3.4 Méthode du gradient conjugué

Dans l'algorithme du gradient à pas optimaux, les directions de descente

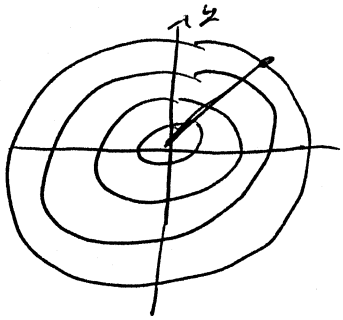
$$d_k = -\nabla f(x_k)$$

vérifient la propriété

$$d_{k+1} \perp d_k .$$

Pour des courbes de niveau très aplaties, la convergence peut être lente à cause du mouvement en zig-zag.

Par contre, pour des courbes de niveau presque sphériques, la convergence est très rapide.



Ex: $\min \frac{x^2+y^2}{2}$

L'alg. converge en 1 itération.

Le problème provient de la notion de "perpendicularité" imposée par $d_{k+1} \perp d_k$. Il est préférable de modifier le produit scalaire de sorte que

$$d_{k+1} \perp_{\text{pour le nouveau produit scalaire}} d_k$$

Dans ce qui suit, nous allons considérer seulement le cas quadratique

$$f(x) = \frac{1}{2}(Ax, x) - (b, x) \qquad \begin{matrix} A > 0 \\ b \in \mathbb{R}^n \end{matrix}$$

Ce qui motive ce choix est le fait que toute fonction (de classe C^2) peut être approchée (localement) par une fonction quadratique. Par conséquent, il est préférable que l'algorithme de minimisation soit performant en premier lieu dans le cas quadratique.

Définition : A- conjugué

Un ensemble de directions $d_0, d_1, \dots, d_k$ est dite A-conjugué (ou conjugué) si

$$(A d_i, d_j) = 0 \qquad \forall i \neq j$$

Remarque : $\langle x, y \rangle = (Ax, y)$ définit un produit scalaire dans $\mathbb{R}^n$ car $A > 0$.
donc $\langle x, y \rangle = 0$ $\Leftrightarrow$ x et y sont A-conjugués.

L'algorithme du gradient conjugué (dans le cas linéaire) permet de construire deux suites à l'itération k

- $\{r_0, r_1, r_2, \dots, r_k\}$ qui forment un système orthogonal
$$r_i \perp r_j \qquad \forall i \neq j.$$
Ceci est plus "fort" que $r_{k+1} \perp r_k$.

- $\{d_0, d_1, \ldots, d_k\}$ qui forme un système A-conjugué

$$(Ad_i, d_j) = 0 \quad \forall i \neq j.$$

On consultera le livre de M. Delfour pour la construction explicite des r_i et des d_i.

Voici le principe de base de l'algorithme du gradient conjugué qui est une méthode de descente basée sur des directions A-conjuguées.

ALGORITHME DU GRADIENT CONJUGUE

- Mise à jour de x_k :

(1) $$x_{k+1} = x_k + \rho_k d_k \quad \text{où } \rho_k > 0$$

ρ_k est choisi comme dans l'alg. du gradient à pas optimal

$$\min_{\rho} f(x_k + \rho d_k)$$

- Mise à jour du résidu r_k

(2) $$r_{k+1} = r_k - \rho_k A r_k$$

Conséquence directe de (1)

- Mise à jour des directions conjuguées d_k

(3) $$d_{k+1} = r_{k+1} + \beta_k d_k$$

où β_k est choisi de sorte que $(Ad_{k+1}, d_k) = 0$

Voici comment évaluer ρ_k :

$$f(x_k + \rho_k d_k) = \min_{\rho} f(x_k + \rho d_k)$$

$$\Rightarrow \left(\nabla f(x_{k+1}), d_k \right) = 0$$

$$\Leftrightarrow \quad (r_{k+1}, d_k) = 0$$

$$\Leftrightarrow \quad (r_k - \rho_k A d_k, d_k) = 0$$

$$\Rightarrow \quad \rho_k = \frac{(r_k, d_k)}{(A d_k, d_k)}$$

Mais

$$d_k = r_k + \beta_{k-1} d_{k-1}$$

$$(r_k, d_k) = (d_k - \beta_{k-1} d_{k-1}, d_k)$$

$$= r_k + \beta_{k-1} (r_{k-1} + \beta_{k-2} d_{k-2})$$

$$\vdots$$

$$d_k = r_k + \sum_{i=0}^{k-1} \alpha_i r_i$$

$$\Rightarrow \left(d_k, r_k \right) = \left(r_k + \sum_{i=0}^{k-1} \alpha_i r_i, r_k \right)$$

$$= (r_k, r_k) \quad \text{car } r_i \perp r_k \ \underline{\forall i < k}$$

$$\therefore \quad \boxed{\rho_k = \frac{\| r_k \|^2}{(A d_k, d_k)}}$$

Maintenant, passons au paramètre β_k.

