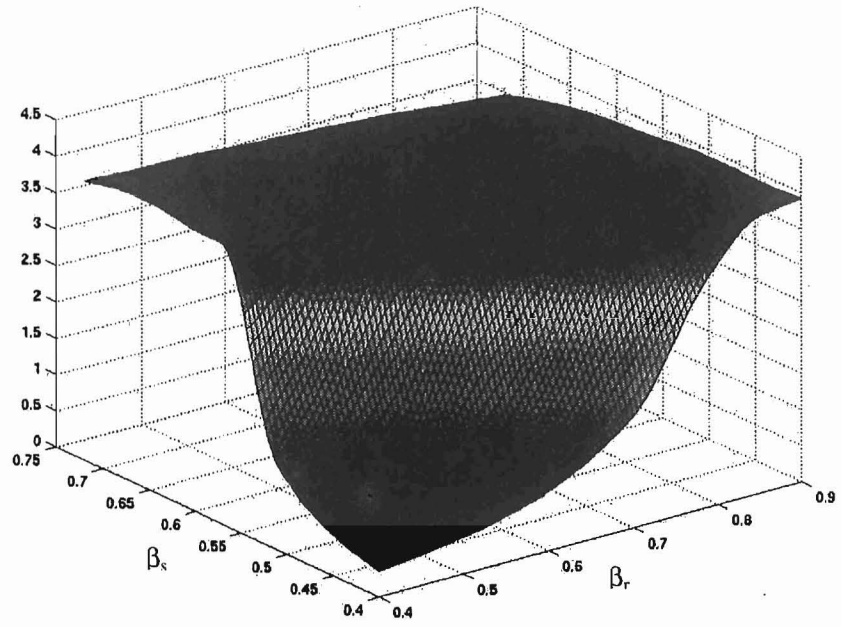


Figure 77 - Surface de réponse régulière du couple magnétique

Nous avons ensuite appliqué sur cette surface de réponse une méthode de résolution basée sur les algorithmes génétiques, avec une configuration de 30 individus et 300 générations.

Le Tableau XXV présente les valeurs obtenues pour les paramètres r_2, β_r et β_s à la fin de l'optimisation et la valeur de la fonction objectif évalué à partir de la surface de réponse (f_{sr}) et par calcul éléments finis (f_{mef}). La Figure 78 nous montre la géométrie du moteur correspondante à ces valeurs.

TABLEAU XXV - VALEURS DES PARAMÈTRES OPTIMISÉS PAR SURFACE DE RÉPONSE RÉGULIÈRE

Paramètre	**Valeur**
r_2	35,0
β_r	0,6687
β_s	0,4407
f_{sr}	4,0583
f_{mef}	3,9266

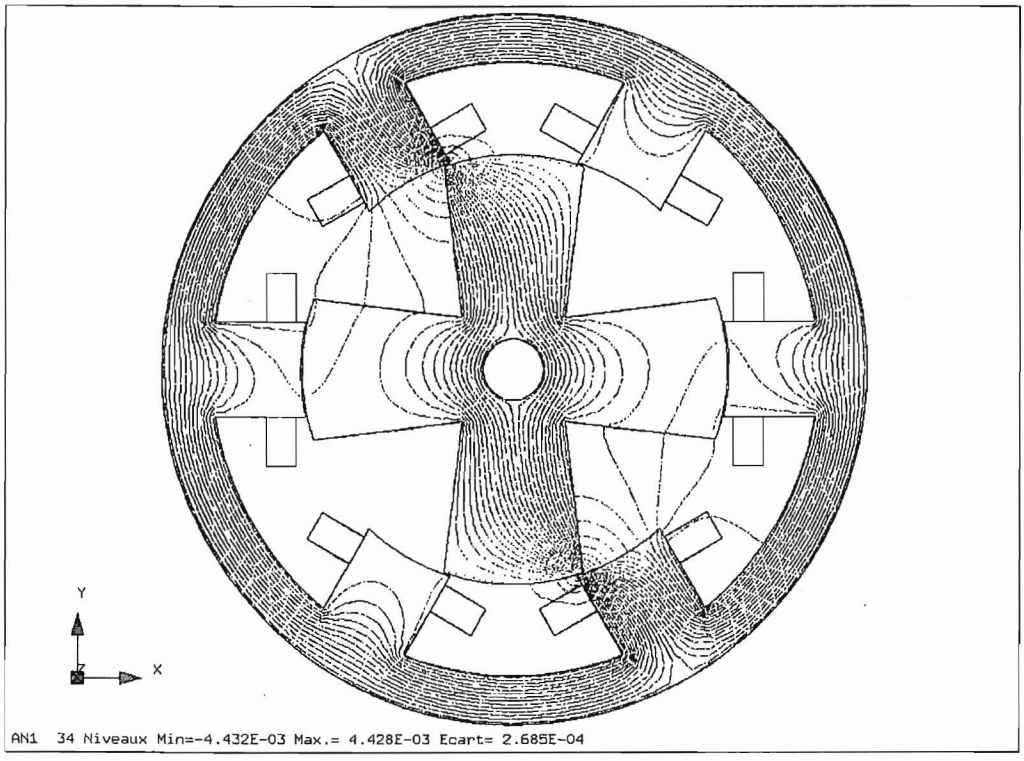


Figure 78 - Moteur optimisé par surface de réponse régulière

- *Optimisation à partir d'une Surface de Réponse Adaptative*

- Adaptativité par rapport à la qualité de l'approximation

Dans une deuxième étape de notre analyse, nous avons utilisé la méthodologie de surface de réponse adaptative par rapport à la qualité de l'approximation présentée dans la section III.2.2.1 pour optimiser les paramètres r_2, β_r et β_s.

L'algorithme adaptatif a fait appel à 79 calculs par éléments finis pour construire la surface de réponse. La Figure 79 présente une coupe de la surface de réponse obtenue pour la valeur maximale de r_2, où nous pouvons percevoir une similarité avec celle obtenue par l'approximation régulière (Figure 77).

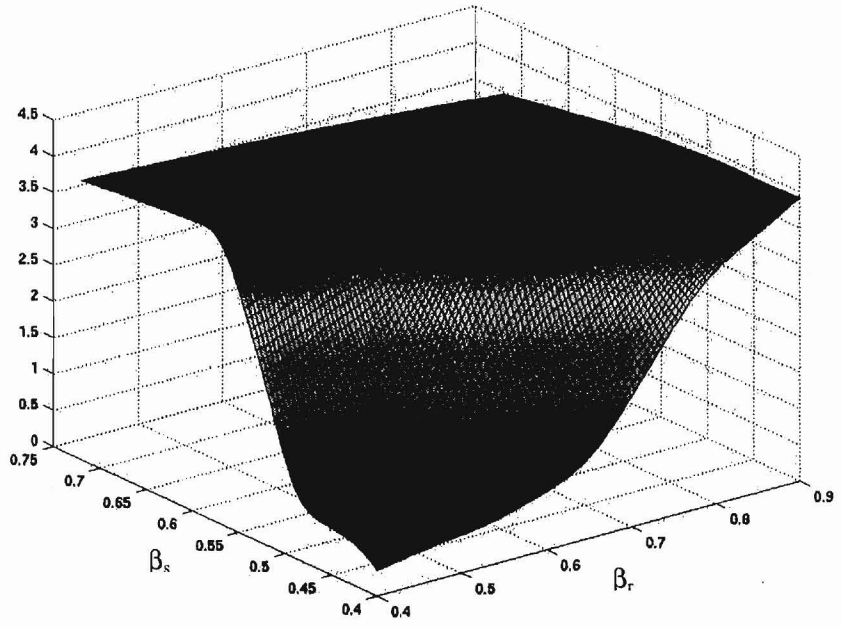


Figure 79 - Surface de réponse adaptative du couple magnétique

Ensuite, nous avons appliqué une méthode de résolution stochastique du type algorithme génétique avec une configuration de 30 individus et 300 générations pour identifier le point d'optimisation global donné par cette surface de réponse.

