Optimisation sans dérivées pour le problème de calage d'historique

Un nouvel algorithme d'optimisation sans dérivées adapté aux fonctions partiellement séparables (DFO-PSOF pour "Derivative Free Optimization for Partially Separable Objective Functions") est présenté dans ce chapitre. Il s'agit d'un algorithme à région de confiance utilisant des modèles d'interpolation quadratique de la fonction objectif. On rappelle qu'une fonction partiellement séparable est une fonction qui peut s'écrire sous la forme :

$$F(x_1, \ldots, x_p) = \sum_{i=1}^{n} f_i(x_{i,1}, \ldots, x_{i,p_i}) \tag{2.1}$$

avec pour tout $i \in \{1, ..., n\}$, $p_i < p$. L'adaptation à la séparabilité partielle de la fonction objectif se fait au niveau de la construction des modèles, similairement à l'adaptation aux fonctions de type moindres carrés [113, 112] : au lieu de construire un modèle pour la fonction objectif globale, la méthode DFO-PSOF construit un modèle pour chaque sous-fonction objectif. Chacune de ces fonctions ne dépendant que d'un nombre très restreint de paramètres, les modèles d'interpolation peuvent en particulier être construits avec seulement un petit nombre de points sans détériorer leur précision.

On présente en section 2.1 un état de l'art des méthodes d'optimisation locales utilisées en ingénierie pétrolière. En section 2.2 sont rappelés les résultats nécessaires à l'introduction du nouvel algorithme. Ce dernier est ensuite décrit en section 2.3 avec la preuve de sa convergence en section 2.4. Enfin, on montre en section 2.5 les premiers résultats numériques obtenus à partir de son implémentation en C++.

2.1 État de l'art

De nombreux tests ont été réalisés dans la littérature et à IFPEN pour déterminer des méthodes d'optimisation efficaces dans le domaine pétrolier afin d'apporter des solutions "optimales" à des problèmes variés tels que le placement des puits [5, 7, 13], la maximisation de la production [73, 31, 114, 109] ou le calage d'historique [11, 18, 36]. Pour simplifier l'écriture, on considère que ces problèmes s'écrivent sous la forme :

$$\begin{gathered} \text{Trouver } (x_1^*, ..., x_p^*) \in [b1, B1] \times ... \times [b_p, B_p] \text{ tel que} \\ (x_1^*, ..., x_p^*) = argmin F(x_1, \ldots, x_p) \end{gathered} \tag{2.2}$$

où $F(x_1, \ldots, x_p) = \sum_{i=1}^{n} f_i(x_1, \ldots, x_p)$ est supposée continûment différentiable avec ∇F continu et Lipschitzien sur le domaine vérifiant les contraintes.

Par exemple, le calage d'historique est un problème de minimisation de l'erreur de production sur chaque puits et peut donc s'écrire sous la forme d'une minimisation d'une somme de fonctions. Les contraintes de bornes sont présentes pour assurer que chacun des paramètres ne varie pas au-delà de tout sens physique ou pour éviter d'ajouter des minimums locaux, la fonctions objectif étant périodique en certains paramètres (les paramètres de déformation graduelle par exemple). La fonction objectif étant par exemple périodique pour les paramètres de déformation graduelle, il est naturel de contraindre ces derniers à rester dans la limite d'une période.

Il existe une très grande diversité de méthodes d'optimisation dans la littérature dont bon nombre ont été appliquées à des problèmes pétroliers. On peut trouver par exemple des méthodes stochastiques [5, 43] ou déterministes [73, 36], des méthodes globales [13, 48, 79] ou locales [11, 70], etc. Les méthodes globales, bien que permettant généralement un meilleur calage, demandent la plupart du temps un nombre d'évaluations beaucoup plus important que les autres méthodes et sont difficilement utilisables pour des cas non synthétiques de réservoir (dans la plupart des cas, elles sont mêmes trop coûteuses pour les cas synthétiques). On se concentrera ainsi dans ce manuscrit sur les méthodes locales. Plus précisément, on décrira plus en détail les méthodes de descente et les méthodes à région de confiance avant de proposer une nouvelle méthode adaptée aux caractéristiques de la fonction objectif du problème de calage d'historique.

2.1.1 Méthodes de descente

Les méthodes présentées ici utilisent une information locale sur la fonction objectif qui garantit la plupart du temps, sous des hypothèses de régularité, une convergence de l'algorithme mais n'offre aucune assurance quant au caractère global de l'optimum trouvé. Les hypothèses de régularité peuvent être difficile à vérifier dans le cas du problème de calage d'historique. Par exemple, la condition de différentiabilité de la fonction objectif peut être très contraignante et entraîner des problèmes de convergence dans des cas réels.

Les algorithmes décrits dans cette section consistent à donner une estimation initiale de l'optimum puis à la mettre à jour en se déplaçant dans une direction de descente, comme illustré ci-dessous :

1. Choisir une estimation initiale de l'optimum x_0 et une tolérance ϵ.
2. À l'itération $k \geq 0$:
   (a) Trouver une direction de descente d_k.
   (b) Mettre à jour l'estimation de l'optimum, $x_{k+1} = x_k + \alpha_k d_k$ avec un pas α_k bien choisi.
3. Conditions d'arrêt (par exemple) : $\frac{\|x_{k+1} - x_k\|}{\|x_0\|} < \epsilon$.

La direction de descente d_k est en général calculée à partir des dérivées au premier et second ordre de la fonction objectif. Un exemple simple est de prendre $d_k = -\nabla F(x_k)$, auquel cas d_k est appelée la direction de plus forte pente (ou steepest descent). Les méthodes de type quasi Newton s'appuient quant à elles sur une estimation de la Hessienne de la fonction objectif en prenant $d_k = -\tilde{H}_k^{-1} \nabla F(x_k)$.

Il est donc important de calculer des approximations fiables des dérivées aux ordres un et deux de la fonction objectif. On présente ci-dessous cinq approches possibles de ce problème, pour la plupart très coûteuses en terme de nombre de simulations.

(i) Approximation du gradient par méthode adjointe

Une méthode courante en problèmes inverses pour calculer le gradient de la fonction objectif consiste à construire un problème adjoint au problème direct et, à partir de la résolution de ces deux problèmes d'obtenir une évaluation du gradient. Les domaines d'application peuvent être très vastes : Oberail et al. [76] utilisent par exemple ce type de méthode dans le milieu de l'imagerie médicale, Jarny et al. [57] l'utilisent pour résoudre des problèmes de conduction tandis que Fisher et al. [40] en présentent une application dans le domaine de l'optimisation de forme. Une telle méthode s'avère très efficace puisqu'une simple simulation du système et une résolution du problème adjoint permettent d'obtenir une bonne estimation du gradient. Cependant, il n'existe pas de méthode systématique pour construire le problème adjoint ; la construction d'un tel problème peut ainsi se révéler très délicate et nécessite une connaissance approfondie du problème direct.

Dans les problèmes inverses issus de l'ingénierie pétrolière, le problème direct correspond à la simulation des écoulements dans le réservoir [78]. Lorsqu'un problème adjoint est disponible, cette méthode offre d'excellents résultats [73, 51, 66, 96]. Cependant, les simulateurs industriels étant très complexes, il est impossible de produire un code adjoint dans la plupart des cas. Il est donc nécessaire de disposer d'une méthode alternative pour calculer le gradient pour le calage d'historique.

(ii) Approximation du gradient par différences finies

On peut estimer le gradient de F en approximant chacune de ses composantes par différences finies décentrées [75] :

$$\frac{\partial F(x)}{\partial x_i} \approx \frac{F(x + h_i e_i) - F(x)}{h_i}$$

où e_i est le $i^{\text{ème}}$ vecteur de la base canonique de $\mathbb{R}^p$.

Cette méthode d'approximation du gradient a notamment été utilisée pour des problèmes pétroliers dans [11, 49, 105]. Elle requiert par contre $(p+1)$ évaluations de la fonction objectif, ce qui devient très coûteux lorsque la dimension du problème augmente, phénomène mis en évidence dans [5] par exemple.

De plus, un problème récurrent avec de telles approximations consiste à choisir correctement le paramètre de pas h_i. En effet, le développement de Taylor de F d'où est issue la formule de différences finies nous incite à choisir h_i le plus petit possible. Toutefois, dans le cas où une erreur e est commise dans l'évaluation de F (ce qui est le cas dans un problème réel de simulation de réservoir), cette erreur se répercute en $\frac{e}{h_i}$ dans le calcul de $\frac{\partial F(x)}{\partial x_i}$. Une petite erreur peut alors avoir un très grand impact sur l'évaluation de ∇F. Le calibrage du paramètre h_i doit alors se faire à partir d'une estimation de l'erreur commise lors de l'évaluation de la fonction objectif, ce qui peut s'avérer très délicat dans la pratique lorsque les problèmes sont bruités.

(iii) Gradient stochastique - SPSA

Une méthode proposée par Spall dans [101, 102] permet de calculer une approximation du gradient d'une fonction à l'aide de perturbations aléatoires. Le gradient stochastique (ou SPSA) basique s'écrit :

$$\nabla F(x) \approx \frac{F(x + \epsilon\delta) - F(x)}{\epsilon\delta}$$

où $\delta = (\delta_1, \ldots, \delta_p)$ représente le vecteur de perturbation et $\epsilon > 0$ est l'amplitude de la perturbation. La notation avec δ en dénominateur correspond ici à une division terme à terme de

vecteurs, c'est à dire que la coordonnées i du gradient estimé de F est :

$$(\nabla F(x))_i \approx \frac{F(x+\epsilon\delta) - F(x)}{\epsilon\delta_i}$$

La $i^{\text{ème}}$ coordonnée de la perturbation δ est choisie dans la méthode de Spall en faisant un tirage aléatoire à partir d'une loi de Bernouilli symétrique ± 1, mais d'autres résultats [65] ont montré qu'il peut être avantageux de choisir une autre loi. A noter que l'écriture avec δ_i en dénominateur interdit de choisir une loi qui donne la possibilité de tirer une des coordonnées de δ à 0, l'une des hypothèses nécessaires étant même de choisir $\frac{1}{\delta_i}$ uniformément borné.

Deux propriétés importantes sont par ailleurs à signaler pour cette estimation du gradient :
— elle propose toujours une direction de descente, en l'occurrence la direction δ,
— son espérance est égale au gradient réel de la fonction objectif.
la première propriété permet d'assurer la convergence d'un algorithme de descente utilisant ce gradient. La seconde permet d'affiner grandement l'approximation du gradient en moyennant plusieurs estimations calculées à partir de différents tirages de δ.

Le principal intérêt de cette méthode est le faible coût pour avoir une estimation du gradient puisque deux évaluations suffisent à l'obtenir. Elle a d'ailleurs été utilisée dans [43, 65] pour des problèmes de calage d'historique, dans [38, 80, 115] pour des problèmes de maximisation de la production et dans [5, 4] pour l'optimisation du placement de puits. Malheureusement, le manque de précision dans ce calcul ralentit souvent beaucoup la vitesse de convergence des algorithmes d'optimisation, particulièrement dans le das où les problèmes traités sont de surcroit bruités.

(iv) Approximation de la Hessienne de la fonction objectif

Pour des raisons de coût de calcul, il n'est en général pas possible de calculer les dérivées d'ordre 2 de la fonction objectif ($\mathcal{O}(p^2)$ simulations). On utilise donc souvent des méthodes de mise à jour séquentielle de la Hessienne de la fonction objectif à partir de calculs effectués pour le gradient. L'algorithme de descente ainsi conçu est appelé algorithme de Quasi-Newton [29, 75, 99] : il s'agit d'une méthode de Newton qui n'utilise pas la valeur exacte de la Hessienne mais une approximation qui s'affine au cours des itérations. L'idée consiste alors à construire la Hessienne H_{k+1} à partir des valeurs de H_k, ∇F^k, ∇F^{k+1}, x_k et x_{k+1}. L'approximation BFGS (Broyden-Fletcher-GoldFarb-Shanno) [14, 41, 47, 99] consiste, en notant :

$$\begin{cases} \delta_k = x_{k+1} - x_k \\ \gamma_k = \nabla F^{k+1} - \nabla F^k \end{cases}$$

à mettre à jour H_{k+1} telle que :

$$H_{k+1} = H_k + \frac{\gamma_k \gamma_k^T}{\gamma_k^T \delta_k} - \frac{H_k \delta_k \delta_k^T H_k^T}{\delta_k^T H_k^T \delta_k}$$

Cette façon de mettre à jour la Hessienne à chaque itération de l'algorithme garantit que si H_k est symétrique définie positive, alors H_{k+1} le sera aussi. On initialise en général H_0 à un multiple de l'identité, de façon à maintenir ces propriétés tout au long de l'algorithme.

Une alternative plus simple peut être utilisée lorsque la fonction objectif s'écrit sous la forme d'une somme de résidus au carré [12, 64] (ce qui est le cas dans le problème de calage

d'historique) :

$$F(x_1, \ldots, x_p) = \sum_{i=1}^{n} r_i^2(x_1, \ldots, x_p)$$

Dans ce cas, les dérivées d'ordre 1 et 2 peuvent s'exprimer en fonction sous la forme suivante, lorsque la matrice Jacobienne s'écrit :

$$J(x) = \begin{bmatrix} \nabla r_1(x)^T \\ \nabla r_2(x)^T \\ \vdots \\ \nabla r_n(x)^T \end{bmatrix}$$

Le gradient devient :

$$\nabla F(x) = \sum_{i=1}^{n} r_i(x) \nabla r_i(x) = J(x)^T r(x)$$

La Hessienne devient quant à elle :

$$\nabla^2 F(x) = J(x)^T J(x) + \sum_{i=1}^{n} r_j(x) \nabla^2 r_i(x)$$

L'approximation de Gauss-Newton consiste à négliger le second terme dans l'expression de la Hessienne : $\nabla^2 F(x) \approx J(x)^T J(x)$. Dans la plupart des cas, le premier terme $J(x)^T J(x)$ est prépondérant par rapport au second terme (particulièrement à l'approche de l'optimum x^*). Cette approximation offre l'avantage de ne pas nécessiter de calculs supplémentaires une fois que le calcul du gradient est effectué et d'approcher généralement beaucoup mieux la Hessienne que la méthode BFGS.

De telles approximations offrent de bons résultats pour les algorithmes de descente du deuxième ordre sans nécessiter d'évaluation supplémentaire par rapport à l'estimation du gradient. Des applications de ces méthodes dans l'ingénierie pétrolière peuvent être trouvées dans [44, 100].

(v) Approximation du gradient d'une fonction partiellement séparable

Les travaux de D. Ding et al. dans [36] ont montré qu'il est possible de réduire le coût en terme d'évaluations de la fonction objectif du calcul du gradient d'une fonction partiellement séparable. Plus précisément, on cherche à calculer le gradient d'une fonction de la forme :

$$F(x_1, \ldots, x_p) = \sum_{i=1}^{n} f_i(x_{i,1}, \ldots, x_{i,p_i})$$

où chaque sous-fonction f_i ne dépend que d'un nombre restreint de paramètres (pour tout i, $p_i < p$). Les dérivées partielles de F s'écrivent alors :

$$\frac{\partial F(x_1, \ldots, x_p)}{\partial x_j} = \sum_{i=1}^{n} \frac{\partial f_i(x_{i,1}, \ldots, x_{i,p_i})}{\partial x_j}$$

Notons p_{max} le nombre maximum de paramètres dont dépendent les f_i. On veut pouvoir calculer simultanément les dérivées partielles de toutes les sous-fonctions objectifs avec seulement L

perturbations de F ($p_{max} \leq L \leq p$ où p est le nombre de paramètres de F). La matrice des perturbations s'écrit :

$$A = \begin{bmatrix} a_{11} & \dots & a_{1p} \\ a_{21} & \dots & a_{2p} \\ \vdots & \ddots & \vdots \\ a_{L1} & \dots & a_{Lp} \end{bmatrix}$$

On peut remarquer que l'approximation du gradient par différences finies correspond au cas où $L = p$ et $A = Id$.

Pour tout $k \in [\![1, \dots, n]\!]$, et pour tout $i \in [\![1, \dots, L]\!]$, on note $B_k = (b_1^k, ..., b_L^k)$ le vecteur des différences de la sous-fonction objectif f_k entre les points perturbés P_i et le point courant P_0 : $b_i^k = f_k(P_i) - f_k(P_0)$. Les dérivées partielles d'une sous-fonction f_k peuvent alors être obtenues en résolvant le problème d'optimisation suivant :

$$\text{Trouver } Y_k^* = argmin J(Y_k)$$

avec

$$J(Y_k) = \frac{1}{2} \|A_k Y_k - B_k\|^2$$

et où A_k correspond à la matrice de perturbations associée à la fonction f_k :

$$A_k = \begin{bmatrix} a_{1,1_k} & \dots & a_{1,p_k} \\ a_{2,1_k} & \dots & a_{2,p_k} \\ \vdots & \ddots & \vdots \\ a_{L,1_k} & \dots & a_{L,p_k} \end{bmatrix}$$

Comme on a choisi $L \geq p_{max}$, la solution du problème aux moindres carrés donne une approximation du gradient de f_k à la seule condition que A_k soit de rang maximal. Pour obtenir une approximation du gradient de F, il suffit donc juste de choisir A de telle sorte que toutes les matrices A_k soient de rang maximal. Une bonne manière de la choisir est de la prendre localement de rang maximal (toutes les sous-matrices sont de rang maximal). Généralement, une matrice dont les éléments ont été générés aléatoirement présente cette caractéristique. Il est donc possible de calculer une approximation du gradient d'une fonction partiellement séparable avec seulement $L = p_{max}$ simulations, ce qui peut grandement améliorer les performances d'un algorithme d'optimisation basé sur le calcul du gradient.

Le choix des perturbations reste cependant problématique. En effet, le bon choix de ces perturbations est en lien direct avec la qualité de l'approximation du gradient. Une idée proposée dans [36] serait de trouver un ensemble de perturbations "optimal" en minimisant l'erreur commise sur l'approximation du gradient d'une fonction connue. Il est cependant difficile de garantir que l'approximation du gradient de la véritable fonction objectif restera pertinente.

2.1.2 Méthodes de recherche directe

Les méthodes de recherche directe consistent à échantillonner la fonction objectif sur un certain nombre de points et à décider des évaluations suivantes à réaliser en utilisant uniquement cet échantillonage. Elles ne nécessitent donc pas de construction de modèle de la fonction objectif ni d'approximation de ses dérivées. On présente deux types de méthodes : la première est basée sur la recherche de directions de descente tandis que la seconde considère l'évaluation de nouveaux points en fonction de la forme de simplexes. Ces méthodes présentent l'avantage de ne pas nécessiter la connaissance des dérivées de la fonction objectif tout en étant très simples à implémenter.

