ISTIL 2ème année, parcours MAM

Optimisation Continue Année 2007/2008

TP 1 : Algorithmes de minimisation sans contrainte

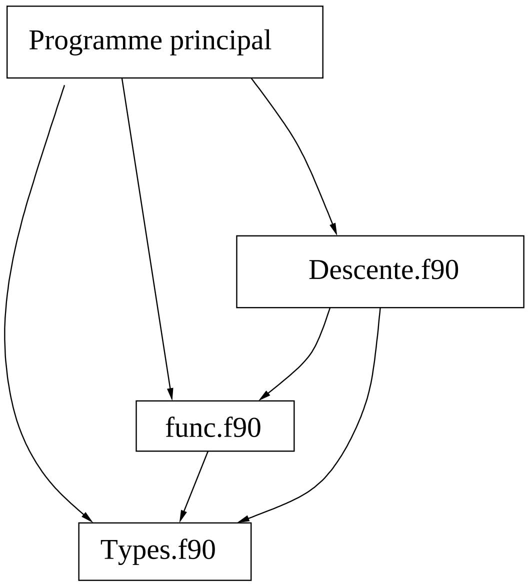
Le langage de programmation est le FORTRAN 90.

1. Gradient à pas constant

Télécharger le squelette de code sur

`http://math.univ-lyon1.fr/~perrier/index.php`

Le schéma des modules de ce programme est le suivant



Le module TYPES.F90 contient un type DONNEES, qui permet de transporter facilement les différents paramètres des méthodes ou des fonctions à travers les différents modules. Le module FUNC.F90 permet de programmer les différentes fonctions que l'on va minimiser. Le module DESCENTE.F90 contient les différentes méthodes de descente que l'on va programmer.

Programmer la méthode de gradient à pas fixe. Tester cette méthode sur les problèmes suivants

1. $f(x,y) = x^2 - 5xy + y^4 - 25x - 8y$
2. $f(x,y) = 5x^2 + 5y^2 - xy - 11x + 11y + 11$
3. $f(x,y) = (x^4 - 3)^2 + y^4$
4. $f(x_1,x_2) = x_1^4 - 4x_1^3 + 6(x_1^2 + x_2^2) - 4(x_1 + x_2)$

2. Recherche linéaire d'un pas optimal

La méthode de gradient à pas fixe oblige à toujours faire le même pas dans la direction de descente. Dans l'idéal, on voudrait faire de grands pas au début, et des pas de plus en plus petits à la fin de l'algorithme. Pour une itération donnée x_k, notons

$$\Phi_k \,:\, \alpha \longmapsto \mathrm{J}(x_k - \alpha \nabla \mathrm{J}(x_k))$$

> **Rappel : règle de Goldstein**
> La *règle de Goldstein* consiste à définir deux règles à partir d'un paramètre $\rho \in \left[0\,;\frac{1}{2}\right]$
>
> $$\begin{array}{ll} (\mathbf{1}) & \Phi(\alpha) \leqslant \Phi(0) + \rho\alpha\Phi'(0) \\ (\mathbf{2}) & \Phi(\alpha) \geqslant \Phi(0) + (1-\rho)\alpha\Phi'(0) \end{array}$$
>
> On considère alors qu'un pas α est correct si (**1**) et (**2**) sont satisfaits. En pratique, on choisit en général $\alpha_g = 0$; on cherche ensuite α_d tel que le critère (**2**) soit violé. Il reste alors à faire une dichotomie dans l'intervalle $[\,\alpha_g\,;\alpha_d\,]$ pour déterminer un α vérifiant les deux conditions (**1**) et (**2**) :
>
> - α est trop grand si $\Phi(\alpha) > \Phi(0) + \rho\alpha\Phi'(0)$
> - α est trop petit si $\Phi(\alpha) < \Phi(0) + (1-\rho)\alpha\Phi'(0)$

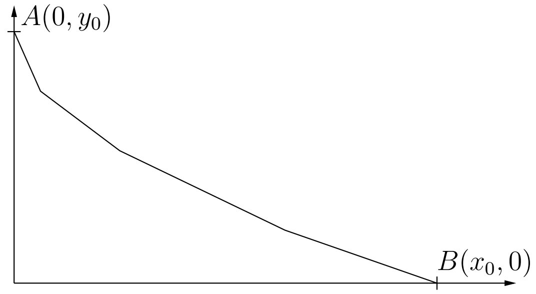
> **Rappel : règle de Wolfe-Powell**
> La *règle de Wolfe-Powell* consiste à se donner deux paramètres $0 < \rho < \sigma < 1$, et à définir la règle suivante
>
> $$(\mathbf{2bis}) \qquad \Phi'(\alpha) \geqslant \sigma\Phi'(0)$$
>
> Un pas α est alors jugé satisfaisant s'il vérifie les règles (**1**) et (**2bis**). Le principe de recherche de α est le même que pour la méthode de Goldstein.

