

IFT 3515  
Fonctions à plusieurs variables  
Optimisation sans contraintes  
Gradient conjugué

Fabian Bastin  
DIRO  
Université de Montréal

# Méthode du gradient conjugué

Cette méthode peut être utilisée quand les exigences mémoire des méthodes quasi-Newton dépassent la mémoire machine disponible, ou alternativement pour résoudre des systèmes d'équations linéaires définis positifs.

De plus, elle est particulièrement adaptée pour résoudre le sous-problème quadratique en région de confiance.

Note : ces transparents se sont fortement inspirés de <http://www.numerical.rl.ac.uk/people/nimg/course/lectures/raphael/lectures/lec5slides.pdf>

## Motivations : exigences mémoire

- L'utilisation de la mémoire dans un ordinateur a longtemps été un sujet de préoccupation en raison du coût de celle-ci et de sa capacité limitée. Aujourd'hui, des défis demeurent comme les modèles peuvent être de grandes tailles, avec plusieurs millions de variables.
- Supposons que nous avons  $n$  variables. Les méthodes quasi-Newton doivent maintenir en mémoire une matrice de dimensions  $n \times n$ , représentant une approximation de la matrice hessienne ou de son inverse, ou le facteur de Cholesky de cette matrice. Dans tous les cas, le coût de stockage est en  $O(n^2)$ .
- La méthode de plus forte pente a seulement des besoins mémoire en  $O(n)$ , pour stocker  $x_k$  et  $\nabla f(x_k)$ , et peut donc traiter des problèmes de plus grandes tailles, mais a tendance à exiger beaucoup plus d'itérations pour converger.

# Propriétés du gradient conjugué

La méthode du gradient conjugué a

- une consommation mémoire en  $O(n)$ ,
- une complexité de calcul en  $O(n)$  par itération,
- mais converge plus rapidement que la méthode de plus forte pente.

# Minimisation de fonctions quadratiques

Soit  $B \succ 0$  une matrice définie positive et considérons le problème

$$\min_{x \in \mathcal{R}^n} f(x) = \frac{1}{2}x^T Bx + b^T x + a.$$

Puisque  $f$  est convexe,  $\nabla f(x) = 0$  est une condition d'optimalité suffisante. Dès lors, ce problème d'optimisation est équivalent à résoudre le système linéaire défini positif

$$Bx = -b$$

qui a pour solution

$$x^* = -B^{-1}b.$$

## Motivation géométrique

Ajouter une constante à la fonction objectif du problème

$$\min_{x \in \mathcal{R}^n} f(x) = \frac{1}{2}x^T Bx + b^T x + a.$$

ne change pas le minimiseur global  $x^* = -B^{-1}b$ . Dès lors, le problème est équivalent à

$$\min_{x \in \mathcal{R}^n} f(x) = \frac{1}{2}(x - x^*)^T B(x - x^*).$$

En effet,

$$\begin{aligned} \frac{1}{2}(x - x^*)^T B(x - x^*) &= \frac{1}{2}x^T Bx + \frac{1}{2}(x^*)^T Bx^* - 2\frac{1}{2}(x^*)^T Bx \\ &= \frac{1}{2}x^T Bx + \frac{1}{2}(x^*)^T Bx^* + b^T (B^{-1})^T Bx \\ &= \frac{1}{2}x^T Bx + \frac{1}{2}(x^*)^T Bx^* + b^T x \end{aligned}$$

## Motivation géométrique

Nous pouvons encore réécrire le problème comme

$$\min_{x \in \mathcal{R}^n} f(x) = \frac{1}{2}(x - x^*)^T B(x - x^*) = \frac{1}{2}y^T y = g(y),$$

où  $y = B^{1/2}(x - x^*)$ . Dès lors le problème d'optimisation apparaît particulièrement simple en termes de  $y$ .

Note : soit une matrice  $A \in \mathcal{R}^{n \times n}$  symétrique.

- $A$  a  $n$  valeurs propres réelles  $\lambda_1 \leq \dots \leq \lambda_n$  et il existe une matrice orthogonale  $Q$  (i.e.  $QQ^T = Q^T Q = I$ ) telle que  $A = Q \text{diag}(\lambda) Q^T$
- $A^{-1} = Q D^{-1} Q^T$ , i.e.,  $A$  est non-singulière ssi  $\forall i, \lambda_i \neq 0$
- $A$  est définie positive ssi  $\forall i, \lambda_i > 0$ , et alors  $A^{1/2} := Q \text{diag}(\lambda^{1/2}) Q^T$  est l'unique matrice symétrique définie positive telle que  $A^{1/2} A^{1/2} = A$ .