Les valeurs des paramètres obtenues à la fin de l'optimisation sont présentées dans le Tableau XXVI, tandis que la Figure 80 présente la géométrie du moteur correspondante à ces valeurs.

TABLEAU XXVI - VALEURS DES PARAMÈTRES OPTIMISÉS PAR SURFACE DE RÉPONSE ADAPTATIVE

Paramètre	**Valeur**
r_2	35,0
β_r	0,6753
β_s	0,4553
f_{sr}	4,0699
f_{mef}	3,9449

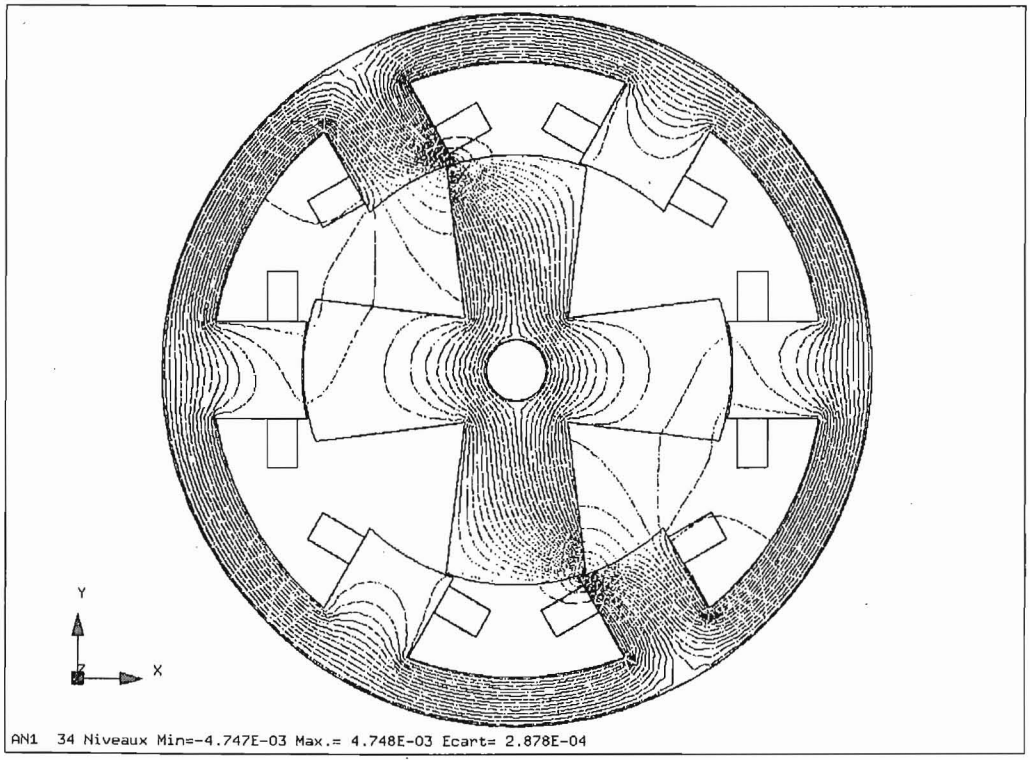


Figure 80 - Moteur optimisé par surface de réponse adaptative par rapport à la qualité

- Adaptativité par rapport à l'optimum

Pour finaliser notre analyse, nous avons utilisé la méthodologie de surface de réponse adaptative par rapport à l'optimum présentée dans la section III.2.2.2 pour optimiser les paramètres r_2, β_r et β_s.

Nous avons appliqué une méthode de résolution stochastique du type algorithme génétique avec une configuration de 30 individus et 300 générations pour identifier le point d'optimisation à chaque itération de l'algorithme adaptatif.

Après 3 itérations du processus adaptatif, nous avons trouvé les valeurs optimales r_2, β_r et β_s. L'algorithme a fait appel à 61 calculs par éléments finis, tandis que les AG ont demandé environ 27000 évaluations de la fonction approchée. Le Tableau XXVII présente les valeurs des paramètres obtenues à la fin de chaque itération du processus, ainsi que la valeur de la fonction objectif approchée. La géométrie résultante de cette configuration de paramètres est montrée dans la Figure 81.

TABLEAU XXVII - ITÉRATIONS DE LA SURFACE DE RÉPONSE ADAPTATIVE PAR RAPPORT À L'OPTIMUM

Itération	Évaluations	r_2	β_r	β_s	f_{sr} (N.m)
1	27	35,0	0,7024	0,5446	4,08
2	44	35,0	0,6913	0,4551	3,93
3	61	35,0	0,6704	0,4390	4,10

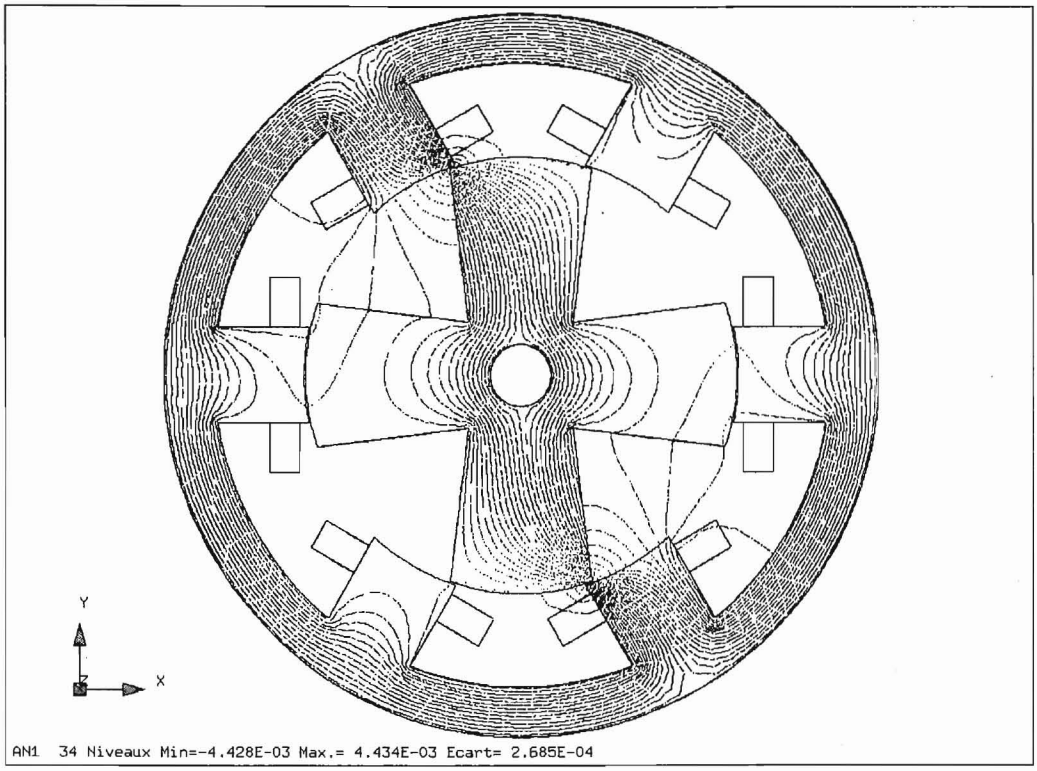


Figure 81 - Moteur optimisé par surface de réponse adaptative par rapport à l'optimum

Pendant l'application de la méthode adaptative, les valeurs des autres paramètres du problème ont été fixées dans celles présentées sur le Tableau XXI.