(i) Recherche directe directionnelle

On commence par donner la définition d'une base positive de $\mathbb{R}^n$ [27, 30].

Définition 1. — *Le cône positif engendré (positive span) par l'ensemble de vecteurs* $\{v_1, ..., v_r\}$ *de* $\mathbb{R}^n$ *est le cône convexe :*

$$\{v \in \mathbb{R}^n \text{ tel qu'on peut écrire } v = \alpha_1 v_1 + ... + \alpha_r v_r \text{ avec } \forall i, \alpha_i \geq 0\}. \tag{2.3}$$

— *Un ensemble engendrant* $\mathbb{R}^n$ *positivement est un ensemble de vecteurs de* $\mathbb{R}^n$ *dont le cône positif engendré est* $\mathbb{R}^n$*.*

— *Un ensemble de vecteurs est dit positivement dépendant si l'un des vecteurs de l'ensemble appartient au cône positif engendré par les autres.*

— *Une base positive de* $\mathbb{R}^n$ *est un ensemble de vecteurs positivement indépendants engendrant positivement* $\mathbb{R}^n$*.*

Les algorithmes de recherche directe directionnelle suivent alors la forme générale décrite dans l'algorithme 1.

Algorithme 1 Algorithme général de recherche directe

1. **Initialisation**
   — Choisir un point x_0 et des réels $\alpha_0 > 0$, $0 < \beta_1 \leq \beta_2 < 1$, $\gamma > 1$ et $\epsilon > 0$.
   — Choisir $\mathcal{D}$ un ensemble fini de bases positives de $\mathbb{R}^n$.
2. **Itération** k
   (a) **Exploration**
   — Choisir une base positive $\mathcal{D}_k \in \mathcal{D}$ et ordonner l'ensemble d'exploration $P_k = \{x_k + \alpha_k d, \text{ où } d \in \mathcal{D}_k\}$.
   — Évaluer la fonction objectif aux points de l'ensemble d'exploration dans l'ordre établi.
   — Si l'un des points d'exploration est tel que

   $$F(x_k + \alpha_k d) < F(x_k)$$

   alors l'itération est fructueuse : arrêter l'exploration et choisir $x_{k+1} = x_k + \alpha_k d$.
   (b) **Mise à jour du pas d'exploration** α_k
   — Si l'itération a été fructueuse, alors augmenter le pas d'exploration : $\alpha_{k+1} \in [\alpha_k, \gamma\alpha_k]$.
   — Sinon, réduire le pas d'exploration : $\alpha_{k+1} \in [\beta_1\alpha_k, \beta_2\alpha_k]$.
   (c) **Test d'arrêt** Arrêter l'algorithme lorsque le pas d'exploration devient trop petit ($\alpha_k < \epsilon$).

Le fait de choisir des bases positives de $\mathbb{R}^n$ permet de garantir qu'il est possible d'explorer l'espace dans son ensemble avec les points d'exploration définis dans l'étape 2(a) de l'algorithme 1. Des informations plus détaillées peuvent être trouvées dans [2, 106]. Kolda et al. [60] ont par ailleurs publié en 2003 une revue des méthodes classiques de ce type.

On peut par exemple considérer le cas de la base positive canonique :

$$\mathcal{D} = [e_1, ..., e_n, -e_1, ... - e_n] \tag{2.4}$$

Le cas le plus simple pour l'algorithme 1 consiste alors à choisir uniquement dans $\mathcal{D}$ la base positive canonique, l'algorithme résultant est appelé recherche directe par coordonnées.

(ii) Recherche directe par simplexe
L'algorithme Nelder-Mead [74] est un algorithme particulier classique de recherche directe. Chaque itération détermine l'itéré suivant en fonction de la valeur de la fonction objectif aux $n+1$ sommets d'un simplexe $Y = \{y_0, y_1, ..., y_n\}$ ordonnés dans un ordre croissant (suivant la fonction objectif à optimiser). Quatre opérations de base fondées sur le centre de gravité $y_g = \sum_{i=0}^{n-1} y_i$ du meilleur simplexe à n sommets sont possibles dans cet algorithme. Elles consistent toutes à calculer un nouveau point y de la forme

$$y = y_g + \delta(y_g - y_n)$$

où δ est un réel. Dans l'algorithme Nelder-Mead,

1. l'opération correspond à une réflexion lorsque $\delta = \delta_r = 1$
2. l'opération correspond à une expansion lorsque $\delta = \delta_e = 2$
3. l'opération correspond à une contraction extérieure lorsque $\delta = \delta_{ce} = 0.5$
4. l'opération correspond à une contraction intérieure lorsque $\delta = \delta_{ci} = -0.5$

Ces coefficients correspondent aux choix standards. Dans un cadre plus général on peut simplement choisir $-1 < \delta_{ci} < 0 < \delta_{ce} < \delta_r < \delta_e$.
L'algorithme a aussi la possibilité d'effectuer un rétrécissement dans le cas où aucune des opérations précédentes n'a donné de point correct. Un rétrécissement consiste à supprimer tous les points du simplexe Y excepté y_0 et à calculer pour tout $i \in \{1, ...n\}$

$$y_i^{ret} = y_0 + \gamma(y_i - y_0)$$

avec $0 < \gamma < 1$. Le nouveau simplexe devient alors $Y^{ret} = \{y_0, y_1^{ret}..., y_n^{ret}\}$. On présente dans l'algorithme 2 les détails de la méthode Nelder-Mead.
L'algorithme Nelder-Mead est populaire grâce à la fois à sa simplicité et à sa capacité à prendre en compte la courbure de la fonction objectif. En l'état, il existe des cas dans lesquels il ne convergera pas vers des points critiques de la fonction objectif. Il est cependant possible de l'adapter pour qu'il converge globalement vers des points critiques [61, 71].

Les algorithmes de recherche directe sont très simples à implémenter et à utiliser (ils nécessitent très peu de paramètres à calibrer). Cependant, leur efficacité n'est souvent pas la meilleure et ils deviennent très vite trop coûteux pour nos applications. On peut néanmoins trouver des applications de ces méthodes à des problèmes pétroliers dans [8, 20, 21]. Des arguments pour l'utilisation de ce type de méthode peuvent être avancés lorsqu'il est possible d'effectuer en parallèle les simulations d'écoulement. L'article [8] en particulier montre que lorsque le nombre de simulations effectuées en parallèle augmente, les méthodes de recherche directe ne donnent pas forcément de moins bons résultats que les méthodes de type SQP, "Sequential Quadratic Programming" présentés brièvement dans la section suivante.

Algorithme 2 Algorithme Nelder-Mead

1. **Initialisation**
   — Choisir un simplexe initial $Y_0 = \{y_0^0, y_1^0, ..., y_n^0\}$.
   — Évaluer F aux points de Y_0.
   — Choisir les constantes $0 < \gamma < 1$ et $-1 < \delta_{ci} < 0 < \delta_{ce} < \delta_r < \delta_e$.
2. **Itération** k
   Fixer $Y = Y_k$ et calculer $y_g = \sum_{i=0}^{n-1} y_i$.
   (a) **Ordonner les sommets de** Y
   Classer les sommets de Y de telle sorte que
   $$F(y_0) \leq F(y_1) \leq ... \leq F(y_n)$$
   (b) **Réflexion**
   — Calculer y_r la réflexion de y_n par y_g.
   — Évaluer $F(y_r)$.
   — Si $F(y_0) \leq F(y_r) < F(y_{n-1})$, alors remplacer y_n par y_r dans Y et terminer l'itération : $Y_{k+1} = \{y_0, y_1, ...y_{n-1}, y_r\}$.
   (c) **Expansion**
   Si $F(y_r) < F(y_0)$, alors
   — Évaluer y_e le point étendu.
   — Si $F(y_e) \leq F(y_r)$, alors remplacer y_n par y_e dans Y et terminer l'itération : $Y_{k+1} = \{y_0, y_1, ...y_{n-1}, y_e\}$.
   — Si $F(y_r) < F(y_e)$, alors remplacer y_n par y_r dans Y et terminer l'itération : $Y_{k+1} = \{y_0, y_1, ...y_{n-1}, y_r\}$.
   (d) **Contraction extérieure**
   Si $F(y_r) < F(y_n)$, alors
   — Évaluer y_{ce} le contracté extérieur.
   — Si $F(y_{ce}) \leq F(y_r)$, alors remplacer y_n par y_{ce} dans Y et terminer l'itération : $Y_{k+1} = \{y_0, y_1, ...y_{n-1}, y_{ce}\}$.
   — Sinon, effectuer un rétrécissement du simplexe (étape 2.(f)).
   (e) **Contraction intérieure**
   Si $F(y_r) \geq F(y_n)$, alors
   — Évaluer y_{ci} le contracté intérieur.
   — Si $F(y_{ci}) \leq F(y_n)$, alors remplacer y_n par y_{ci} dans Y et terminer l'itération : $Y_{k+1} = \{y_0, y_1, ...y_{n-1}, y_{ci}\}$.
   — Sinon, effectuer un rétrécissement du simplexe (étape 2.(f)).
   (f) **Rétrécissement du simplexe**
   $Y_{k+1} = \{y_0 + \gamma(y_i - y_0), \text{ pour } i = 0, ..., n\}$.

2.1.3 Méthodes de type région de confiance

Ces méthodes consistent à construire des modèles quadratiques successifs de la fonction objectif dont l'optimisation sur une région dite de "confiance" est simple et peut être appelée avec un coût marginal. C'est particulièrement le cas pour les problèmes pétroliers puisque l'optimisation de tels modèles quadratiques prend au maximum quelques secondes tandis que l'évaluation de la fonction objectif demande plusieurs heures de temps CPU.

Un algorithme de région de confiance s'écrit souvent sous la forme [25, 27, 75] de l'algorithme 3

Algorithme 3 Un algorithme de type région de confiance

1. **Initialisation**
   Choisir x_0 un point, $\Delta_0 > 0$ une taille de région de confiance autour de x_0 et construire un modèle quadratique initial m_0 de la fonction objectif.
2. **Itération k**
   — S'assurer de la correction du modèle
   — Lorsque le modèle est correct, calculer $x_k^+ = \min_{x \in B(x_k, \Delta_k)} m_k(x)$
   — Mettre à jour le modèle en évaluant F au point x_k^+
   — Si x_k^+ est un meilleur point que x_k, définir $x_{k+1} = x_k^+$, sinon $x_{k+1} = x_k$
3. **Condition d'arrêt**
   Arrêter l'algorithme lorsque le gradient du modèle m_k devient trop petit.

Plusieurs méthodes sont applicables pour calculer les modèles quadratiques, on en présente brièvement deux ci-dessous.

(i) Estimation du gradient et de la Hessienne de F
L'idée ici est d'utiliser des approximations du gradient et de la Hessienne de la fonction objectif pour construire les modèles quadratiques. Le modèle m_k à l'itération k de l'algorithme à région de confiance s'écrit alors simplement :

$$m_k(x_k + x) = F(x_k) + \nabla F(x_k)^T x + \frac{1}{2} x^T \nabla^2_{xx} F(x_k) x$$

Ce type de méthode à région de confiance fait partie des méthodes SQP (Sequential Quadratic Programming) [15, 16, 108]. Une revue générale de la littérature dans le domaine a été effectuée par Gould et al. [50]. Typiquement, ce type d'algorithmes produit de bons résultats dans les cas pétroliers [46, 110]. Ces méthodes fournissent d'excellents résultats lorsque les gradients et Hessiennes de la fonction objectif sont bons. Néanmoins, les mêmes problèmes de coût et de précision de l'estimation de ces quantités que pour les algorithmes de descente peuvent survenir en l'absence de problème adjoint.

(ii) Interpolation quadratique
Afin de s'affranchir des problèmes liés aux évaluations des dérivées de la fonction objectif, on peut créer les modèles en interpolant F sur des points correctement choisis. Ce type d'optimisation sans dérivées donne souvent de très bons résultats sur des problèmes pétroliers [116, 63, 114]. C'est sur ce type de méthodes que nous avons choisi de travailler en raison de ses qualités et du fait qu'elles ne nécessitent pas d'évaluation de dérivées. Nous les décrivons avec beaucoup plus de détails dans la section suivante.

2.2 Description du fonctionnement des méthodes de type région de confiance avec interpolation quadratique

2.2.1 Un algorithme simpliste

Dans une optique d'optimisation par une méthode de type région de confiance avec des modèles d'interpolation quadratique, l'algorithme simpliste 4 est l'un de ceux pouvant être conçu intuitivement.

Algorithme 4 Un algorithme simpliste de type région de confiance avec modèle d'interpolation

1. **Initialisation**
   - Choisir un point x_0 et un rayon de région de confiance Δ_0 initiaux.
   - Sélectionner un ensemble d'interpolation initial Y_0 contenant q points dont x_0 et évaluer F en les points de Y_0.
   - Construire le modèle initial m_0 qui interpole F aux points de Y_0.
   - Fixer les constantes $\epsilon > 0$ et $0 < \gamma_{low} < 1$.
2. **Itération** k
   (a) **Test d'arrêt**
   - Calculer le gradient du modèle au point courant x_k : $g_k = \nabla m_k(x_k)$
   - Si $\|g_k\| \leq \epsilon$, arrêter l'algorithme, sinon passer à l'étape suivante

   (b) **Calcul du nouveau point**
   Calculer un point x_k^+ de telle sorte que :
   $$m_k(x_k^+) = \min_{x \in B(x_k, \Delta_k)} m_k(x)$$
   Évaluer $F(x_k^+)$
   i. **Itération fructueuse :** Si $F(x_k^+) < F(x_k)$, remplacer un point de Y_k par x_k^+ et choisir $x_{k+1} = x_k^+$.
   ii. **Itération infructueuse :** Si $F(x_k^+) \geq F(x_k)$, choisir $x_{k+1} = x_k$.
   iii. **Gestion de la région de confiance :**
   - Si $x_{k+1} = x_k$, choisir $\Delta_{k+1} = \gamma_{low}\Delta_k$
   - Si $x_{k+1} = x_k^+$, choisir $\Delta_{k+1} \geq \Delta_k$

   (c) **Mise à jour du modèle**
   Calculer le nouveau modèle d'interpolation m_{k+1} centré sur x_{k+1} et utilisant Y_{k+1}. Retourner ensuite à l'étape 2.

Un tel algorithme peut fonctionner correctement et parfois donner de très bons résultats numériques, comme présenté dans l'article de Fasano et al. [39]. Cependant, Scheinberg et Toint ont montré dans [98] qu'il n'est pas possible d'ignorer complètement la qualité de la géométrie des ensembles d'interpolation pour assurer la convergence de l'algorithme. En effet, dans l'algorithme 4, l'obtention d'un mauvais point d'essai x_k^+ peut être due à deux causes distinctes :

- soit la région de confiance est trop grande pour qu'un modèle quadratique puisse être une bonne approximation de F ;
- soit la géométrie de l'ensemble d'interpolation n'est pas assez bonne pour construire un modèle quadratique approchant correctement F.

En l'état, seule la première cause est prise en compte dans l'algorithme au-dessus ; il n'y a en effet aucun moyen d'améliorer la géométrie des ensembles d'interpolation si cela devient nécessaire.

La figure 2.1 issue de [98] montre un cas en dimension 2 pour lequel l'algorithme tend à n'ajouter que des points sur une unique droite.

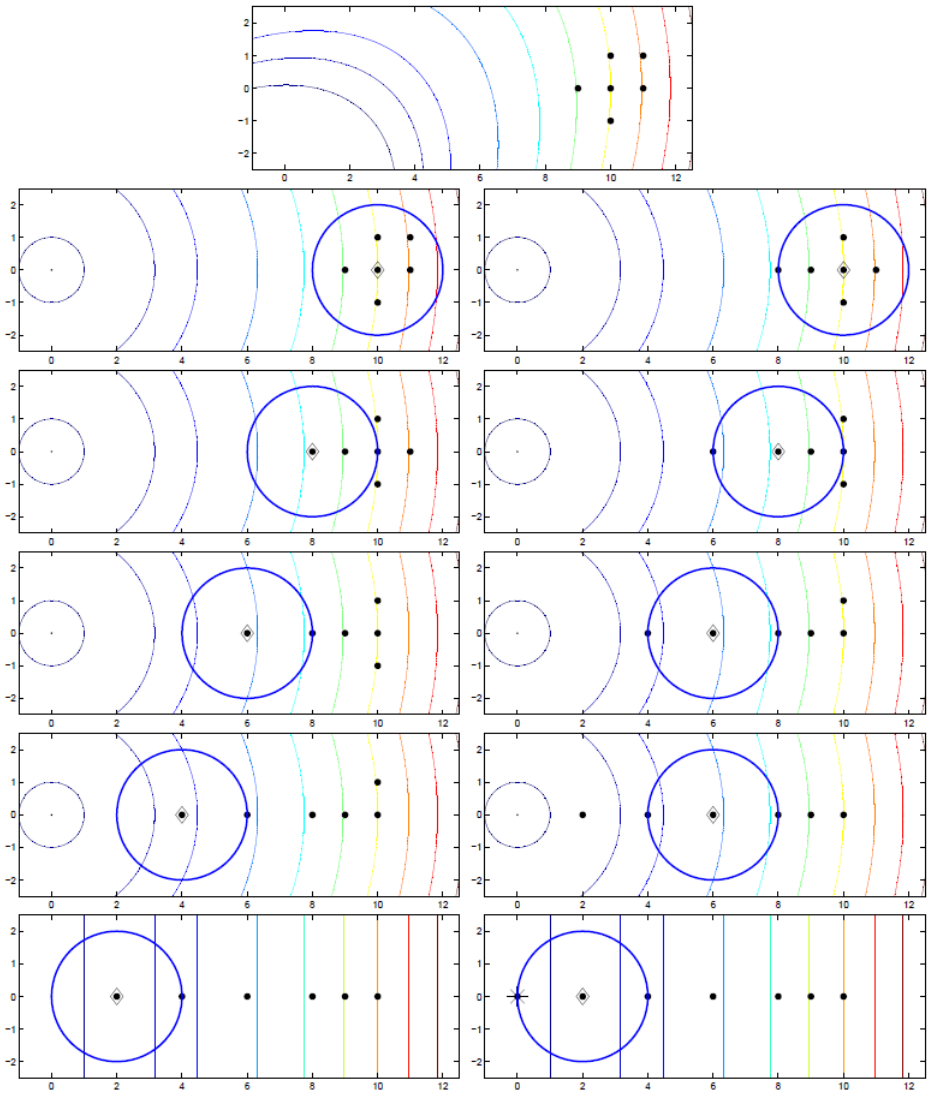


FIGURE 2.1 – Exemple d'une fonction et d'un ensemble d'interpolation qui mène l'algorithme à construire des modèles dégénérés (issu de [98]).