# Principes généraux

Nous souhaitons construire une séquence itérative  $\{x_k\}_{k \in \mathcal{N}}$  telle que la séquence correspondante  $\{y_k\}_{k \in \mathcal{N}} = \{B^{1/2}(x_k - x^*)\}_{k \in \mathcal{N}}$  se comporte de manière appropriée.

Considérons l'itéré courant  $x_k$  appliquons une recherche linéaire exacte

$$\begin{aligned}\alpha_k &= \arg \min_{\alpha} f(x_k + \alpha_k d_k) \\ &= \arg \min_{\alpha} \frac{1}{2} (x_k + \alpha_k d_k - x^*)^T B (x_k + \alpha_k d_k - x^*)\end{aligned}$$

à partir de  $x_k$  dans la direction  $d_k$ .



# Principes généraux

Comme  $B$  est définie positive, il s'agit d'un problème univarié strictement convexe, et la solution peut se calculer en annulant la dérivée par rapport à  $\alpha$  :

$$\frac{d}{d\alpha} f(x_k + \alpha_k d_k) = 0$$

# Principes généraux

De manière équivalente

$$\alpha_k d_k^T B d_k + d_k^T B x_k + d_k^T b = 0$$

ou encore

$$\alpha_k d_k^T B d_k + d_k^T \nabla f(x_k) = 0$$

et donc

$$\alpha_k = -\frac{d_k^T \nabla f(x_k)}{d_k^T B d_k}$$

# Principes généraux

En termes de  $y_k$ , cela donne

$$\begin{aligned}\alpha_k &= \arg \min_{\alpha} g(y_k + \alpha p_k) \\ &= \arg \min_{\alpha} \frac{1}{2} (y_k + \alpha p_k)^T (y_k + \alpha p_k) \\ &= \arg \min_{\alpha} \frac{1}{2} \|y_k\|^2 + \alpha p_k^T y_k + \frac{1}{2} \alpha^2 \|p_k\|^2\end{aligned}$$

où en choisissant  $p_k = B^{1/2} d_k$ , et  $\alpha_k = -\frac{p_k^T y_k}{\|p_k\|^2}$ , on retrouve le problème précédent.

## Directions conjuguées

Si nous posons

$$y_{k+1} = y_k + \alpha_k p_k,$$

alors

$$y_{k+1}^T p_k = y_k^T p_k + \alpha_k p_k^T p_k = y_k^T p_k - \frac{p_k^T y_k}{\|p_k\|^2} \|p_k\|^2 = 0.$$

Autrement dit,  $y_{k+1}$  est orthogonal à  $p_k$ , et ceci indépendamment de la localisation de  $x_k$ .

Comme  $y_{k+1} = B^{1/2}(x_{k+1} - x^*)$ , nous avons aussi

$$(x_{k+1} - x^*)^T B^{1/2} p_k = 0$$

et donc

$$x_{k+1} \in \pi_k := x^* + B^{-1/2} p_k^\perp$$

## Directions conjuguées

Plus généralement, la résolution du problème

$$\alpha^* = \arg \min_{\alpha} f(x + \alpha d)$$

dans la direction  $d = \pm d_k$  conduit à un point  $x^+ = x + \alpha^* d$  dans l'hyperplan  $\pi_k$ , aussi nous ne devrions plus sortir de  $\pi_k$  à partir de l'itération  $k + 1$ .

L'exigence que toutes les itérations ultérieures à l'itération  $k$  doivent être conduites à l'intérieur de  $\pi_k$  mènent à la condition

$$p_j \perp p_k$$

pour tout  $j > k$ , ou, de manière équivalente, pour tout  $j \geq k + 1$ ,

$$d_k^T B d_j = 0.$$

Si cette relation tient, nous disons que  $d_k$  et  $d_j$  sont  $B$ -conjuguées, qui revient à l'orthogonalité par rapport au produit scalaire euclidien défini par  $B$ .

## Directions conjuguées

En répétant l'argumentation à partir de l'itération  $k + 2$ , on peut affirmer que  $x_j$  appartiendra à l'hyperplan  $\pi_{k+1}$  pour  $j \geq k + 2$ .