IV.3.2.3 - Comparaison des Résultats

Dans cette dernière étape de notre analyse, nous avons effectué la comparaison des résultats obtenus dans les différentes approches d'optimisation par surface de réponse que nous avons réalisés.

Le Tableau XXVIII présente les valeurs des paramètres r_2, β_r et β_s obtenues dans les trois différentes approches, ainsi que la valeur de la fonction objectif et le nombre d'évaluations correspondants.

TABLEAU XXVIII - COMPARAISON DES RÉSULTATS OBTENUS

Approche	r_2	β_r	β_s	f_{mef} **(N.m)**	**Évaluations**
Configuration Initiale	25,0	0,6283	0,5585	2,7634	-
SR Régulière	35,0	0,6687	0,4407	3,9266	729
SR Adaptative Qualité	35,0	0,6753	0,4553	3,9449	79
SR Adaptative Optimum	35,0	0,6704	0,4390	3,9739	61

Nous pouvons observer que les résultats sont restés très proches l'un des autres et que nous avons obtenu un gain important concernant la valeur de la fonction objectif par rapport à celle de la configuration initiale.

En ce qui concerne le nombre d'évaluations de la fonction objectif, les approches adaptatives ont montré leur grand avantage par rapport à l'approche de surface de réponse régulière, car le nombre de calculs par éléments finis demandé par cette dernière est resté environ dix fois plus important.

IV.4 - Optimisation d'un Contacteur Électromagnétique

IV.4.1 - Description du Problème

Dans cette troisième et dernière application, nous avons optimisé la forme d'un contacteur électromagnétique dans le but de maximiser la force exercée sur sa palette, en respectant, en même temps, une valeur limite définie pour son poids. Cette application a été proposée par les ingénieurs de Cedrat Recherche pour tester les outils d'optimisation implémentés dans ce travail.

La géométrie du contacteur est constituée par deux colonnes cylindriques dans lesquelles se situent les bobines inductrices, dont chaque enroulement porte 450 ampères-tours. Entre ces deux colonnes, nous avons une colonne rectangulaire sur laquelle nous trouvons un aimant permanent d'induction rémanente égale à 0,85 Tesla.

Les différentes parties qui composent le contacteur sont présentées sur la Figure 82. Nous pouvons vérifier que la géométrie présente une symétrie dans le plan ZOX, ce qui nous permet de n'utiliser que la moitié du dispositif lors de son analyse [Cedrat 1999].

Nous avons alors utilisé la géométrie représentée dans la Figure 83, où nous pouvons observer aussi le maillage en éléments finis sur les faces du dispositif. Ce problème a été analysé par le logiciel éléments finis Flux3D® avec une formulation magnétostatique tridimensionnelle non-linéaire.

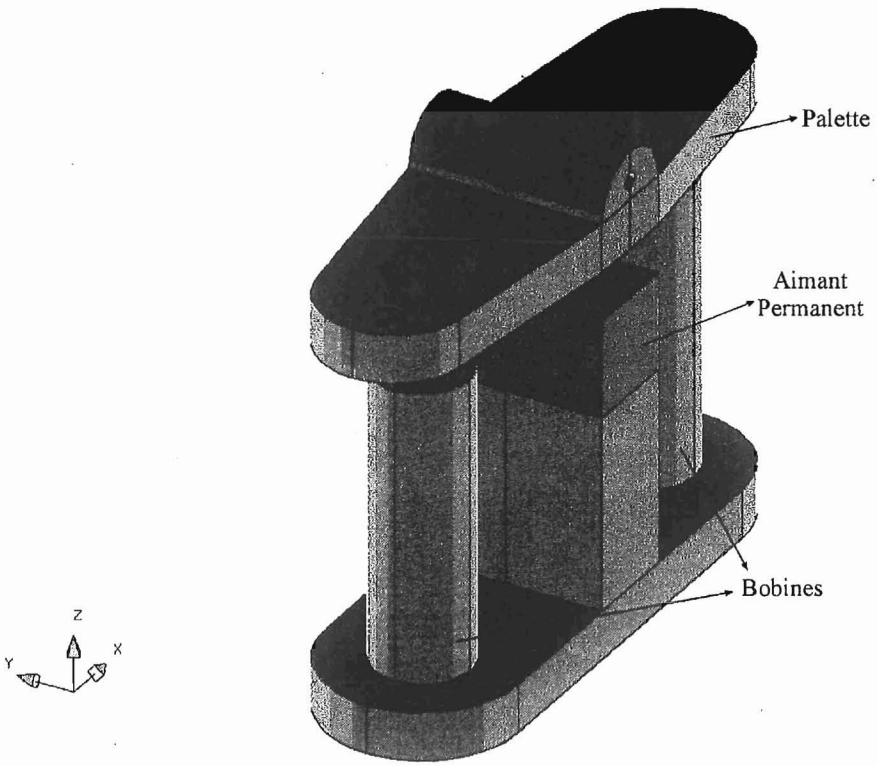


Figure 82 - Géométrie complète du contacteur

Les simulations par éléments finis en trois dimensions demandent souvent des temps de calcul assez importants. Dans le cas particulier de cette application, une simulation correspond à environ 10 minutes de résolution sur un Pentium® III - 1.0 GHz.

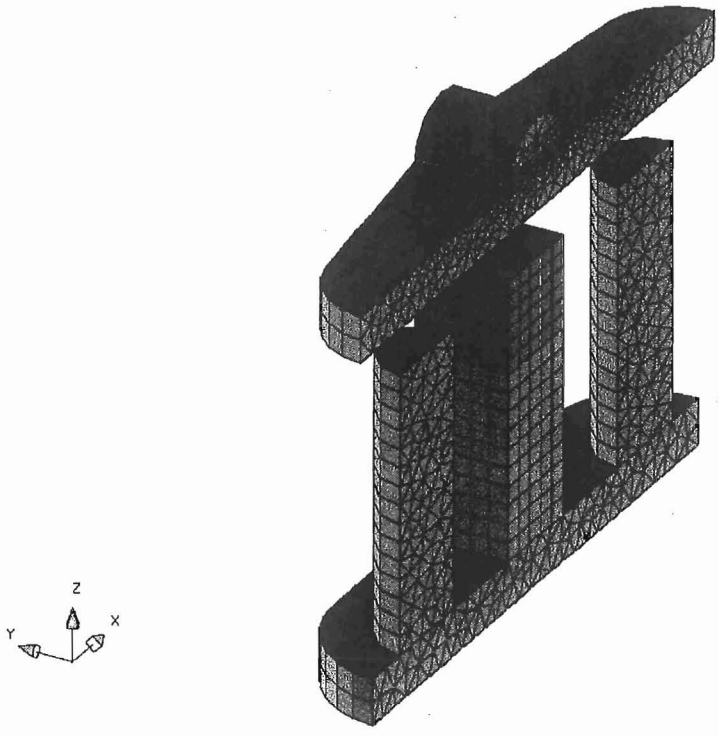


Figure 83 - Géométrie utilisée dans la simulation

IV.4.1.1 - Paramètres à optimiser

La géométrie du contacteur est décrite par deux paramètres *R* et *H* choisis de façon à modifier la forme des colonnes cylindriques où se situent les bobines d'alimentation et aussi de la colonne rectangulaire sur laquelle l'aimant permanent est placé.