Programmer ces deux méthodes. Comparer l'efficacité de ces deux méthodes, avec celle du gradient à pas variable, en terme de temps CPU et de nombre d'itérations. Étudier la sensibilité de ces performances en fonction du (des) paramètre(s).

3. Profil optimal d'un « Schuss »

Toto est élève de l'ISTIL et va au week-end ski à l'Alpe d'Huez. Il dispose d'une pelle et d'une quantité infinie de neige. Aider Toto à trouver le profil entre le point A, de coordonnées $(y_\mathrm{A}, 0)$ et le point B, de coordonnées $(0, x_\mathrm{B})$ permettant de minimiser le temps de parcours.

Mise en équation

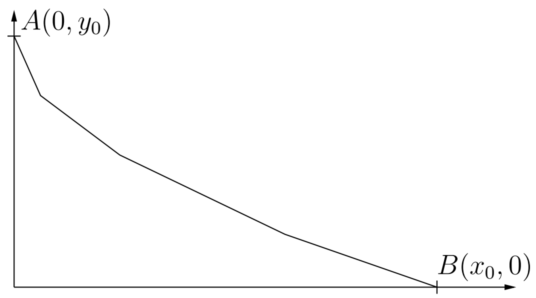


Toto se modélise comme une masse ponctuelle, soumise uniquement à son poids et à la réaction du sol, qui est sans frottement. Notons v sa vitesse en un point donné, et y son ordonnée. Alors le théorème de l'énergie cinétique donne

$$\frac{1}{2}mv^2 + mgy = mgy_0$$

d'où
$$v = \sqrt{2g(y_\mathrm{A} - y)}$$

Notons x_k une suite de point équirépartis dans $[\,0\,;x_\mathrm{B}\,]$, avec $x_0 = 0$ et $x_{\mathrm{N}+1} = x_\mathrm{B}$. Notons h le pas de discrétisation ; on a donc pour tout k, $h = x_{k+1} - x_k$. Le parcours de Toto est supposé affine entre les points $y(x_k)$ et $y(x_{k+1})$.



Sur le segment $[\,x_k\,;x_{k+1}\,]$, on a

$$\sin\theta = \frac{1}{\sqrt{1 + \left(\dfrac{y_{k+1} - y_k}{h}\right)^2}}$$

Lorsqu'on se déplace d'un accroissement $\mathrm{d}x$, la distance $\mathrm{d}\ell$ parcourue est égale à

$\mathrm{d}x/\sin\theta$. On a alors

$$\begin{aligned}
t(k+1)-t(k) &= \int_{x_k}^{x_{k+1}} \frac{\mathrm{d}\ell}{v} \\
&= \int_{x_k}^{x_{k+1}} \frac{\mathrm{d}x}{v\sin\theta} \\
&= \frac{1}{\sin\theta}\int_{x_k}^{x_{k+1}} \frac{\mathrm{d}x}{v} \\
&= \frac{1}{\sin\theta}\int_{x_k}^{x_{k+1}} \frac{\mathrm{d}x}{\sqrt{2g\left(y_0 - \frac{y_{k+1}-y_k}{h}(x-x_k) - y_k\right)}} \\
&= -\frac{1}{\sin\theta}\,\frac{h}{g(y_{k+1}-y_k)}\left(\sqrt{2g(y_0-y_{k+1})} - \sqrt{2g(y_0-y_k)}\right)
\end{aligned}$$

Après simplifications (en particulier, utilisation de la quantité conjuguée pour faire disparaitre la différence des racines), on obtient

$$\boxed{t(k+1)-t(k) = \frac{h}{\sqrt{2g}}\,\frac{2\sqrt{1+\left(\frac{y_{k+1}-y_k}{h}\right)^2}}{\sqrt{y_0-y_{k+1}}+\sqrt{y_0-y_k}}}$$

Comparer les méthodes de gradient sur cet exemple.

Raffinement de maillage

Comparer en terme de temps CPU les méthodes suivantes :

1. Recherche d'un minimum pour un maillage contenant 2^n points, avec un point de départ quelconque.
2. Recherche d'un minimum pour un maillage contenant 2 points, puis 4 points, puis 8 points, puis, … 2^n points, où la minimisation pour le maillage contenant 2^n est obtenue en commençant avec la solution obtenue pour 2^{n-1} points.