Introduction aux problèmes d'optimisation numérique déterministe.

J-B. Blanchard[1,a]

[1]Den-Service de thermo-hydraulique et de mécanique des fluides (STMF), CEA, Université Paris-Saclay, F-91191, Gif-sur-Yvette, France

Résumé Cette note introduit la notion d'optimisation numérique déterministe en discutant leur intérêt dans le cadre de l'expérimentation réelle et de la simulation numérique. Ces techniques, utiles en elles-mêmes pour l'aide à la décision, le dimensionnement, sont aussi très utilisées pour construire des modèles de substitution (pour déterminer la meilleure valeur des paramètres). Cette note n'est en aucun cas exhaustive mais, à l'inverse, elle expose les notions couramment discutées ainsi que quelques techniques usuellement utilisées.

1 Introduction

Cette partie pose les bases du vocabulaire nécessaire à la bonne définition du problème. La plupart du contenu de cette note est issu de la vaste littérature sur le sujet, qu'elle soit en français [1] ou en anglais [2, 3, 4].

1.1 Notations mathématiques

Un problème d'optimisation est une recherche d'**optimum**, *i.e.* du maximum ou du minimum d'une fonction. Sachant que la quête d'un maximum équivaut à chercher le minimum de la fonction de signe opposée, un problème d'optimisation est toujours ramené à une minimisation et est formellement écrit comme :

$$\min_{x\in X} f(x), \quad \text{pour } \mathrm{f}:\mathbb{R}^{\mathrm{n}}\to\mathbb{R}^{\mathrm{q}} \tag{1}$$

avec les contraintes

$$h(x)=0, \quad \text{pour } \mathrm{h}:\mathbb{R}^{\mathrm{n}}\to\mathbb{R}^{\mathrm{m}} \tag{2}$$

$$g(x)\leq 0, \quad \text{pour } \mathrm{g}:\mathbb{R}^{\mathrm{n}}\to\mathbb{R}^{\mathrm{p}} \tag{3}$$

sachant que X est un sous-ensemble de $\mathbb{R}^n$ ($X\subseteq\mathbb{R}^n$). Par la suite, dans le cas où $q=1$, la *vraie valeur* de x pour laquelle la fonction f est minimale sera notée x^*.

1.2 Vocabulaire

Plusieurs variables et fonctions ont été introduites dans les équations précédentes, on les appelle souvent :

- **variables de décision / conception** : ce sont les composantes du système sur lesquelles on peut agir afin de changer la configuration / le fonctionnement du problème. Elles sont représentées comme un vecteur $x=(x_1\dots x_n)^T$ de dimension n, ($n\geq 1$).
- **fonction objectif / coût / critère** : ce sont les mesures de l'état du système que l'on cherche à minimiser avec l'optimisation (de dimension q, $q\geq 1$).
- **contraintes** : ce sont les limites imposées sur les variables de décisions, voire les objectifs, et qui sont représentatives des circonstances du problème. On sépare généralement les contraintes d'inégalité (de dimension p) de celles d'égalité (de dimension m).

Il existe des sous-ensembles classiques des problèmes d'optimisation, parmi lesquels on peut citer les cas où :

- $m+p=0$. Ces optimisations sont dites **sans contrainte**. Dans le cas contraire (si $m\neq 0$ et / ou $p\neq 0$), on parle d'optimisation **sous contrainte**.
- $q=1$. On parle ici d'optimisation **mono-critère**. Si $q>1$, on peut distinguer deux configurations : soit on combine les critères dans une seule fonction amalgamante (ramenant à $q=1$), au risque de pas pouvoir aisément interpréter les résultats, soit on souhaite minimiser les critères conjointement et on parle alors de problème **multi-critères**, *c.f.* section 4.

Finalement les méthodes sont souvent séparées entre celles :

- **locales**, itératives et sensibles aux **minima locaux** (*i.e.* les x_0 pour lesquels il existe un voisinage $\mathcal{V}_{x_0}$ tel que $\forall x\in\mathcal{V}_{x_0}\bigcap X, f(x)\geq f(x_0)$) ;
- **globales**, insensibles aux minima locaux, mais plus complexes, qui cherchent à identifier le **minimum global** (*i.e.* les x_0 tels que $\forall x\in X, f(x)\geq f(x_0)$).

Ces notions d'optimum locaux et globaux sont illustrées dans le cas simple ($n=1$, $q=1$, $p=m=0$) sur la fig. 1.