Le paramètre *R* définit le rayon de chaque colonne cylindrique, tandis que le paramètre *H* permet de définir leur hauteur et celle de la colonne rectangulaire. La Figure 84 montre la localisation de ces deux paramètres, tandis que leurs limites de variation sont présentées dans le Tableau XXIX.

TABLEAU XXIX - LIMITES DE VARIATION DES PARAMÈTRES

Paramètre	Valeur Minimale (mm)	Valeur Maximale (mm)	Valeur Initiale (mm)
R	15,0	35,0	20,0
H	40,0	100,0	60,0

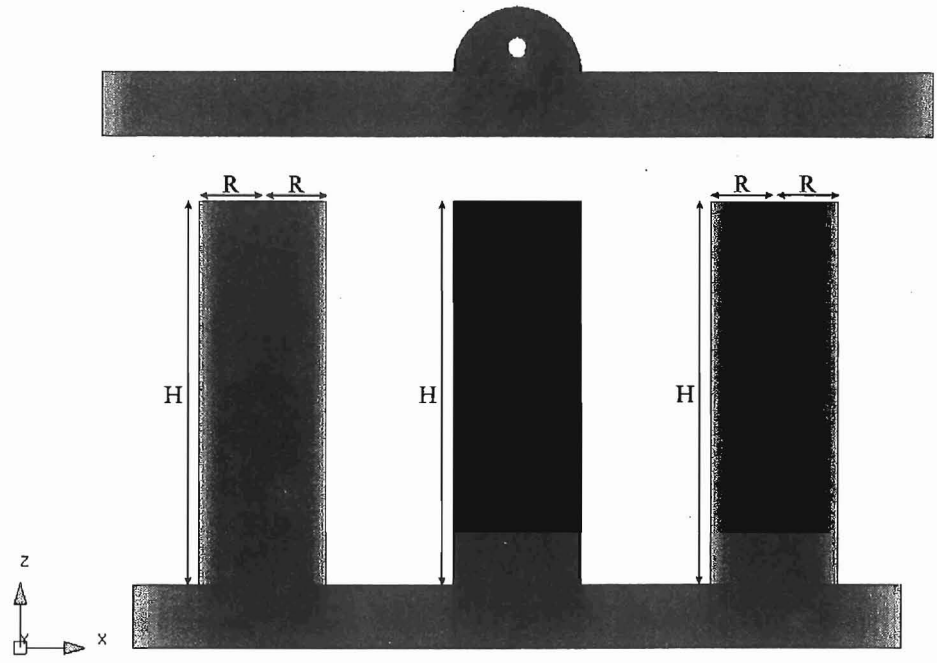


Figure 84 - Paramètres à optimiser

IV.4.1.2 - Fonction Objectif

L'objectif de notre application est d'obtenir une force maximale sur la palette du contacteur, sans dépasser la valeur limite définie pour le poids du dispositif. Ce problème représente alors un problème d'optimisation contraint dont la fonction objectif et la fonction contrainte sont celles données par l'équation (67).

$$\begin{cases} f(R,H) = F \\ g(R,H) = P \leq 1{,}20 \end{cases} \tag{67}$$

où

F est la force [N] exercée sur la palette du contacteur

P est le poids [kg] du dispositif.

IV.4.2 - Résolution du Problème

À la différence des autres applications présentées dans ce chapitre, nous n'avons pas utilisé la méthode des Plans d'Expériences pour identifier les paramètres qui ont une influence significative sur la valeur de la fonction objectif *f*, car leur nombre est déjà restreint. Nous avons donc résolu ce problème en effectuant directement l'optimisation des paramètres.

IV.4.2.1 - Optimisation des Paramètres

- *Construction de la Surface de Réponse*

Pour optimiser les paramètres R et H, nous avons utilisé la méthodologie d'optimisation présentée dans la section III.2, basée sur la création d'une surface de réponse construite à partir de la variation de ces deux paramètres sur une grille régulière constituée par 9 points par direction auxquels nous avons associé des éléments diffus de premier ordre.

- Surface de Réponse de la Force

La Figure 85 présente la surface de réponse obtenue pour la force exercée sur la palette du contacteur par rapport aux paramètres R et H.

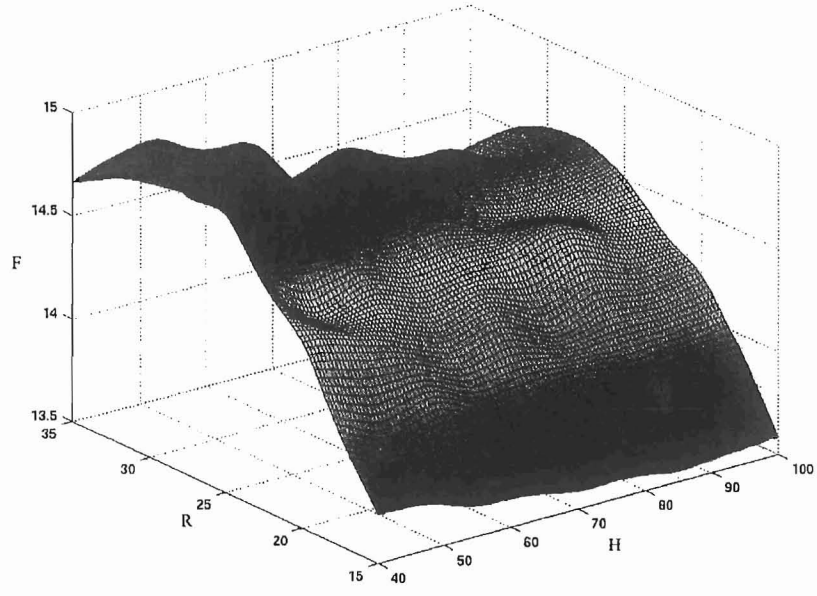


Figure 85 - Surface de réponse de la force exercée sur la palette du contacteur

Nous pouvons observer sur cette surface de réponse, la présence de quelques oscillations qui semblent être de minima locaux, mais qui en réalité représentent des bruits numériques liés au calcul de la force par éléments finis.

Ces bruits numériques sont issus de la qualité du maillage éléments finis utilisé (environ 18.000 éléments tétraèdres, hexaèdres et pyramides d'ordre 2). Une augmentation du nombre

d'éléments pourrait diminuer ou même éliminer la présence de ces bruits. Cependant, cela entraînerait une forte augmentation du temps de calcul qui est déjà onéreux (environ 13,5 heures pour 81 simulations).

Nous avons alors décidé d'utiliser cette surface de réponse dans la résolution de ce problème, quoique l'approximation obtenue ne soit pas parfaite, car le principal objectif de cette application est de plutôt montrer la démarche d'utilisation de l'approche d'optimisation par surface de réponse dans la résolution d'un problème contraint. De toute façon, la solution obtenue peut apporter une information utile pour la localisation d'un optimum.

- Surface de Réponse du Poids

Il s'agit d'une optimisation sous contrainte et la valeur de la fonction contrainte dépend aussi de la simulation par éléments finis (en fait, le poids du dispositif est calculé à partir de l'intégrale de sa densité massique sur son volume). Il devient alors nécessaire de construire une deuxième surface de réponse correspondant à la variation du poids par rapport aux paramètres du problème. Cette surface de réponse, construite aussi sur une grille régulière de 9 points par direction, est représentée dans la Figure 86, et nous pouvons vérifier qu'elle est beaucoup plus lisse que celle concernant la force, car le calcul du poids ne dépend pas beaucoup de la qualité du maillage.