Il suit que la dimension de l'espace de recherche  $\pi_k$  diminue de 1 à chaque itération, aussi l'algorithme se termine après  $n$  itérations.

Dès lors, nous devons choisir des directions de recherche mutuellement  $B$ -conjuguées :

$$d_i^T B d_j = 0, \forall i \neq j.$$

# Convergence

## Théorème

Soit

$$f(x) := \frac{1}{2}x^T Bx + b^T x + a,$$

où  $B \succ 0$ . Pour  $k = 0, \dots, n-1$ , soit  $d_k$  choisi tel que

$$d_i^T B d_j = 0, \quad \forall i \neq j.$$

Soient  $x_0 \in \mathcal{R}^n$  et

$$x_{k+1} = x_k + \alpha_k d_k$$

pour  $k = 0, \dots, n-1$ , où

$$\alpha_k = \arg \min_{\alpha} f(x_k + \alpha d_k)$$

Alors  $x_n$  est le minimiseur global de  $f$ .

## Calcul de $\alpha_k$

Comme le problème est quadratique, le calcul de  $\alpha_k$  est aisé. Par définition

$$\alpha_k = \arg \min_{\alpha} f(x_k + \alpha d_k)$$

Nous pouvons annuler la dérivée par rapport à  $\alpha$  :

$$\frac{d}{d\alpha} f(x_k + \alpha d_k) = d_k^T \nabla f(x_k + \alpha d_k)$$

Or

$$\nabla f(x_k + \alpha d_k) = B(x_k + \alpha d_k) + b$$

et donc nous obtenons un système linéaire en  $\alpha$  :

$$d_k^T (B(x_k + \alpha d_k) + b) = 0$$



## Calcul de $\alpha_k$

Le système peut se réécrire comme

$$\begin{aligned}\alpha d_k^T B d_k &= -d_k^T (B x_k + b) \\ &= -d_k^T \nabla f(x_k)\end{aligned}$$

et donc

$$\begin{aligned}\alpha &= -\frac{d_k^T (B x_k + b)}{2 d_k^T B d_k} \\ &= -\frac{d_k^T \nabla f(x_k)}{d_k^T B d_k}\end{aligned}$$

# Directions

Comment choisir des directions de recherche  $B$ -conjuguées ?

## Lemme (Orthogonalisation de Gram-Schmidt)

Soient  $v_0, \dots, v_{n-1} \in \mathcal{R}^n$  des vecteurs linéairement indépendants, et soient  $d_0, \dots, d_{n-1}$  définis récursivement comme

$$d_k = v_k - \sum_{j=0}^{k-1} \frac{d_j^T B v_k}{d_j^T B d_j} d_j.$$

Alors  $d_i^T B d_k = 0$  pour tout  $i \neq k$ .

# Directions

## Démonstration.

Induction sur  $k$ .

- Pour  $k = 0$ , il n'y a rien à prouver.
- Supposons que

$$d_i^T B d_j = 0$$

pour tout  $i, j \in \{0, \dots, k-1\}$ ,  $i \neq j$ .

- Pour  $i < k$ ,

$$d_i^T B d_k = d_i^T B v_k - d_i^T B \sum_{j=0}^{k-1} \frac{d_j^T B v_k}{d_j^T B d_j} d_j = d_i^T B v_k - d_i^T B v_k = 0$$

- L'indépendance linéaire des  $v_j$  garantit qu'autant des  $d_j$  n'est nul, et dès lors  $d_j^T B d_j > 0$  pour tout  $j$ .



## Plus forte pente

Malheureusement, cette procédure exigerait de stocker les vecteurs  $d_j$  ( $j < k$ ) en mémoire, aussi la méthode consommerait un espace mémoire en  $O(n^2)$ .

Nous pouvons conserver une exigence de stockage en  $O(n)$  si nous choisissons pour  $v_k$  la direction de plus forte pente.

### Lemme (Orthogonalité)

Choisissons  $d_0 = -\nabla f(x_0)$  et pour  $k = 1, \dots, n-1$ , calculons  $d_k$  comme

$$d_k = -\nabla f(x_k) - \sum_{j=0}^{k-1} \frac{d_j^T B(-\nabla f(x_k))}{d_j^T B d_j} d_j.$$

Alors  $\nabla f(x_j)^T \nabla f(x_k) = 0$  et  $d_j^T \nabla f(x_k) = 0$  pour  $j < k$ .