1.3 Hypothèses de travail

Les méthodes et algorithmes présentés dans cette note ne sont pas universels et reposent sur quelques hypothèses.

Continuité : les contraintes et les objectifs sont modélisés par des fonctions continues des variables de décisions. Cette hypothèse exclut les problèmes en nombres entiers (dits **d'optimisation discrète** ou **combinatoire**). Deux

a. e-mail : jean-baptiste.blanchard@cea.fr

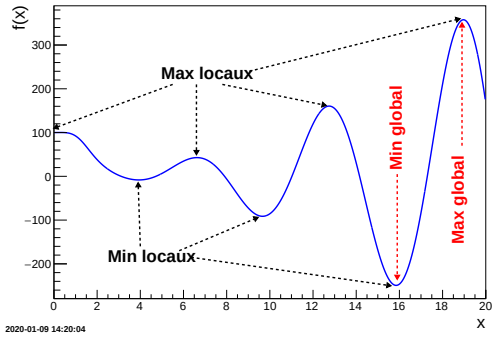


FIGURE 1: Illustration des optimums locaux et globaux dans le domaine de définition d'une fonction jouet f.

conséquences directes sont alors liées à la nature flottante des grandeurs [5] :

— la nécessité de définir une **tolérance** ε pour considérer qu'une solution (notée x_C) est valide (au sens $|x_C - x^*| < \varepsilon$) [1]. C'est un **critère d'arrêt** obligatoire car l'obtention de la valeur exacte x^* n'est pas garantie (même si elle est calculable analytiquement).

— la nécessité d'obtenir le résultat de la fonction avec une précision suffisante, dans les cas d'algorithmes à gradients (surtout pour les approximations par différences finies, *c.f.* section 2.2).

Déterministe : les contraintes et les objectifs sont des fonctions dont le résultat ne contient pas de partie stochastique (deux estimations identiques donnent les mêmes résultats). La modélisation d'incertitude et / ou de phénomènes intrinsèquement aléatoires peut être prise en compte par une **optimisation robuste** [6], limitant l'impact des variations en question, mais cet aspect ne sera pas abordé par la suite.

Finalement, une hypothèse, nécessaire lorsque les algorithmes choisis reposent sur les notions de gradient (voire de hessienne), sera la **différentiabilité** (voire double différentiabilité), qui implique de toute manière la continuité.

Les procédures mono-critères, basées sur le comportement mathématique de la modélisation du problème, sont d'abord introduites (*c.f.* section 2), suivi d'heuristique et de méthodes utilisant des modèles de substitution (*c.f.* section 3) avant d'ouvrir sur les cas multi-critères (*c.f.* section 4).

2 Recherche de solutions mathématiques

Cette partie sera illustrée à partie d'un exemple courant de la littérature : la fonction de Rosenbrock. Elle est présentée dans la fig. 2 et est définie ici, comme $f : \mathbb{R}^2 \to \mathbb{R}$:

$$f(x_1,x_2) = a \times (x_2 - x_1^2)^2 + b \times (1.0 - x_1)^2 \qquad (4)$$

1. La valeur de x^* étant rarement connue, la tolérance est alors définie comme une mesure de la convergence des itérations entre elles.

Il existe un grand nombre d'algorithmes pour minimiser cet exemple simple ($n = 2$, $q = 1$ et $p = m = 0$). Ceux qui seront utilisés ici sont issus du package NLopt [7]. Nous allons introduire les concepts généraux, dans ce qui peut être considéré comme une origine commune historique, avant de discuter les possibles développements et améliorations.

2.1 Méthode analytique historique dite de Newton

Le principe de cette méthode repose sur la recherche de **points critiques**, *i.e.* les $x_C \in X$ tels que le gradient soit nul ($\nabla f(x_C) = 0$), caractérisant les minima, maxima et les points selles. Une autre condition suffisante est utilisée pour s'assurer du caractère minimum (local) : vérifier que la matrice hessienne ($\nabla^2 f(x_C)$) est définie positive.