On doit utiliser le fait que

$$(A d_{k+1}, d_k) = 0$$

$$\Leftrightarrow \quad (d_{k+1}, A d_k) = 0$$

$$\Leftrightarrow \quad (r_{k+1} + \beta_k d_k, A d_k) = 0$$

$$\Rightarrow \quad \beta_k = \frac{-(A d_k, r_{k+1})}{(A d_k, d_k)}$$

Mais, grâce à la mise à jour (2) de r_k,

$$A d_k = \frac{r_{k+1} - r_k}{-\rho_k}$$

$$\Rightarrow \quad \beta_k = \frac{1}{\rho_k} \frac{(r_{k+1} - r_k, r_{k+1})}{(A d_k, d_k)}$$

$$= \frac{\|r_{k+1}\|^2}{\rho_k (A d_k, d_k)} \qquad \text{car } r_k \perp r_{k+1}$$

$$= \frac{\|r_{k+1}\|^2}{\frac{\|r_k\|^2}{(A d_k, d_k)} (A d_k, d_k)}$$

$$= \frac{\|r_{k+1}\|^2}{\|r_k\|^2}$$

En résumé, voici l'algorithme du gradient conjugué avec la phase d'initialisation.

Pour x_0 donné. Poser $r_0 = b - Ax_0$ et $d_0 = r_0$.

Pour $k = 0, 1, 2, \ldots$ jusqu'à convergence, faire :

- $\alpha_k = \dfrac{(r_k, r_k)}{(Ad_k, d_k)}$
- $x_{k+1} = x_k + \alpha_k d_k$
- $r_{k+1} = r_k - \alpha_k A d_k$
- $\beta_k = \dfrac{\|r_{k+1}\|^2}{\|r_k\|^2}$
- $d_{k+1} = r_{k+1} + \beta_k d_k$

Remarques : — le critère d'arrêt est habituellement de la forme : $\|r_k\| < \varepsilon$.

— étant donné que l'algorithme construit un système orthogonal $\{r_0, r_1, \ldots, r_k\}$, l'algo. doit converger en au plus n itérations car il est impossible d'avoir un système orthogonal de plus de n éléments à moins qu'un des $r_i = 0$.

— C'est une méthode très efficace pour résoudre des systèmes linéaires

$$Ax = b.$$

avec $A > 0$.

3.5 Algorithme de Newton

Soit $f : U \to \mathbb{R}$ une fonction numérique de classe C^2 définie sur un ouvert $U \subset \mathbb{R}^n$.

Une condition nécessaire pour qu'un point $a \in U$ soit un minimum (local ou global) est que :

$$\nabla f(a) = 0$$

Par conséquent, une façon de calculer les points minimisant est de résoudre l'équation

$$F(x) = \nabla f(x) = 0.$$

Cette équation forme un système de n équations à n variables

$$\begin{cases} F_1(x_1, x_2, \dots, x_{n-1}) = 0 \\ F_2(x_1, x_2, \dots, x_{n-1}) = 0 \\ \vdots \\ F_n(x_1, x_2, \dots, x_{n-1}) = 0 \end{cases}$$

Par la suite, nous allons voir comment calculer numériquement une solution du système $F(x) = 0$.

Cas d'une fonction d'une variable réelle :

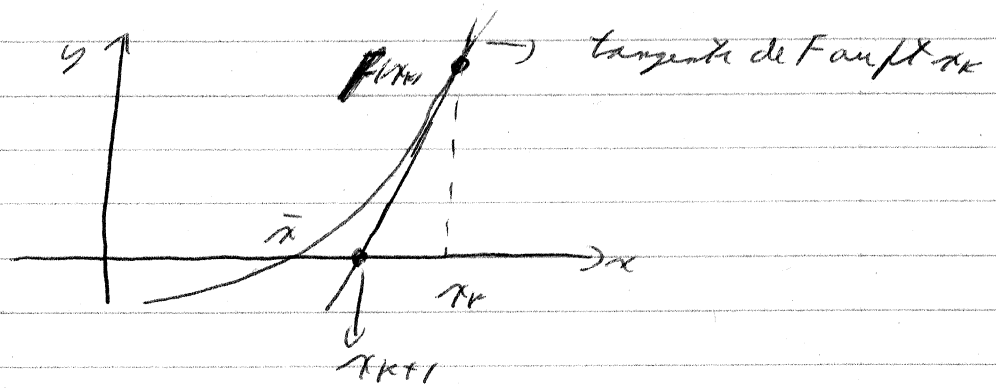
Nous allons présenter l'algorithme de Newton pour la résolution d'une équation à une variable

$$F(x) = 0$$

Le principe est le suivant:

Etant donné une approximation x_k de la racine de $F(x)=0$, on désire améliorer cette valeur pour produire un nouvel itéré x_{k+1}.

