Annexe C

Algorithmes d'optimisation non-linéaire sans contrainte

Sommaire

L'objectif de ce chapitre est de présenter différents types d'algorithmes d'optimisation. Deux classes d'algorithmes seront présentés : les algorithmes déterministes et les algorithmes probabilistes (§C.4). La première classe d'algorithmes peut être décomposée en deux sous classes : les algorithmes basés sur le gradient de la fonction à optimiser (§C.1) et les algorithmes sans calcul de gradients (§C.2).

On considère le problème d'optimisation sans contrainte suivant :

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{C.1}$$

avec $f : \mathbb{R}^n \mapsto \mathbb{R}$ une fonction continue dans un espace muni d'un produit scalaire $\langle \cdot, \cdot \rangle$ et de sa norme associée. On note $\boldsymbol{\nabla} f$ le gradient et $H = \boldsymbol{\nabla}^2 f$ le hessien de la fonction f au point courant $\boldsymbol{x}$.

Dans ce qui suit les différents algorithmes d'optimisation seront testés sur une fonction test de type Rosenbrock. La fonction Rosenbrock "banana", représentée sur la figure C.1, présente une vallée qui mène à son minimum : à cet effet, les différents algorithmes montreront leur capacité à suivre cette vallée en un minimum d'itérations.

La fonction de Rosenbrock banana $\mathcal{R} : \mathbb{R}^2 \mapsto \mathbb{R}$ à minimiser est la suivante :

$$\mathcal{R}(x, y) = (x - 1)^2 + p\,(x^2 - y^2)^2.$$

La profondeur de la vallée est fonction du paramètre p qui est ici choisi égal à 10. Le minimum de la fonction $\mathcal{R}$ est $x = y = 1$.

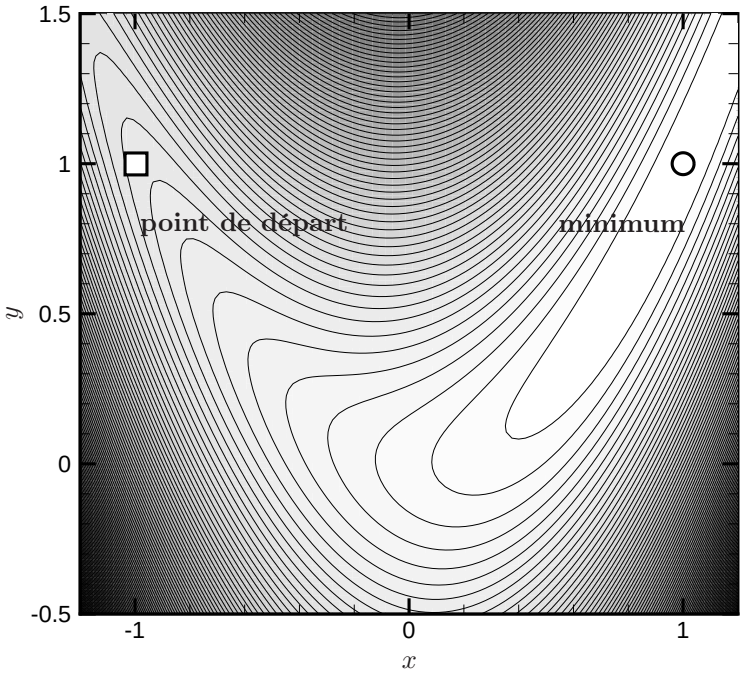


FIGURE C.1 – *Isovaleurs de la fonction de Rosenbrock banana. Le point de départ des algorithmes déterministes est* $(x\,;\,y) = (-1\,;\,1)$ *et le minimum est localisé au point* $(x\,;\,y) = (1\,;\,1)$.

C.1 Algorithmes à directions de descente

Une direction de descente $\boldsymbol{d}$ est une direction de l'espace $\mathbb{R}^n$ qui vérifie $\langle \boldsymbol{\nabla} f, \boldsymbol{d} \rangle < 0$.

Par définition de la dérivée, si $\boldsymbol{d}$ est une direction de descente alors pour tout $\alpha > 0$ suffisamment petit, on a :

$$f(\boldsymbol{x} + \alpha \boldsymbol{d}) < f(\boldsymbol{x}). \tag{C.2}$$

Au moins localement, la fonction f diminue en effectuant un déplacement dans la direction $\boldsymbol{d}$. Les méthodes à directions de descentes suive ce principe et construisent une suite d'itérés $\{\boldsymbol{x}_k\}_{k\geq 1}$ approchant la solution $\boldsymbol{x}^\star$ du problème (C.1) de la façon suivante :

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k.$$

Le paramètre α_k est le pas à effectuer le long de la direction de descente $\boldsymbol{d}_k$ au point courant $\boldsymbol{x}_k$.

Un algorithme à directions de descente est donc déterminé par les paramètres $\boldsymbol{d}$ et α : la façon dont la direction de descente $\boldsymbol{d}$ est calculée donne son nom à l'algorithme et la façon dont le pas α est déterminé est appelée recherche linéaire (§C.1.1) et peut se déterminer de différentes façon à préciser.

Un modèle d'algorithme basé sur des directions de descente est fourni par l'algorithme 10.

Algorithme 10 (Algorithme à directions de descente)

Choix d'un itéré initial $\boldsymbol{x}_1 \in \mathbb{R}^n$ *et d'un petit paramètre* ε. *Initialisation* $k = 1$.

1. *faire test de convergence : si* $\|\boldsymbol{\nabla} f(\boldsymbol{x}_k)\|_2 < \varepsilon$ *arrêt de l'algorithme*

2. *déterminer une direction de descente* $\boldsymbol{d}_k$
3. *déterminer un pas* $\alpha_k > 0$ *assez petit tel que la fonction* f *décroisse suffisamment*
4. *déterminer un nouvel itéré* $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$
5. *poser* $k = k + 1$ *et retourner à l'étape* 1.

C.1.1 La recherche linéaire

La recherche linéaire consiste à déterminer le pas α_k à effectuer le long d'une direction de descente $\boldsymbol{d}_k$. Des valeurs de ce pas peuvent être obtenues par différentes méthodes : par recherche linéaire exacte, par la méthode d'Armijo ou de Goldstein et par la méthode de Wolfe.

Le pas α_k est souvent choisi afin de vérifier les deux conditions suivantes :

– la fonction f doit décroître suffisamment le long de la direction de descente ;

– le pas α_k ne doit pas être trop petit.

Le premier point est habituellement réalisé en forçant l'inégalité (C.2) en rajoutant un terme négatif β_k au second membre :

$$f(\boldsymbol{x} + \alpha \boldsymbol{d}) \leq f(\boldsymbol{x}) + \beta_k. \tag{C.3}$$

Cette condition est habituellement vérifiée par des pas α_k petits. Or de trop petits pas peuvent entraîner une fausse convergence : le second point doit donc également être satisfait.

Dans ce qui suit la notation $h_k(\alpha) = f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)$ sera parfois utilisée.

Recherche linéaire exacte

La recherche linéaire exacte consiste à déterminer le pas optimal, c'est à dire le pas qui minimise la fonction f le long de la direction de descente $\boldsymbol{d}_k$. Le pas α_k est donc solution du problème :

$$\alpha_k = \arg\min_{\alpha \geq 0} f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k). \tag{C.4}$$

Le pas optimal ainsi déterminé est le pas de Cauchy. Le problème (C.4) est appelé problème de Cauchy. Le pas optimal obtenu est correspond au minimum global de la fonction f. Il est des fois préférable de choisir le premier minimum local atteint par la fonction f, soit :

$$\alpha_k = \inf\{\alpha \geq 0 \mid f'_\alpha(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)\}. \tag{C.5}$$

Dans ce cas on parle de règle de Curry. Le pas de Curry ainsi obtenu n'est pas optimal d'un point de vue global, mais rentre tout de même dans la classe des méthodes de recherche linéaire dites *exactes*.

Il faut cependant préciser que les règles de Cauchy et de Curry sont applicables que dans des cas particuliers, où au moins localement le hessien de la fonction à minimiser est défini positif : dans tout autre cas, les pas de Cauchy et de Curry n'existent pas sur un intervalle semi-infini. Ce type de recherche linéaire demande en général beaucoup de temps de calcul sans permettre d'améliorer grandement la convergence de l'algorithme. D'autres règles de calcul, moins restrictives, du pas α_k peuvent alors être utilisées. Ces conditions seront alors plus facilement vérifiées car elles n'autorisent pas une unique solution pour le pas α_k, mais un ensemble d'intervalles solutions comme il est illustré par les règles d'Armijo, de Goldstein et de Wolfe.