L'idée de la méthode de Newton est de substituer localement la fonction f par un modèle quadratique tel que

$$m_{\hat{x}}(\hat{x}+d) \simeq f(\hat{x}) + d^T \nabla f(\hat{x}) + \frac{1}{2} d^T \nabla^2 f(\hat{x}) d,$$

où $d = (x - \hat{x})$ et l'annulation de son gradient équivaut à

$$\begin{aligned} 0 = \nabla f(\hat{x}) + \nabla^2 f(\hat{x}) d \Leftrightarrow d &= -\nabla^2 f(\hat{x})^{-1} \nabla f(\hat{x}) \\ \Leftrightarrow x &= \hat{x} - \nabla^2 f(\hat{x})^{-1} \nabla f(\hat{x}) \end{aligned}$$

Il faut alors choisir un point de départ pour x_k ($k = 0$), estimer $f(x_k)$, construire le modèle quadratique, trouver la direction d_k de variation et recommencer à partir de $x_{k+1} = x_k + d_k$ jusqu'à atteindre la tolérance.

Cette méthode est rarement utilisée car elle nécessite l'estimation et l'inversion de la hessienne à chaque itération, mais elle reste une base à bien d'autres algorithmes [1].

2.2 Méthodes à gradients

Les algorithmes utilisant les gradients pour chercher la direction privilégiée sont appelés algorithmes **de descente**. Parmi ces derniers, on appelle méthodes de **quasi-Newton** ceux qui évitent de calculer l'inverse de la hessienne. Ils tentent généralement d'approximer cette dernière par des processus itératifs, en travaillant sur l'annulation du gradient.

La fig. 2 présente, en traits pointillés, deux manières de faire cela. La première combine les *méthodes sécantes* avec la *mise à jour de Broyden*, pour obtenir une matrice H_k inversée par *factorisation de Cholesky*. Le résultat est appelé algorithme BFGS [7] et est représenté en vert. La seconde utilise la méthode des *gradient conjugués* pour définir une suite de *directions conjuguées* permettant d'annuler le gradient selon l'approximation quadratique. La solution présentée en bleue est appelée NEWTON dans la librairie NLopt [7] et elle effectue un pré-conditionnement et tronque le nombre de directions pour accélérer le calcul.

Il est possible, si la fonction étudiée ne fournit pas directement les gradients, d'approximer ces derniers par différences finies autour de l'itération x_k, ce qui implique d'une

part que la précision de la fonction soit suffisamment grande pour que la dérivée partielle estimée soit significative et, d'autre part, que chaque estimation représente en fait $2n+1$ appels à la fonction (c'est ce qui est fait pour la fig. 2).

2.3 Méthodes directes

Les méthodes directes, quant à elles, ne font ni l'estimation du gradient, ni celle de la hessienne. La plus connue de ces méthodes est celle dite du SIMPLEX, représentée dans la fig. 2 en noir (et appelé NELDERMEAD dans [7]). Elle construit un bloc de $n+1$ estimations et calcule ensuite leur barycentre. Plusieurs solutions sont testées sur la droite reliant ce barycentre et l'estimation la moins performante du bloc, cette dernière étant remplacée par la meilleure solution, définissant un nouveau bloc et donc un nouveau barycentre. La seconde solution, représentée en rouge et appelé BOBYQA dans [7] est un des rares algorithmes directs utilisant toujours l'approximation quadratique. Il définit une *région de confiance* autour de x_k et construit le modèle en minimisant la *norme de Frobénius* de la partie quadratique.

2.4 Approfondissement théorique et pratique

Les méthodes citées jusqu'ici sont toutes locales et itératives ce qui implique plusieurs choses. Elles nécessitent un second critère d'arrêt, le **nombre maximal d'itération**, si la toléance n'est jamais atteinte. La position de l'itération x_{k+1} nécessite le résultat de l'estimation x_k, ce qui rend impossible la parallélisation des calculs. Une manière d'utiliser plusieurs ressources est le **multistart**, *i.e.* choisir plusieurs points de départ (1 par ressource) et faire autant d'optimisation, testant ainsi la robustesse aux minima locaux.

Certains algorithmes peuvent gérer nativement les contraintes, mais en règle générale, il existe plusieurs solutions pour intégrer ces dernières. On peut définir comme objectif le **Lagrangien** du problème, $L(x,\lambda,\mu) = f(x) + \lambda^T h(x) + \mu^T g(x)$, où $\lambda \in \mathbb{R}^m$ et $\mu \in \mathbb{R}^p$ sont appelés **multiplicateurs de Lagrange**. Cette définition introduit la notion de problème **primal/dual** et est parfois complétée par des **termes de pénalisation**, ce qui peut donner, pour un problème avec contrainte d'égalité seulement, un **Lagrangien augmenté** du type $L_c(x,\lambda) = L(x,\lambda) + \frac{c}{2}||h(x)||^2$, où $c \in \mathbb{R}^+$.