## Plus forte pente

### Démonstration.

Notons que  $\nabla f(x_k) = Bx_k + b$  pour tout  $k$ .

Par induction sur  $k$ , prouvons que  $d_j^T \nabla f(x_k) = 0$  pour  $j < k$ .

- C'est évident pour  $k = 0$ . Supposons que le résultat tient pour  $k$ . Alors, pour  $j = 0, \dots, k - 1$ ,

$$\begin{aligned} d_j^T \nabla f(x_{k+1}) &= d_j^T (Bx_{k+1} + b) \\ &= d_j^T (B(x_k + \alpha_k d_k) + b) \\ &= d_j^T \nabla f(x_k) + \alpha_k d_j^T B d_k \\ &= 0. \end{aligned}$$

- De plus,  $d_k^T \nabla f(x_{k+1}) = 0$  est la condition d'optimalité de premier ordre pour la recherche linéaire  $\min_{\alpha} f(x_k + \alpha d_k)$ , définissant  $x_{k+1}$ .

## Plus forte pente

### Démonstration.

La définition de  $d_k$  implique que, pour tout  $k$ ,

$$\text{span}(d_0, \dots, d_k) = \text{span}(\nabla f(x_0), \dots, \nabla f(x_k)).$$

Pour  $j < k$ , il existe dès lors certains  $\lambda_1, \dots, \lambda_j$  tels que

$$\nabla f(x_j) = \sum_{i=0}^j \lambda_i d_i,$$

et nous avons

$$\nabla f(x_j)^T \nabla f(x_k) = \sum_{i=0}^j \lambda_i d_i^T \nabla f(x_k) = 0.$$

## Mise à jour des directions

Rappelons que

$$d_k = -\nabla f(x_k) - \sum_{j=0}^{k-1} \frac{d_j^T B(-\nabla f(x_k))}{d_j^T B d_j} d_j.$$

Comme  $\nabla f(x_k) = Bx_k + b$ ,

$$\nabla f(x_{j+1}) - \nabla f(x_j) = B(x_{j+1} - x_j) = B(x_j + \alpha_j d_j - x_j) = \alpha_j B d_j.$$

Dès lors

$$\begin{aligned} d_k &= -\nabla f(x_k) - \sum_{j=0}^{k-1} \frac{(\nabla f(x_{j+1}) - \nabla f(x_j))^T (-\nabla f(x_k))}{(\nabla f(x_{j+1}) - \nabla f(x_j))^T d_j} d_j \\ &= -\nabla f(x_k) + \sum_{j=0}^{k-1} \frac{\nabla f(x_{j+1})^T \nabla f(x_k) - \nabla f(x_j)^T \nabla f(x_k)}{\nabla f(x_{j+1})^T d_j - \nabla f(x_j)^T d_j} d_j \end{aligned}$$

## Mise à jour des directions

Du dernier lemme,  $\nabla f(x_j)^T \nabla f(x_k) = 0$  et  $\nabla f(x_k)^T d_j = 0$  si  $j < k$ , et donc

$$\begin{aligned} d_k &= -\nabla f(x_k) + \frac{\nabla f(x_k)^T \nabla f(x_k)}{-\nabla f(x_{k-1})^T d_{k-1}} d_{k-1} \\ &= -\nabla f(x_k) - \frac{\|\nabla f(x_k)\|^2}{\nabla f(x_{k-1})^T d_{k-1}} d_{k-1} \end{aligned}$$

Dès lors

$$d_{k-1}^T \nabla f(x_{k-1}) = -\|\nabla f(x_{k-1})\|^2$$

et

$$d_k = -\nabla f(x_k) + \frac{\|\nabla f(x_k)\|^2}{\|\nabla f(x_{k-1})\|^2} d_{k-1}$$

C'est la règle du **gradient conjugué** pour la mise à jour de la direction de recherche.



## Considérations pratiques

- Dans le calcul de  $d_k$ , nous devons seulement garder deux vecteurs et un scalaire stockés dans la mémoire centrale :  $d_{k-1}$ ,  $x_k$  et  $\|\nabla f(x_{k-1})\|^2$ .
- Les registres occupés par ces données sont réécrits durant le calcul du nouvel itéré par  $d_k$ ,  $x_{k+1}$  et  $\|\nabla f(x_k)\|^2$ .
- La méthode se termine en au plus  $n$  itérations.
- De plus, en général,  $x_k$  donne une bonne approximation de  $x^*$  après quelques itérations, et les itérations restantes sont utilisées pour affiner le résultat.