Méthodes d'Armijo et de Goldstein

Ce type de méthodes se base sur la condition (C.3). La façon de procéder est alors de demander à f de décroire autant qu'une proportion $\omega_1 \in]0{,}1[$ de ce que ferai le modèle linéarisé de f autour du point $\boldsymbol{x}_k$. On a alors affaire à la condition d'Armijo :

$$f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k) \leq f(\boldsymbol{x}_k) + \omega_1 \, \alpha_k \, \langle \boldsymbol{\nabla} f_k, \boldsymbol{d}_k \rangle. \tag{C.6}$$

Ici, la constante ω_1 est choisie arbitrairement et est habituellement prise très petite (de l'ordre de 10^{-4}) afin que la condition d'Armijo (C.6) soit plus aisément satisfaite. Le choix de cette constante est indépendant du type de problème étudié, mais il est souvent préférable de choisir $\omega_1 < 0{,}5$.

Il est évident qu'un pas α_k très petit satisfait la condition d'Armijo (C.6). Or nous avons énoncé qu'il était dangereux d'accepter un pas α_k trop petit sous peine de converger vers un point non-stationnaire. Il faut donc une méthode permettant de déterminer un pas qui ne soit pas trop petit. Deux méthodes sont couramment utilisées : la méthode d'Armijo à rebroussement et la méthode de Goldstein.

Méthode d'Armijo à rebroussement

La méthode d'Armijo à rebroussement, dans sa version simplifiée, consiste à choisir pour pas de recherche linéaire dans la direction $\boldsymbol{d}_k$ le pas $\alpha_k = \tau^i$ où $\tau \in]0,1[$ est une constante et i est le plus petit entier positif vérifiant la condition d'Armijo (C.6). Typiquement, α_k est le plus grand réel de la suite décroissante (d'où le nom de rebroussement) $\{\tau^i\}_{i\in\mathbb{N}}$ qui vérifie la condition (C.6). Le pas ainsi détreminé ne sera donc *a priori* pas trop petit. Pratiquement, cette version simplifiée de la méthode d'Armijo à rebroussement n'est pas toujours satisfaisante, et on lui préfère souvent une version améliorée utilisant les informations sur la fonction f calculée au pas précédent.

La méthode d'Armijo à rebroussement est présentée par l'algorithme 11 :

Algorithme 11 (Méthode d'Armijo à rebroussement)

Initialisations : choix d'un pas $\alpha_k^1 > 0$ *et d'un paramètre* $\tau \in]0,1[$. $i = 1$.

1. *Test : le pas* α_k^i *est accepté s'il vérifie la relation* (C.6) *:*

$$f(\boldsymbol{x}_k + \alpha_k^i \boldsymbol{d}_k) \leq f(\boldsymbol{x}_k) + \omega_1\, \alpha_k^i\, \langle \boldsymbol{\nabla} f_k, \boldsymbol{d}_k \rangle,$$

Sinon :

2. *On choisit* $\alpha_k^{i+1} \in [\tau\alpha_k^i, (1-\tau)\alpha_k^i]$
3. *On pose* $i = i + 1$ *et* $\alpha_k = \alpha_k^i$*, puis on retourne à l'étape* 1.

Le paramètre τ est ici laissé au choix de l'utilisateur et est habituellement choisi égal à 10^{-2}. L'étape 2 de l'algorithme précédent est souvent effectuée par interpolation. Le pas obtenu par cet algorithme est appelé pas d'Armijo.

Méthode de Goldstein

Dans la méthode de Goldstein la détermination du pas α_k doit vérifier l'équation (C.6) et l'équation suivante :

$$f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k) \geq f(\boldsymbol{x}_k) + \omega_1'\, \alpha_k\, \langle \boldsymbol{\nabla} f_k, \boldsymbol{d}_k \rangle. \tag{C.7}$$

On choisit $\omega_1' \in]0,1[$, typiquement $\omega_1' = 0{,}99$. On peut démontrer qu'il est possible d'obtenir un pas, appelé pas de Goldstein, vérifiant les équations (C.6) et (C.7) si f_k est continue et bornée inférieurement.

Méthode de Wolfe

La méthode de Wolfe consiste toujours à déterminer un pas α_k, appelé pas de Wolfe, vérifiant la condition (C.6). Ici, afin d'empêcher le pas α_k d'être trop petit, il lui est imposé de vérifier la condition supplémentaire

$$\langle \boldsymbol{\nabla} f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k), \boldsymbol{d}_k \rangle \geq \omega_2\, \langle \boldsymbol{\nabla} f_k, \boldsymbol{d}_k \rangle \tag{C.8}$$

tout en imposant $0 < \omega_1 < \omega_2 < 1$.

Toutes les méthodes de recherche linéaire s'effectuent le long de directions de descente. Différentes façons de déterminer ces directions de descente sont abordées : la méthode du gradient (§C.1.2), la méthode du gradient conjugué (§C.1.3), la méthode de Newton (§C.1.4) et sa généralisation, la méthode de quasi-Newton (§C.1.5).

C.1.2 Méthode du gradient

Cette méthode utilise pour direction de descente l'opposée du gradient de la fonction f au point courant $\boldsymbol{x}$, soit

$$\boldsymbol{d}_k = -\boldsymbol{\nabla} f(\boldsymbol{x}_k).$$

Cette direction est évidemment direction de descente car on a $\langle \boldsymbol{\nabla} f(\boldsymbol{x}_k), \boldsymbol{d}_k \rangle = -\|\boldsymbol{\nabla} f(\boldsymbol{x}_k)\|^2 < 0$.

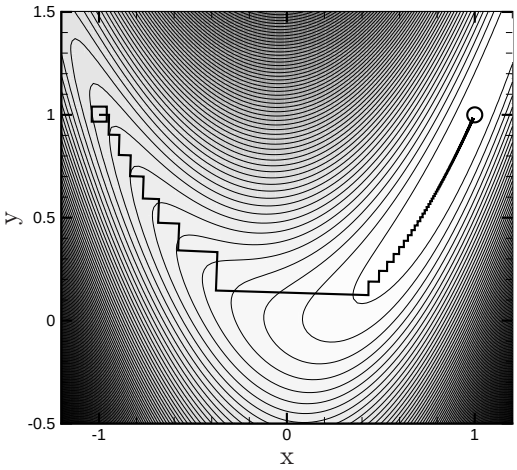


FIGURE C.2 – *Algorithme du gradient à pas optimal appliqué à la fonction Rosenbrock.*

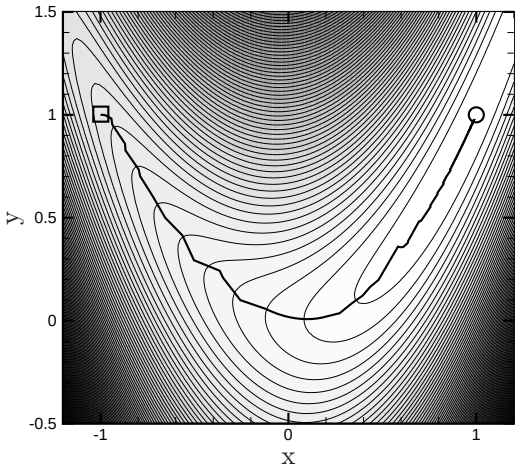


FIGURE C.3 – *Algorithme du gradient à pas d'Armijo appliqué à la fonction Rosenbrock.*

La figure C.2 montre l'évolution de l'algorithme du gradient à pas optimal lors de la minimisation de la fonction Rosenbrock. La convergence est réalisée à l'aide de 406 directions de descente le long desquelles un grand nombre de recherches linéaires est effectué. Le minimum obtenu est $f_{min} = 6{,}13 \times 10^{-5}$ au point $(x\,;\,y) = (0{,}992\,;\,0{,}984)$.

La figure C.3 montre l'évolution de l'algorithme du gradient à pas d'Armijo lors de la minimisation de la fonction Rosenbrock. La convergence est dans ce cas obtenue en 312 directions de descente avec en moyenne 2 étapes de recherche linéaire par direction. Le minimum obtenu est $f_{min} = 10{,}8 \times 10^{-5}$ au point $(x\,;\,y) = (0{,}990\,;\,0{,}979)$.

Dans les deux cas le critère de convergence choisi est $\|\boldsymbol{\nabla} f(\boldsymbol{x}_k)\|_2 < \varepsilon$ avec $\varepsilon = 0.01$. Il est évident qu'un ε plus grand diminue grandement le nombre de directions de descentes, mais les résultats sont largement moins précis. Ce paramètre est laissé au choix des utilisateurs, mais ici afin de comparer toutes les méthodes basées sur le gradient de la fonction f, ce paramètre sera pris égal à 10^{-2}.

C.1.3 Méthodes de gradient conjugué non linéaire

Les méthodes de gradient conjugué non linéaire ne sont pas uniques, mais se déclinent sous plusieurs formes. Le cœur de ces méthodes est présenté par l'algorithme 12.