Les modèles m_k successifs n'ont alors plus aucune chance d'être de bonnes approximations de la fonction objectif quelle que soit la taille des régions de confiance et mènent l'algorithme à s'arrêter sur des points non stationnaires. Il est donc nécessaire pour un algorithme de type région de confiance utilisant des modèles d'interpolation de disposer d'un moyen de garantir la qualité de la géométrie des ensembles de points d'interpolation. On présentera dans les sous-sections suivantes une mesure de la qualité des ensembles d'interpolation ainsi qu'une procédure séquentielle permettant d'améliorer ces géométries.

2.2.2 Construction des modèles quadratiques

Deux méthodes sont possibles ici et présentées ci-dessous.

2.2.2.1 Décomposition dans une base de polynômes quadratiques

Pour construire un modèle d'interpolation quadratique de F, on commence par choisir une base $\phi = \{\phi_1, ..., \phi_{L_p}\}$ de $\mathcal{P}_2^p$ (espace des polynômes à p variables de degré ≤ 2), où $L_p = \frac{(p+1)(p+2)}{2}$ est la dimension de $\mathcal{P}_2^p$. On aura donc besoin d'évaluer F en L_p points $Y = \{y_1, \ldots, y_{L_p}\}$ de $\mathbb{R}^p$ pour être capable de construire un modèle d'interpolation $m \in \mathcal{P}_2^p$ unique. On définit :

$$M(\phi, Y) = \begin{bmatrix} \phi_1(y_1) & \phi_2(y_1) & \cdots & \phi_{L_p}(y_1) \\ \phi_1(y_2) & \phi_2(y_2) & \cdots & \phi_{L_p}(y_2) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_1(y_{L_p}) & \phi_2(y_{L_p}) & \cdots & \phi_{L_p}(y_{L_p}) \end{bmatrix}$$
$$\alpha_\phi = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{L_p} \end{bmatrix} \text{ et } F(Y) = \begin{bmatrix} F(y_1) \\ F(y_2) \\ \vdots \\ F(y_{L_p}) \end{bmatrix}. \tag{2.5}$$

Les coordonnées α_ϕ dans ϕ du modèle m interpolant F aux points de Y sont solution du problème linéaire :

$$M(\phi, Y)\alpha_\phi = F(Y) \tag{2.6}$$

m existe et est unique si et seulement si la matrice $M(\phi, Y)$ n'est pas singulière, c'est à dire si tous les points de Y ne résident pas sur une même variété quadratique. On définit pour cela la notion d'unisolvance :

Définition 2 (unisolvance)**.** *Soient Y un ensemble de L_p points d'interpolation dans $\mathbb{R}^p$ et ϕ une base de $\mathcal{P}_2^p$. On dit que Y est unisolvant ("poised" en anglais) si la matrice $M(\phi, Y)$ n'est pas singulière.*

2.2.2.2 Calcul des polynômes de Lagrange

Une autre façon équivalente de construire un modèle d'interpolation quadratique est de considérer des bases de polynômes de Lagrange.

Définition 3. *Une base $\{l_1, ..., l_{L_p}\}$ de $\mathcal{P}_2^p$ est une base de Lagrange pour l'ensemble de points $Y = \{y_1, \ldots, y_{L_p}\}$ si elle vérifie :*

$$\forall (i,j) \in [\![1, ..., L_p]\!]^2,\ l_j(y_i) = \delta_{ij} = \begin{cases} 1 \text{ si } i = j \\ 0 \text{ si } i \neq j \end{cases} \tag{2.7}$$

Chaque polynôme d'une base de Lagrange est un polynôme qui interpole une fonction qui vaut 1 sur un des points d'interpolation et s'annule sur les autres. Il en découle que, par définition de l'unisolvance, si un ensemble de points d'interpolation Y est unisolvant, la base de Lagrange qui lui est associée existe et est unique. L'unique modèle qui interpole une fonction F aux points de Y peut alors être construit simplement grâce à l'expression :

$$m(x) = \sum_{i=1}^{L_p} F(y_i) l_i(x) \tag{2.8}$$

De façon équivalente au paragraphe précédent, on peut montrer que l'ensemble d'interpolation Y est unisolvant si et seulement si la base de polynômes de Lagrange qui lui est associée existe et est unique.

Numériquement, la construction des polynômes de Lagrange nécessite la résolution d'un système linéaire pour chacun des L_p points de l'ensemble d'interpolation Y, c'est donc un processus d'une complexité en $\mathcal{O}(L_p^3)$. D'un autre côté, si les points de Y ne sont remplacés que un par un, il est possible de mettre à jour ces polynômes en un coût de l'ordre de $\mathcal{O}(L_p^2)$. Pour les applications dans lesquelles l'évaluation de la fonction objectif en un point est très long devant la résolution d'un système linéaire, ce coût de calcul est marginal. Lorsque ce n'est pas le cas, l'optimisation du code de calcul de ces polynômes de Lagrange est nécessaire pour assurer de bonnes performances des algorithmes d'optimisation.

2.2.3 Λ-unisolvance des ensembles d'interpolation

Dans un algorithme d'optimisation sans dérivées, il est nécessaire que les modèles d'interpolation conduisent à une bonne approximation de la fonction objectif sur un domaine fermé $B \subset \mathbb{R}^p$. On a montré dans la section précédente que pour pouvoir construire un modèle d'interpolation, l'ensemble des points d'interpolation Y devait être unisolvant. Pourtant, cette condition seule d'unisolvance ne donne aucune information sur la qualité du modèle sur B. On a donc besoin d'une façon de mesurer la qualité de la répartition des points de Y dans B. Une mesure fréquemment utilisée est celle de Λ-unisolvance.

Définition 4 (Λ-unisolvance)**.** *Soit B un sous-ensemble fermé de $\mathbb{R}^p$, $Y = \{y_1, y_2, \ldots, y_{L_p}\}$ un ensemble de points d'interpolation unisolvant, $\Lambda > 0$ et $\{l_1, \ldots, l_{L_p}\}$ les polynômes de Lagrange associés à Y. Y est dit Λ-unisolvant ("Λ-poised") si :*

$$\max_{1 \leq i \leq L_p} \max_{x \in B} |l_i(x)| \leq \Lambda$$

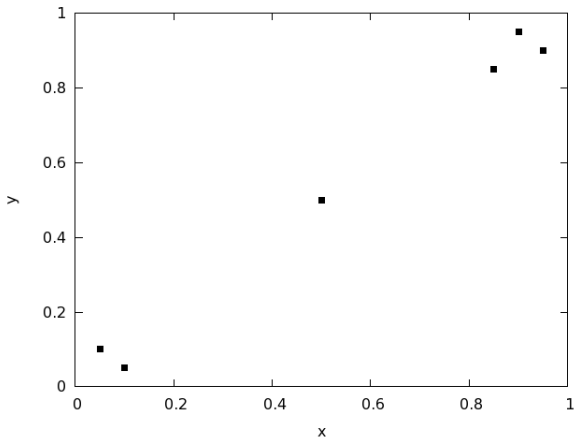
La Λ-unisolvance peut être interprétée comme une sorte de mesure d'inverse de la distance à un ensemble non-unisolvant ($\Lambda \geq 1$ et plus Λ est grand, plus l'ensemble d'interpolation est proche de la dépendance linéaire). La figure 2.2 montre par exemple deux ensembles de points unisolvants en dimension 2. Le premier est "presque" non-unisolvant (tous les points sont proches d'une même droite) alors que le second est bien réparti dans la boule $B(0, 1)$. La Λ-unisolvance donne une bonne indication sur la qualité de ces sous-ensembles. En effet, Le premier ensemble est Λ-unisolvant pour $\Lambda \geq 440$ tandis que le second l'est pour $\Lambda \geq 1$.

Il est intéressant de remarquer que cette définition n'impose pas de condition sur la position des points d'interpolation par rapport au domaine B. Cependant, plus ils en sont éloignés, plus la Λ-unisolvance de l'ensemble Y a des chances d'être élevée. C'est pourquoi dans la plupart des algorithmes d'optimisation fondés sur des modèles d'interpolation, on cherchera d'abord à remplacer les points éloignés de la région de confiance.

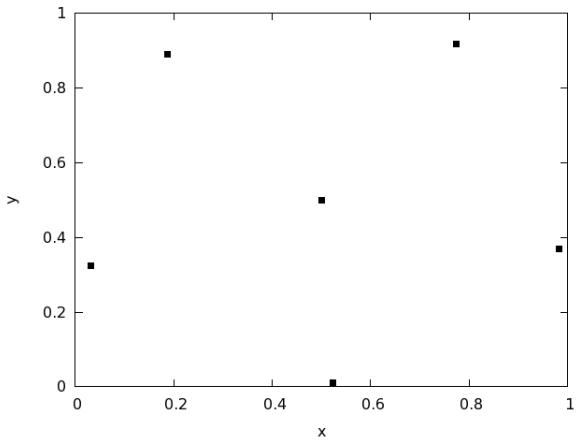
2.2.4 Modèles d'interpolation sous-déterminés

Les modèles construits avec l'une des méthodes de la section 2.2.2 sont faciles à déterminer et généralement assez précis sur une région contenant les points d'interpolation. En revanche, ils nécessitent l'évaluation de F en L_p points, ce qui est beaucoup trop coûteux pour être utilisé dans la pratique. Afin d'être capable d'utiliser nos processus d'optimisation, on a donc besoin de construire des modèles d'interpolation avec un nombre plus restreint de points $q < L_p$. Avec moins de L_p points et pour une base ϕ, la matrice $M(\phi, Y)$ comprend plus de colonnes que de lignes et le système (2.6) ne possède pas de solution unique. Une des possibilités est de choisir la solution α de norme euclidienne minimale. Si $M(\phi, Y)$ est de rang maximal, son pseudo-inverse existe et la solution de norme minimale est donnée par la formule :

$$\alpha = M(\phi, Y)^T [M(\phi, Y) M(\phi, Y)^T]^{-1} F(Y). \tag{2.9}$$



(a) Mauvaise Λ-unisolvance ($\Lambda \geq 440$)



(b) Bonne Λ-unisolvance ($\Lambda \geq 1$)

FIGURE 2.2 – Un ensemble d'interpolation présentant une mauvaise Λ-unisolvance (a) et un autre présentant un bonne Λ-unisolvance (b) (voir [27]).

Cette solution est robuste par rapport aux petites variations de l'ensemble d'interpolation. Elle est cependant dépendante du choix de la base ϕ. Il est donc possible d'obtenir de meilleurs ou de moins bons modèles suivant la base dans laquelle ils sont construits. Identifier la meilleure de ces bases pour un cas donné n'est pas aisé, le choix naturel se porte donc sur la base sur laquelle de nombreux résultats ont été montrés : la base canonique $\bar{\phi}$ de $\mathcal{P}_2^p$.

Dans une optique d'optimisation, il est souvent désirable de construire des modèles avec une partie linéaire précise puis d'ajouter progressivement l'information sur la courbure de la fonction objectif à l'aide de nouveaux points d'interpolation. Si $\bar{\phi} = \{1, x_1, \ldots, x_p, \frac{1}{2}x_1^2, x_1x_2, \ldots, \frac{1}{2}x_p^2\}$ est la base canonique de $\mathcal{P}_2^p$, $\bar{\phi}_L = \{1, \ldots, x_p\}$ sa partie linéaire et $\bar{\phi}_Q = \{\frac{1}{2}x_1^2, x_1x_2, \ldots, \frac{1}{2}x_p^2\}$ sa partie quadratique, on peut écrire le modèle d'interpolation recherché sous la forme :

$$m(x) = \alpha_L^T \bar{\phi}_L(x) + \alpha_Q^T \bar{\phi}_Q(x). \tag{2.10}$$

Si Y contient au moins $p+2$ points, on définit alors le modèle d'interpolation (α_L^*, α_Q^*) comme étant la solution du problème :

$$\begin{cases} (\alpha_L^*, \alpha_Q^*) = argmin \dfrac{1}{2} \|\alpha_Q\|^2 \\ \text{sous la contrainte : } M(\bar{\phi}_L, Y)\alpha_L + M(\bar{\phi}_Q, Y)\alpha_Q = F(Y). \end{cases} \tag{2.11}$$

On appelle la solution de ce problème, le modèle de norme de Frobenius minimale. En effet, en utilisant la base canonique $\bar{\phi}$ et en séparant la solution en $\alpha = (\alpha_L, \alpha_Q)$, minimiser $\|\alpha_Q\|$ est équivalent à minimiser la norme de Frobenius de la Hessienne du modèle m.

Le problème (2.11) possède une unique solution si et seulement si la matrice suivante est non singulière :

$$F(\bar{\phi}, Y) = \begin{bmatrix} M(\bar{\phi}_Q, Y)M(\bar{\phi}_Q, Y)^T & M(\bar{\phi}_L, Y) \\ M(\bar{\phi}_L, Y)^T & 0 \end{bmatrix}. \tag{2.12}$$

Si cette condition est vérifiée, Y est dit unisolvant au sens de la norme de Frobenius. Dans la suite du document, tous les modèles sous-déterminés considérés seront des modèles de norme de Frobenius minimale.

Comme pour le cas de modèles déterminés, on peut définir les polynômes de Lagrange au sens de la norme de Frobenius pour des ensembles Y contenant moins de L_p points : $\alpha_i^*(x) = \alpha_{L,i}^{*T}\Phi_L(x) + \alpha_{Q,i}^{*T}\Phi_Q(x)$, avec $(\alpha_{L,i}^*, \alpha_{Q,i}^*)$ solution du problème

$$\begin{cases} (\alpha_{L,i}^*, \alpha_{Q,i}^*) = argmin \frac{1}{2} \|\alpha_{Q,i}\|^2 \\ \text{sous la contrainte : } M(\bar{\phi}_L, Y)\alpha_{L,i} + M(\bar{\phi}_Q, Y)\alpha_{Q,i} = I_{y_i}(Y). \end{cases} \tag{2.13}$$

où I_{y_i} est la fonction indicatrice du point y_i. Chaque composante de la solution $\alpha_i = (\alpha_{L,i}, \alpha_{Q,i})$ interpole une fonction qui vaut 1 sur un des points de Y et 0 sur les autres. Cette propriété généralise bien les polynômes de Lagrange au cas sous-déterminé.

De la même façon que pour le cas entièrement déterminé, il est possible de définir les modèles d'interpolation de norme de Frobenius minimale à l'aide des polynômes de Lagrange :

$$m(x) = \sum_{i=1}^{q} F(y_i) l_i(x). \tag{2.14}$$

Ici encore, la construction des polynômes de Lagrange nécessite la résolution d'un système linéaire pour chaque point d'interpolation et est donc un processus de complexité $\mathcal{O}(q^3)$. Le processus de mise à jour de ces polynômes lorsqu'un point de Y est remplacé est quant à lui de complexité $\mathcal{O}(q^2)$.

On peut enfin définir de façon analogue au cas déterminé la notion de Λ-unisolvance :

Définition 5. *Soit B un sous-ensemble fermé de $\mathbb{R}^p$, $Y = \{y_1, y_2, \ldots, y_q\}$ un ensemble de points d'interpolation unisolvant au sens de la norme de Frobenius ($q \leq L_p$), $\Lambda > 0$ et $\{l_1, \ldots, l_q\}$ les polynômes de Lagrange associés à Y. Y est dit Λ-unisolvant (Λ-poised) si :*

$$\max_{1 \leq i \leq q} \max_{x \in B} |l_i(x)| \leq \Lambda$$

Comme auparavant, la Λ-unisolvance donne un moyen de mesurer la distance de l'ensemble d'interpolation Y à un ensemble non-unisolvant.

2.2.5 Initialisation des modèles quadratiques

Il est en général trop coûteux d'évaluer F en L_p points lors de la construction du premier modèle quadratique m_0 dans un algorithme sans dérivées du type de l'algorithme 4. C'est pourquoi on travaille souvent avec un modèle sous-déterminé de norme de Frobenius minimale. De bons résultats numériques sont généralement obtenus lorsqu'on choisit un nombre de points d'interpolation initial égal à $2p + 1$. Il est possible de commencer avec moins de points d'interpolation (deux points peuvent suffire pour débuter l'algorithme) mais le choix de $2p+1$ points nous a paru raisonnable pour garantir un bon modèle initial. Afin de maximiser la précision du modèle m_0, on cherche à répartir au mieux ces points dans la région de confiance. Plus précisément, étant donnés initialement un point x_0 et un rayon de région de confiance Δ_0, on choisit l'ensemble d'interpolation initial $Y = \{y_0, y_1, ..., y_{2p}\}$ comme suit :

$$y_0 = x_0 \text{ et } \forall i \in \{0, 1, ..., p-1\} \begin{cases} y_{i+1} = y_0 + \Delta_0 e_i \\ y_{i+p+1} = y_0 - \Delta_0 e_i \end{cases} \tag{2.15}$$

où e_i représente le $i^{\text{ème}}$ vecteur de la base canonique de $\mathbb{R}^p$.
Un tel ensemble d'interpolation est Λ-unisolvant dans $B(x_0, \Delta_0)$ pour $\Lambda = 1$.

2.2.6 Précision des modèles et amélioration de la Λ-unisolvance

2.2.6.1 Modèles entièrement linéaires

Pour assurer la convergence d'un algorithme de région de confiance, il est nécessaire de garantir que le rayon de la région de confiance ne tende pas vers 0 hors des points stationnaires. Si les modèles locaux sont construits à l'aide des dérivées de la fonction objectif, il est facile de montrer qu'une diminution de la taille de la région de confiance induit une amélioration de la précision du modèle avec la proposition suivante :

Proposition 2. *On suppose que la fonction objectif F est deux fois continûment dérivable avec une Hessienne continue et Lipschitzienne. Si on considère un modèle quadratique construit à partir des dérivées de F sur une boule $B(x_0, \Delta)$, alors il existe des constantes k_{eh}, k_{eg} et k_{ef} telles que*

— *l'erreur entre la Hessienne du modèle et celle de F vérifie*

$$\forall x \in B(x_0, \Delta),\ \left\|\nabla^2 F(x) - \nabla^2 m(x)\right\| \leq k_{eh}\Delta,$$

— *l'erreur entre le gradient du modèle et celui de F vérifie*

$$\forall x \in B(x_0, \Delta),\ \|\nabla F(x) - \nabla m(x)\| \leq k_{eg}\Delta^2,$$

— *l'erreur entre le modèle et F vérifie*

$$\forall x \in B(x_0, \Delta),\ \|F(x) - m(x)\| \leq k_{ef}\Delta^3.$$

Dans le cas où les modèles sont construits sans prendre en compte d'information sur la dérivée, on va chercher à avoir des propriétés similaires à la proposition 2. Pour cela, on commence par introduire la notion de modèles entièrement linéaires.