## Calcul de $\alpha_k$

Nous avons obtenu

$$\alpha_k = -\frac{d_k^T \nabla f(x_k)}{d_k^T B d_k}$$

Or, pour  $k > 1$ ,

$$d_k = -\nabla f(x_k) + \frac{\|\nabla f(x_k)\|^2}{\|\nabla f(x_{k-1})\|^2} d_{k-1}$$

et nous pouvons réécrire le calcul de  $\alpha_k$  comme

$$\alpha_k = -\frac{(-\nabla f(x_k) + \frac{\|\nabla f(x_k)\|^2}{\|\nabla f(x_{k-1})\|^2} d_{k-1})^T \nabla f(x_k)}{d_k^T B d_k}$$

En utilisant la propriété  $d_{k-1}^T \nabla f(x_k) = 0$ , l'égalité devient

$$\alpha_k = \frac{\|\nabla f(x_k)\|_2^2}{d_k^T B d_k}$$

## Algorithme : gradients conjugués

Étape 0. Soient  $x_0 \in \mathcal{R}^n$ ,  $d_0 := -\nabla f(x_0)$ ,  $k := 0$ .

Étape 1. Calculer

$$\alpha_k = \arg \min_{\alpha} f(x_k + \alpha d_k)$$

et poser

$$x_{k+1} = x_k + \alpha_k d_k.$$

Étape 2. Si  $k < n - 1$ , calculer

$$d_{k+1} = -\nabla f(x_{k+1}) + \frac{\|\nabla f(x_{k+1})\|^2}{\|\nabla f(x_k)\|^2} d_k$$

Poser  $k := k + 1$ . Si  $k = n$ , arrêter ;  $x^* = x_n$ . Sinon, retourner à l'étape 1.

## Remarque d'implémentation

On pourra remplacer

$$\frac{\|\nabla f(x_{k+1})\|^2}{\|\nabla f(x_k)\|^2}$$

par

$$\frac{\nabla f(x_{k+1})^T B d_k}{d_k^T B d_k}$$

En effet, comme  $\nabla f(x_{k+1})^T \nabla f(x_k) = 0$ ,  $d_k^T \nabla f(x_{k+1}) = 0$  et

$$B d_k = \frac{1}{\alpha_k} (\nabla f(x_{k+1}) - \nabla f(x_k)),$$

nous avons

$$\begin{aligned} \frac{\nabla f(x_{k+1})^T B d_k}{d_k^T B d_k} &= \frac{\nabla f(x_{k+1})^T (\nabla f(x_{k+1}) - \nabla f(x_k))}{d_k^T (\nabla f(x_{k+1}) - \nabla f(x_k))} \\ &= - \frac{\nabla f(x_{k+1})^T \nabla f(x_{k+1})}{d_k^T \nabla f(x_k)} \end{aligned}$$

# Vitesse de convergence

Conn, Gould, Toint, théorème 5.1.7

## Théorème

*L'erreur résiduelle  $\epsilon_k = x_k - x^*$  des itérés générés par l'algorithme du gradient conjugué satisfait l'inégalité*

$$\frac{\|\epsilon_k\|_B}{\|\epsilon_0\|_B} \leq 2 \left( \frac{\sqrt{\kappa(B)} - 1}{\sqrt{\kappa(B)} + 1} \right)^k$$

*où  $\kappa(B)$  est le conditionnement de  $B$ .*

# Conditionnement

Le conditionnement d'une matrice normale  $B$  est le rapport entre les valeurs absolues de la plus grande et de la plus petite valeur propre :

$$\kappa(B) = \frac{|\lambda_{\max}(B)|}{|\lambda_{\min}(B)|}$$

Note : En algèbre linéaire, une matrice carrée  $A$  à coefficients complexes est une matrice normale si elle commute avec sa matrice adjointe  $A^*$ , c'est-à-dire si  $AA^* = A^*A$ . Si tous les coefficients sont réels,  $A^* = A^T$ , et toute matrice symétrique est normale.

## Algorithme du gradient conjugué préconditionné

Idée : appliquer le gradient conjugué après un changement linéaire de coordonnées au moyen d'une matrice  $R$  inversible :  $x = R^{-1}y$ .