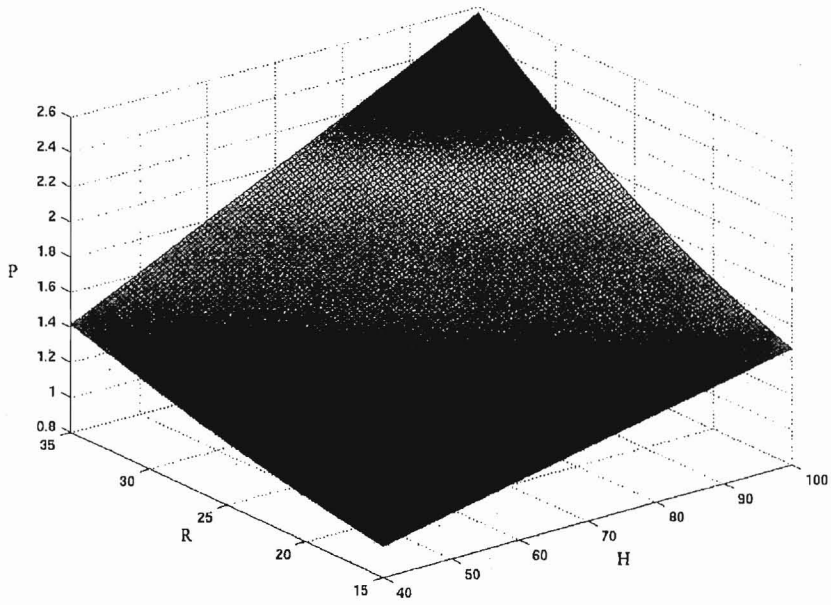


Figure 86 - Surface de réponse du poids du contacteur

Le fait d'avoir utilisé la même discrétisation pour construire les deux surfaces de réponse nous a permis d'avoir la valeur de la fonction objectif et de la fonction contrainte pour chaque point de la grille en n'effectuant qu'une seule analyse par éléments finis. De cette façon, nous n'avons effectué que 81 ($=9^2$) calculs par éléments finis pour créer les deux surfaces de réponse.

- *Application de la Méthode de Résolution*

Une fois que les surfaces de réponse ont été construites, nous avons appliqué sur elles une méthode de résolution basée sur les algorithmes génétiques, avec une population de 30 individus et 300 générations.

La valeur d'adaptation de chaque individu est donnée par la fonction Φ de l'équation (68), qui correspond à la valeur de la fonction objectif évaluée à partir de la surface de réponse de la force, augmentée d'une valeur de pénalité provenant de la surface de réponse du poids, lorsque la fonction contrainte n'est pas satisfaite.

$$\begin{cases} \Phi(R,H) = f_{sr}(R,H) + r[\max[0, g_{sr}(R,H)]]^2 \\ g_{sr}(R,H) = P - 1{,}20 \end{cases} \tag{68}$$

où

r (=1000) est le coefficient de pénalité

Le Tableau XXX présente les valeurs obtenues pour les paramètres R et H à la fin de l'optimisation, ainsi que les fonctions objectif et contrainte évaluées par les respectives surfaces de réponse (f_{sr} et g_{sr}) et par calculs éléments finis (f_{mef}, g_{mef}). Nous pouvons vérifier à partir de ces résultats que la fonction contrainte a été satisfaite.

TABLEAU XXX - VALEURS DES PARAMÈTRES OPTIMISÉS AVEC CONTRAINTE

Paramètre	Valeur
R	25,53 mm
H	42,66 mm
f_{sr}	14,872 N
f_{mef}	14,878 N
g_{sr}	1,20 kg
g_{mef}	1,20 kg

La Figure 87 et la Figure 88 montrent respectivement la visualisation en deux et en trois dimensions de la géométrie du contacteur correspondante aux valeurs optimisées.

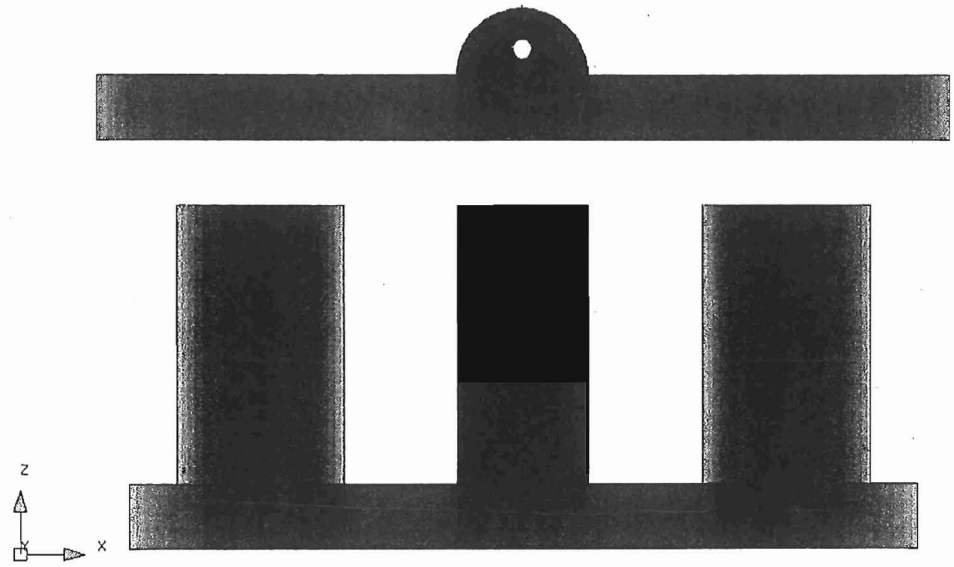


Figure 87 - Contacteur optimisé par surface de réponse - visualisation 2D

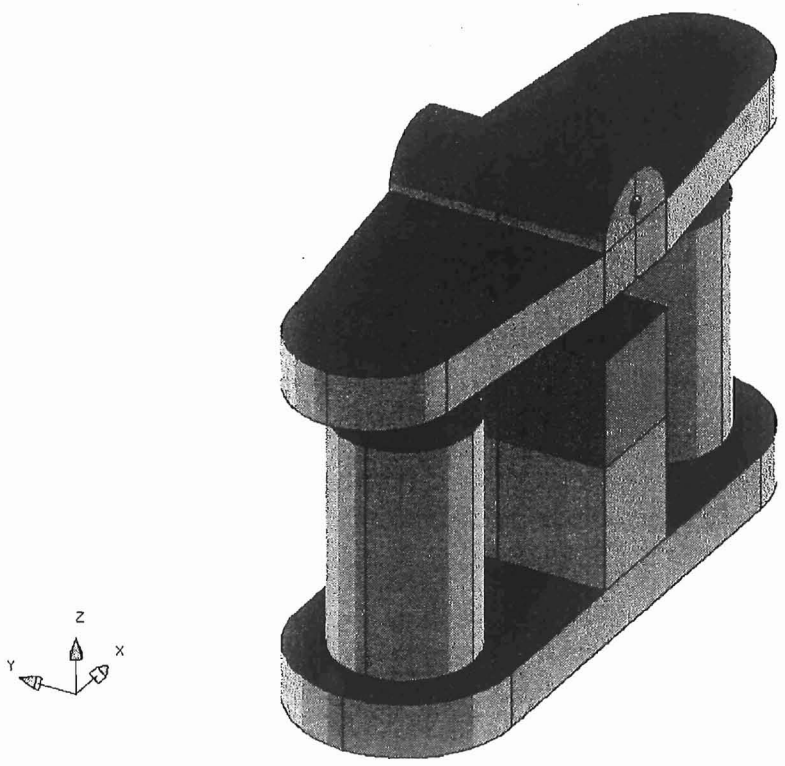


Figure 88 - Contacteur optimisé par surface de réponse - visualisation 3D

Le Tableau XXXI présente les valeurs d'évaluation par éléments finis des fonctions objectif et contrainte correspondantes à la configuration initiale des paramètres. Nous pouvons vérifier, à partir de ces valeurs, que la configuration de paramètres du Tableau XXX nous apporte un gain sur la valeur de la fonction objectif.