Définition 6. *Soient un ensemble $S \subset \mathbb{R}^p$ et un rayon maximal Δ_{max} donnés. On suppose que la fonction objectif F est continûment dérivable avec un gradient continu et Lipschitzien sur $\cup_{x\in S} B(x, \Delta_{max})$. Un ensemble de modèles $\mathcal{M}$ est appelé classe de modèles entièrement linéaire si :*

1. *Il existe des constantes k_{eg} et k_{ef} telles que pour tout $x_0 \in S$ et $\Delta \leq \Delta_{max}$, il existe un modèle m de $\mathcal{M}$ tel que*
   — *l'erreur entre le gradient du modèle et celui de F vérifie*

   $$\forall x \in B(x_0, \Delta),\ \|\nabla F(x) - \nabla m(x)\| \leq k_{eg}\Delta$$

   — *l'erreur entre le modèle et F vérifie*

   $$\forall x \in B(x_0, \Delta),\ \|F(x) - m(x)\| \leq k_{ef}\Delta^2.$$

   Un tel modèle m est dit entièrement linéaire sur $B(x_0, \Delta)$.
2. *Pour cette classe $\mathcal{M}$, il existe un algorithme d'amélioration de modèle qui peut, en un nombre fini d'étapes :*
   — *soit garantir qu'un modèle $m \in \mathcal{M}$ est entièrement linéaire sur $B(x_0, \Delta)$*
   — *soit trouver un modèle $\tilde{m} \in \mathcal{M}$ qui est entièrement linéaire sur $B(x_0, \Delta)$.*

Disposer d'une famille de modèles vérifiant cette définition est nécessaire pour les preuves de convergence d'algorithmes à région de confiance. On peut alors garantir qu'un algorithme utilisant des modèles entièrement linéaires converge vers un point critique du premier ordre. On peut de la même façon définir les modèles entièrement quadratiques.

Définition 7. *Soient un ensemble S et un rayon maximal Δ_{max} donnés. On suppose que la fonction objectif F est deux fois continûment dérivable avec une Hessienne continue Lipschitz sur $\cup_{x\in S} B(x, \Delta_{max})$. Un ensemble de modèle $\mathcal{M}$ est appelé classe de modèle entièrement quadratique si*

1. *Il existe des constantes k_{eg}, k_{ef} et k_{eh} telles que pour tout $x \in S$ et $\Delta \leq \Delta_{max}$, il existe un modèle m de $\mathcal{M}$ tel que*
   - *l'erreur entre la Hessienne du modèle et celle de F vérifie*

$$\forall x \in B(x_0, \Delta),\ \left\|\nabla^2 F(x) - \nabla^2 m(x)\right\| \leq k_{eh}\Delta$$

   - *l'erreur entre le gradient du modèle et celui de F vérifie*

$$\forall x \in B(x_0, \Delta),\ \|\nabla F(x) - \nabla m(x)\| \leq k_{eg}\Delta^2$$

   - *l'erreur entre le modèle et F vérifie*

$$\forall x \in B(x_0, \Delta),\ \|F(x) - m(x)\| \leq k_{ef}\Delta^3.$$

2. *Pour cette classe $\mathcal{M}$, il existe un algorithme d'amélioration de modèle qui peut en un nombre fini d'étapes*
   - *soit garantir qu'un modèle $m \in \mathcal{M}$ est entièrement quadratique sur $B(x_0, \Delta)$*
   - *soit trouver un modèle $\tilde{m} \in \mathcal{M}$ qui est entièrement quadratique.*

Cette propriété plus forte permet d'assurer un comportement du modèle identique à un modèle construit avec des informations sur les dérivées. Elle donne ainsi la possibilité d'apporter des preuves de convergences de certains algorithmes vers un point critique d'ordre 2.

2.2.6.2 Lien avec la Λ-unisolvance

Il est possible de montrer qu'il existe des bornes similaires à celles données dans la définition 6 sur les erreurs entre des modèles construits avec un ensemble Λ-unisolvant et la fonction interpolée [27]. On travaille sous l'hypothèse suivante :

Hypothèse 1. *On suppose que la fonction objectif F est deux fois continûment dérivable avec une Hessienne continue et Lipschitzienne sur un domaine ouvert S contenant le point x_0 et la région de confiance $B(x_0, \Delta)$.*

Le premier résultat dans le cas de modèles déterminés est le suivant :

Proposition 3. *Sous l'hypothèse 1, on suppose qu'on dispose d'un point x_0 et d'un ensemble d'interpolation $Y = \{y_0, ..., y_{L_p}\}$ entièrement déterminé et Λ-unisolvant sur $B(x_0, \Delta)$ pour un certain Λ. Alors, il existe des constantes k_{eh}, k_{eg} et k_{ef} ne dépendant que de Λ, L_p et de la constante de Lipschitz de $\nabla^2 F$ telles que*

- *l'erreur entre la Hessienne du modèle et celle de F vérifie*

$$\forall x \in B(x_0, \Delta),\ \left\|\nabla^2 F(x) - \nabla^2 m(x)\right\| \leq k_{eh}\Delta,$$

- *l'erreur entre le gradient du modèle et celui de F vérifie*

$$\forall x \in B(x_0, \Delta),\ \|\nabla F(x) - \nabla m(x)\| \leq k_{eg}\Delta^2,$$

- *l'erreur entre le modèle et F vérifie*

$$\forall x \in B(x_0, \Delta),\ \|F(x) - m(x)\| \leq k_{ef}\Delta^3.$$

Dans le cas de modèles sous-déterminés, on ne peut que vérifier l'entière linéarité :

Proposition 4. *Sous l'hypothèse 1, on suppose qu'on dispose d'un point x_0 et d'un ensemble d'interpolation $Y = \{y_0, ..., y_q\}$ sous-déterminé et Λ-unisolvant sur $B(x_0, \Delta)$ pour un certain Λ. Alors, il existe des constantes k_{eg} et k_{ef} ne dépendant que de Λ, q et de la constante de Lipschitz de $\nabla^2 F$ telles que*

— *l'erreur entre le gradient du modèle et celui de F vérifie*

$$\forall x \in B(x_0, \Delta),\ \|\nabla F(x) - \nabla m(x)\| \leq k_{eg}\Delta,$$

— *l'erreur entre le modèle et F vérifie*

$$\forall x \in B(x_0, \Delta),\ \|F(x) - m(x)\| \leq k_{ef}\Delta^2.$$

Il est aussi possible de montrer les bornes suivantes :

Proposition 5. *Étant données une boule $B(x_0, \Delta)$, un ensemble d'interpolation Y Λ-unisolvant sur $B(x_0, \Delta)$ et sa base de polynômes de Lagrange associée $\{l_1, \ldots, l_q\}$, il existe des constantes $k_f > 0$ et $k_g > 0$ telles que, pour un modèle d'interpolation quadratique m et un point quelconque $y \in B(x_0, \Delta)$:*

$$\begin{cases} \|F(y) - m(y)\| \leq k_f \sum_{j=1}^{q} \|y_j - y\|^2 |l_j(y)| \\ \|\nabla F(y) - \nabla m(y)\| \leq k_g \Lambda \Delta. \end{cases}$$

La Λ-unisolvance d'un modèle permet donc concrètement d'estimer la qualité d'un modèle, ce qui s'avère très utile dans le cadre d'un algorithme d'optimisation.

2.2.6.3 Amélioration de la géométrie des ensembles d'interpolation

On a besoin à ce stade d'un algorithme capable, pour un ensemble d'interpolation Y donné, de :

— déterminer si Y est Λ-unisolvant ou non sur une boule B,

— modifier Y de façon à ce qu'il soit Λ-unisolvant sur B.

L'algorithme 5 donne une façon de modifier Y itérativement pour le rendre unisolvant en utilisant les polynômes de Lagrange.
On peut montrer le résultat suivant pour l'algorithme 5 [27] :

Lemme 1. *L'algorithme 5 se termine en un nombre fini d'itérations avec un ensemble Y Λ-unisolvant sur B.*

On dispose ainsi d'un outil permettant d'assurer en temps fini la Λ-unisolvance d'un ensemble d'interpolation donné. On peut donc à présent construire un algorithme beaucoup plus robuste que l'algorithme 4.

2.2.7 Un algorithme classique : NEWUOA

Étant donné un modèle quadratique, les propriétés des sections précédentes nous apportent un moyen de déterminer s'il est entièrement linéaire et de l'améliorer itérativement s'il ne l'est pas. Il est donc possible de remédier aux défauts de l'algorithme simpliste numéro 4. Dans cet algorithme, la réduction de la taille de la région de confiance ne garantissait pas une amélioration de la qualité du modèle. Cependant, ce problème disparait si on ajoute la possibilité de garantir

Algorithme 5 Un algorithme d'amélioration de la Λ-unisolvance

1. **Initialisation**
   — Générer un ensemble d'interpolation $Y = \{y_1, ..., y_q\}$.
   — Choisir une boule fermée $B \subset \mathbb{R}^p$ et une constante $\Lambda > 1$.
   — Construire les polynômes de Lagrange de norme de Frobenius minimale associés à Y, notés $\{l_1, ..., l_q\}$.
   — Fixer $k = 0$.
2. **Itération** k
   — Estimer
   $$\Lambda_k = \max_{1 \leq i \leq q} \max_{x \in B} l_i(x).$$
   — Si $\Lambda_k \leq \Lambda$, Y est Λ-unisolvant : sortir de l'algorithme.
   — Si $\Lambda_k > \Lambda$, soit i un indice tel que
   $$\max_{x \in B} |l_i(x)| > \Lambda$$
   et x_k un point de B tel que $|l_i(x_k)| > \Lambda$.
   Mettre à jour Y en remplaçant le point y_i par x_k.
   — Mettre à jour les polynômes de Lagrange.

l'entière linéarité des modèles. En effet, comme dans le cas des modèles calculés avec les dérivées des fonctions objectifs, il ne sera pas possible pour la région de confiance de diminuer indéfiniment hors des points critiques de F.

On peut noter que dans une optique d'optimisation, il n'est pas nécessaire de garantir la Λ-unisolvance du modèle à toutes les itérations de l'algorithme. Si à une certaine itération k le modèle a bien prédit la diminution de la fonction objectif, on pourra accepter le point d'essai x_k^+ sans vérifier si le modèle est proche de la fonction objectif sur toute la région de confiance. On introduit donc la quantité $\rho_k = \frac{F(x_k) - F(x_k^+)}{m_k(x_k) - m_k(x_k^+)}$ qui nous servira de critère pour déterminer si une itération est fructueuse ou non. On décrit dans l'algorithme 6 un algorithme classique très proche de l'algorithme NEWUOA de Powell [84, 86], mais aussi de l'algorithme SQA développé à IFPEN ([63]).

Algorithme 6 Algorithme NEWUOA ou SQA (variante IFPEN)

1. **Initialisation**
   - Choisir un point initial x_0 et un rayon de région de confiance Δ_0.
   - Choisir un ensemble d'interpolation Y_0 contenant $t \in [\![p+2, L_p]\!]$ points, dont x_0, et évaluer F sur tous les points de Y_0.
   - Construire le modèle d'interpolation de F initial, noté m_0.
   - Choisir les constantes $\epsilon > 0$, $\epsilon_0 > 0$, $q = 0$, $0 < \nu < 1$, $\gamma_{inc} \geq 1$, $0 < \gamma_{low} < 1$, $0 < \mu < 1$, $0 < \theta < 1$ et $\Lambda > 1$.
2. **Étape critique**
   (a) Fixer $\widehat{m_q} = m_k$.
   (b) Si $\|\nabla \widehat{m_q}(x_k)\| < \epsilon$, arrêter l'algorithme et retourner x_k.
   (c) Si $\|\nabla \widehat{m_q}(x_k)\| < \epsilon_q$, fixer $\epsilon_{q+1} = \mu \|\nabla \widehat{m_q}(x_k)\|$, construire un nouveau modèle Λ-unisolvant $\widehat{m_{q+1}}$ dans $B(x_k, \epsilon_{q+1})$ et incrémenter q.
   (d) Si q a été incrémenté, choisir $m_k = \widehat{m_q}$, $\Delta_k = \theta \|\nabla m_k(x_k)\|$ et recommencer l'étape du test critique à partir du pas 2(b).
3. **Iteration k**
   Calculer le point d'essai x_k^+ tel que :
   $$m_k(x_k^+) = \min_{x \in B(x_k, \Delta_k)} m_k(x)$$
   Évaluer $F(x_k^+)$, $\rho_k = \frac{F(x_k) - F(x_k^+)}{m_k(x_k) - m_k(x_k^+)}$, les polynômes de Lagrange $\{l_{k,1}, ..., l_{k,t}\}$ associés à Y_k et $y_{k,max} = arg\max_{y_{k,j} \in Y_k} \left\|y_{k,j} - x_k^+\right\|^2 \left|l_{k,j}(x_k^+)\right|$.
   (a) **Itération fructueuse :** Si $\rho_k \geq \nu$ ou si le modèle m_k est entièrement linéaire (Y_k est Λ-unisolvant) et $\rho_k \geq 0$, choisir $x_{k+1} = x_k^+$ et remplacer $y_{k,max}$ par x_k^+ dans Y_{k+1}.
   (b) **Itération infructueuse :** Si $\rho_k < \nu$, choisir $x_{k+1} = x_k$ et tenter de remplacer un point de Y_k en suivant une itération de l'algorithme 5. Le choix des nouveaux points se fait pour les variantes de Powell à l'intérieur d'une région de confiance de rayon plus petit que Δ_k. Si aucun point de Y_k n'a pu être remplacé, alors Y_k est Λ-unisolvant et on peut garantir l'entière linéarité de m_k.
   (c) **Gestion de la région de confiance :**
   - Si $\rho_k \geq \nu$, choisir $\Delta_{k+1} = \gamma_{inc}\Delta_k$.
   - Si $\rho_k < \nu$ et m_k est entièrement linéaire, choisir $\Delta_{k+1} = \gamma_{low}\Delta_k$.
   - Choisir $\Delta_{k+1} = \Delta_k$ dans les autres cas.
4. **Mise à jour du modèle**
   Calculer le nouveau modèle d'interpolation m_{k+1} centré autour de x_{k+1} à partir des points de Y_{k+1}, puis retourner à l'étape 2.

2.2.8 Auto-correction de la géométrie des ensembles d'interpolation

Comme observé au début de la section 2.2, la figure 2.1 issue de [98] est un exemple qui montre que les étapes d'amélioration de la géométrie de l'ensemble d'interpolation sont nécessaires pour assurer la convergence d'un algorithme à région de confiance vers un point critique de F. Cependant, Fasano et al. ont montré dans [39] que de très bons résultats numériques pouvaient être obtenus en les supprimant. Scheinberg et Toint [98] ont donc développé en 2010

un algorithme basé sur des modèles d'interpolation qui, sans ignorer totalement la qualité des modèles, réduit considérablement le nombre d'évaluations de la fonction objectif réalisé dans le seul but d'améliorer le modèle. L'idée exploitée pour parvenir à ce résultat est simplement d'étudier l'effet de l'addition du point d'essai x_k^+ à l'ensemble d'interpolation à chaque itération de l'algorithme. Il s'avère que pour la plupart des itérations, le point d'essai améliore la géométrie de l'ensemble d'interpolation et retire donc le besoin d'évaluer la fonction objectif en davantage de points.

Pour pouvoir décrire l'algorithme de Scheinberg et Toint [98], on se place dans un contexte d'optimisation sans dérivées avec des modèles d'interpolation du type de NEWUOA (algorithme 6). A l'itération k, le modèle est noté m_k, l'ensemble d'interpolation Y_k, le point d'essai x_k^+, le rayon de la région de confiance Δ_k et les polynômes de Lagrange associés à Y_k $\{l_{k,1}, ..., l_{k,q}\}$. On introduit pour une itération k et deux constantes $\Lambda > 1$ et $\beta > 1$ données :

— l'ensemble F_k des points d'interpolation éloignés de la région de confiance :

$$F_k = \{y_{k,j} \in Y_k \text{ tels que } \|y_{k,j} - x_k\| > \beta\Delta_k \text{ et } l_{k,j}(x_k^+) \neq 0\}, \tag{2.16}$$

— l'ensemble C_k des points proches de la région de confiance qui permettent une amélioration de la géométrie de l'ensemble d'interpolation s'ils sont remplacés par le point d'essai x_k^+

$$C_k = \{y_{k,j} \in Y_k \backslash \{x_k\} \text{ tels que } \|y_{k,j} - x_k\| \leq \beta\Delta_k \text{ et } l_{k,j}(x_k^+) \geq \Lambda\}. \tag{2.17}$$

Le résultat fondamental sur lequel Scheinberg et Toint se basent pour montrer la convergence de leur algorithme vers un point critique du premier ordre est le suivant.

Lemme 2. *Si la fonction objectif F est continûment dérivable et que son gradient est continu et Lipschitzien alors, pour tout $\Lambda > 1$, il existe une constante k_Λ telle que si l'itération k est infructueuse et que :*

$$\begin{cases} F_k = \emptyset, \\ \Delta_k < k_\Lambda \|\nabla m_k(x_k)\|, \end{cases}$$

alors :

$$C_k \neq \emptyset. \tag{2.18}$$

Ce résultat stipule que si tous les points de l'ensemble d'interpolation sont suffisamment proches de la région de confiance ($F_k = \emptyset$) et que le rayon de la région de confiance est suffisamment petit ($\Delta_k < k_\Lambda \|\nabla m_k(x_k)\|$), alors il existe un point dans Y_k dont le remplacement par x_k^+ améliorera la qualité du modèle ($C_k \neq \emptyset$). L'algorithme 7 exploite cette propriété et utilise le point d'essai de chaque itération pour tenter d'améliorer la géométrie de l'ensemble d'interpolation du modèle.