Algorithme 12 (Méthode du gradient conjugué)

On se fixe $\boldsymbol{x}_0$, puis on calcule $f_0 = f(\boldsymbol{x}_0)$ et $\boldsymbol{\nabla} f_0 = \boldsymbol{\nabla} f(\boldsymbol{x}_0)$. L'algorithme est initialisé par une étape du gradient simple. On a $\boldsymbol{d}_0 = \boldsymbol{\nabla} f_0$.

Tant qu'un critère de convergence n'est pas vérifié :

1. *Détermination d'un pas α_k par la méthode de recherche linéaire de son choix. Calcul d'un nouvel itéré $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$;*
2. *Evaluation d'un nouveau gradient $\boldsymbol{\nabla} f_{k+1}$;*
3. *Calcul du paramètre β_{k+1} par la méthode de son choix (voir ci-dessous) ;*
4. *Construction d'une nouvelle direction de descente $\boldsymbol{d}_{k+1} = -\boldsymbol{\nabla} f_{k+1} + \beta_{k+1} \boldsymbol{d}_k$*

5. Incrémentation : k=k+1 ;

Plusieurs méthodes existent pour calculer le terme β_{k+1} de l'étape 3. Parmi elles se dégagent trois méthodes : la méthode de Fletcher-Reeves (§C.1.3) la méthode de Polack-Ribière (§C.1.3) et une variante de la méthode de Fletcher-Reeves, la méthode de Hesteness-Stiefel (§C.1.3). Il est à noter que cette dernière méthode est particulièrement efficace dans le cas où la fonction f est quadratique et où la recherche linéaire est effectuée de manière exacte.

Fletcher-Reeves

La méthode de Fletcher-Reeves consiste à calculer β_{k+1} de la manière suivante :

$$\beta_{k+1} = \frac{\boldsymbol{\nabla} f_{k+1}^T \boldsymbol{\nabla} f_{k+1}}{\boldsymbol{\nabla} f_k^T \boldsymbol{\nabla} f_k}.$$

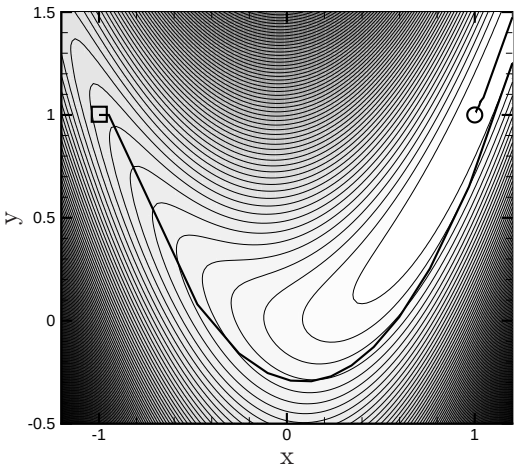


FIGURE C.4 – *Algorithme du gradient conjugué de Fletcher-Reeves à pas optimal appliqué à la fonction Rosenbrock.*

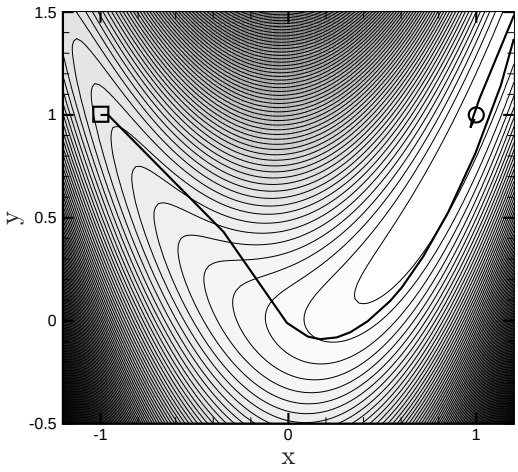


FIGURE C.5 – *Algorithme du gradient conjugué de Fletcher-Reeves à pas d'Armijo appliqué à la fonction Rosenbrock.*

La figure C.4 représente l'évolution de la méthode de gradient conjugué de Fletcher-Reeves à pas optimal. La convergence est réalisée en 25 itérations. Le minimum obtenu est $f_{min} = 3{,}47 \times 10^{-5}$ au point $(x\,;\,y) = (1{,}006\,;\,1{,}001)$.

La figure C.5 représente l'évolution de la méthode de gradient conjugué de Fletcher-Reeves à pas d'Armijo. La convergence est réalisée en 30 itérations. Le minimum obtenu est $f_{min} = 9{,}71 \times 10^{-5}$ au point $(x\,;\,y) = (0{,}990\,;\,0{,}980)$.

Polack-Ribière

La méthode de Polack-Ribière consiste à calculer β_{k+1} de la manière suivante :

$$\beta_{k+1} = \frac{\boldsymbol{\nabla} f_{k+1}^T (\boldsymbol{\nabla} f_{k+1} - \boldsymbol{\nabla} f_k)}{\boldsymbol{\nabla} f_k^T \boldsymbol{\nabla} f_k}.$$

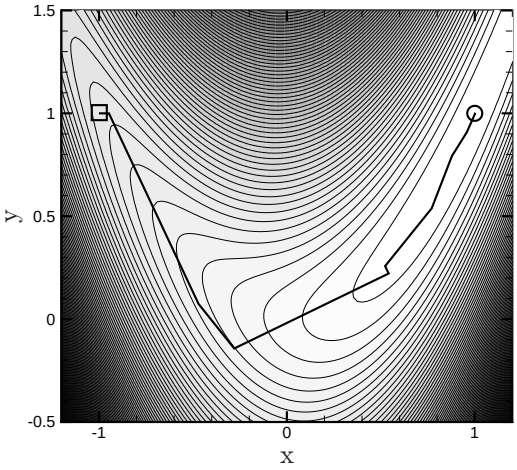


FIGURE C.6 – *Algorithme du gradient conjugué de Polack-Ribière à pas optimal appliqué à la fonction Rosenbrock.*

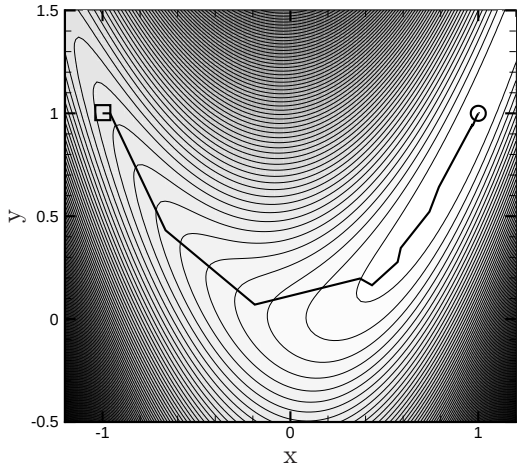


FIGURE C.7 – *Algorithme du gradient conjugué de Polack-Ribière à pas d'Armijo appliqué à la fonction Rosenbrock.*

La figure C.6 représente l'évolution de la méthode de gradient conjugué de Polack-Ribière à pas optimal. La convergence est réalisée en 10 itérations. Le minimum obtenu est $f_{min} = 1{,}16 \times 10^{-6}$ au point $(x\,;\,y) = (0{,}999\,;\,0{,}998)$.

La figure C.7 représente l'évolution de la méthode de gradient conjugué de Polack-Ribière à pas d'Armijo. La convergence est réalisée en 16 itérations. Le minimum obtenu est $f_{min} = 1{,}21 \times 10^{-7}$ au point $(x\,;\,y) = (1{,}000\,;\,1{,}001)$.

Hestenes-Stiefel

La méthode de Hestenes-Stiefel consiste à calculer β_{k+1} de la manière suivante :

$$\beta_{k+1} = \frac{\boldsymbol{\nabla} f_{k+1}^T (\boldsymbol{\nabla} f_{k+1} - \boldsymbol{\nabla} f_k)}{(\boldsymbol{\nabla} f_{k+1} - \boldsymbol{\nabla} f_k)^T \boldsymbol{d}_k}.$$

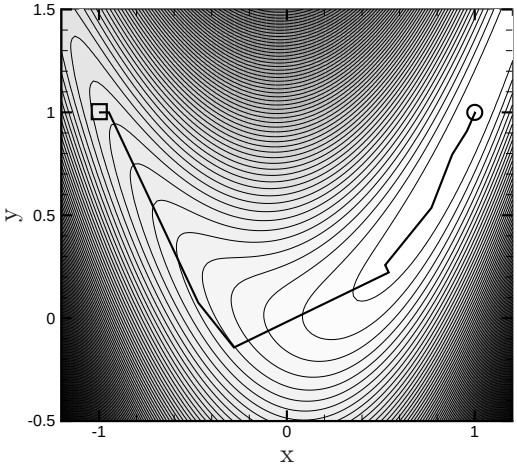


FIGURE C.8 – *Algorithme du gradient conjugué de Hestenes-Stiefel à pas optimal appliqué à la fonction Rosenbrock.*

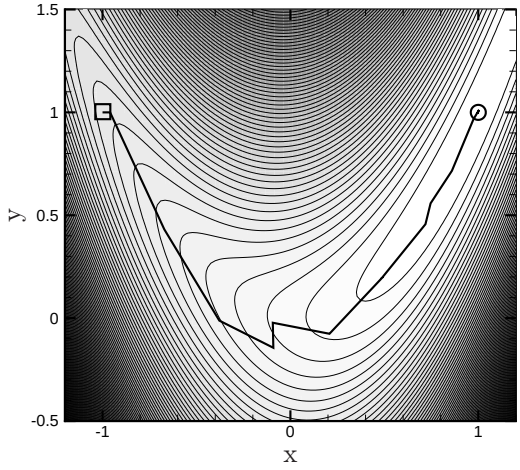


FIGURE C.9 – *Algorithme du gradient conjugué de Hestenes-Stiefel à pas d'Armijo appliqué à la fonction Rosenbrock.*

La figure C.8 représente l'évolution de la méthode de gradient conjugué de Hestenes-Stiefel à pas optimal. La convergence est réalisée en 10 itérations. Le minimum obtenu est $f_{min} = 1{,}27 \times 10^{-6}$ au point $(x\,;\,y) = (0{,}999\,;\,0{,}998)$.