On choisit x_{k+1} de la manière suivante



Un calcul donne

$$x_{k+1} = x_k - \frac{F(x_k)}{F'(x_k)}$$

C'est l'algorithme de Newton qui fonctionne sous l'hypothèse que

$$F'(\bar{x}) \neq 0 \Rightarrow F'(x_k) \neq 0$$

si les x_k sont suffisamment près de $\bar{x}$.

Une autre façon de présenter cet algorithme est basée sur l'approximation linéaire de F autour du pt. x_k

$$F(x_k + \Delta x) \cong F(x_k) + F'(x_k)\,\Delta x$$

On désire corriger la valeur de x_k de sorte que

$$F(x_k + \Delta x) = 0$$

Au lieu d'imposer cette égalité à F, on l'impose sur son approximation linéaire

$$F(x_k) + F'(x_k)\,\Delta x = 0$$

On obtient

$$\Delta x = \frac{-F(x_k)}{F'(x_k)}$$

et $x_{k+1} = x_k + \Delta x$

d'où le même algorithme que précédemment.

Exemple: écrire l'algo. de Newton pour

$$x^2 = 2$$

$$F(x) = x^2 - 2 \quad \Rightarrow \quad F'(x) = 2x$$

$$x_{k+1} = x_k - \frac{(x_k^2 - 2)}{2x_k} = \frac{x_k + 2/x_k}{2}$$

Généralisation à n variables :

Soit $F : U \longrightarrow \mathbb{R}^n$ une fonction dérivable à valeur vectorielle.

Définition: matrice Jacobienne en $x = x_0$

$$\left(DF(x_0)\right)_{ij} = \left(\frac{\partial F_i}{\partial x_j}(x_0)\right)_{i,j=1}^{n}$$

$$DF = \begin{matrix} \\ i \end{matrix} \begin{pmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\ & \vdots & \\ \frac{\partial F_i}{\partial x_j} & \cdots & \\ & \vdots & \\ \frac{\partial F_n}{\partial x_1} & \cdots & \frac{\partial F_n}{\partial x_n} \end{pmatrix}_{n \times n} \quad j$$

Formule d'approximation linéaire de F autour d'un point donné x_0.

$$F(x_0 + \Delta x) \tilde{=} F(x_0) + DF(x_0)\,\Delta x$$

↓ produit matrice-vecteur

A l'aide de cet outils, nous sommes prêts à traiter de l'algorithme de Newton.

On désire corriger la valeur x_k d'une première approximation du système $F(x) = 0$

$$0 = F(x_k + \Delta x) \tilde{=} F(x_k) + DF(x_k)\,\Delta x$$

On impose :

$$F(x_k) + DF(x_k)\,\Delta x = 0$$

$$\Rightarrow \quad \Delta x = -\left[DF(x_k)\right]^{-1} F(x_k)$$

Algorithme de Newton

- x_0 donné
- $x_{k+1} = x_k - \left[DF(x_k)\right]^{-1} F(x_k)$

On doit faire l'hypothèse que la matrice jacobienne soit inversible au point $\bar{x}$ i.e. $F(\bar{x}) = 0$

$DF(\bar{x})$ inversible $\Rightarrow$ $DF(x_k)$ inversible pour tous les x_k suffisamment près de $\bar{x}$.

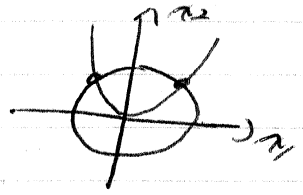
Le critère d'arrêt est de la forme

1) $\| F(x_k) \| < \epsilon$

et/ou 2) $\| x_{k+1} - x_k \| < \epsilon$

Exemple :

$$\begin{cases} x_1^2 + x_2^2 - 1 = 0 \\ x_2 - x_1^2 = 0 \end{cases}$$



$\Rightarrow$ admet 2 solutions.