Utiliser le gradient conjugué pour résoudre

$$R^{-T}BR^{-1}y = R^{-T}c$$

avec  $c = -\frac{1}{2}b$ .

Soit  $y^*$  la solution de ce nouveau problème d'optimisation. On obtient la solution du problème de départ en prenant

$$x^* = R^{-1}y^*.$$

Le but recherché est que la matrice de transformation inversible  $R$  soit choisie de sorte à regrouper les valeurs problèmes de la matrice hessienne transformée  $\bar{B} := R^{-T}BR^{-1}$

## Algorithme du gradient conjugué préconditionné

Cependant, nous ne voulons pas former explicitement  $\bar{B} = R^{-T} B R^{-1}$  et  $\bar{c} = R^{-T} c$  comme  $\bar{B}$  pourrait être dense alors que  $B$  serait creuse. Nous souhaitons plutôt une méthode implicite.

Cas extrême :  $R = B$ .

On cherche un compromis entre une convergence accélérée, au coût additionnel de la multiplication par  $R^{-1}$  à chaque itération.

Le but est de trouver  $R$  facile à inverser et qui approxime bien  $B$ , en regroupant les valeurs de propres de  $R^{-1}B$ .



## Algorithme GC pour le problème transformé

Étant donné  $y_0$ , poser  $\bar{g}_0 = \bar{B}y_0 + \bar{c}$  et  $\bar{d}_0 = -\bar{g}_0$ . Pour  $k = 0, 1, 2, \dots$ , jusqu'à convergence, faire

$$\alpha_k = \frac{\|\bar{g}_k\|_2^2}{\bar{d}_k^T \bar{B} \bar{d}_k}$$

$$y_{k+1} = y_k + \alpha_k \bar{d}_k$$

$$\bar{g}_{k+1} = \bar{g}_k + \alpha_k \bar{B} \bar{d}_k$$

$$\beta_k = \frac{\|\bar{g}_{k+1}\|_2^2}{\|\bar{g}_k\|_2^2}$$

$$\bar{d}_{k+1} = -\bar{g}_{k+1} + \beta_k \bar{d}_k$$

On va tâcher de revenir à un algorithme en termes de  $x$ .

## Algorithme GC préconditionné

Posons  $y_k = Rx_k$ ,  $\bar{d}_k = Rd_k$ , alors

$$y_{k+1} = y_k + \alpha_k \bar{d}_k$$

devient

$$Rx_{k+1} = Rx_k + \alpha_k Rd_k$$

et en prémultipliant par  $R^{-1}$

$$x_{k+1} = x_k + \alpha_k d_k$$

Similairement, prenons  $\bar{g}_k = R^{-T} g_k$ . Alors

$$\bar{g}_{k+1} = \bar{g}_k + \alpha_k \bar{B} \bar{d}_k$$

se réécrit

$$g_{k+1} = g_k + \alpha_k B d_k$$

## Algorithme GC préconditionné

De même

$$\|\bar{g}_k\|_2^2 = \|R^{-T}g_k\|_2^2 = \langle R^{-T}g_k, R^{-T}g_k \rangle = \langle g_k, R^{-1}R^{-T}g_k \rangle$$

Finalement,

$$\bar{d}_{k+1} = -\bar{g}_{k+1} + \beta_k \bar{d}_k$$

devient

$$Rd_{k+1} = -R^{-T}g_{k+1} + \beta_k Rd_k$$

ou

$$d_{k+1} = -R^{-1}R^{-T}g_{k+1} + \beta_k d_k$$

## Algorithme GC préconditionné

Les constantes deviennent quant à elles

$$\alpha_k = \frac{\langle g_k, R^{-1}R^{-T}g_k \rangle}{\langle Rd_k, R^{-T}BR^{-1}Rd_k \rangle} = \frac{\langle g_k, R^{-1}R^{-T}g_k \rangle}{\langle d_k, Bd_k \rangle}$$

et

$$\beta_k = \frac{\langle g_{k+1}, R^{-1}R^{-T}g_{k+1} \rangle}{\langle g_k, R^{-1}R^{-T}g_k \rangle}$$