La figure C.9 représente l'évolution de la méthode de gradient conjugué de Hestenes-Stiefel à pas d'Armijo. La convergence est réalisée en 17 itérations. Le minimum obtenu est $f_{min} = 9{,}47 \times 10^{-8}$ au point $(x\,;\,y) = (1{,}000\,;\,1{,}000)$.

On constate que les méthodes de Polack-Ribière et de Hesteness-Stiefel sont bien plus efficaces que la méthode de Fletcher-Reeves, du moins dans le cas où la fonction étudiée est la fonction Rosenbrock "banana".

Il faut à nouveau préciser que dans chaque cas les directions de descentes nécessaires sont moins nombreuses avec un pas optimal que dans le cas du pas d'Armijo : cependant ce dernier est obtenu bien plus rapidement, faisant intervenir beaucoup moins d'évaluation de la fonction f. Dans chaque cas on peut estimer que le temps de calcul est au minimum 10 fois plus court lorsque la méthode d'Armijo est utilisée.

C.1.4 Méthode de Newton

Soient $\boldsymbol{\nabla} f_k$ le gradient et $B_k = \boldsymbol{\nabla}^2 f_k$ le hessien de fonction f au point courant $\boldsymbol{x}_k$.
La méthode de Newton consiste alors à prendre pour direction de descente la direction

$$\boldsymbol{d}_k = -B_k^{-1} \boldsymbol{\nabla} f_k.$$

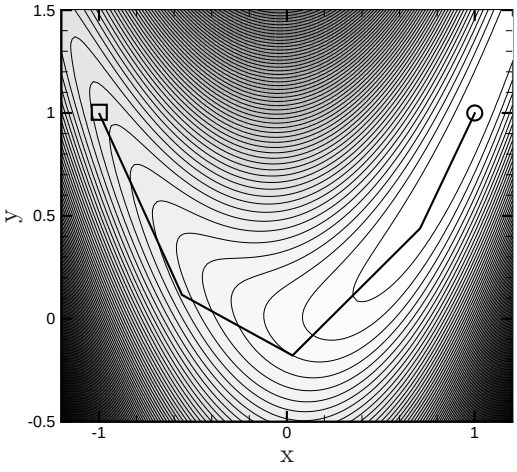


FIGURE C.10 – *Algorithme de Newton à pas optimal appliqué à la fonction Rosenbrock.*

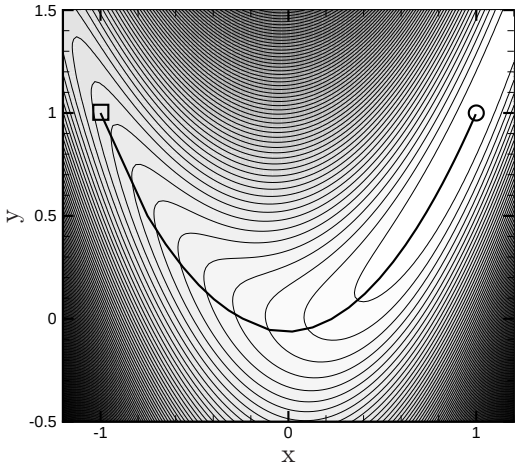


FIGURE C.11 – *Algorithme de Newton à pas d'Armijo appliqué à la fonction Rosenbrock.*

La figure C.10 représente l'évolution de la méthode de Newton à pas optimal. La convergence est réalisée en 7 itérations. Le minimum obtenu est $f_{min} = 2{,}74 \times 10^{-12}$ au point $(x\,;\,y) = (1{,}000\,;\,1{,}000)$.

La figure C.11 représente l'évolution de la méthode de Newton à pas d'Armijo. La convergence est réalisée en 7 itérations. Le minimum obtenu est $f_{min} = 1{,}61 \times 10^{-5}$ au point $(x\,;\,y) = (0{,}996\,;\,0{,}991)$.

La méthode de Newton avec recherche linéaire suivant une méthode d'Armijo à rebroussement montre une capacité à suivre exactement la vallée des minima jusqu'au minimum global de la fonction f.

C.1.5 Méthode de quasi-Newton

Souvent, dans la pratique, le hessien B^{-1} est très difficile à évaluer dans le cas ou la fonction f n'est pas analytique. Le gradient quant à lui est toujours plus ou moins accessible (méthodes inverses). On souhaite donc ne pas calculer exactement le hessien, mais simplement en évaluer une approximation. Parmi les méthodes d'approximation du hessien trois sont retenues ici : la méthodes *BFGS* pour Broyden - Fletcher - Goldfarb -

Shanno (§C.1.5), la méthode *DFP* pour Davidon - Fletcher - Powell (§C.1.5) et la méthode *SR1* pour Matrice Symétrique de Rang 1 (Symmetric Rank One, §C.1.5).

Dans ce quit suit on note $\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$ et $\boldsymbol{y}_k = \boldsymbol{\nabla} f_{k+1} - \boldsymbol{\nabla} f_k$ et on choisit une matrice B_0 définie positive (pour des raison de convexité) : la matrice identité est habituellement choisie.

Actualisation du Hessien par la méthode BFGS

Dans cette approche l'approximation du hessien est donné par

$$B_{k+1} = B_k + \frac{\boldsymbol{y}_k \boldsymbol{y}_k^T}{\boldsymbol{y}_k^T \boldsymbol{s}_k} - \frac{B_k \boldsymbol{s}_k \boldsymbol{s}_k^T B_k}{\boldsymbol{s}_k^T B_k \boldsymbol{s}_k}.$$

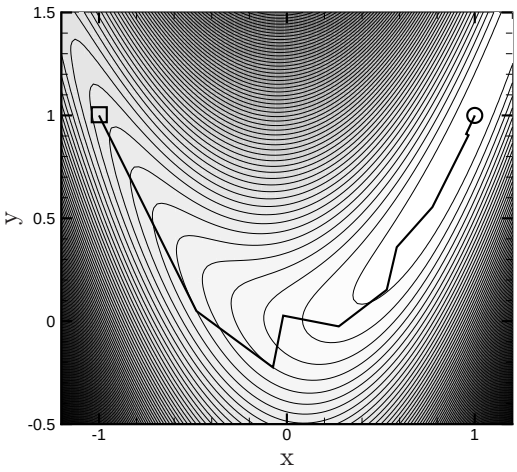


FIGURE C.12 – *Algorithme de quasi-Newton BFGS à pas optimal appliqué à la fonction Rosenbrock.*

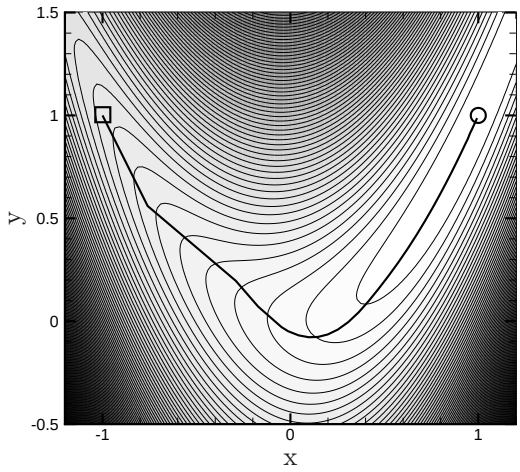


FIGURE C.13 – *Algorithme de quasi-Newton BFGS à pas d'Armijo appliqué à la fonction Rosenbrock.*

La figure C.12 représente l'évolution de l'algorithme de quasi Newton *BFGS* à pas optimal. La convergence est réalisée en 13 itérations. Le minimum obtenu est $f_{min} = 5{,}40 \times 10^{-11}$ au point $(x\,;\,y) = (1{,}000\,;\,1{,}000)$.

