ENS CACHAN, DÉPARTEMENT DE MATHÉMATIQUES

OPTIMISATION NUMÉRIQUE M1 HADAMARD
TP 1 – GRADIENT, LEVENBERG-MARQUARDT, NELDER-MEAD

FLORIAN DE VUYST, ADRIEN LE COËNT

1. Gradient et règle d'Armijo pour la recherche linéaire approchée

Les méthodes d'optimisation sont en général testées et validées sur les cas tests de référence. Parmi ces cas, on trouve la minimisation sur la fonction de Rosenbrock (ou encore fonction dite "banane") de $\mathbb{R}^2$ dans $\mathbb{R}$, $\boldsymbol{u} = (u_1, u_2)$, définie par

$$J(\boldsymbol{u}) = (1 - u_1)^2 + p\,(u_2 - {u_1}^2)^2.$$

avec $p = 10$. Représenter par lignes de niveaux la fonction J dans le rectangle $[-1.5, 1.5] \times [-0.5, 1.5]$. On constate que le minimum global $\boldsymbol{u}^\star = (1, 1)$ se trouve dans une vallée escarpée. Calculer sur papier $\nabla J(\boldsymbol{u})$.

On considère l'initialisation $\boldsymbol{u}^0 = (-1, 1)$ on on souhaite appliquer la méthode de gradient suivante

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k - \rho\,\frac{\nabla J(\boldsymbol{u}^k)}{\|\nabla J(\boldsymbol{u}^k)\|},$$

avec $\rho = 0.01$. Étudier numériquement le comportement de la suite. On représentera sur les lignes de niveaux les itérés $\boldsymbol{u}^k$ par des ronds et la trajectoire en ligne brisée. Compter le nombre d'évaluations de J nécessaire pour atteindre le critère d'arrêt

$$\frac{\|\nabla J(\boldsymbol{u}^k)\|}{\|\nabla J(\boldsymbol{u}^0)\|} \leq 10^{-4}$$

et afficher le nombres d'itérations k.

Règle d'Armijo. On utilise maintenant une méthode de gradient à pas variable

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k - \rho^k\,\nabla J(\boldsymbol{u}^k)$$

utilisant une recherche linéaire approchée. La règle dite d'Armijo définit l'algorithme suivant pour calculer ρ^k :

(1) Paramètre d'entrée : $m \in (0, 1)$. On prendra $m = 0.01$.

(2) $\rho^k \leftarrow \rho^0$ (on prendra $\rho^0 = 10$).

(3) Tant que

$$J(\boldsymbol{u}^k - \rho^k \nabla J(\boldsymbol{u}^k)) > J(\boldsymbol{u}^k) + m\rho^k \|\nabla J(\boldsymbol{u}^k)\|^2$$

faire $\rho^k \leftarrow \rho^k/2$.

Programmer à nouveau la méthode de gradient avec règle d'Armijo, compter le nombre d'évaluation de J et afficher le nombre d'itérations k. Tracer les itérés comme indiqués ci-dessus.

2. Régression non linéaire, algorithme de Levenberg-Marquardt

Dans les problèmes classiques de calibrage de modèles, où le but est de trouver les paramètres β d'un certain modèle $y = f(x, \beta)$ permettant le meilleur ajustement aux observations $\{(x_i, y_i)\}_{i=1,...,m}$, on cherche à minimiser les écarts aux moindres carrés

$$J(\beta) = \frac{1}{2} \sum_{i=1}^{m} \left[y_i - f(x_i, \boldsymbol{u}) \right]^2 .$$

Dans cet exercice, on considère l'approximation définie comme superposition de K gaussiennes

$$f(x; \boldsymbol{u}) = \sum_{j=1}^{K} \alpha_j \, \exp\left(-\frac{1}{2} \frac{(x - x_j^0)^2}{\sigma_j^2} \right).$$

Les paramètres à ajuster ici sont les α_i, les σ_i et les x_i^0, et on a

$$\boldsymbol{u} = (\alpha_1, ..., \alpha_K, \, \sigma_1, ..., \sigma_K, \, x_1^0, ...x_K^0)^T \in \mathbb{R}^{3K}.$$