Algorithme 7 Un algorithme sans dérivées utilisant une propriété d'auto-correction des ensembles d'interpolation

1. **Initialisation**
   — Choisir un point initial x_0 et un rayon de région de confiance Δ_0.
   — Choisir un ensemble d'interpolation initial Y_0 contenant x_0 et évaluer F sur tous les points de Y_0.
   — Construire le modèle initial m_0 interpolant F aux points de Y_0.
   — Choisir les constantes $\epsilon > 0$, $\epsilon_0 > 0$, $0 < \nu < 1$, $\gamma_{inc} \geq 1$, $0 < \gamma_{low} < 1$, $\Lambda > 1$, $0 < \mu < 1$, $\theta > 0$ et $\beta \geq 1$.
   — choisir $v_0 \neq x_0$, fixer $q = 0$ et $k = 0$.
2. **Test critique**
   (a) Définir $\widehat{m_q} = m_k$.
   (b) Si $\|\nabla \widehat{m_q}(x_k)\| < \epsilon$, arrêter l'algorithme et retourner x_k.
   (c) Si $\|\nabla \widehat{m_q}(x_k)\| < \epsilon_q$, fixer $\epsilon_{q+1} = \mu \|\nabla \widehat{m_q}(x_k)\|$, construire un nouveau modèle $\widehat{m_{q+1}}$ Λ-unisolvant dans $B(x_k, \epsilon_{q+1})$ et incrémenter q.
   (d) Si q a été incrémenté, choisir $m_k = \widehat{m_q}$, $v_q = x_k$, $\Delta_k = \theta \|\nabla m_k(x_k)\|$ et recommencer l'étape du test critique à partir du pas 2(b).
3. **Itération k**
   Calculer le point d'essai x_k^+ tel que :

$$m_k(x_k^+) = \min_{x \in B(x_k, \Delta_k)} m_k(x).$$

   Évaluer $F(x_k^+)$ et $\rho_k = \frac{F(x_k) - F(x_k^+)}{m_k(x_k) - m_k(x_k^+)}$.
   (a) **Itération fructueuse :** Si $\rho_k \geq \nu$, définir $x_{k+1} = x_k^+$, $\Delta_{k+1} = \gamma_{inc} \Delta_k$ et $Y_{k+1} = Y_k \backslash \{y_{k,r}\} \cup \{x_k^+\}$ avec $y_{k,r}$ tel que :

$$y_{k,r} = arg \max_{y_{k,j} \in Y_k} \left\| y_{k,j} - x_k^+ \right\|^2 \left| l_{k,j}(x_k^+) \right|.$$

   (b) **Remplacer un point d'interpolation éloigné :** Si $\rho_k < \nu$, et que, soit $x_k \neq v_q$, soit $\Delta_k \leq \epsilon_q$ et l'ensemble $F_k \neq \emptyset$, alors choisir $x_{k+1} = x_k$, $\Delta_{k+1} = \Delta_k$ et $Y_{k+1} = Y_k \backslash \{y_{k,r}\} \cup \{x_k^+\}$ où $y_{k,r}$ est un point de F_k, par exemple :

$$y_{k,r} = arg \max_{y_{k,j} \in F_k} \left\| y_{k,j} - x_k^+ \right\|^2 \left| l_{k,j}(x_k^+) \right|$$

   (c) **Remplacer un point d'interpolation proche :** Si $\rho_k < \nu$, et que, soit $x_k \neq v_q$, soit $\Delta_k \leq \epsilon_q$, l'ensemble $F_k = \emptyset$ et l'ensemble $C_k \neq \emptyset$, alors choisir $x_{k+1} = x_k$, $\Delta_{k+1} = \Delta_k$ et $Y_{k+1} = Y_k \backslash \{y_{k,r}\} \cup \{x_k^+\}$ où $y_{k,r}$ est un point de C_k, par exemple :

$$y_{k,r} = arg \max_{y_{k,j} \in C_k} \left\| y_{k,j} - x_k^+ \right\|^2 \left| l_{k,j}(x_k^+) \right|$$

   (d) **Réduire la région de confiance :** Si $\rho_k < \nu$, et que, soit $x_k = v_q$ et $\Delta_k > \epsilon_q$, soit $F_k \cup C_k = \emptyset$, alors choisir $x_{k+1} = x_k$, $\Delta_{k+1} = \gamma_{low} \Delta_k$ et $Y_{k+1} = Y_k$.
4. **Mise à jour du modèle :** Calculer le nouveau modèle d'interpolation m_{k+1} centré autour de x_{k+1} à partir des points de Y_{k+1}. Incrémenter k et retourner à l'étape 2.

L'idée exploitée par l'algorithme 7 est de faire en sorte de se mettre le plus souvent possible dans la situation des hypothèses du lemme 2. Les étapes 3(b), 3(c) et 3(d) décrivent les différents cas qui se présentent lorsqu'une itération infructueuse survient :

(i) s'il existe des points d'interpolation éloignés de la région de confiance, on en remplace un par le point d'essai,

(ii) s'il n'y a pas de point éloigné mais que C_k est non vide, alors on peut améliorer la géométrie de Y_k en remplaçant un point de C_k par x_k^+,

(iii) si aucun des deux cas précédents ne survient, on peut réduire la région de confiance sans difficulté puisque :
   — soit le modèle n'est pas entièrement linéaire et on se rapproche des conditions du lemme 2, lequel garantira que C_k sera non vide lors d'une prochaine itération,
   — soit le modèle est entièrement linéaire et la réduction de la région de confiance diminuera l'écart entre le modèle et la fonction objectif.

On peut donc voir que, comme pour l'algorithme NEWUOA (algorithme 6), il n'est pas possible pour le rayon de la région de confiance de diminuer indéfiniment hors des points critiques de F.

2.3 Une nouvelle méthode de type région de confiance pour les fonctions partiellement séparables : l'algorithme DFO-PSOF

On présente dans cette section une méthode d'optimisation sans dérivées développée durant la thèse de façon à être adaptée aux fonctions partiellement séparables. On rappelle qu'une fonction est dite partiellement séparable si elle peut s'écrire sous la forme :

$$\begin{gathered} F(x_1, \ldots, x_p) = \sum_{i=1}^{n} f_i(x_{i,1}, \ldots, x_{i,p_i}) \\ \text{avec } \forall i \in \{1, ..., n\}, p_i < p \end{gathered} \tag{2.19}$$

L'objectif ici est d'exploiter au maximum cette propriété de façon à réduire le nombre d'évaluations de la fonction objectif pour atteindre un optimum local. Au lieu de créer un modèle quadratique pour la fonction objectif complète F comme dans l'algorithme 6 NEWUOA, l'idée principale consiste à créer un modèle pour chaque sous-fonction f_i : à l'itération k de l'algorithme, le modèle m_k de F s'écrit alors

$$m_k(x_1, ..., x_p) = \sum_{i=1}^{n} m_k^i(x_{i,1}, \ldots, x_{i,p_i}) \tag{2.20}$$

On espère ainsi avoir besoin de moins de points pour construire le modèle tout en réduisant l'erreur commise entre le modèle et la fonction objectif.

La manière la plus simple d'obtenir un algorithme exploitant la propriété de la fonction objectif serait de choisir un algorithme classique du type de NEWUOA et d'adapter chaque étape pour gérer un modèle par sous-fonction. A une itération k, on pourrait par exemple :

(a) Si l'itération est fructueuse, remplacer un point dans chaque sous-ensemble d'interpolation Y_k^i par x_k^+.

(b) Si l'itération n'est pas fructueuse et que tous les ensembles d'interpolation ne sont pas entièrement linéaires, remplacer un point de chaque ensemble d'interpolation non entièrement linéaire en suivant une itération de l'algorithme 5.

Plusieurs problèmes apparaissent avec une telle adaptation de NEWUOA. D'abord, l'initialisation du premier modèle m_0 peut potentiellement requérir plus d'évaluations de la fonction objectif si la séparabilité partielle de F est prise en compte. En effet, si dans NEWUOA on choisit d'initialiser m_0 avec $t(p)$ points (par exemple $2p+1$ points), on devra, pour garder la même précision dans l'algorithme adapté, initialiser chaque sous-modèle avec $t(p_i)$ points ($2p_i+1$ points dans l'exemple). Une telle initialisation nécessite l'évaluation de $\sum_{i=1}^{n} t(p_i)$ points. Ce nombre est dépendant de la structure de F et peut très largement excéder $t(p)$, ce qui n'est pas acceptable dans une optique de réduction du coût de l'optimisation.

De plus, afin de maintenir la qualité de tous les sous-modèles au cours de l'optimisation, l'algorithme adapté pourrait requérir à chaque itération l'évaluation de F en plusieurs points. Le coût de telles évaluations peut très vite devenir élevé sans garantir une accélération de la convergence de l'algorithme. Colson et Toint [23] on proposé un moyen de réduire le nombre de points à évaluer dans une étape similaire mais leur algorithme nécessite toujours l'évaluation potentielle de la fonction objectif en de multiples points à chaque itération.

Nous proposons dans la suite de la section une méthode pour initialiser le modèle m_0 de façon plus efficace ainsi qu'une généralisation de la propriété d'auto-correction de la géométrie aux fonctions partiellement séparables.

2.3.1 Initialisation de l'algorithme DFO-PSOF

Pour faire en sorte que le modèle initial m_0 soit suffisamment précis dans un algorithme comme NEWUOA, il est courant de travailler avec des ensembles d'interpolation contenant $2p+1$ points. Ce choix pour le nombre de points à évaluer initialement offre un bon compromis entre coût et précision du modèle puisque qu'il permet d'utiliser de l'information sur la courbure de la fonction objectif tout en restant de l'ordre de p. Dans cette partie, on initialisera donc chacun des sous-modèles avec $2p_i + 1$ points.

(i) Un exemple simple (cas entièrement séparable)
On commence par considérer un exemple très simple dans lequel F ne dépend que de 2 paramètres x_1 et x_2, et peut s'écrire sous la forme :

$$F(x_1, x_2) = f_1(x_1) + f_2(x_2) \tag{2.21}$$

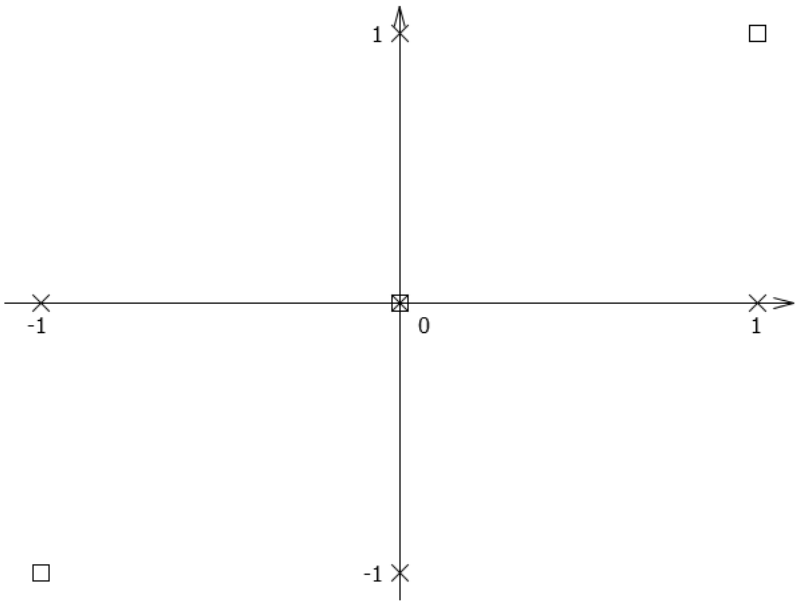


FIGURE 2.3 – Les croix sont les points à évaluer pour construire un modèle quadratique avec $2p+1$ points sans exploiter la séparabilité partielle tandis que les carrés sont ceux utilisés pour construire les modèles de f_1 et f_2.

On veut construire un modèle d'interpolation m de F autour de $x_0 = (0,0)$ dans la boule $B(x_0, 1)$. Pour construire directement m avec $2p + 1 = 5$ points, on choisit généralement les points représentés par les croix sur la figure 2.3. Avec les mêmes points, on peut construire les modèles initiaux m_1 et m_2 de f_1 et f_2 : l'initialisation de m_1 se fait avec les 3 points $(-1, 0)$, $(0, 0)$ et $(1, 0)$ tandis que celle de m_2 se fait à l'aide des points $(0, -1)$, $(0, 0)$ and $(0, 1)$. Cependant, cette façon de faire ne prend pas en compte le fait que f_1 est indépendant de x_2 et que f_2 est indépendant de x_1. En fait, sans perdre d'information, on peut réduire les points d'interpolation aux seuls 3 points représentés par des carrés sur la figure 2.3 : le point $(1, 1)$ peut remplacer à la fois les points $(0, 1)$ et $(1, 0)$ tandis que $(-1, -1)$ peut remplacer $(0, -1)$ et $(-1, 0)$ pour la construction des modèles m_1 and m_2.

C'est cette façon de faire que l'on va généraliser à l'ensemble des fonctions partiellement séparables dans la suite. Pour cela, on commence par chercher à déterminer les indépendances entre les différentes variables à l'aide d'un graphe coloré. Une fois ces indépendances identifiées, on perturbe les points d'interpolation simultanément dans toutes les directions mutuellement indépendantes.

(ii) Coloration des variables

Afin de généraliser l'exemple précédent, on commence par modéliser les dépendances des sous-fonctions de F par un graphe. Ce graphe suit les règles suivantes [58] :

— chaque paramètre x_i est représenté par un nœud i,

— s'il existe une sous-fonction f_k de F qui dépend à la fois des variables x_i et x_j , une arête est créée entre les nœuds i et j du graphe.

Exemple : on veut construire le graphe de dépendance de la fonction

$$F(x_1, x_2, x_3, x_4) = f_1(x_1, x_2) + f_2(x_2, x_3) + f_3(x_3, x_4) + f_4(x_4, x_1) + f_5(x_2, x_4) \tag{2.22}$$

Le graphe qui lui est associé est schématisé dans la figure 2.4.

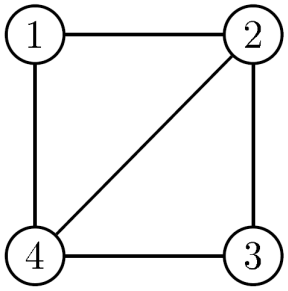


FIGURE 2.4 – Graphe de dépendance de la fonction F définie en (2.22).

Afin d'identifier les différentes indépendances entre les variables, on colore le graphe de dépendance [28, 22]. On cherche donc à attribuer une couleur à chaque nœud du graphe de dépendance de la fonction objectif en suivant les deux règles suivantes :

— on colore avec le moins de couleurs différentes possibles,

— deux nœuds reliés par un arête ne peuvent pas avoir la même couleur.

Avec un graphe de dépendance coloré, il devient facile d'identifier les variables indépendantes entre elles. En effet, deux nœuds i et j du graphe auront la même couleur seulement s'il n'existe pas de sous-fonction de F qui dépende à la fois de x_i et de x_j. On montre en figure 2.5 la coloration du graphe de dépendance de F.

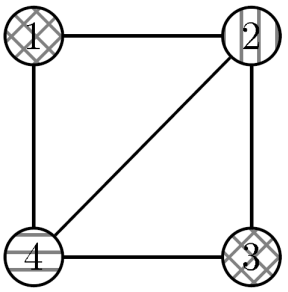


FIGURE 2.5 – Graphe de dépendance coloré de la fonction F (2.22).

Pour la fonction F, les paramètres x_1 et x_3 sont indépendants (il n'existe pas de sous-fonction de F qui dépende à la fois de x_1 et de x_3), on peut donc attribuer la même couleur aux nœuds 1 et 3.

(iii) Initialisation des sous-modèles
Pour que les sous-modèles m_0^i aient une certaine précision, on veut toujours qu'ils soient chacun construits avec $2p_i + 1$ points d'interpolation. Afin d'y parvenir, on exploite les indépendances des différents paramètres. On dit que deux paramètres x_1 et x_2 sont indépendants si les sous-fonctions qui dépendent d'un des paramètres ne dépendent pas de l'autre. Par conséquent, la valeur du paramètre x_2 des points d'interpolation servant à construire les modèles dépendants de x_1 n'a pas d'influence. La coloration du graphe de dépendance de la fonction objectif nous permet d'identifier ces indépendances.

Soit c le nombre de couleurs différentes utilisées pour colorer le graphe de dépendance de F. On choisit seulement $2c + 1$ points $Y_0 = \{y_0, ..., y_{2c}\}$ pour initialiser tous les sous-modèles sur $B(x_0, \Delta_0)$:

- $y_0 = x_0$
- $\forall i \in \{1, ..., c\}$, on appelle c_i la $i^{\text{ème}}$ couleur et I_i l'ensemble des indices des points colorés par la couleur c_i
- Les points choisis sont alors

$$\forall i \in \{0, ..., c-1\} \begin{cases} y_{i+1} = y_0 + \Delta_0 \sum_{j \in I_i} e_j \\ y_{i+c+1} = y_0 - \Delta_0 \sum_{j \in I_i} e_j \end{cases} \tag{2.23}$$

Le nombre de nœuds du graphe de dépendances étant le nombre de paramètres dont dépend F, on a toujours $c \leq p$. On est donc parvenu à réduire systématiquement le nombre de points à évaluer pour la construction du modèle initial m_0 sans influer sur sa précision. Ce nombre d'évaluation dépend en fait uniquement de la séparation de F. Un exemple simple permettant de comprendre le gain potentiel apporté par cette façon de faire est celui des fonctions entièrement séparables. Si F peut s'écrire sous la forme

$$F(x_1, ..., x_p) = f_1(x_1) + \cdots + f_p(x_p) \tag{2.24}$$

dans ce cas, tous les paramètres sont indépendants et sont donc colorés de la même couleur. Cette propriété implique que, quelque soit le nombre de paramètres p dont dépend F, on pourra toujours construire le premier modèle m_0 à l'aide de seulement 3 points. En fait, moins il y a d'interactions entre les variables, moins le nombre de points requis pour construire m_0 sera élevé.

2.3.2 Une propriété d'auto-correction de la géométrie des ensembles d'interpolation

On a vu en début de section 2.3 que le maintien de la qualité de tous les sous-modèles dans un algorithme à région de confiance pouvait avoir un coût beaucoup trop important pour pouvoir être utilisé dans la pratique. Afin de réduire ce coût, on généralise le lemme 2 de la section précédente qui stipule qu'un algorithme basé sur des modèles d'interpolation corrige généralement de lui-même la géométrie de l'ensemble d'interpolation.