La figure C.13 représente l'évolution de l'algorithme de quasi Newton *BFGS* à pas d'Armijo. La convergence est réalisée en 53 itérations. Le minimum obtenu est $f_{min} = 2{,}54 \times 10^{-5}$ au point $(x\,;\,y) = (0{,}995\,;\,0{,}990)$.

Actualisation du Hessien par la méthode DFP

Dans cette approche l'approximation du hessien est donné par

$$B_{k+1} = B_k + \frac{\boldsymbol{s}_k \boldsymbol{s}_k^T}{\boldsymbol{s}_k^T \boldsymbol{y}_k} - \frac{B_k \boldsymbol{y}_k \boldsymbol{y}_k^T B_k}{\boldsymbol{y}_k^T B_k \boldsymbol{y}_k}.$$

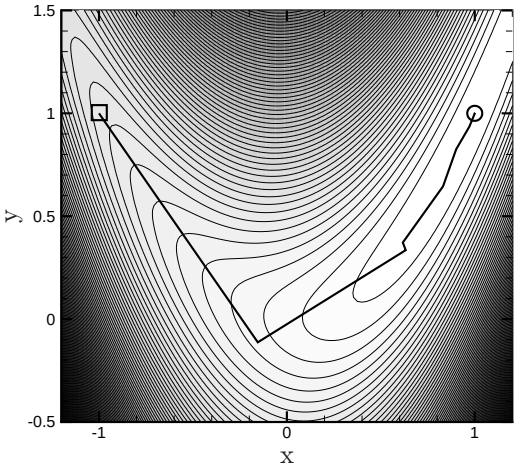


FIGURE C.14 – *Algorithme de quasi-Newton DFP à pas optimal appliqué à la fonction Rosenbrock.*

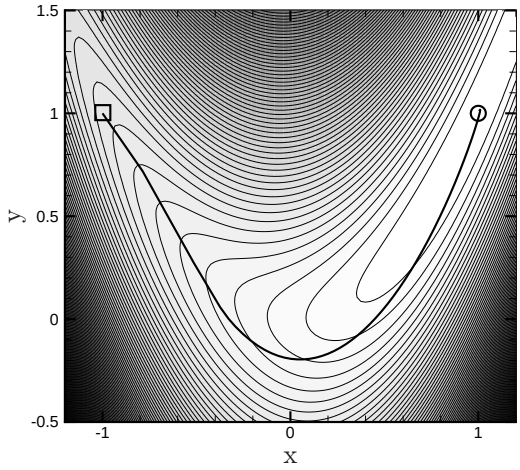


FIGURE C.15 – *Algorithme de quasi-Newton DFP à pas d'Armijo appliqué à la fonction Rosenbrock.*

La figure C.14 représente l'évolution de l'algorithme de quasi Newton *DFP* à pas optimal. La convergence est réalisée en 10 itérations. Le minimum obtenu est $f_{min} = 3{,}05 \times 10^{-11}$ au point $(x\,;\,y) = (1{,}000\,;\,1{,}000)$.

La figure C.15 représente l'évolution de l'algorithme de quasi Newton *DFP* à pas d'Armijo. La convergence est réalisée en 137 itérations. Le minimum obtenu est $f_{min} = 3{,}52 \times 10^{-5}$ au point $(x\,;\,y) = (1{,}002\,;\,1{,}003)$.

Actualisation du Hessien par la méthode SR1

Dans cette approche l'approximation du hessien est donné par

$$B_{k+1} = B_k + \frac{(\boldsymbol{y}_k - B_k \boldsymbol{s}_k)(\boldsymbol{y}_k - B_k \boldsymbol{s}_k)^T}{(\boldsymbol{y}_k - B_k \boldsymbol{s}_k)^T \boldsymbol{s}_k}.$$

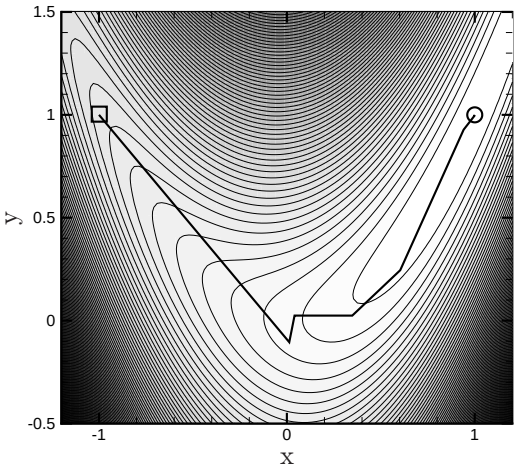


FIGURE C.16 – *Algorithme de quasi-Newton SR1 à pas optimal appliqué à la fonction Rosenbrock.*

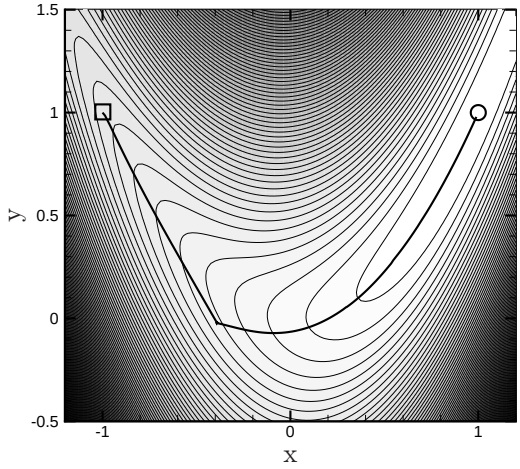


FIGURE C.17 – *Algorithme de quasi-Newton SR1 à pas d'Armijo appliqué à la fonction Rosenbrock.*

La figure C.16 représente l'évolution de l'algorithme de quasi Newton *SR1* à pas optimal. La convergence est réalisée en 10 itérations. Le minimum obtenu est $f_{min} = 5{,}91 \times 10^{-8}$ au point $(x\,;\,y) = (1{,}000\,;\,1{,}000)$.

La figure C.17 représente l'évolution de l'algorithme de quasi Newton *SR1* à pas d'Armijo. La convergence est réalisée en 46 itérations. Le minimum obtenu est $f_{min} = 9{,}5 \times 10^{-5}$ au point $(x\,;\,y) = (0.990\,;\,0.979)$.

C.2 Algorithmes sans calcul de gradient

C.2.1 Méthodes du simplexe

Soit une fonction $f : \mathbb{R}^n \mapsto \mathbb{R}$. Un simplexe est un ensemble de $n+1$ points.

Processus de résolution par la méthode du simplexe de Nelder et Mead :

Étape 0 : Initialisation.

On pose $k = 0$. On construit un simplexe initial. On évalue la valeur de la fonction f en chaque point du simplexe.

Étape 1 : Réflexion.

On cherche $\boldsymbol{x}^k_{max} = \arg\max f(\boldsymbol{x}^k)$ puis on calcule le barycentre $\bar{\boldsymbol{x}}^k$ des n points restants. On évalue ensuite la position $\boldsymbol{x}^k_{ref}$ du point réfléchi de $\boldsymbol{x}^k_{max}$ par rapport à $\bar{\boldsymbol{x}}^k$ de la manière suivante :

$$\boldsymbol{x}^k_{ref} = (1+\alpha)\bar{\boldsymbol{x}}^k - \alpha \boldsymbol{x}^k_{max}.$$

Le point $\boldsymbol{x}^k_{ref}$ remplace donc le point $\boldsymbol{x}^k_{max}$ pour former un nouveau simplexe.

Souvent on désire que le point réfléchi $\boldsymbol{x}^k_{ref}$ soit le symétrique du point $\boldsymbol{x}^k_{max}$ par rapport au barycentre $\bar{\boldsymbol{x}}^k$ et on prend alors $\alpha = 1$.

Étape 2 : Expansion.

Si $f(\boldsymbol{x}^k_{ref}) \leq f(\boldsymbol{x}^k_i)$, $\forall i$, c'est à dire si l'étape de réflexion donne un point correspondant à une valeur de la fonction inférieure aux valeurs des autres point du simplexe, on essaye d'aller plus loin dans cette direction. On expand alors le sommet $\boldsymbol{x}^k_{exp}$ par la relation suivante :

$$\boldsymbol{x}^k_{exp} = \gamma \boldsymbol{x}^k_{ref} + (1-\gamma)\bar{\boldsymbol{x}}^k$$

où γ est laissé au choix de l'utilisateur et varie généralement entre 1 et 2 suivant la convexité de la fonction f.

Si $f(\boldsymbol{x}^k_{exp}) < f(\boldsymbol{x}^k_{ref})$ on remplace le point $\boldsymbol{x}^k_{ref}$ par le point $\boldsymbol{x}^k_{exp}$ dans le nouveau simplexe. Dans le cas contraire on conserve le point $\boldsymbol{x}^k_{ref}$.

Étape 3 : Contraction.