TABLEAU XXXI - ÉVALUATION DE LA CONFIGURATION DE PARAMÈTRES INITIAUX

Paramètre	Valeur
R	20,0 mm
H	60,0 mm
f_{mef}	14,20 N
g_{mef}	1,26 kg

- *Vérification de l'influence de la Fonction Contrainte*

Pour vérifier l'influence de la fonction contrainte sur la solution du problème, nous avons effectué l'optimisation du contacteur sans prendre en compte la fonction contrainte de l'équation (67). Cela correspond à résoudre le problème non contraint de l'équation (69).

$$f(R,H) = F \tag{69}$$

Le Tableau XXXII présente les valeurs des paramètres obtenues après l'application des algorithmes génétiques sur la surface de réponse de la force, représentée dans la Figure 85, ainsi que les valeurs des fonctions objectif et contrainte correspondantes.

Nous pouvons vérifier que les résultats sont différents de ceux du Tableau XXX et que la contrainte du problème original n'a pas été pas satisfaite, ce qui nous permet de valider l'approche d'optimisation par surface de réponse dans la résolution de problèmes contraints.

TABLEAU XXXII - VALEURS DES PARAMÈTRES OPTIMISÉS SANS CONTRAINTES

Paramètre	Valeur
R	30,18 mm
H	40,00 mm
f_{sr}	14,920 N
f_{mef}	14,922 N
g_{mef}	1,28 kg

IV.5 - Conclusion

Nous avons présenté deux différentes applications de l'approche basée sur la construction d'une surface de réponse associée à l'analyse par les plans d'expériences. Cette approche a été appliquée dans l'optimisation de problèmes non contraints en électrotechnique à plusieurs paramètres liés à la simulation par éléments finis: un moteur à réluctance variable et le problème 25 du TEAM Workshop.

Ces applications nous ont permis de valider les considérations concernant l'analyse des résultats issus de l'utilisation des plans d'expériences fractionnaires dans l'identification des paramètres influents sur la valeur de la fonction objectif. Elles ont montré aussi que nous pouvons avoir des résultats d'optimisation assez fiables avec un nombre restreint d'évaluations de la fonction objectif.

Nous avons appliqué aussi l'approche d'optimisation par surface de réponse dans l'optimisation d'un problème contraint tridimensionnel concernant la forme d'un contacteur électromagnétique. Cette application nous a permis de valider l'utilisation de cette approche dans la résolution de problèmes contraints.

Cependant, nous avons vérifié dans cette application, que la surface de réponse par éléments diffus peut, tout de même, présenter des oscillations dues aux bruits numériques de la simulation par éléments finis. L'explication la plus plausible à ces oscillations reste liée à la qualité insuffisante du maillage éléments finis que nous avons utilisé pour limiter les temps de calcul.

L'application de techniques de lissage de la surface de réponse se présente alors comme une bonne voie de recherche pour améliorer la précision de la solution, ainsi que l'application du calcul distribué pour réduire le temps de calcul et permettre l'utilisation de maillages plus fins, surtout dans des applications en trois dimensions.

Conclusion Générale

Conclusion Générale

Ce travail nous a permis de constater que l'optimisation de dispositifs électromagnétiques analysés par des outils de simulation numérique demande un traitement spécial, car les difficultés liées aux caractéristiques du dispositif, aux contraintes de l'outil de simulation et surtout aux besoins de l'utilisateur sont souvent nombreuses.

Parmi ces difficultés, nous avons vu que la diversité des problèmes traités, l'échange des données entre l'outil de simulation et l'outil d'optimisation, les erreurs de la méthode de simulation dues aux petits intervalles de discrétisation de l'espace, la recherche d'une solution globale et surtout le temps de calcul onéreux pour atteindre une solution se présentent comme les plus expressives.

Nous avons alors proposé des solutions à ces difficultés, dans le but d'implémenter un outil d'optimisation générique, performant et capable de traiter avec robustesse les problèmes d'optimisation liés à la simulation numérique.

L'utilisation des concepts de la Programmation Orientée Objet pour implémenter l'architecture logiciel de cet outil d'optimisation nous a apporté des solutions aux difficultés concernant, entre autres, la diversité des problèmes d'optimisation, l'échange des données entre l'outil de simulation et l'outil d'optimisation et aussi la grande variété de méthodes d'optimisation existantes.

Une étude des propriétés des problèmes d'optimisation et des caractéristiques des méthodes d'optimisation les plus classiques était indispensable au développement de cette architecture et le résultat de cette étude a été reproduit dans le premier chapitre.

En même temps, la nouvelle approche d'optimisation basée sur la construction d'une surface de réponse en utilisant la Méthode des Éléments Diffus et la Méthode des Plans d'Expériences nous a permis d'avoir des solutions aux difficultés concernant la recherche d'une solution globale, les erreurs de la méthode de simulation dues aux petits intervalles de discrétisation dans l'espace, le gradient non accessible de la fonction objectif et le temps de calcul onéreux.

Nous avons constaté que la Méthode des Éléments Diffus présente plusieurs avantages dans la construction de la surface de réponse, parmi lesquels nous avons mis en évidence sa facilité de mise en œuvre et sa capacité de fournir des surfaces de réponses adaptatives.

Nous avons aussi montré que la Méthode des Plans d'Expériences peut nous orienter dans la construction de la surface de réponse en fournissant des informations précieuses concernant l'influence des paramètres sur la valeur de la fonction objectif, à partir d'un nombre restreint d'évaluations de la fonction.

Enfin, les applications en électrotechnique présentées dans le dernier chapitre nous ont permis de vérifier la performance de cette nouvelle approche dans la résolution de problèmes contraints et à plusieurs paramètres.

Il reste encore de nombreuses recherches à faire dans le domaine de l'optimisation liée à la simulation par éléments finis. Le traitement des problèmes à objectifs multiples, l'étude de nouvelles méthodes et de nouvelles techniques de construction de surfaces de réponse, l'application de méthodes d'optimisation hybrides (stochastiques + déterministes), le traitement de paramètres discrets et l'utilisation du calcul distribué pour diminuer encore plus le temps de calcul, représentent des voies prometteuses pour l'avenir de l'optimisation et de la Conception Assistée par Ordinateur.

Bibliographie

Bibliographie

[Abakar 1998] A. Abakar, J-L. Coulomb, Y. Maréchal, "*Radial Basis Function Network for Acceleration of Genetic Algorithm Optimization Procedures*", Proceedings of 9th IGTE, Symposium on Numerical Field Calculation in Electrical Engineering, Graz, Austria, 1998.

[Alotto 1996] P. Alotto, A. Caiti, G. Molinari, M. Repetto, "*A Multiquadrics-Based Algorithm for the Acceleration of Simulated Annealing Optimization Procedures*", IEEE Transactions on Magnetics, vol. 32, n. 3, pp. 1198-1201, 1996.

[Alotto 1998] P. Alotto, C. Eranda, B. Brandstätter, G. Fürntratt, C. Magele, G. Molinari, M. Nervi, K. Preis, M. Repetto, K. R. Richter, "*Stochastic Algorithms in Electromagnetic Optimization*", IEEE Transactions on Magnetics, vol. 34, n. 5, pp. 3674-3684, 1998.