Avant de donner quelques notations et définitions, on remarque que chaque sous-modèle m_k^i ne dépend que de vecteurs de dimension p_i appartenant au sous-espace $Vect\{e_{i,1}, \ldots, e_{i,p_i}\}$. Un sous-modèle m_k^i est donc construit à partir de l'évaluation de f_i aux projections orthogonales

des points d'interpolation sur le sous-espace associé. Afin de ne pas alourdir les notations, on garde ces projections implicites dans toute la suite.

On introduit à présent la notion de sous-modèle "dominant" :

Définition 8. *Dans un algorithme à région de confiance adapté aux fonctions partiellement séparables, à l'itération k, le sous-modèle m_k^j est appelé sous-modèle dominant si*

$$j = arg \max_{i \in \{1,\dots,n\}} \big(m_k^i(x_k) - m_k^i(x_k^+)\big). \tag{2.25}$$

À l'itération k, le sous-modèle dominant est simplement celui qui a le plus contribué à la diminution du modèle global m_k.

De façon similaire à la section 2.2.8, on introduit les ensembles F_k^i et C_k^i :

Définition 9. *On se place à l'itération k d'un algorithme à région de confiance adapté aux fonctions partiellement séparables. Pour tout indice i, si Y_k^i est l'ensemble d'interpolation du sous-modèle m_k^i, $\beta \geq 1$ et $\Lambda > 1$ des réels et $\{l_{k,h}\}_h$ les polynômes de Lagrange associés à Y_k^i, alors les ensembles F_k^i et C_k^i sont respectivement définis par :*

— *l'ensemble des points d'interpolation de m_k^i éloignés de la région de confiance*

$$F_k^i = \{y_{k,h} \in Y_k^i \text{ tels que } \|y_{k,h} - x_k\| > \beta\Delta_k \text{ et } l_{k,h}(x_k^+) \neq 0\}, \tag{2.26}$$

— *l'ensemble des points d'interpolation de m_k^i qui permettent d'améliorer le modèle s'ils sont remplacés par le point d'essai x_k^+*

$$C_k^i = \{y_{k,h} \in Y_k^i \backslash \{x_k\} \text{ tels que } \|y_{k,h} - x_k\| \leq \beta\Delta_k \text{ et } l_{k,h}(x_k^+) \geq \Lambda\}. \tag{2.27}$$

On définit ensuite les quantités ρ_k et ρ_k^i pour chaque sous-modèle m_k^i qui servent de critère pour décider si la qualité des modèles est suffisante :

Définition 10. *Dans un algorithme à région de confiance adapté aux fonctions partiellement séparables, si x_k^+ est le point d'essai proposé à l'itération k et $i \leq n$ est l'indice d'une sous-fonction f_i, on définit :*

$$\rho_k = \frac{F(x_k) - F(x_k^+)}{m_k(x_k) - m_k(x_k^+)} \tag{2.28}$$

et

$$\rho_k^i = \frac{f_i(x_k) - f_i(x_k^+)}{m_k^i(x_k) - m_k^i(x_k^+)} \tag{2.29}$$

Pour simplifier les notations dans l'algorithme, pour une itération k et un indice $i \in \{1, ..., n\}$ d'une sous-fonction f_i, on définit enfin p_k^i et L_p^i.

Définition 11. *Soit p_k^i le nombre de points d'interpolation utilisés pour construire le sous-modèle m_k^i et L_p^i le nombre de points d'interpolation nécessaire pour que le modèle m_k^i soit déterminé :*

$$\begin{cases} p_k^i = \left|Y_k^i\right| \\ L_p^i = \frac{(p_i+1)(p_i+2)}{2}. \end{cases} \tag{2.30}$$

Avec les notations précédentes, on pourra énoncer un résultat d'auto-correction de la géométrie des ensembles d'interpolation. Ce résultat (énoncé et démontré en section 2.4) stipule que, sous certaines conditions sur la région de confiance, il est possible d'améliorer la Λ-unisolvance de l'ensemble d'interpolation du sous-modèle dominant en y remplaçant un point par le point d'essai x_k^+ proposé par l'algorithme à l'itération k. On montrera en section 2.4 que cette propriété est suffisante pour montrer la convergence de l'algorithme associé, présenté ci-dessous (algorithme 8).

2.3.3 Description de l'algorithme DFO-PSOF

Avec les propriétés et définitions précédentes, on peut à présent décrire le nouvel algorithme à région de confiance adapté aux fonctions partiellement séparables (algorithme 8), appelé DFO-PSOF ("Derivative Free Optimization for Partially Separable Functions").

Algorithme 8 Algorithme d'optimisation sans dérivées adapté aux fonctions partiellement séparables (DFO-PSOF).

---

1. **Initialisation**
   - Choisir un point initial x_0 et un rayon de région de confiance Δ_0.
   - Choisir des ensembles d'interpolation unisolvants Y_0^i suivant la méthode montrée en section 2.3.1 et évaluer F sur tout les points des ensembles Y_0^i.
   - Construire les modèles initiaux m_0^i interpolant les sous-fonctions f_i sur les ensembles Y_0^i.
   - Construire le modèle initial global $m_0 = \sum_{i=1}^{n} m_0^i$.
   - Choisir les constantes $q = 0$, $0 < \nu < 1$, $0 < \gamma < 1$, $0 < \mu \leq 1$, $\beta > 1$, $\epsilon_0 > 0$, $\epsilon > 0$, $\theta > 0$ et $\Lambda > 1$
   - Fixer k à 0 et choisir $v_0 \neq x_0$.
2. **Test critique**
   (a) Fixer $\widehat{m_q} = m_k$.
   (b) Si $\|\nabla \widehat{m_q}(x_k)\| < \epsilon$, arrêter l'algorithme et retourner x_k.
   (c) Si $\|\nabla \widehat{m_q}(x_k)\| < \epsilon_q$ fixer $\epsilon_{q+1} = \mu \|\nabla \widehat{m_q}(x_k)\|$, calculer un nouveau modèle Λ-unisolvant $\widehat{m_{q+1}}$ dans $B(x_k, \epsilon_{q+1})$ et incrémenter q.
   (d) Si q a été incrémenté, choisir $m_k = \widehat{m_q}$, $v_q = x_k$, $\Delta_k = \theta \|\nabla m_k(x_k)\|$ et recommencer l'étape critique à partir du pas 2(b).
3. **Itération k**
   (a) **Calculer le point d'essai :**
      - Calculer x_k^+ tel que $m_k(x_k^+) = \min_{x \in B(x_k, \Delta_k)} m_k(x)$.
      - Calculer pour tout $i \in \{1, \ldots, n\}$, $f_i(x_k^+)$, $f(x_k^+)$, ρ_k^i, et ρ_k.
   (b) **Suivre la procédure de gestion des sous-modèles (algorithme 9)**
   (c) **Gestion de la région de confiance :**
   Soit j l'indice du sous-modèle dominant m_k^j :
      - Si $\rho_k \geq \nu$, choisir $x_{k+1} = x_k^+$ et $\Delta_{k+1} \geq \Delta_k$.
      - Sinon, si $C_k^j \cup F_k^j = \emptyset$, choisir $x_{k+1} = x_k$ et $\Delta_{k+1} = \Delta_k$.
      - Sinon, si $\rho_k^j > \nu$ et au moins un point a été ajouté dans un sous-modèle dans l'étape 2 de la procédure de gestion des ensembles d'interpolation (algorithme 9), choisir $x_{k+1} = x_k$ et $\Delta_{k+1} = \Delta_k$.
      - Sinon si $\rho_k^j > \nu$ et aucun point n'a été ajouté à l'étape 2 de l'algorithme de la procédure de gestion des ensembles d'interpolation (algorithme 9), choisir $x_{k+1} = x_k$ et $\Delta_{k+1} = \gamma \Delta_k$.
      - Sinon si soit $x_k = v_q$ et $\Delta_k \geq \epsilon_q$ soit $C_k^j \cup F_k^j = \emptyset$, choisir $x_{k+1} = x_k$ et $\Delta_{k+1} = \gamma \Delta_k$.
   (d) **Incrémenter k, mettre à jour les modèles et retourner à l'étape 2**

Dans cet algorithme, on essaie d'exploiter au maximum un équivalent du lemme 2, en l'occur-

Algorithme 9 Gestion de l'ensemble d'interpolation des sous-modèles à l'itération k de l'algorithme 8.

1. **Pour tous les sous-modèles m_k^i, faire :**
   (a) Si m_k^i n'est pas entièrement déterminé ($p_k^i < L_p^i$),
   choisir $Y_{k+1}^i = Y_k^i \cup \{x_k^+\}$.
   (b) Sinon, si l'itération k est fructueuse ($\rho_k > \nu$),
   choisir $Y_{k+1}^i = Y_k^i \backslash \{y_{k,r}\} \cup \{x_k^+\}$ où $r = arg\max_h \left\|y_{k,h} - x_k^+\right\|^2 \left|l_{k,h}(x_k^+)\right|$.
   (c) Sinon, si $\rho_k^i \geq \nu$ et que soit $x_k \neq v_q$, soit $\Delta_k < \epsilon_q$,
   choisir $Y_{k+1}^i = Y_k^i \backslash \{y_{k,r}\} \cup \{x_k^+\}$ où $r = arg\max_h \left\|y_{k,h} - x_k^+\right\|^2 |l_{k,h}(x_k)|$.
   (d) Sinon, si soit $x_k \neq v_q$, soit $\Delta_k < \epsilon_q$ et que $F_k^i \neq \emptyset$,
   remplacer un point de F_k^i par x_k^+ dans Y_k^i.
   (e) Sinon, si soit $x_k \neq v_q$, soit $\Delta_k < \epsilon_q$ et que $C_k^i \neq \emptyset$
   remplacer un point de C_k^i par x_k^+ dans Y_k^i.
2. **Si : soit $x_k \neq v_q$, soit $\Delta_k < \epsilon_q$, et que $\rho_k < \nu$ et $\rho_k^j \geq \nu$, améliorer les sous-modèles non dominants :**
   Pour tous les sous-modèles tels que $\rho_k^i < \nu$, fixer $y_{k,r} = arg\max_{y_{k,h} \in Y_k^i} \left\|y_{k,h} - x_k^+\right\|^2 \left|l_{k,h}(x_k^+)\right|$. Si Y_k^i n'es pas Λ-unisolvant, remplacer $y_{k,r}$ en suivant une itération de l'algorithme 5.

rence le lemme 4 énoncé ultérieurement en section 2.4 concernant la propriété d'auto-correction de la géométrie des sous-ensembles d'interpolation. À chaque itération, soit on fait en sorte de se rapprocher des hypothèses du lemme 4, soit on garantit qu'au moins le sous-modèle dominant de l'itération est amélioré. L'algorithme ne requiert donc presque jamais l'évaluation de F en de nouveaux points uniquement pour assurer la qualité des modèles. Des points supplémentaires sont toutefois nécessaires dans le rare cas où l'itération globale est infructueuse mais le sous-modèle dominant est précis (étape 2 de l'algorithme permettant la gestion des ensembles d'interpolation, algorithme 9). Dans ce cas, on recommande d'améliorer tous les sous-modèles non-dominants qui ne sont pas assez prédictifs (tels que $\rho_k^i < \nu$). Il est cependant intéressant de noter que la convergence de l'algorithme peut toujours être prouvée avec l'amélioration d'un unique sous-modèle. Cela signifie que si un sous-modèle non dominant a été amélioré à l'étape 1 de l'algorithme 9, aucun point supplémentaire n'a besoin d'être évalué dans l'étape 2 du même algorithme.

Un autre point intéressant à signaler est le fait que, similairement à l'algorithme proposé par Tröltzsch dans [107], tant que les sous-modèles ne sont pas entièrement déterminés, l'algorithme DFO-PSOF continue d'ajouter les points d'essai dans leurs ensembles d'interpolation respectifs. L'une des spécificités du cas des fonctions partiellement séparables est en effet le fait qu'un sous-modèle est entièrement déterminé lorsqu'il est construit avec L_p^i points. Si F est correctement séparée (p_i est relativement petit devant p pour tout i), L_p^i est beaucoup plus petit que L_p, ce qui entraine une obtention beaucoup plus rapide de modèles entièrement déterminés. Dans ce cas, les modèles quadratiques de la fonction objectif seront précis avec beaucoup moins d'évaluations de la fonction objectif que dans un algorithme d'optimisation sans dérivées classique.

2.4 Preuve de convergence de l'algorithme DFO-PSOF

On se place dans le cadre des définitions de la section 2.3.2 et sous les hypothèses suivantes :

Hypothèse 2. *On suppose que :*
- *F est continûment dérivable et son gradient est continu et Lipschitzien de constante de Lipschitz $\frac{1}{2}L$,*
- *il existe une constante $k_{low} \in \mathbb{R}$ telle que $F(x) \geq k_{low}$ pour tout $x \in \mathbb{R}^p$,*
- *il existe une constante $k_H \geq L$ telle que pour toutes les itérations de l'algorithme, $1 + \|H_k\| \leq k_H$ où $H_k = \nabla^2 m_k(x_k)$.*

Le résultat de convergence de l'algorithme 8 vers un point critique d'ordre 1 est le suivant :

Théorème 1. *Sous les hypothèses précédentes, la suite $(x_k)_{k\in\mathbb{N}}$ construite dans l'algorithme 8 (DFO-PSOF) est telle que :*

$$\lim_{k\to+\infty} \inf \|\nabla F(x_k)\| = 0$$

Dans l'algorithme 8, le point d'essai est obtenu à l'itération k par minimisation du modèle m_k sur la région de confiance. On ne requiert en fait ici que la condition de Cauchy de descente, à savoir l'existence de $0 < k_c \leq 1$ tel que pour tout $k \in \mathbb{N}$:

$$m_k(x_k) - m_k(x_k^+) \geq k_c \|g_k\| \min\left(\frac{\|g_k\|}{1 + \|H_k\|}, \Delta_k\right) \tag{2.31}$$

où g_k est le gradient du modèle m_k en x_k et H_k la Hessienne.
Pour démontrer le théorème, on aura besoin du Lemme issu de [98] et rappelé ci-dessous :

Lemme 3. *Supposons que pour des réels $\{\alpha_i\}_{1\leq i\leq t}$ tels que*

$$\sigma_{abs} = \sum_{i=1}^{t} |\alpha_i| > 2\sum_{i=1}^{t} \alpha_i = \sigma > 0,$$

si on définit

$$i^* = arg \max_{i=1,\dots,t} |\alpha_i| \ \ et\ j^* = arg \max_{\substack{j=1,\dots,t \\ j\neq i^*}} |\alpha_j|$$

alors

$$|\alpha_{j^*}| \geq \frac{\sigma_{abs} - 2\sigma}{2t} \tag{2.32}$$

On montre maintenant le résultat central sur la nature auto-correctrice de l'algorithme 8 :

Lemme 4. *À l'itération k de l'algorithme 8, pour tout $\Lambda > 1$, il existe une constante k_Λ telle que :*

$$si \begin{cases} \text{- } j \text{ est l'indice du sous-modèle dominant} \\ \text{- } \Delta_k \leq k_\Lambda \|g_k\| \\ \text{- } \rho_k^j < \nu \\ \text{- } F_k^j = \emptyset \end{cases}$$

alors

$$C_k^j \neq \emptyset.$$

Démonstration. On montre le résultat pour la valeur particulière de k_Λ suivante :

$$k_\Lambda = \min\left(\frac{1}{k_H}, \frac{(1-\nu)\frac{k_c}{n}}{2k_f(\beta+1)^2(L_p^{max}\Lambda+1)}\right)$$

où $L_p^{max} = max_{i\in[\![1,\dots,n]\!]}L_p^i$ et k_f est la constante donnée dans la proposition 5. Sous les hypothèses et notations du lemme 4, on a

$$\rho_k^j = \frac{f_j(x_k) - f_j(x_k^+)}{m_k^j(x_k) - m_k^j(x_k^+)} < \nu.$$

Comme les sous-modèles interpolent les sous-fonctions f_i, on a aussi $\forall i \in \{1,...,n\}$, $f_i(x_k) = m_k^i(x_k)$. Cela implique

$$\begin{gathered} f_j(x_k^+) > (1-\nu)m_k^j(x_k) + \nu m_k^j(x_k^+), \\ \text{et donc } \left|f_j(x_k^+) - m_k^j(x_k^+)\right| > (1-\nu)\left|m_k^j(x_k) - m_k^j(x_k^+)\right|. \end{gathered}$$

De plus, la proposition 5 donne l'existence d'une constante $k_f > 0$ tel que

$$\left|f_j(x_k^+) - m_k^j(x_k^+)\right| \leq k_f \sum_{h=1}^{p_k^j} \left\|y_{k,h}^j - x_k^+\right\|^2 \left|l_{k,h}^j(x_k^+)\right|.$$

D'un autre côté, les hypothèses donnent $F_k^j = \emptyset$, donc

$$\forall h \in \{1,...,p_k^j\} \text{ tel que } l_{k,h}^j(x_k^+) \neq 0,\ \left\|y_{k,h}^j - x_k\right\| \leq \beta\Delta_k.$$

Ce qui donne, $\forall h \in \{1,...,p_k^j\}$ tel que $l_{k,h}^j(x_k^+) \neq 0$,

$$\left\|y_{k,h}^j - x_k^+\right\| \leq \left\|y_{k,h}^j - x_k\right\| + \left\|x_k^+ - x_k\right\| \leq (\beta+1)\Delta_k$$

et donc

$$(1-\nu)\left|m_k^j(x_k) - m_k^j(x_k^+)\right| < \left|f_j(x_k^+) - m_k^j(x_k^+)\right| \leq k_f(\beta+1)^2\Delta_k^2 \sum_{h=1}^{p_k^j}\left|l_{k,h}^j(x_k^+)\right|.$$

La condition de Cauchy donne quant à elle :

$$m_k(x_k) - m_k(x_k^+) \geq k_c \left\|g_k\right\| \min\left[\frac{\|g_k\|}{1+\|H_k\|}, \Delta_k\right].$$

Or, avec notre choix de k_Λ, on a par hypothèse :

$$\Delta_k \leq \frac{\|g_k\|}{k_H} \leq \frac{\|g_k\|}{1+\|H_k\|}$$

Donc, la condition de Cauchy peut se réécrire :

$$m_k(x_k) - m_k(x_k^+) \geq k_c \left\|g_k\right\| \Delta_k.$$

L'indice j correspondant au sous-modèle dominant, on a d'autre part

$$m_k^j(x_k) - m_k^j(x_k^+) \geq \frac{1}{n}(m_k(x_k) - m_k(x_k^+))$$

ce qui implique, avec la condition de Cauchy :

$$m_k^j(x_k) - m_k^j(x_k^+) \geq \frac{k_c}{n} \|g_k\| \Delta_k$$

et

$$\sum_{h=1}^{p_k^j} \left| l_{k,h}^j(x_k^+) \right| \geq \frac{(1-\nu)\frac{k_c}{n} \|g_k\|}{k_f(\beta+1)^2 \Delta_k},$$

et donc, avec l'hypothèse prise sur Δ_k,

$$\sum_{h=1}^{p_k^j} \left| l_{k,h}^j(x_k^+) \right| \geq 2(L_p^{max}\Lambda + 1) \geq 2(L_p^j\Lambda + 1) \geq 2(p_k^j\Lambda + 1).$$

Les $l_{k,h}^j$ étant des polynômes de Lagrange, on a aussi

$$\sum_{h=1}^{p_k^j} l_{k,h}^j(x_k^+) = 1.$$

Le lemme 3 rappelé précédemment garantit que si $m = arg\max_{h=1,\ldots,p_k^j} \left| l_{k,h}^j(x_k^+) \right|$,

$$\left| l_{k,r}^j(x_k^+) \right| \geq \Lambda \text{ pour } r = arg \max_{\substack{h=1,\ldots,p_k^j \\ h \neq m}} \left| l_{k,h}^j(x_k^+) \right|.$$

Cette dernière inégalité combinée au fait que $F_k^j = \emptyset$ prouve que $C_k^j \neq \emptyset$.