Si $f(\boldsymbol{x}^k_{ref}) \geq f(\boldsymbol{x}^k_i)$, $\forall i$, c'est à dire si l'étape de réflexion donne encore un point correspondant à la valeur maximum de f sur le simplexe, on contracte alors le mouvement :

$$\boldsymbol{x}^k_{con} = \beta \boldsymbol{x}^k_{ref} + (1-\beta)\bar{\boldsymbol{x}}^k.$$

Le paramètre β est laissé au choix de l'utilisateur et varie en général entre 0.2 et 1.

Si $f(\boldsymbol{x}^k_{con}) < f(\boldsymbol{x}^k_{ref})$ on remplace le point $\boldsymbol{x}^k_{ref}$ par le point $\boldsymbol{x}^k_{con}$ dans le nouveau simplexe. Dans le cas contraire on décide d'effectuer un mouvement de **réduction** afin d'adapter le simplexe à la topologie du problème étudié.

Étape 4 : Réduction.

Cette étape est nécessaire lorsque l'étape 3 a échoué. On effectue alors une homothétie autour du point du simplexe $\boldsymbol{x}^k_{min}$ correspondant à la valeurs de f la plus faible. Les autres sommets d'indice i sont alors remplacés par :

$$\boldsymbol{x}_i^k = \frac{\boldsymbol{x}_i^k + \boldsymbol{x}_{min}^k}{2}.$$

Si $f(\boldsymbol{x}_{ref}^k) > f(\boldsymbol{x}_{max}^k)$ le simplexe initial subit une réduction : les étapes 1, 2 et 3 sont alors ignorées.

Si $f(\boldsymbol{x}_{ref}^k) < f(\boldsymbol{x}_{max}^k)$ alors le simplexe réfléchi subit une réduction : les étapes 2 et 3 sont dans ce cas ignorées.

La figure C.18 représente l'évolution du processus d'optimisation par la méthode du simplexe appliquée à la fonction Rosenbrock banana. On constate qu'après une mise en route difficile (le simplexe essai de trouver sa forme la mieux adaptée à la topologie du problème) la méthode du simplexe permet de suivre très convenablement la vallée proposée par la fonction étudiée. La convergence est effectuée en 80 itérations, ce qui est tout à fait convenable. On décide d'arrêter le processus d'optimisation par la méthode du simplexe lorsque l'aire du simplexe (pour $f : \mathbb{R}^2 \mapsto \mathbb{R}$) devient inférieure à un seuil fixé (ici 10^{-5}). En effet en se rapprochant du minimum le simplexe subit des réductions successives. Dans notre étude les valeurs de α et β sont respectivement fixées à 1,2 et à 0,7.

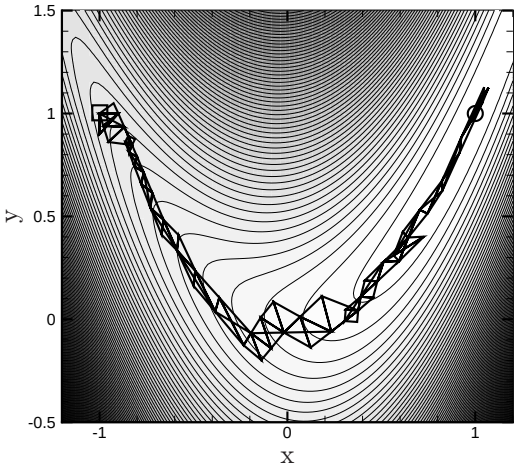


FIGURE C.18 – *Algorithme du simplexe de Nelder et Mead appliqué à la fonction Rosenbrock.*

Le choix des constantes α et β est très important dans le déroulement du processus de résolution car elles influent directement sur la topologie du simplexe. Pour ne pas prendre trop de risques, un choix de ces paramètres autour de 1 (on modifie très légèrement la forme du simplexe) peut souvent être judicieux et peut éviter une fausse convergence due aux rétrécissements trop brutaux de la taille du simplexe.

Les algorithmes de types simplexe fonctionnent donc *a priori* bien, du moins sur certaines configurations : il n'existe cependant aucune preuve de convergence mathématique vers le minimum d'une fonction $f : \mathbb{R}^n \mapsto \mathbb{R}$ pour $n \geq 2$ (Lagarias *et al.*, 1998).

C.2.2 Méthodes de recherche multi-directionnelle

Ce type de méthodes consiste à effectuer des recherches unidirectionnelles suivant des directions orthogonales, formant une base de l'espace considéré, idéalement choisies. Supposons toujours que l'on désire minimiser une fonction $f : \mathbb{R}^n \mapsto \mathbb{R}$: on a donc à effectuer n recherches unidirectionnelles.

Algorithme à base canonique

La méthode la plus évidente pour un problème de recherche multi-directionnelle est de choisir pour directions de recherche $\{\boldsymbol{\xi}_i\}_{i=1,\dots,n}$ la base canonique $\{\boldsymbol{e}_i\}_{i=1,\dots,n}$ de $\mathbb{R}^n$. On est donc amené à réaliser une minimisation unidirectionnelle dans chaque direction successive, et ceci jusqu'à convergence. Ce type d'algorithme présente le défaut majeur d'avoir un comportement oscillant (recherche suivant des directions orthogonales) menant au minimum. Cette constatation a déjà été observée pour l'algorithme du gradient à pas optimal, où deux directions successives (gradients) étaient orthogonales.

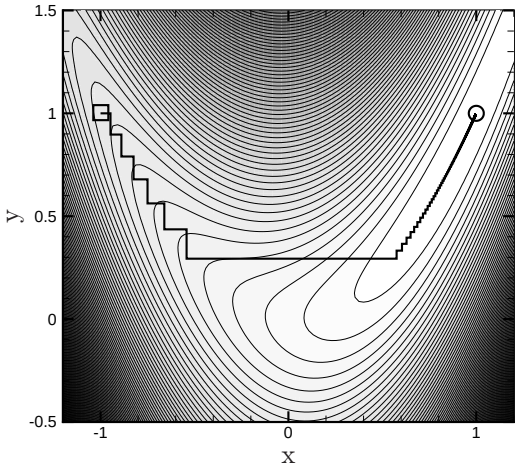


FIGURE C.19 – *Algorithme de recherche multi-directionnelle appliqué à la fonction Rosenbrock.*

Algorithme de Rosenbrock

C.3 Méthodes à régions de confiance

L'idée des méthodes à régions de confiance consiste à remplacer la résolution du problème (C.1) par la résolution d'une sequence de sous-problèmes plus simples où la fonction f est remplacée une fonction modèle m.

La fonction modèle est habituellement choisie quadratique de façon à simplifier la résolution des sous problèmes (§C.3.1). Il arrive souvent des situations où le gradient de la fonction f n'est pas disponible analytiquement, et il convient de l'approcher numériquement dans l'optique de déterminer un modèle quadratique de la fonction f (§C.3.2). Enfin, des éléments algorithmiques sur des fonctions modèles quelconques seront présentés (§C.3.3).

C.3.1 Fonctions modèles quadratiques basées sur un gradient exact

La fonction quadratique m^{quad} est construite comme étant la troncature du développement en série de Taylor de f autour du point courant $\boldsymbol{x}_k$ à l'ordre 2. La fonction quadratique modèle de la fonction f utilisée durant le processus itératif à l'étape k est

$$m_k^{quad}(\boldsymbol{x}_k + \boldsymbol{r}) = f(\boldsymbol{x}_k) + \boldsymbol{\nabla} f(\boldsymbol{x}_k)^T \boldsymbol{r} + \frac{1}{2} \boldsymbol{r}^T H_k \, \boldsymbol{r} \tag{C.9}$$

où $\boldsymbol{\nabla} f$ est le gradient et la matrice symétrique H représente le hessien (exact ou approximé) de la fonction objective f au point courant $\boldsymbol{x}$. Une contraite de région de confiance est ajoutée à la fonction m_k^{quad} définie en (C.9)

$$\|\boldsymbol{r}\| \leq \Delta_k \tag{C.10}$$

où $\Delta_k > 0$ est appelé le rayon de la région de confiance au point $\boldsymbol{x}_k$, définissant une région dans laquelle la fonction m_k^{quad} est supposée modéliser correctement la fonction objective f. Ce rayon de confiance Δ_k évolue au cour du processus itératif traduisant la capacité de le fonction m_k^{quad} à représenter les non-linéarités de la fonction objective f.

Au point courant $\boldsymbol{x}_k$, le sous problème à résoudre est :

$$\boldsymbol{r}_k = \arg\min_{\boldsymbol{r} \in \mathbb{R}^n} m_k^{quad}(\boldsymbol{x}_k + \boldsymbol{r}) \quad \text{sous la contrainte} \quad \|\boldsymbol{r}\| \leq \Delta_k \tag{C.11}$$

pour un rayon de confiance Δ_k déterminé au préalable. La solution $\boldsymbol{r}_k$ du problème C.11 est en général une solution approchée.