3 Solution heuristique et modèle de substitution

Cette partie introduit deux mécanismes mono-critères différents de ceux discutés précédemment, à savoir une heuristique et une méthode basée sur un modèle de substitution.

3.1 Recuit simulé

En métallurgie, la re-cuisson est une étape où on chauffe un métal pour le laisser se refroidir lentement améliorant

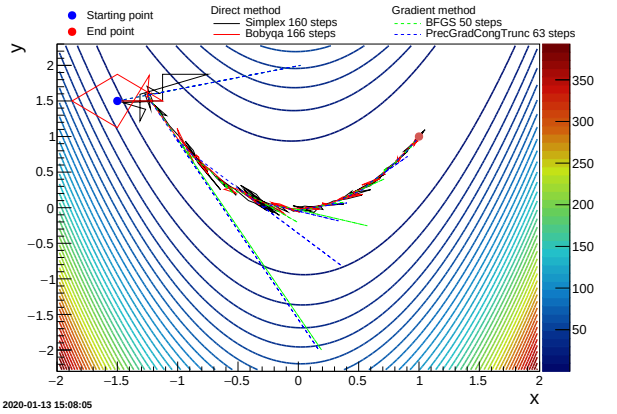


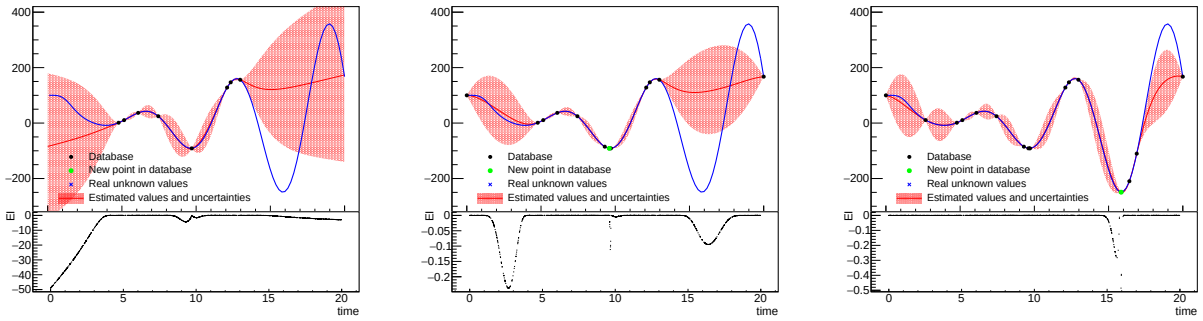
FIGURE 2: Évolution de la fonction de Rosenbrock pour $a = 100$ et $b = 1$ et résultats d'algorithmes d'optimisation, partant d'un point de départ commun (bleu) et arrivant au minimum théorique attendu (rouge).

ainsi ses propriétés en changeant sa structure cristalline. Le recuit simulé est une extension des recherches locales directes, discutées en section 2.3, qui peut accepter des itérations avec des valeurs d'objectifs plus grands. Ainsi pour l'itération x_k et y au voisinage de x_k si $f(y) \leq f(x_k)$ la solution est acceptée comme prochaine itération, mais si au contraire $f(y) > f(x_k)$, cette dernière sera acceptée avec la probabilité $\exp(-\frac{f(y)-f(x_k)}{T})$, où T est appelée **température** du système qui va diminuer au fur et à mesure du processus d'optimisation.

3.2 Efficient Global Optimisation (EGO)

Cette méthode utilise le krigeage [8], qui fournit, à partir d'un échantillon de résultats d'une fonction, une prédiction du comportement de cette dernière ainsi qu'un intervalle de confiance. Le critère *Expected Improvement*, ou EI, combinant les incertitudes de prédiction avec la distance entre les prédictions et le vrai minimum connu de la fonction, est utilisé définissant ainsi la méthode globale dite EGO [9]. Le principe, discuté ci-dessous et illustré dans la fig. 3, est plus lourd à mettre en place que les techniques de la section 2, mais il limite le nombre d'appels à la fonction, ce qui est un avantage quand cette dernière est couteuse.