[Baker 1985] J. E. Baker, "*Adaptive Selection Methods for Genetic Algorithms*", Proceedings of ICGA85 - 1st International Conference on Genetic Algorithms and Their Applications, New Jersey, pp. 101-111, 1985.

[Baker 1987] J. E. Baker, "*Reducing Bias and Inefficiency in the Selection Algorithm*", Proceedings of ICGA87 - 2nd International Conference on Genetic Algorithms and Their Applications, New Jersey, pp. 14-21, 1987.

[Brent 1973] R. P. Brent, "*Algorithms for Minimization Without Derivatives*", Prentice-Hall, 1973.

[Buzzi-Ferraris 1993] G. Buzzi-Ferraris, "Scientific C++: Building Numerical Libraries the Object-Oriented Way", Addison-Wesley Publishing Company, 1993.

[Cedrat 1999] Cedrat Electrical Engineering, "Flux3D® version 3.1 - Magnetostatics Tutorial", 1999.

[Caroll 1961] C. W. Caroll, "*The Created Response Surface Technique for Optimizating Nonlinear Restrained Systems*", Operational Research, n. 9, pp. 169-184, 1961.

[Cherruault 1999] Y. Cherruault, "*Optimisation: Méthodes Locales et Globales*", Presses Universitaires de France, ISBN 2-130-49910-4, 1999.

[Ciarlet 1982] P. G. Ciarlet, "*Introduction à l'Analyse Matricielle et à l'Optimisation*", Masson, ISBN 2-225-68893-1, 1982.

[Costa 2001a] M. C. Costa, S. Giurgea, J-L. Coulomb, Y. Maréchal, A. B. Dietrich, S. I. Nabeta, "*Application of Experimental Design Method in the Screening of Electromagnetic Devices Parameters*", à apparaitre dans les proceedings de la 13th Conference on the Computation of Electromagnetic Fields - COMPUMAG 2001, Evian, France, julliet 2001.

[Costa 2001b] M. C. Costa, J-L. Coulomb, Y. Maréchal, "*Parameters Screening of TEAM Workshop Problem 25 by the Application of Experimental Design Method*", à apparaître dans les proceedings du XI International Symposium on Theoretical Electrical Engineering - ISTET 2001, Linz, Autriche, août, 2001.

[Costa 2001c] M. C. Costa, J-L. Coulomb, Y. Maréchal, S. I. Nabeta, "*An Adaptive Method Applied to the Diffuse Element Approximation in Optimization Process*", à apparaître dans IEEE Transactions on Magnetics, septembre 2001.

[Coulomb 1981] J-L. Coulomb, "*Analyse Tridimensionnelle des Champs Électriques et Magnétiques par la Méthode des Éléments Finis*", Thèse de Doctorat, Institut National Polytechnique de Grenoble, 1981.

[Culioli 1994] J. C. Culioli, "*Introduction à l'Optimisation*", Ellipses, ISBN 2-729-89428-4, 1994.

[Davis 1989] L. Davis, "*Adapting Operator Probabilities in Genetic Algorithms*", Proceedings of ICGA89 - 3rd International Conference on Genetic Algorithms and Their Applications, pp. 61-69, 1989.

[DeJong 1975] K. A. De Jong, "*An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*", Doctorat, University of Michigan, 1975.

[DePaula 2000] P. P. De Paula, "*Aspectos de Projeto e Operação de Geradores e Motores de Relutância Chaveados*", Thèse de Doctorat, Université de São Paulo, Brésil, 2000.

[Demonsant 1996] J. Demonsant, "*Comprendre et Mener des Plans d'Expériences*", Afnor, ISBN 2-124-75032-1, 1996.

[Dennis 1987] J. E. Dennis, R. B. Schnabel, "*Numerical Methods for Unconstrained Optimization and Nonlinear Equations*", Prentice-Hall, ISBN 0-898-71364-1, 1987.

[Duarte 1995] C. A. Duarte, "*A Review of Some Meshless Methods to Solve Partial Differential Equations*", Texas Institute for Computational and Applied Mathematics, rapport 95-06, 1995.

[Fiacco 1968] A. V. Fiacco, G. P McCormick, "*Nonlinear Programming Sequential Unconstrained Minimization Techniques*", John Wiley, New York, 1968.

[Fletcher 1964] R. Fletcher, C. M. Reeves, "*Function minimization by conjugate gradients*", Computer Journal, n. 7, pp. 148-154, 1964.

[Fletcher 1987] R. Fletcher, "*Practical Methods of Optimization*", John Wiley & Sons, ISBN 0-471-49463-1, 1987.

[Fleury 1986] C. Fleury, "*Structural Optimization: A New Dual Method Using Mixed Variables*", International Journal on Numerical Methods in Engineering, vol. 23, pp. 409-428, 1986.

[Fogel 1994] L. J. Fogel, "*Evolutionary Programming in Perspective: The Top-Down View*", in [Zurada 1994], pp. 135-146, 1994.

[Gen 1997] M. Gen, R. Cheng, "*Genetic Algorithms and Engineering Design*", John Wiley & Sons, ISBN 0-471-12741-8, 1997.

[Gillon 1997] F. Gillon, "*Modélisation et Optimisation par Plans d'Expériences d'un Moteur à Commutations Électroniques*", Thèse de Doctorat, Université de Lille I, 1997.

[Gillon 2000] F. Gillon, P. Brochet, "*Screening and Response Surface Method applied to the Numerical Optimisation of Electromagnetic Devices*", IEEE Transactions on Magnetics, vol. 36, n. 4, pp. 1163-1166, 2000.

[Giurgea 2000] S. Giurgea, "*Le Plan d'Expériences Utilisé dans l'Optimisation de Dispositifs Électromagnétiques*", Rapport de Stage, Institut National Polytechnique de Grenoble, 2000.

[Glover 1989] F. Glover, "*Tabu Search - Part I*", ORSA Journal on Computing, vol. 1, n. 3, pp. 190-206, 1989.

[Glover 1990] F. Glover, "*Tabu Search - Part II*", ORSA Journal on Computing, vol. 2, n. 1, pp. 4-32, 1990.

[Goldberg 1989] D. E. Goldberg, "*Genetic Algorithms in Search, Optimization and Machine Learning*", Addison Wesley, ISBN 0-201-15767-5, 1989.

[Han 1977] S. P. Han, "*A Globally Convergent Method for Nonlinear Programming*", JOTA, vol. 22, n. 3, pp. 297-309, 1977.

[Hérault 1999] C. Hérault, Y. Maréchal, JL. Coulomb, "*A Meshless Approximation for the Acceleration of Stochastic Algorithm in Optimization Procedure*", Proceedings of COMPUMAG 99, pp. 174-175, 1999.

[Hérault 2000] C. Hérault, "*Vers une Simulation sans Maillage des Phénomènes Électromagnétiques*", Thèse de Doctorat, Institut National Polytechnique de Grenoble, 2000.

[Hestenes 1969] M. R. Hestenes, "*Multiplier and Gradient Methods*", Journal of Optimization Theory and Applications, n. 4, pp. 303-320, 1969.

[Holland 1975] J. H. Holland, "*Adaptation in Natural and Artificial System*", The University of Michigan Press, ISBN 0-472-08460-7, 1975.

[Holzner 1993] S. Holzner, "*C Programming*", Ed. Campus, 1993.

[Horstmann 1999] C. S. Horstmann, G. Cornell, "*Core Java 2: Volume 1 – Fundamentals*", Prentice Hall PTR/Sun Microsystems Press, ISBN 0-130-81933-6, 1999.

[Hu 1992] N. Hu, "*Tabu Search Method with random moves for globally optimal design*", International Journal for Numerical Methods in Engineering, vol. 35, n. 5, pp. 1055-1070, 1992.