Si la valeur de la fonction objective f évaluée au point $\boldsymbol{x}_k+\boldsymbol{r}_k$ est assez petite devant la valeur de f évaluée au point $\boldsymbol{x}_k$ alors le nouveau point $\boldsymbol{x}_k+\boldsymbol{r}_k$ est accepté. Dans le cas contraire, si $f(\boldsymbol{x}_k+\boldsymbol{r}_k) > f(\boldsymbol{x}_k)$, on résout à nouveau le problème C.11 avec un rayon de confiance Δ_k plus petit. On passe ensuite à l'étape $k+1$, avec réactualisation de la fonction quadratique modèle m_{k+1}^{quad} : on est donc amené à calculer un nouveau gradient $\boldsymbol{\nabla} f_{k+1}$ et un nouvel Hessien H_{k+1} et à déterminer un nouveau rayon de confiance Δ_{k+1}. La détermination du nouveau rayon de confiance est basé sur l'observation de la comparaison entre la décroissance réelle de la fonction f entre deux étapes successives et la décroissance prédite par la fonction modèle.

On introduit donc la réduction réelle

$$\mathcal{R}_k = f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + \boldsymbol{r}_k)$$

et la réduction prédite par la fonction modèle

$$\mathcal{P}_k = m^{quad}(\boldsymbol{x}_k) - m^{quad}(\boldsymbol{x}_k + \boldsymbol{r}_k).$$

L'algorithme modèle pour une méthode à régions de confiance est présenté par l'algorithme 13.

Algorithme 13 (Méthodes à région de confiance)

Initialisations : Choisir ε petit, $0 < \eta_1 < \eta_2 < 1$, $0 < \gamma_1 \leq \gamma_2 < 1 \leq \gamma_3$, un rayon de confiance initial $\Delta_0 > 0$ et un itéré initial $\boldsymbol{x}_k$. Poser $k = 0$.

1. *Test de convergence : si $\|\boldsymbol{\nabla} f(\boldsymbol{x}_k)\| < \varepsilon$ et si H_k est semi-défini positif, on arrête le calcul. Sinon on va à l'étape 2.*
2. *Construire le modèle quadratique m_k^{quad} est déterminer une solution (approchée) $\boldsymbol{r}_k$ du problème*

$$\min_{\|\boldsymbol{r}\| \leq \Delta_k} m_k^{quad}(\boldsymbol{x}_k + \boldsymbol{r}).$$

3. *Calculer $f(\boldsymbol{x}_k + \boldsymbol{r})$ et*

$$\rho_k = \frac{\mathcal{R}_k}{\mathcal{P}_k}.$$

4. *Évaluer un nouveau rayon de confiance :*
   - *Si $\rho_k \geq \eta_2$ on pose $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{r}_k$ et on choisit $\Delta_{k+1} \in [\Delta_k, \gamma_3 \Delta_k]$.*
   - *Si $\eta_1 \leq \rho_k < \eta_2$ on pose $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{r}_k$ et on choisit $\Delta_{k+1} \in [\gamma_2 \Delta_k, \Delta_k]$.*
   - *Si $\rho_k < \eta_1$ on pose $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ et on choisit $\Delta_{k+1} \in [\gamma_1 \Delta_k, \gamma_2 \Delta_k]$.*
5. *Poser $k = k + 1$ et retourner à l'étape 1.*

D'autres tests de convergence peuvent être effectués à l'étape 2 de l'algorithme précédent. On choisit ici de vérifier approximativement les conditions d'optimalité au second ordre.

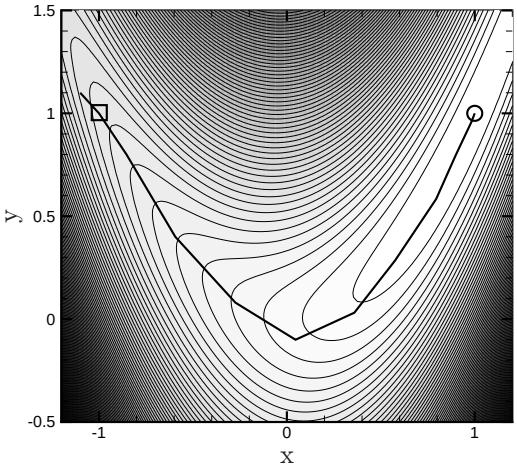


FIGURE C.20 – *Algorithme à régions de confiance basé sur un gradient et un hessien exact appliqué à la fonction de Rosenbrock.*

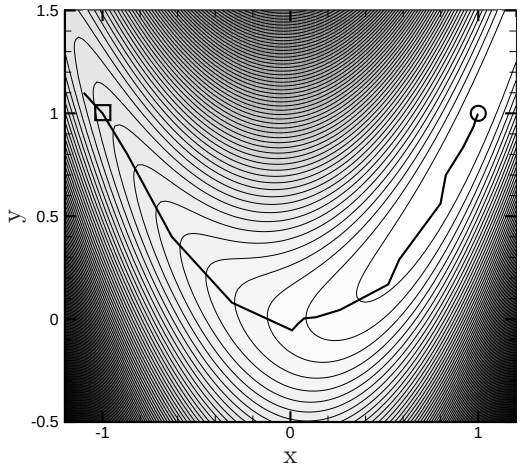


FIGURE C.21 – *Algorithme à régions de confiance basé sur un gradient exact et un hessien BFGS appliqué à la fonction de Rosenbrock.*

La figure C.20 représente l'évolution de l'algorithme à régions de confiance basé sur un gradient et un hessien exacts appliqué à la fonction de Rosenbrock. La convergence est réalisée en 14 itérations. Le minimum obtenu est $f_{min} = 3{,}39 \times 10^{-8}$ au point $(x\,;\,y) = (1{,}000\,;\,0{,}999)$.

La figure C.21 représente l'évolution de l'algorithme à régions de confiance basé sur un gradient exact et un hessien approximé par une méthode BFGS appliqué à la fonction de Rosenbrock. La convergence est réalisée en 27 itérations. Le minimum obtenu est $f_{min} = 2{,}28 \times 10^{-8}$ au point $(x\,;\,y) = (1{,}000\,;\,1{,}000)$.

C.3.2 Fonctions modèles quadratiques basées sur un gradient inexact

C.3.3 Fonctions modèles quelconques

C.4 Algorithmes génétiques

Les algorithmes génétiques, contrairement au méthodes déterministes présentées ci dessus, sont des méthodes aléatoires basées sur les probalités. Comme leur nom l'indique ce type d'algorithme se base sur l'évolution biologique d'une population d'individus. Chaque individus est caractérisée par un génome (séquance de chromozomes ou allèles). La population évolue par sélection (seulement les individus les mieux adaptés au milieu extérieur survivent), par croisement (génèrant des enfants comportant des séquences de chromozomes de leur parents) et par mutation d'une partie de leur génome.

Ce type d'algorithme est très adapté à la minimisation (maximisation) d'une fonction présentant plusieurs minima (locaux ou globaux). Le coût de calcul est cependant très important car tous les individus de chaque génération doivent être évalués, faisant autant de fois appel au calcul de la fonctionnelle à optimiser.

Un algorithme génétique se base sur les étapes suivantes :

Codage

On rappelle que l'on cherche $\boldsymbol{x} = \arg\min f(\boldsymbol{x})$ avec $f : \mathbb{R}^n \mapsto \mathbb{R}$ et $\boldsymbol{x} = [x_1, x_2, \cdots, x^n]$. Chaque x_i, $i = 1, \cdots, n$ correspond à une valeur réelle qui peut être codée à l'aide d'un alphabet : l'alphabet le plus simple, composé de deux caractères, est l'alphabet binaire (composé de 0 et de 1). Ce nombre, une fois codé, est appelé chromosome. Chaque caractère du chromosome (0 ou 1) est appelé gène (ou allèle). Si l'on cherche $\boldsymbol{x}$ dans un domaine borné, alors on a $x_i^{inf} < x_i < x_i^{sup}$, $\forall i$. Le nombre de gènes par composante x_i nécessaires au codage binaire est alors le plus petit entier m_i vérifiant :

$$10^{N_i}(x_i^{sup} - x_i^{inf}) \leq 2^{m_i} - 1$$

où N_i est le nombre de chiffres significatifs désiré. Le point $\boldsymbol{x}$ est alors construit en ajoutant bout à bout tous les codage des x_i. La chaîne ainsi constituée comprend alors $m = \sum_{i=1}^{n} m_i$ caractères.

A l'aide de ce codage on génère aléatoirement une population initiale de n_{ind} individus. Une fois ces individus générés, les opérateurs de sélection, de croisement et de mutation peuvent être appliqués à cette population.

Sélection

Cet opérateur a pour but la selection des individus les mieux adaptés au milieu extérieur, i.e. les individus correspondant aux valeur les plus faibles (*resp.* élevées) dans le cas d'un minimisation (*resp.* maximisation).