La matrice Jacobienne :

$$DF = \begin{pmatrix} 2x_1 & 2x_2 \\ -2x_1 & 1 \end{pmatrix} \Rightarrow \det DF = 2x_1 + 4x_2^2 x_1$$

Il est clair que $\det DF \neq 0$ aux 2 solutions de $F(x)=$

L'alg. de Newton s'écrit

$$\begin{pmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{pmatrix} = \begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \end{pmatrix} + \begin{pmatrix} 2x_1^{(k)} & 2x_2^{(k)} \\ -2x_1^{(k)} & 1 \end{pmatrix}^{-1} \begin{pmatrix} 1 - x_1^{(k)2} - x_2^{(k)} \\ x_1^{(k)2} - x_2^{(k)} \end{pmatrix}$$

Rappel :

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \left(\frac{1}{ad - bc} \right)$$

$$\Rightarrow \begin{pmatrix} 2x_1 & 2x_2 \\ -2x_1 & 1 \end{pmatrix}^{-1} = \frac{1}{2x_1 + 4x_2^2 x_1} \begin{pmatrix} 1 & -2x_2 \\ 2x_1 & 2x_1 \end{pmatrix}$$

Maintenant revenons au problème du calcul approché du minimum de f

$$\min_{x \in U} f(x) = f(\bar{x})$$

On pose $F(x) = Df(x) = 0$

La matrice Jacobienne de F correspond à la matrice Hessienne

$$H_f(x) = DF(x) = \left(\frac{\partial^2 f}{\partial x_i \partial x_j} \right)$$

qui est toujours symétrique.

L'algorithme de Newton s'écrit :

$$\begin{cases} x_0 \text{ donné} \\ x_{k+1} = x_k - \left[H_f(x_k) \right]^{-1} Df(x_k) \end{cases}$$

Remarques :

- cet algorithme est très sensible au choix du point de départ
- si l'algorithme converge, elle converge de manière quadratique

$$e_{k+1} \leq C \, e_k^2 \qquad C = \text{constante} > 0$$

où e_k désigne l'erreur à l'itération k

$$e_k = \| x_k - \bar{x} \|$$

- On doit exiger que la matrice Jacobienne soit inversible au voisinage du pt $\bar{x}$.

  Ceci est vérifié si

  $$H_f(\bar{x}) > 0 \quad \left(\text{condition suffisante du 2}^{\text{nd}} \text{ ordre}\right)$$

Exemple:

Ecrire l'algo. de Newton pour

$$\min_{(x,y)\in\mathbb{R}^2} (x-2)^4 + (x-2y)^2$$

$$f(x,y) = (x-2)^4 + (x-2y)^2 \quad (f \text{ pas quadratique})$$

$$f_x = 4(x-2)^3 + 2(x-2y)$$

$$f_y = 4(2y-x)$$

$$H = \begin{pmatrix} 12(x-2)^2+2 & -4 \\ -4 & 8 \end{pmatrix}$$

Newton:

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} - \begin{pmatrix} 12(x_k-2)^2+2 & -4 \\ -4 & 8 \end{pmatrix}^{-1} \begin{pmatrix} 4(x_k-2)^3 + 2(x_k-2y_k) \\ 4(2y_k - x_k) \end{pmatrix}$$

Au point critique $(2,1)$, on a que

$$H(2,1) = \begin{pmatrix} 2 & -4 \\ -4 & 8 \end{pmatrix} \text{ n'est pas inversible}$$

$\Rightarrow$ la convergence de Newton n'est pas assurée !

Donc, on ne peut conclure sur l'efficacité de l'algo. de Newton dans ce cas !

Finalement, pour conclure, montrons que les itérés x_k produits par l'algo. de Newton vérifient la propriété :

$$f(x_{k+1}) \leq f(x_k)$$

Faisons l'hypothèse que

$$H_f(\bar{x}) > 0 \Rightarrow H_f(x_k) > 0$$

pour les x_k près de $\bar{x}$.

$$\Rightarrow \left[H_f(x_k)\right]^{-1} > 0.$$

Utilisons la formule de Taylor pour f autour de x_k

$$f(x_{k+1}) = f(x_k) + \left(\nabla f(x_k), x_{k+1} - x_k\right) + \frac{1}{2}\left(H(x_{k+1} - x_k), x_{k+1} - x_k\right)$$

↳ évalué en un pt. intermédiaire !

Or $x_{k+1} = x_k - M_k \nabla f(x_k)$ où $M_k = \left[H(x_k)\right]^{-1} > 0$

$$\Rightarrow f(x_{k+1}) = f(x_k) - \left(M_k \nabla f(x_k), \nabla f(x_k)\right) + \frac{1}{2}(H -, -)$$

$$\lesssim f(x_k) - \left(M_k \nabla f(x_k), \nabla f(x_k)\right)$$ ↳ terme petit (négligeable)

$$\leq f(x_k) \quad \text{car } M_k > 0$$