En introduisant les fonctions de résidu $\varphi_i = y_i - f(x_i, \boldsymbol{u})$, on a

$$J(\boldsymbol{u}) = \frac{1}{2} \sum_{i=1}^{m} [\varphi_i(\boldsymbol{u})]^2$$

et

$$\nabla J(\boldsymbol{u}) = \sum_{i=1}^{m} \nabla \varphi_i(\boldsymbol{u}) \, \varphi_i(\boldsymbol{u}).$$

La méthode de Gauss-Newton est une méthode de type quasi-Newton où la matrice Hessienne $\nabla^2 J$ de J est approchée simplement par le second terme

$$\nabla^2 J(\boldsymbol{u}) \approx \sum_{i} \nabla \varphi_i(\boldsymbol{u}) \nabla \varphi_i(\boldsymbol{u})^T.$$

Cette matrice est symétrique, positive mais pas toujours définie, sinon très souvent mal conditionnée.

L'algorithme de **Levenberg-Marquardt** (LM) est une version régularisée de l'algorithme de Gauss-Newton. Étant donné un coefficient de régularisation $\mu > 0$, on considère la matrice de quasi-Newton

$$S(\boldsymbol{u}) = \sum_{i=1}^{m} \nabla \varphi_i(\boldsymbol{u}) \nabla \varphi_i(\boldsymbol{u})^T + \mu \, \mathrm{I}.$$

On résume donc l'algorithme de Levenberg-Marquardt :

(1) *Initialisation :* On choisit un vecteur β^0 quelconque.

(2) *Itération k :* on calcule

$$S^k = \sum_{i=1}^{m} \nabla \varphi_i(\boldsymbol{u}^k) \nabla \varphi_i(\boldsymbol{u}^k)^T + \mu \mathrm{I}$$

et la direction de recherche

$$\boldsymbol{d}^k = - \left(S^k \right)^{-1} \nabla J(\boldsymbol{u}^k).$$

(3) *Recherche linéaire :* on cherche le minimim exact ou approché de

$$\rho^k = \arg\min_{\rho} J(\beta + \rho \, \boldsymbol{d}^k).$$

On considèrera ici la recherche linéaire approchée avec la règle d'Armijo.

(4) *Mise à jour :*

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \rho^k \, \boldsymbol{d}^k.$$

(5) Critère d'arrêt :

$$\frac{\|\nabla J(\boldsymbol{u}^k)\|}{\|\nabla J(\boldsymbol{u}^0)\|} \leq 10^{-4},$$

si oui, fin de boucle sur k.

(6) $\boldsymbol{u}^\star \leftarrow \boldsymbol{u}^k$.

Mettre en oeuvre la méthode. On considèrera les valeurs numériques suivantes pour les x_i et y_i :

x_i	1	2	3	4	5	6	7	8
y_i	0.127	0.2	0.3	0.25	0.32	0.5	0.7	0.9

On initialisera par $\alpha_1 = \alpha_2 = 1$, $\sigma_1 = \sigma_2 = 1$, $x_1^0 = 3$, $x_2^0 = 6$ et on choisira $K = 2$ et

$$\mu = \mu^k = 10^{-4}\,\lambda_{max}\left(\sum_{i=1}^{m} \nabla\varphi_i(\boldsymbol{u}^k)\nabla\varphi_i(\boldsymbol{u}^k)^T\right)$$

(où $\lambda_{max}(A)$ désigne le plus grand module des v.p. de la matrice A).

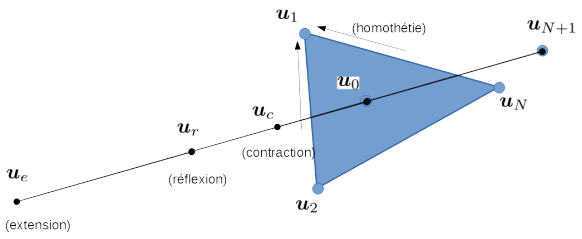
Pour la visualisation, tracer les points de données $\{(x_i, y_i)\}_i$ et la fonction d'approximation

$$x \mapsto f(x, \boldsymbol{u}^\star)$$

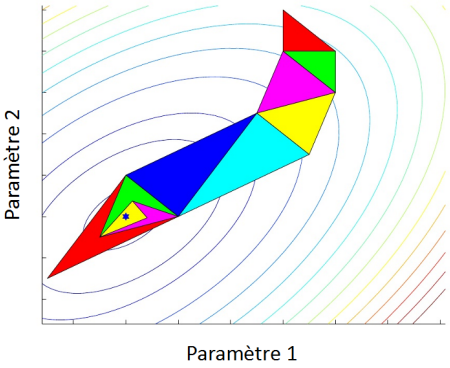
sur l'intervalle $x \in [0, 10]$ (on discrétisera sur 200 points) au cours des itérations $k$de la boucle d'optimisation.