□

Le lemme suivant fait le lien entre la qualité des sous-modèles et la qualité du modèle global. Plus précisément, si tous les sous-modèles sont Λ-unisolvants et que la région de confiance est suffisamment petite, alors l'itération ne peut pas être infructueuse.

Lemme 5. *À l'itération k, si tous les ensembles d'interpolation Y_k^i sont Λ-unisolvants et*

$$\Delta_k < \min\left((1-\nu)\frac{k_c}{k_{ef}n}, \frac{1}{k_H} \right) \|g_k\|$$

où k_{ef} est la constante donnée par la proposition 4, alors $\rho_k > \nu$.

Démonstration. Par définition

$$\begin{aligned} \rho_k &= \frac{f(x_k) - f(x_k^+)}{m_k(x_k) - m_k(x_k^+)} = \frac{\sum_{i=1}^n \left(f_i(x_k) - f_i(x_k^+) \right)}{m_k(x_k) - m_k(x_k^+)} \\ &= \frac{\sum_{i=1}^n \left(f_i(x_k) - m_k^i(x_k) - f_i(x_k^+) + m_k^i(x_k^+) \right)}{m_k(x_k) - m_k(x_k^+)} + 1, \end{aligned}$$

La condition de Cauchy (2.31) donne comme précédemment :

$$m_k(x_k) - m_k(x_k^+) > k_c \|g_k\| \Delta_k,$$

tandis que les propriétés d'interpolation donnent :

$$f_i(x_k) = m_k^i(x_k),$$

et la Λ-unisolvance garantit l'entière linéarité d'après la proposition 4 :

$$\forall i \in \{1, \ldots, n\}, \left| f_i(x_k^+) - m_k^i(x_k^+) \right| \leq k_{ef} \Delta_k^2.$$

Les équations et inégalités précédentes impliquent alors que :

$$\begin{aligned} \left| \frac{\sum_{i=1}^n \left(f_i(x_k) - m_k^i(x_k) - f_i(x_k^+) + m_k^i(x_k^+) \right)}{m_k(x_k) - m_k(x_k^+)} \right| &< \frac{k_{ef} \Delta_k^2 n}{k_c \left\| g_k \right\| \Delta_k} \\ &< (1 - \nu) \end{aligned}$$

ce qui prouve donc le résultat annoncé, à savoir $\rho_k > \nu$. □

Avec ces premiers résultats, on peut à présent suivre les étapes classiques de preuve de convergence des méthodes à région de confiance. On commence par montrer que le rayon de la région de confiance Δ_k ne peut pas devenir arbitrairement petit hors d'un point critique.

Lemme 6. *Supposons qu'il existe $c > 0$ et $k_0 \geq 0$ tels que pour $k \geq k_0$:*

$$\|g_k\| \geq c,$$

alors il existe une constante $k_\Delta > 0$ telle que pour tout $k \geq k_0$

$$\Delta_k > k_\Delta.$$

Démonstration. Supposons que pour une itération $k \geq k_0$, $\Delta_k < \min\left((1-\nu)\frac{k_c}{k_{ef} n}, \frac{1}{k_H}, k_\Lambda, \mu \right) c$.

Soit k_q l'indice du modèle créé par la dernière étape de test critique, on a

$$\Delta_k < \mu c \leq \mu \left\| g_{k_q} \right\| = \epsilon_q,$$

donc $\Delta_k < \epsilon_q$.
Seuls cinq cas peuvent alors survenir :

1. $\rho_k \geq \nu$: dans ce cas, l'étape de gestion de la région de confiance (étape 3(c)) garantit que $\Delta_{k+1} \geq \Delta_k$.
2. $\rho_k < \nu$, $\rho_k^j \geq \nu$ et au moins un des sous-modèles n'est pas Λ-unisolvant : dans ce cas l'algorithme suit l'étape d'amélioration des sous-modèles non dominants (étape 2 de l'algorithme 9) ce qui garantit que $\Delta_{k+1} = \Delta_k$.
3. $\rho_k < \nu$, $\rho_k^j \geq \nu$ et tous les sous-modèles sont Λ-unisolvants : les hypothèses sur Δ_k impliquent que les conditions du lemme 5 sont remplies, ce qui implique que $\rho_k > \nu$. Une contradiction est levée, ce cas ne peut donc jamais se produire dans ces conditions.
4. $\rho_k < \nu$, $\rho_k^j < \nu$ et $F_k^j \neq \emptyset$: dans ce cas, l'étape de gestion de la région de confiance (étape 3(c)) entraine $\Delta_{k+1} = \Delta_k$.
5. $\rho_k < \nu$, $\rho_k^j < \nu$ et $F_k^j = \emptyset$: on a $\Delta_k < k_\Lambda \left\| g_k \right\|$, le lemme 4 implique donc que $C_k^j \neq \emptyset$ et par conséquent $\Delta_{k+1} = \Delta_k$.

Aucun de ces cas ne cause une diminution de la région de confiance. Une condition pour que la région de confiance se réduise est donc :

$$\Delta_k \geq \min\left((1-\nu)\frac{k_c}{k_{ef}n}, \frac{1}{k_H}, k_\Lambda, \mu\right) c$$

ce qui prouve donc le résultat avec $k_\Delta = \gamma \min\left((1-\nu)\frac{k_c}{k_{ef}n}, \frac{1}{k_H}, k_\Lambda, \mu\right) c$ □

On considère à présent le cas où le nombre d'itérations fructueuses est fini :

Lemme 7. *Supposons que seul un nombre fini d'itérations fructueuses ($\rho_k \geq \nu$) ne survienne. Alors*

$$\lim_{k\to+\infty} \inf \|g_k\| = 0$$

Démonstration. Sous les hypothèses précédentes, toutes les itérations deviennent infructueuses pour k grand : $\exists k_0 \geq 0$ tel que $\forall k \geq k_0$, $\rho_k < \nu$. Cela implique qu'à partir du rang k_0, les itérés x_k de l'algorithme deviennent invariants, c'est à dire $\forall k \geq k_0$, $x_{k+1} = x_k = x_*$.
Afin de trouver une contradiction, on suppose qu'il existe $c > 0$ tel que pour tout $k \geq k_0$, $\|g_k\| > c$. Le lemme 6 donne

$$\Delta_{k+1} > k_\Delta > 0.$$

Comme $\forall k \geq k_0$, $\rho_k < \nu$, la suite $(\Delta_k)_{k\geq k_0}$ est décroissante et minorée et est donc convergente. On appelle $\Delta_\infty = \lim_{k\to\infty} \Delta_k \geq k_\Delta$.
$\|g_k\|$ étant minoré par $c > 0$, il n'est pas possible pour l'algorithme de boucler infiniment dans l'étape critique (étape 2). De plus, les étapes dans lesquels Δ_k diminue entrainent :

$$\Delta_{k+1} = \gamma\Delta_k \text{ avec } \gamma < 1.$$

Puisque (Δ_k) converge vers une valeur strictement positive, l'algorithme ne peut pas boucler infiniment dans ces étapes. Les seuls étapes restantes dans l'algorithme sont :

- $\rho_k^j > \nu$ et il existe des sous-modèles non dominants et non Λ-unisolvants tels que $\rho_k^i < \nu$: comme x_k est constant pour $k \geq k_0$ et que le nombre d'étapes nécessaires à l'obtention d'un modèle Λ-unisolvant est fini, l'algorithme ne peut entrer qu'un nombre fini de fois dans cette étape.
- $\rho_k^j \leq \nu$ et $F_k^j \neq \emptyset$: l'algorithme ne peut entrer dans cette étape qu'un nombre fini de fois avant que tous les points d'interpolation de tous les sous-modèles ne soient dans $B(x_*, \Delta_\infty)$.
- $\rho_k^j \leq \nu$ et $C_k^j \neq \emptyset$. À chaque itération, l'algorithme remplace un point de Y_k^j tel que le polynôme de Lagrange associé possède la propriété $\left|l(x_k^+)\right| \geq \Lambda$, ce qu'il est impossible de faire indéfiniment.

On a levé une contradiction, on peut donc conclure que

$$\lim_{k\to+\infty} \inf \|g_k\| = 0.$$

□

Examinons à présent le cas où un nombre infini d'itérations fructueuses se présente :

Lemme 8. *Si un nombre infini d'itérations fructueuses arrive, on a*

$$\lim_{k\to+\infty} \inf \|g_k\| = 0.$$

Démonstration. De façon à lever une contradiction, on suppose qu'il existe $c > 0$ tel que pour tout $k \geq k_0$, $\|g_k\| > c$. Les hypothèses du lemme 6 sont remplies. Il existe donc $k_\Delta > 0$ tel que pour tout k suffisamment grand

$$\Delta_k > k_\Delta.$$

Pour toutes les itérations fructueuses, on a

$$F(x_k) - F(x_k^+) > \nu(m_k(x_k) - m_k(x_k^+)).$$

De plus, la condition de Cauchy (2.31) garantit que le point d'essai x_k^+ est choisi tel que

$$m_k(x_k) - m_k(x_k^+) > k_c \|g_k\| \min\left(\frac{\|g_k\|}{1 + \|H_k\|}, \Delta_k\right).$$

Par conséquent

$$F(x_k) - F(x_k^+) > \nu k_c c \min\left(\frac{c}{k_H}, k_\Delta\right) = k_d > 0.$$

En additionnant toutes ces inégalités, comme il y a un nombre infini d'itérations fructueuses, on peut conclure

$$\lim_{k\to\infty} F(x_k) = -\infty,$$

ce qui contredit le fait que F est minorée.
On a levé une contradiction, on peut donc en conclure que :

$$\lim_{k\to+\infty} \inf \|g_k\| = 0.$$

□

Pour pouvoir finaliser, on a besoin de deux résultats supplémentaires issus de [98] :

Lemme 9. *Pour chaque itération, on a*

$$\left|F(x_k^+) - m_k(x_k^+)\right| \leq \|\nabla F(x_k) - g_k\| \, \Delta_k + k_H \Delta_k^2.$$

Démonstration. Comme montré dans [98], pour que cette inégalité se réalise, on a seulement besoin que $F(x_k) = m_k(x_k)$ pour chaque itéré x_k. Cette propriété est assurée par l'algorithme 8. □

Lemme 10. *Supposons que*

(i) $g_k \neq 0$,

(ii) $\|\nabla F(x_k) - g_k\| \leq \frac{1}{2} k_c (1 - \nu) \|g_k\|$,

(iii) $\Delta_k \leq \frac{k_c}{2k_H}(1 - \nu) \|g_k\|$.

Alors l'itération k est fructueuse.

Démonstration. Avec le lemme précédent, ce résultat est une conséquence directe de la condition de Cauchy. □

Nous avons à présent tous les outils pour montrer le résultat final, en l'occurrence la convergence de la limite inf de $\|\nabla F(x_k)\|$ vers 0 (théorème 1) :

Démonstration. Afin de trouver une contradiction, supposons qu'il existe c_g et $k_0 \geq 0$ tels que

$$\forall k \geq k_0,\ \|\nabla F(x_k)\| \geq c_g.$$

Les deux lemmes 7 et 8 garantissent que quel que soit $\epsilon_q > 0$, il existe une itération k_q tel que $\left\|g_{k_q}\right\| < \epsilon_q$ au début de l'étape du test critique (étape 2). Cela implique qu'un nouveau jeu de sous-modèles est généré à l'itération k_q. Comme c'est vrai pour tout ϵ_q, les suites (k_q) et (ϵ_q) sont infinies et $\lim_{q\to\infty} \epsilon_q = 0$.
Considérons q suffisamment grand pour que

$$\epsilon_q \leq \frac{1}{2}\min\left(\frac{k_c(1-\nu)}{2k_g\Lambda}, \gamma\theta, \frac{\gamma k_c(1-\nu)}{2k_H}\right)c_g. \tag{2.33}$$

Après l'étape du test critique de l'itération k_q, le modèle m_{k_q} est Λ-unisolvant sur $B(x_{k_q}, \mu\epsilon_q)$. La proposition 5 entraine donc

$$\left\|\nabla F(x_{k_q}) - g_{k_q}\right\| \leq k_g\Lambda\mu\epsilon_q < k_g\Lambda\epsilon_q.$$

L'inégalité (2.33) ci-dessus implique alors

$$\left\|\nabla F(x_{k_q}) - g_{k_q}\right\| < k_g\Lambda\epsilon_q < \frac{1}{4}k_c(1-\nu)c_g < \frac{1}{4}k_c(1-\nu)\left\|\nabla F(x_{k_q})\right\|$$

et donc, a fortiori

$$\left\|\nabla F(x_{k_q}) - g_{k_q}\right\| \leq \frac{1}{2}\left\|\nabla F(x_{k_q})\right\|.$$

On a à présent l'inégalité

$$\left\|g_{k_q}\right\| = \left\|\nabla F(x_{k_q}) + g_{k_q} - \nabla F(x_{k_q})\right\| \geq \left\|\nabla F(x_{k_q})\right\| - \left\|\nabla F(x_{k_q}) - g_{k_q}\right\| \geq \frac{1}{2}\left\|\nabla F(x_k)\right\|,$$

ce qui implique

$$\left\|\nabla F(x_{k_q}) - g_{k_q}\right\| \leq \frac{1}{2}k_c(1-\nu)\left\|g_{k_q}\right\|.$$

Par conséquent, pour une telle valeur de q, il n'y a pas de boucle infinie dans l'étape du test critique. On a aussi, à la fin de l'étape du test critique de l'itération k_q,

$$\Delta_{k_q} = \theta\left\|g_{k_q}\right\| \geq \frac{1}{2}\theta c_g \geq \frac{\epsilon_q}{\gamma} > \epsilon_q.$$

Comme un nouveau modèle est généré par l'étape du test critique à l'itération k_q, on a $v_q = x_{k_q}$. De plus, la structure de l'algorithme entraine qu'après une telle génération, le modèle ne peut être modifié que si une itération fructueuse arrive ou que la région de confiance diminue trop ($\Delta_k < \epsilon_q$). L'algorithme peut diminuer la région de confiance pour les itérations $k \geq k_q$ jusqu'à

ce que $\Delta_k < \frac{k_c(1-\nu)c_g}{2k_H}$. Les hypothèses du lemme 10 sont alors vérifiées et une itération fructueuse $k_s \geq k_q$ arrive avec

$$\Delta_{k_s} \geq \frac{\gamma k_c(1-\nu)c_g}{4k_H} = \Delta_{lim} \geq \epsilon_q$$

Puisque le modèle n'a pas été modifié entre les itérations k_q et k_s, on a

$$\|g_{k_s}\| = \|g_{k_q}\| \geq \frac{1}{2}c_g$$

Puisque k_s est une itération fructueuse, le point d'essai $x_{k_s}^+$ est accepté en tant qu'itéré suivant x_{k_s+1} et

$$\frac{F(x_{k_s}) - F(x_{k_s+1})}{m_k(x_{k_s}) - m_k(x_{k_s+1})} > \nu$$

ce qui implique, avec la condition de Cauchy :

$$F(x_{k_s}) - F(x_{k_s+1}) > \nu \|g_{k_s}\| k_c \min\left(\frac{\|g_{k_s}\|}{1+\|H_{k_s}\|}, \Delta_{lim}\right) > \frac{1}{2}\nu c_g k_c \min\left(\frac{\frac{1}{2}c_g}{k_H}, \Delta_{lim}\right) > 0.$$

Ce scénario est répété pour chaque q tel que

$$\epsilon_q \leq \frac{1}{2}\min\left(\frac{k_c(1-\nu)}{2k_g\Lambda}, \gamma\theta, \frac{\gamma k_c(1-\nu)}{2k_H}\right)c_g,$$

ce qui entraine $\liminf_{k\to+\infty} F(x_k) = -\infty$. L'hypothèse que F est minorée est alors contredite. La conclusion suit :

$$\lim_{k\to+\infty} \inf \|\nabla F(x_k)\| = 0$$

□

2.5 Premiers résultats numériques avec l'algorithme DFO-PSOF

L'algorithme DFO-PSOF a été entièrement implémenté en C++ et intégré dans l'outil d'optimisation interne à IFP Énergies Nouvelles HUBopt (un diagramme de classe est présenté en annexe). Cette implémentation a permis de tester l'algorithme à la fois indépendamment sur des fonctions tests analytiques et sur des problèmes industriels de calage d'historique. On présente ici les premiers tests qui nous ont permis de valider numériquement cet algorithme.