Une méthode bien adaptée pour une sélection est une méthode dite de tournoi. Lors de cette étapes des individus seront supprimés et d'autres seront dupliqués afin que la population soit toujours constituée de n_{ind} individus. On organise alors n_{ind} tournois de n_{part} individus choisis aléatoirement dans la population. Un individus peut être choisi pour plusieurs tournois. Le vainqueur du tournoi est l'individu qui est le mieux adapté à ce que l'on désire obtenir. A ce moment, uniquement les individus présentant les meilleurs caractéristiques (séquences de chromosomes) sont conservés.

Croisement

L'idée du croisement est de créer des enfants. On associe donc aléatoirement des individus de la population deux à deux afin de former des couples. Chaque couple va donner naissance à deux enfants comportant chacun des séquences chromosomiques de chaque parent.

Une méthode efficace pour générer des enfant est la méthode dite *des masques*. Un masque est un chromosome de m gènes constitué de 0 et de 1 (dans le cas d'un codage binaire), et est généré aléatoirement pour chaque couple. Le premier enfant est généré de la manière suivante : lorsque le masque comporte un 1 le gène correspondant est la copie du premier parent (choisi au hasard), et lorsque le masque comporte un 0 le gène est la copie du gène du second parent. Le second enfant est généré de manière identique en inversant les 1 et les 0 du masque.

On dispose toujours de n_{ind} individus, *a priori* mieux adaptés au milieu extérieur.

Mutation

L'opérateur de mutation consiste à modifier, avec une faible probabilité p_m, certains gènes de la nouvelle population.

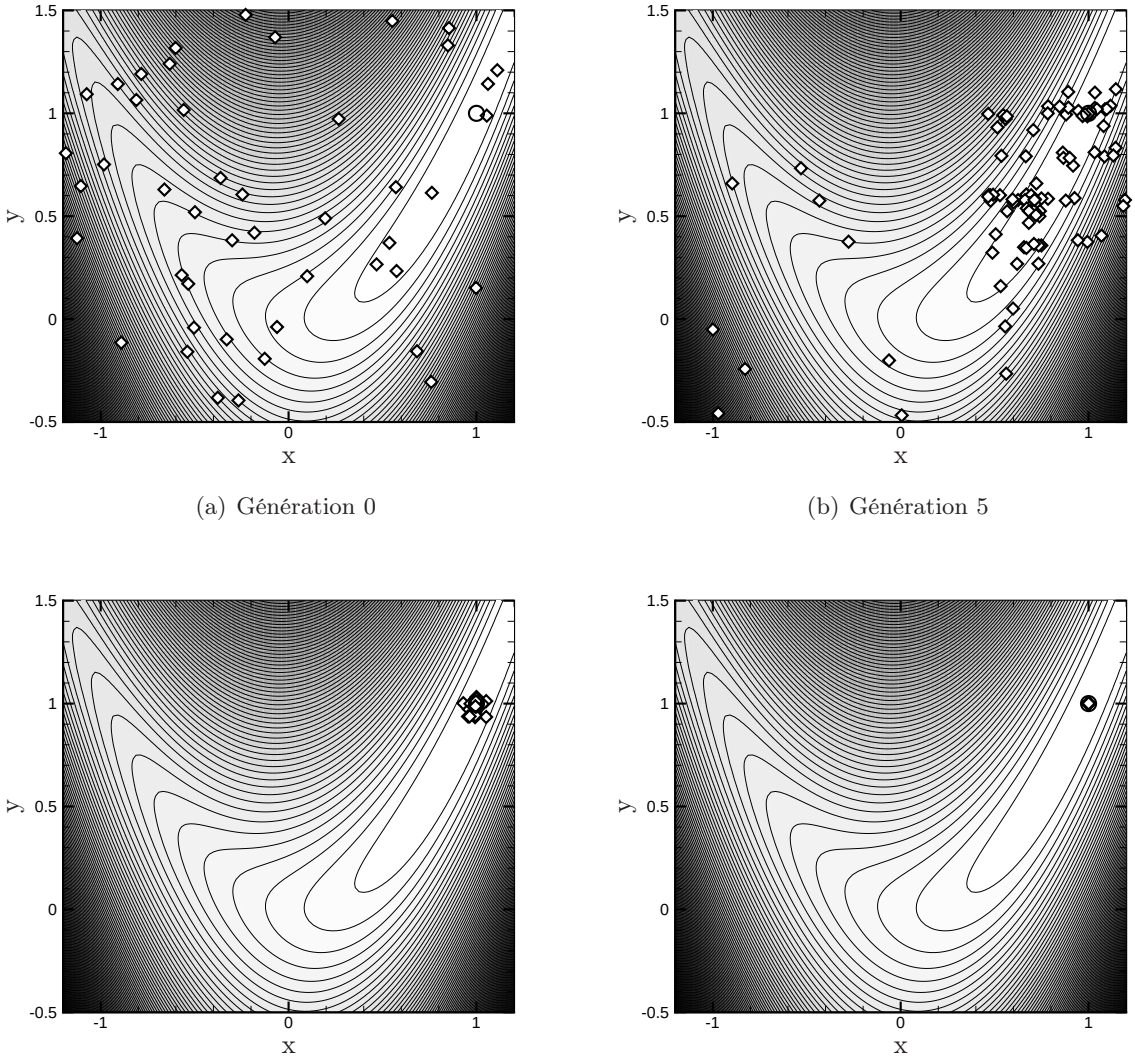
Elitisme

Cet opérateur est introduit afin que le meilleur individu de la population survive au 3 étapes mentionnées ci dessus à savoir sélection, croisement et mutation. En effet, si le meilleur individu n'est pas autorisé à participer au tournoi de sélection (s'il n'est pas sélectionné), il disparaîtra. Par croisement, il est possible que des séquences de gènes favorables disparaîtront. Il en est de même pour l'opérateur de mutation. On peut donc penser isoler le meilleur individu avant la sélection et de le réintroduire aléatoirement dans la population à la place d'un autre individu après l'étape de mutation.

Tous les opérateurs définis ci dessus sont donnés à titre d'exemples et il existe bien d'autres manières de les appliquer.

Exemple d'algorithme génétique

On choisit pour cette exemple une population initiale de 100 individus. La selection est réalisée par tournois de 4 individus, le croisement est effectué en échangeant aléatoirement des séquences de gènes des deux parents, et la probabilité de mutation est fixée à 0,025. Le codage utilisé est le codage Gray.



(a) Génération 0

(b) Génération 5

(c) Génération 10

(d) Génération 15

FIGURE C.22 – *Evolution d'un algorithme génétique appliqué à la fonction de Rosenbrock.*

Le minimum obtenu ici correspond exactement au minimum de la fonction Rosenbrock banana, c'est à dire le point de coordonnée (1,00 ; 1,00), et est obtenu en 15 générations. Il est inutile de préciser qu'il s'agit ici d'un cas très favorable et que le minimum exact est rarement atteint, et quand il est atteint il faut en général bien plus de générations.

Etude paramétrique

On décide de tester ici l'influence du nombre d'individus présents dans la population. On se limite à 500 évaluations de la fonctionnelle à minimiser. Trois cas sont proposés :

- 25 générations de 20 individus,
- 10 générations de 50 individus,
- 5 générations de 100 individus.

On évalue alors le minimum de la fonctionnelle obtenu au cours des générations dans les trois cas précédents. Les résultats correspondent à la moyenne statistique réalisée sur 100 essais, et ceci étant réalisé à trois reprise dans chaque cas. On constate sur la figure C.23 que la configuration comportant 50 individus évoluant sur 10 générations semble être la plus favorable. Il semblerai donc qu'il existe un nombre d'individus optimal. On voit donc bien la capacité d'évolution de l'algorithme génétique.

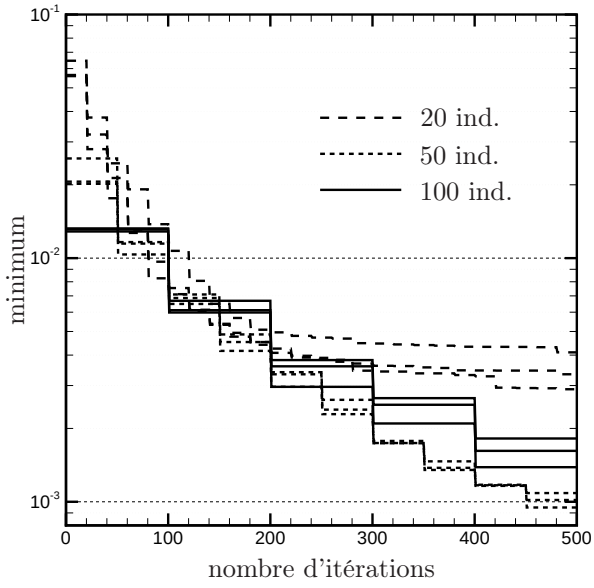


FIGURE C.23 – *Influence du nombre d'individus présents dans la population initiale d'un algorithme génétique.*