3. Méthode sans gradient. Algorithme de Nelder-Mead

Pour certaines applications il est difficile d'évaluer le gradient. Il est connu que les techniques de différences finies conduisent à des méthodes d'optimisation peu robustes. L'algorithme du simplexe de Nelder et Mead est une approche sans gradient (gradient-free) qui fait appel à l'évolution et l'adaptation d'un simplexe.



(A) Opérations élémentaires.



(B) Exemple d'évolution en dimension 2.

FIGURE 1. Algorithme du simplexe de Nelder et Mead

Pour résoudre

$$\min_{\boldsymbol{u}\in\mathbb{R}^N} J(\boldsymbol{u}),$$

la méthode de Nelder-Mead construit un polytope à $(N+1)$ côtés selon l'heuristique suivante :

(1) Initialisation : on choisit $(N+1)$ points formant un simplexe : $\{\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_N, \boldsymbol{u}_{N+1}\}$.

(2) On réindexe les points de façon que $J(\boldsymbol{u}_1) \leq J(\boldsymbol{u}_2) \leq ... \leq J(\boldsymbol{u}_N) \leq J(\boldsymbol{u}_{N+1})$.

(3) On calcule $\boldsymbol{u}_0$ comme le centre de gravité de $(\boldsymbol{u}_1, ..., \boldsymbol{u}_N)$.

(4) (réflexion) On calcule le point réfléchi
$$\boldsymbol{u}_r = \boldsymbol{u}_0 + (\boldsymbol{u}_0 - \boldsymbol{u}_{N+1}).$$
(5) Test $J(\boldsymbol{u}_r) < J(\boldsymbol{u}_N)$. Si oui, on calcule (étirement)
$$\boldsymbol{u}_e = \boldsymbol{u}_0 + 2(\boldsymbol{u}_0 - \boldsymbol{u}_{N+1}).$$
Test $J(\boldsymbol{u}_e) < J(\boldsymbol{u}_r)$. Si oui, $\boldsymbol{u}_{N+1} \leftarrow \boldsymbol{u}_e$, sinon $\boldsymbol{u}_{N+1} \leftarrow \boldsymbol{u}_r$ et retour à (2).
(6) (contraction) Test $J(\boldsymbol{u}_N) < J(\boldsymbol{u}_r)$. Si oui, calcul de
$$\boldsymbol{u}_c = \boldsymbol{u}_{N+1} + \frac{1}{2}(\boldsymbol{u}_0 - \boldsymbol{u}_{N+1}).$$
Test $J(\boldsymbol{u}_c) \leq J(\boldsymbol{u}_N)$. Si oui, $\boldsymbol{u}_{N+1} \leftarrow \boldsymbol{u}_c$ et retour à (2).
(7) Sinon, on applique une contraction générale de rapport $\frac{1}{2}$ et de centre $\boldsymbol{u}_1$, puis on retourne à 2.
(8) + critère d'arrêt.

Dans de très rares cas, le simplexe de Nead-Mead dégénère (écrasement ou alignement des points) et il convient aussi de réinitialiser (restart) l'algorithme au voisinage du dernier point courant.

Programmer l'algorithme de Nelder-Mead et appliquer la méthode au cas de la dimension 2 avec la fonction de Rosenbrock du 1er exercice. Dans ce cas, le simplexe est un triangle. On initialisera le simplexe avec les points
$$\boldsymbol{u}_1 = (-1, 1.5), \quad \boldsymbol{u}_2 = (-1, -0.5), \quad \boldsymbol{u}_3 = (1, 0.5).$$
On affichera les différents triangles au cours des itérations. On comptera le nombre d'évaluations nécessaire pour atteindre la convergence numérique. On utilisera le même critère d'arrêt que dans l'exercice 1.