2.5.1 Fonctions analytiques partiellement séparables

On montre ici les résultats numériques obtenus sur des fonctions analytiques partiellement séparables, en l'occurrence :

$$
\begin{aligned}
DQDRTIC(x) &= \sum_{i=1}^{p-2} (x_i^2 + 100x_{i+1}^2 + 100x_{i+2}^2) \\
LIARWHD(x) &= \sum_{i=1}^{p} 4(x_i^2 - x_1)^2 + (x_i - 1)^2 \\
BDQRTIC(x) &= \sum_{i=1}^{p-4} ((-4x_i + 3)^2 + (x_i^2 + 2x_{i+1}^2 + 3x_{i+2}^2 + 4x_{i+3}^2 + 5x_n^2)^2) \\
ARWHEAD(x) &= \sum_{i=1}^{p-1} ((x_i^2 + x_n^2)^2 - 4x_i + 3) \\
ROSENBROCK(x) &= \sum_{i=1}^{p-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2).
\end{aligned}
$$

Lorsque le nombre de variables p est suffisamment grand, chacune de ces fonctions est difficile à minimiser avec un algorithme d'optimisation sans dérivées classique. Cependant, elles s'écrivent toutes comme la somme de sous-fonctions qui dépendent d'un relativement petit nombre de paramètres. Le nouvel algorithme présenté ici est donc supposé donner de bons résultats comparativement aux méthodes classiques.

On teste d'abord la capacité de l'algorithme DFO-PSOF à traiter des problèmes en grande dimension : l'une des qualités souhaitée de cet algorithme est que sa vitesse de convergence dépend plus de la propriété de séparation de la fonction objectif que du nombre total de paramètres. La figure 2.6 montre le nombre d'évaluations de la fonction objectif nécessaire pour arriver à la convergence de l'algorithme en fonction de la dimension p du problème. On a choisi ici de stopper l'algorithme lorsque la région de confiance devient trop petite ($\Delta_k \leq 10^{-3}$). Il est clair sur cette figure que le nombre total de paramètres n'influe que marginalement sur le nombre d'évaluations nécessaire pour atteindre un optimum. Ce résultat valide numériquement les intuitions qui ont mené à la création de l'algorithme. Il est très important lorsqu'on considère l'application industrielle de calage d'historique : si on est capable de correctement identifier les séparations de la fonction objectif, on pourra en effet augmenter le nombre de paramètres sans accroitre le coût de calcul.

Les résultats donnés par DFO-PSOF ont également été comparés à ceux obtenus avec l'algorithme SQA [63]. SQA est un algorithme d'optimisation sans dérivées, ne prenant pas en compte la séparabilité de la fonction objectif, développé par les équipes IFPEN et est dans le cas sans contraintes très proche de l'algorithme NEWUOA de Powell [86] (voir algorithme 6). Ces résultats sont montrés dans les deux tableaux de la table 2.1. Le premier compare les performances des deux algorithmes lorsque les fonctions tests dépendent de 10 paramètres tandis que le second les compare pour des fonctions tests qui dépendent de 50 paramètres.

Ces tableaux montrent l'efficacité de l'algorithme DFO-PSOF sur ces fonctions tests : le nombre de points requis pour parvenir à la convergence de l'algorithme est bien inférieur lorsque la séparabilité partielle des fonctions est prise en compte. Ceci est particulièrement évident sur les fonctions dépendant de 50 paramètres, le nombre d'évaluations requis par DFO-PSOF étant systématiquement au moins un ordre de grandeur en dessous du nombre d'évaluations requis par SQA.

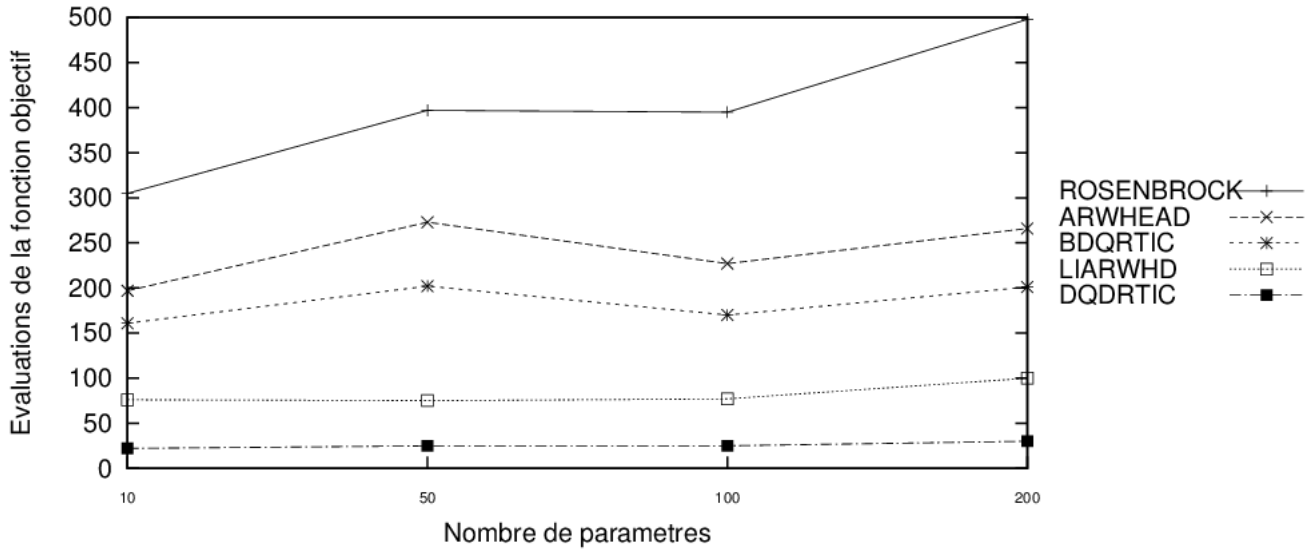


FIGURE 2.6 – Nombre d'évaluations de la fonction objectif pour arriver à convergence en fonction de la dimension du problème pour chacune des fonctions tests avec l'algorithme DFO-PSOF.

Le cas de la fonction DQDRTIC est très intéressant à analyser plus précisément. En effet, l'algorithme DFO-PSOF trouve l'optimum presque immédiatement quelle que soit la dimension du problème (entre 18 et 20 évaluations de la fonction objectif) tandis que SQA requiert beaucoup plus de calculs pour trouver la solution. En fait, la fonction étant elle-même quadratique, par définition de l'interpolation quadratique, si suffisamment de points sont présents dans l'ensemble d'interpolation, le modèle devient exact. Le fait que les tailles des ensembles d'interpolation ne soient pas fixes dans DFO-PSOF combiné au fait que chaque sous-fonction ne dépend que de 3 paramètres rend les modèles exacts en très peu de simulations : les sous-modèles sont exacts à partir du moment où chaque ensemble d'interpolation comprend $\frac{4*5}{2} = 10$ points. Quel que soit le nombre total de paramètres présents, l'algorithme convergera en un nombre presque constant d'évaluations. D'un autre côté, même avec un ensemble d'interpolation de taille variable, un algorithme d'optimisation classique n'obtiendra de modèle exact qu'après un beaucoup plus grand nombre de simulations (pour 50 paramètres, on a besoin de $\frac{51*52}{2} = 1326$ évaluations). La méthode SQA, comme NEWUOA, travaille avec un nombre fixe de points d'interpolations ($2p + 1$ points) et ne construit donc jamais de modèles exactes de la fonction objectif dans ce cas. Ceci explique que le nombre d'évaluations de la fonction objectif nécessaire à la convergence puisse être très supérieur à 1326.

2.5.2 Un cas simple de calage d'historique

L'algorithme DFO-PSOF est testé ici sur le cas décrit en section 1.4.1. On rappelle ci-dessous ses caractéristiques, la figure 2.7 montrant la carte de perméabilité du modèle de référence :

(i) 2500 m dans les directions x et y,

(ii) 10 m dans la direction z,

(iii) discrétisé uniformément par 50x50x1 mailles de 50 m dans les directions x et y et de 10 m dans la direction z,

(iv) 25 puits verticaux :
— 13 producteurs (PRO-1,PRO-3,...,PRO-25)

10 paramètres	SQA		DFO-PSOF	
Fonction	Valeur au meilleur point	Nombre d'itérations	Valeur au meilleur point	Nombre d'itérations
DQDRTIC	4,30E-12	839	7,30E-16	19
LIARWHD	3.51e-09	198	1,26E-09	48
BDQRTIC	18,2881	484	18,2880	198
ARWHEAD	3,19E-09	172	1,19E-09	37
ROSENBROCK	4,86E-09	618	9,20E-09	312
50 paramètres	SQA		DFO-PSOF	
Fonction	Valeur au meilleur point	Nombre d'itérations	Valeur au meilleur point	Nombre d'itérations
DQDRTIC	3,05E-13	4465	7,62E-16	18
LIARWHD	6,53e-09	521	1,90E-09	72
BDQRTIC	178,489	2651	178,489	210
ARWHEAD	8,83E-08	3220	6,70E-07	48
ROSENBROCK	4,05E-08	4057	1,32E-08	383

TABLE 2.1 – Comparaison des méthodes SQA et DFO-PSOF pour l'optimisation des fonctions décrites en début de section pour 10 paramètres (haut) et pour 50 paramètres (bas).

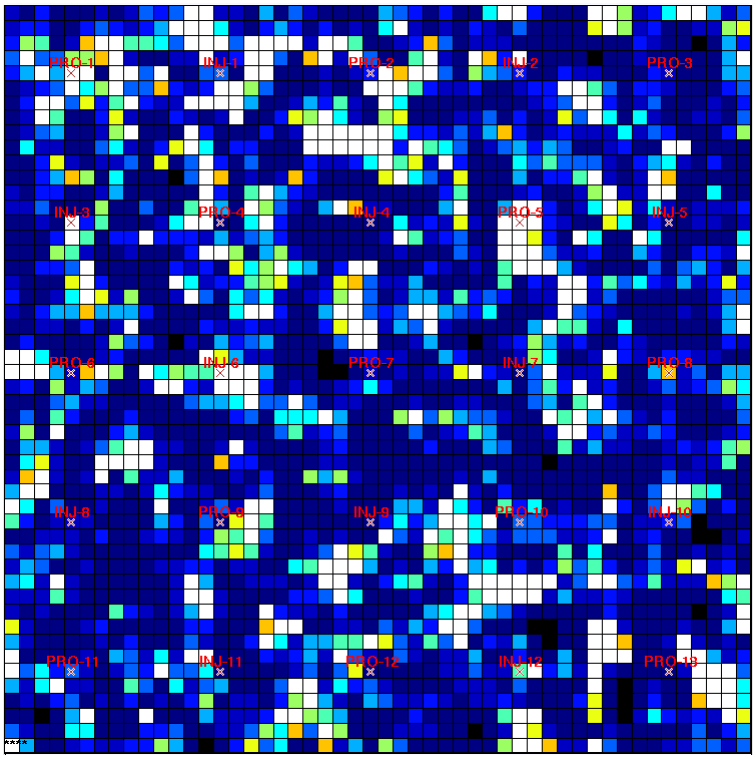


FIGURE 2.7 – Champ synthétique comprenant 13 puits producteurs et 12 puits injecteurs [37].

— 12 injecteurs (INJ-2,INJ-4,...,INJ-24)

(v) réservoir hétérogène : longueur de corrélation de 150 m dans les directions x et y,

(vi) pression initiale dans le réservoir de 250 bars,

(vii) pression en fond de puits imposée :

— 320 bars pour les injecteurs
— 180 bars pour les producteurs.

Un historique de données de production de 4000 jours est supposé connu.

On applique pour ce cas une paramétrisation par déformation graduelle généralisée décrite en section 1.2. 25 réalisations aléatoires indépendantes du modèle sont générées et la meilleure est choisie en tant que réalisation de base. On choisit de créer une zone autour de chaque puits et d'introduire pour chaque zone un paramètre de déformation graduelle locale. De façon à limiter le nombre total de paramètres, la taille des zones est fixée à 5 mailles de modèle (250m), ce qui fait donc un total de 25 paramètres à optimiser. La fonction objectif étant partiellement séparable, on considère qu'un paramètre n'a d'influence que sur les données de production du puits au centre de la zone qui lui est associée et sur les données de production des 4 puits les plus proches. Par exemple, comme illustré sur la figure 2.8, le paramètre associé à la zone autour du puits PRO_7 n'influence en vertu de cette supposition que les données de production des puits PRO_7, INJ_4, INJ_6, INJ_7 et INJ_9. Chaque sous-fonction objectif ne dépend donc au maximum que de 5 paramètres.

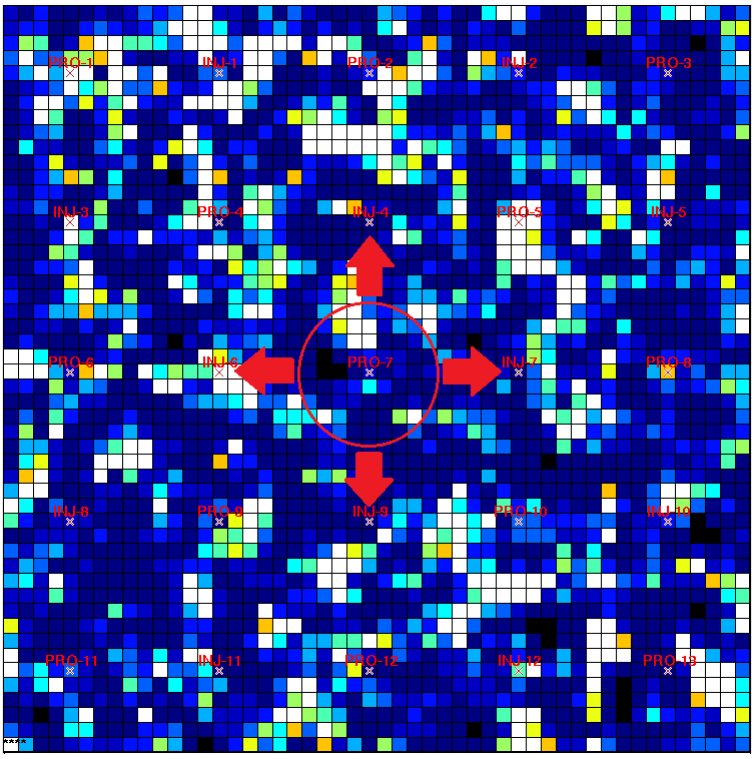


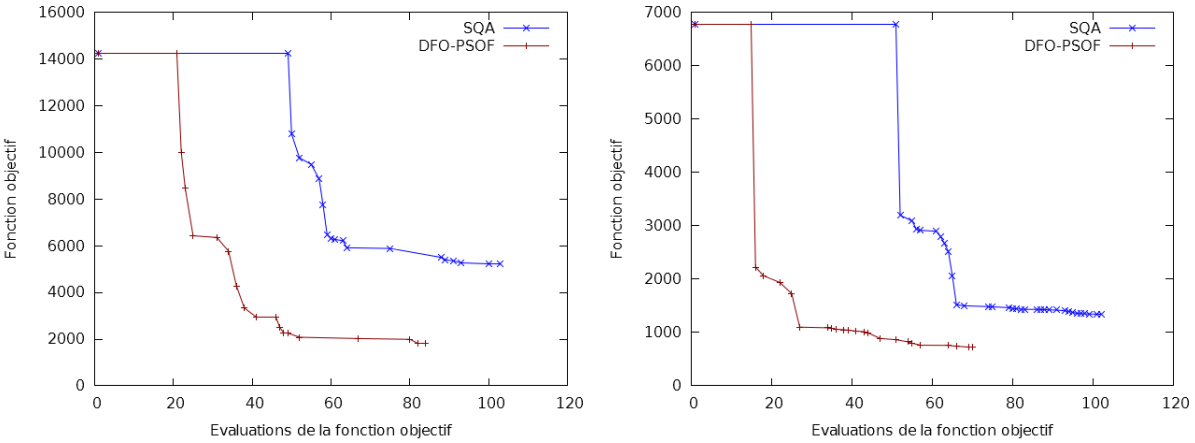
FIGURE 2.8 – Influence du paramètre associé au puits PRO_7.

On présente en figure 2.9 les résultats obtenus dans deux cas différents : dans chacun de ces cas, 25 réalisations indépendantes du modèle sont générées et les réalisations à combiner sont choisies en fonction des valeurs des fonctions objectifs locales. Les résultats des calages d'historique donnés par l'algorithme DFO-PSOF sont comparés avec ceux donnés par l'algorithme SQA [63]. Dans les deux cas, la nouvelle méthode DFO-PSOF donne plus rapidement de meilleurs résultats que SQA. Plus précisément :

(i) **dans le premier cas :**

— Pour obtenir la même valeur de fonction objectif que SQA en 105 itérations, DFO-PSOF ne requiert que 25 évaluations.
— La valeur de la fonction objectif à la fin de l'optimisation est de 5220.6 avec SQA et seulement de 1818.1 avec DFO-PSOF, soit un gain relatif de plus de 65%.

(ii) **dans le second cas :**
— Pour obtenir la même valeur de fonction objectif que SQA en 102 itérations, DFO-PSOF ne requiert que 27 évaluations.
— La valeur finale de la fonction objectif à la fin de l'optimisation est de 1340.2 avec SQA et seulement de 728.6 avec DFO-PSOF, soit un gain relatif de plus de 45%.



(a) Premier cas (b) Second cas

FIGURE 2.9 – Comparaison de calages d'historique par les méthodes SQA et DFO-PSOF.

Ces résultats valident l'hypothèse de séparabilité partielle dans ce cas synthétique simple. Par conséquent, comme on a pu le voir pour les cas analytiques, les modèles construits par le nouvel algorithme nécessitent moins de points d'interpolation que ceux construits avec la méthode SQA, ce qui implique une convergence plus rapide de DFO-PSOF en terme de nombre de simulations. Par ailleurs, la structure de la fonction objectif étant prise en compte, les modèles quadratiques locaux sont plus proches de la fonction objectif, ce qui peut constituer une explication au fait que les points finaux obtenus avec DFO-PSOF soient également meilleurs.

2.6 Bilan du chapitre

Dans cette section, on a cherché à exploiter lors de l'optimisation la structure particulière de la fonction objectif du problème de calage d'historique. En considérant le caractère local des paramètres de déformation graduelle généralisée, on peut supposer que la fonction objectif est partiellement séparable. Après une brève revue des méthodes d'optimisation sans dérivées utilisées dans le domaine, une méthode sans dérivées de type région de confiance utilisant cette structure nommée DFO-PSOF a été introduite. La convergence de cette méthode vers un point critique d'ordre un a été prouvée théoriquement et de premiers résultats numériques ont montré son efficacité tant sur des fonctions analytiques que sur un problème simple de calage d'historique.