- Fig. 3a : l'échantillon initial (points noirs) est utilisé pour construire le krigeage (ligne et bande rouges). L'EI (ligne noire dans la partie inférieure) est alors calculé et minimisé en utilisant une base de test via un algorithme génétique (VIZIR [4]). Le minimum est estimé par la fonction et intégré à l'échantillon.
- Fig. 3b : après trois estimations, les bords du domaine ont été testés et l'algorithme vérifie les abord du "meilleur" minimum actuel (la dernière estimation par la fonction est représentée en vert).



(a) étape 0 (b) étape 4 (c) étape 8

FIGURE 3: Minimisation de la fonction présentée dans la fig. 1 par EGO. L'étape (a) construit la base initiale, la (b) montre que l'algorithme intensifie les recherches autour du meilleur minimum identifié, mais est capable de ressortir pour tester les régions où l'incertitude de prédiction est grande, ce qui est fait en (c) identifiant ainsi le minimum global.

— Fig. 3c : après quelques estimations, l'algorithme est sorti des abords du minimum local pour tester la zone de grandes incertitudes de prédiction. Un nouveau "meilleur" minimum est trouvé.

L'arrêt de cette procédure repose généralement sur un critère de qualité du modèle de substitution [9].

4 Cas des problèmes multi-critères

Pour le **multi-critères** ($q \geq 2$), deux solutions peuvent être incomparables, nécessitant l'introduction du principe d'**optimalité**, par exemple **lexicographique** si l'importance des objectifs est hiérarchisable. Sinon, une solution x_1 **domine** une solution x_2 quand on peut écrire, pour $q = 2$, $f_1(x_1) < f_1(x_2)$ et $f_2(x_1) < f_2(x_2)$. L'idée est ici de trouver une famille de solutions envisageables, dites **non-dominées**, qui forme, dans l'espace des variables, le **jeu de Pareto** et, dans l'espace des critères, le **front de Pareto** (c'est l'**optimalité de Pareto** reposant sur la notion de compromis).

Pour obtenir un front de Pareto, on peut utiliser des algorithmes évolutionnaires. Ces derniers génèrent un ensemble de solutions, les testent, et sélectionnent les meilleurs afin de re-générer de nouvelles solutions. Le mécanisme de génération est souvent inspiré de la nature (algorithme génétique, à essaim…). Un algorithme évolutionnaire appliqué à la fonction f de la fig. 1 fournira la solution optimale ($\simeq 15.9$). Si on rajoute la valeur de x comme objectif, les critères $(x, f(x))$ sont antagonistes sur les portions décroissantes de $f(x)$. Un front de Pareto à 200 solutions, obtenu par VIZIR [4], est ainsi représenté dans la fig. 4.

5 Conclusion

Le concept d'optimisation a été introduit en présentant plusieurs méthodologies, afin de donner des pistes et des références expliquant les grands principes.

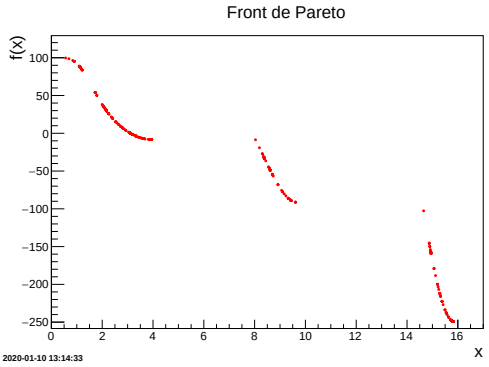


FIGURE 4: Front de Pareto issu d'un algorithme génétique, minimisant $(x, f(x))$, où f est la fonction de la fig. 1.

Références

1. M. Bierlaire, *Introduction à l'optimisation différentiable*. PPUR presses polytechniques, 2006.
2. E. K. Chong and S. H. Zak, *An introduction to optimization*, vol. 76. John Wiley & Sons, 2013.
3. C. T. Kelley, *Iterative methods for linear and nonlinear equations*, vol. 16. Siam, 1995.
4. J.-B. Blanchard, "Methodological reference guide for uranie v3.11.0," tech. rep., CEA, DEN/DANS/DM2S/STMF/LGLS/RT/17-006/A, 2017. *Updated with every new release.*
5. U. Drepper, "What every programmer should know about memory," *Red Hat, Inc*, vol. 11, p. 2007, 2007.
6. A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust optimization*, vol. 28. Princeton University Press, 2009.
7. S. G. Johnson, "The nlopt nonlinear-optimization package." http ://ab-initio.mit.edu/nlopt.
8. J.-B. Blanchard, "Introduction aux modèles de substitution," *I3P book of notice*, 2020.
9. D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.