[Janikow 1991] Z. Janikow, Z. Michalewicz, "*An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms*", Proceedings of ICGA91 - 4th International Conference on Genetic Algorithms and Their Applications, pp. 31-36, 1991.

[Kadded 1993] K. Kadded, "*Optimisation de forme de machines électriques à l'aide d'un logiciel éléments finis et de la méthode des pénalités intérieurs étendues*", Thèse de Doctorat, Institut National Polytechnique de Grenoble, 1993.

[Kirkpatrick 1983] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, "*Optimization by Simulated Annealing*", Science, 220, pp. 671-680, 1983.

[Kowalik 1968] J. Kowalik, M. R. Osborne, "*Methods for Unconstrained Optimization Problems*", Modern analytic and computational methods in Science and Mathematics, Richard Bellman Ed., ISBN 0-444-00041-0, 1968.

[Koza 1992] J. R. Koza, "*Genetic Programming*", Cambridge, MA, MIT Press, 1992.

[Lemay 2000] L. Lemay, R. Cadenhead, "*Le Programmeur Java 2*", Campus Press, ISBN 2-744-00886-9, 2000.

[Marih 1994] M. Marih, "*Mise en Œuvre de l'Approximation Diffuse et de la Méthode des Éléments Diffus pour la Résolution des Problèmes de Mécanique*", Thèse de Doctorat, Université de Technologie de Compiègne, 1994.

[Meropolis 1953] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, "*Equation of state calculations by fast computing machines*", Journal of Chemical Physics, vol. 21, pp. 1087-1092, 1953.

[Merrouche 1997] A. Merrouche, "*Vers une Mise en Oeuvre Automatique de la Conception Optimale: Architecture de Système et Outils Logiciels pour l'Optimisation de Forme de Pièces 3D*", Thèse de Doctorat, Université de Technologie de Compiègne, 1997.

[Michalewicz 1994] Z. Michalewicz, "*Genetic Algorithms + Data Structures = Evolution Programs*", Springer Verlag, ISBN 3-540-58090-5, 1994.

[Nayroles 1991] B. Nayroles, G. Touzot, P. Villon, "*La Méthode des Éléments Diffus*", Compte rendu à l'Academie de Sciences, 313, série II, PP. 133-138, Paris, France, 1991.

[Nelder 1965] J. A. Nelder, R. Mead, "*A Simplex Method for Function Minimization*", Computer Journal, vol. 7, pp. 308-312, 1965.

[Pahner 2000] Pahner, U., Hameyer K., "*Adaptive Coupling of Differential Evolution and Multiquadrics Approximation for the Tuning of the Optimization Process*", IEEE Transactions on Magnetics, vol. 36, n. 4, pp. 1047-1051, 2000.

[Pillet 1997] M. Pillet, "*Les Plans d'Expériences par la Méthode TAGUCHI*", Les Editions d'Organization, ISBN 2-70-812031-X, 1997.

[Polak 1971] E. Polak, "*Computational Methods in Optimization: A Unified Approach*", Academic Press, New York, 1971.

[Powell 1965] M. J. D. Powell, "*An efficient method for finding the minimum of a function of several variables without calculating derivations*", Computer Journal, vol. 7, pp. 155-162, 1965.

[Powell 1969] M. J. D. Powell, "*A Method for Nonlinear constraints in Minimization Problems*", Optimization, Academic Press, New York, pp. 283-298, 1969.

[Press 1992] W. H. Press, "*Numerical Recipes in C: The Art of Scientific Computing*", Cambridge University Press, ISBN 0-521-43108-5, 1992.

[Rao 1996] S. S. Rao, "*Engineering Optimization: Theory and Practice*", John Wiley & Sons, ISBN 0-471-55034-5, 1996.

[Rechenberg 1994] I. Rechenberg, "*Evolution Strategy*", in [Zurada 1994], pp. 147-159, 1994.

[Rockaffelar 1973] R. T. Rockaffelar, "*A Dual Approach to Solving Nonlinear Programming Problems by Unconstrained Optimization*", Mathematical Programming, vol. 12, n. 6, pp. 555-562, 1973.

[Ryan 1974] D. M. Ryan, "*Penalty and Barrier Functions*", P. E. Gill and Murray, Academic Press, 1974.

[Sado 1991] G. Sado, M-C. Sado, "*Les plans d'Expériences: de l'expérimentation à l'Assurance Qualité*", Afnor Technique, ISNB 2-124-50311-1, 1991.

[Saldanha 1992] R. R. Saldanha, "*Optimisation en Electromagnétisme par application conjointe des Méthodes de Programmation Non Linéaire et de la Méthode des Éléments Finis*", Thèse de Doctorat, Institut National Polytechnique de Grenoble, 1992.

[Sareni 1999] B. Sareni, "*Méthodes d'Optimisation Multimodales Associées à la Modélisation Numérique en Electomagnetisme*", Thèse de Doctorat, École Centrale de Lyon, 1999.

[Saludjian 1997] L. Saludjian, "*Optimisation en Electrotechnique par Algorithmes Génétiques*", Thèse de Doctorat, Institut National Polytechnique de Grenoble, 1997.

[Schimmerling 1998] P. Schimmerling, J-C. Sisson, A. Zaïdi, "*Pratique des Plans d'Expériences*", Lavoisier Tec & Doc, ISBN 2-743-00239-5, 1998.

[Schwefel 1995] H. P. Schwefel, "*Evolution and Optimum Seeking*", John Wiley & Sons, ISBN 0-471-57148-2, 1995.

[Shor 1977] N. Z. Shor, "*Cut-off Method with Space Extension in Convex Programming Problems*", Cybernetics, vol. 13, n. 1, pp. 94-96, 1977.

[Stroustrup 1997] B. Stroutstrup, "*The C++ Programming Language*", Addison-Wesley, ISBN 0-201-88954-4, 1997.

[Svanberg 1987] K. Svanberg, "*The Method of Moving Asymptotes. A New Method for Structural Optimization*", International Journal on Numerical Methods in Engineering, vol. 24, pp. 359-373, 1987.

[Syswerda 1989] G. Syswerda, "*Uniform Crossover in Genetic Algorithms*", Proceedings of ICGA89 - 3rd International Conference on Genetic Algorithms and Their Applications, pp. 2-9, 1989.

[Takahashi 1996] N. Takahashi, K. Ebihara, K. Yoshida, T. Nakata, K. Ohashi, K. Miyata, "*Investigation of Simulated Annealing Method and its Application to Optimal Design of die Mold for Orientation of Magnetic Powder*", IEEE Transactions on Magnetics, vol. 32, n. 3, pp. 1210-1213, 1996.

[Üler 1995] G. F. Üler, O. A. Mohammed, C. S. Koh, "*Design Optimization of Electrical Machines Using Genetic Algorithms*", IEEE Transaction on Magnetics, vol. 31, n. 3, pp. 2008-2011, 1995.

[Vasconcelos 1994] J. A. Vasconcelos, "*Optimisation de Forme des Structures Électromagnétiques*", Thèse de Doctorat, École Centrale de Lyon, 1994.

[Weeber 1992] K. Weeber, "*Parametrisation, efficacité et formulations pour l'optimisation de la forme de dispositifs électromagnétiques avec des éléments finis*", Thèse de Doctorat, Institut National Polytechnique de Grenoble, 1992.

[Zurada 1994] J. M. Zurada, R. J. Marks, C. J. Robinson, "*Computational Intelligence: Imitating Life*", IEEE Press, New Jersey, 1994.

Liste de Figures

Liste de Figures

Liste de Tableaux

Liste de Tableaux