En définissant

$$M = R^T R$$

et

$$v_k = M^{-1}g_k$$

nous obtenons l'algorithme suivant

## Algorithme GC préconditionné

Étant donné  $x_0$ , posons  $g_0 = Bx_0 + c$ . Soit  $v_0 = M^{-1}g_0$  et  $d_0 = -v_0$ . Pour  $k = 0, 1, 2, \dots$ , jusqu'à convergence, faire

$$\alpha_k = \frac{\langle g_k, v_k \rangle}{d_k^T B d_k}$$

$$x_{k+1} = x_k + \alpha_k d_k$$

$$g_{k+1} = g_k + \alpha_k B d_k$$

$$v_{k+1} = M^{-1}g_{k+1}$$

$$\beta_k = \frac{\langle g_{k+1}, v_{k+1} \rangle}{\langle g_k, v_k \rangle}$$

$$d_{k+1} = -v_{k+1} + \beta_k d_k$$

$M$  est appelé préconditionneur.

## Vitesse de convergence

Notons tout d'abord que les valeurs propres de  $M^{-1}B$  sont les mêmes que celles de  $R^{-T}BR^{-1}$ . En effet, si  $\lambda$  est valeur propre de  $M^{-1}B$ , il existe un  $u$  tel que

$$M^{-1}Bu = \lambda u$$

et nous pouvons écrire

$$R^{-1}R^{-T}Bu = \lambda u$$

ou

$$R^{-T}Bu = \lambda Ru$$

Soit  $\nu := Ru$ . Nous obtenons

$$R^{-T}BR^{-1}\nu = \lambda\nu$$

# Vitesse de convergence

## Théorème

*L'erreur résiduelle  $\epsilon_k = x_k - x^*$  des itérés générés par l'algorithme du gradient conjugué préconditionné satisfait l'inégalité*

$$\frac{\|\epsilon_k\|_B}{\|\epsilon_0\|_B} \leq 2 \left( \frac{\sqrt{\kappa(M^{-1}B)} - 1}{\sqrt{\kappa(M^{-1}B)} + 1} \right)^k$$

*où  $\kappa(M^{-1}B)$  est le conditionnement de  $M^{-1}B$ .*

## Corollaire

*Si  $M^{-1}B$  a  $\ell$  valeurs propres distinctes, l'algorithme du gradient conjugué préconditionné se terminera avec  $x_j = x^*$  pour un certain  $j \leq \ell$ .*

## Produit matrice hessienne et vecteur

Il est à noter dans l'algorithme qu'il n'est jamais nécessaire de connaître  $B$  explicitement, seul son "effet" est important, i.e. étant donné un vecteur  $y$ , nous devons pouvoir calculer  $Bv$ . Remarquons que

$$\begin{aligned}\nabla_x(\nabla_x f(x)v) &= \nabla_x \left( \sum_{i=1}^n \frac{df(x)}{dx_i} v_i \right) \\ &= \sum_{i=1}^n \nabla_x \frac{df(x)}{dx_i} v_i \\ &= \sum_{i=1}^n \begin{pmatrix} \frac{d^2 f(x)}{dx_i dx_1} \\ \frac{d^2 f(x)}{dx_i dx_2} \\ \vdots \\ \frac{d^2 f(x)}{dx_i dx_n} \end{pmatrix} v_i \\ &= Hv.\end{aligned}$$



## Produit matrice hessienne et vecteur

Autrement dit, il n'est pas nécessaire de stocker  $B$ , mais alors il faut être capable d'évaluer le gradient du produit scalaire entre  $\nabla_x f(x)$  et  $v$ , et ce pour tout nouvel itéré  $x$ .

Dans certains cas, on pourra formuler analytiquement ce produit, sinon on utilisera les approches de différentiation automatique.

## Méthode de Fletcher-Reeves

Cet algorithme peut être adapté pour la minimisation d'une fonction arbitraire  $f \in C^1$  est alors appelé méthode de Fletcher-Reeves. Les différences principales sont les suivantes :

- les recherches linéaires exactes doivent être remplacées par des recherches linéaires pratiques ;
- un critère d'arrêt  $\|\nabla f(x_k)\| < \epsilon$  doit être utilisé pour garantir que l'algorithme se termine en temps fini ;
- puisque le second lemme est seulement établi pour des fonctions quadratiques convexes, le caractère conjugué de  $d_k$  ne peut être que réalisé approximativement et il convient de réinitialiser  $d_k$  à  $-\nabla f(x_k)$  périodiquement.

## Algorithme de Fletcher-Reeves

Étape 0. Choisir  $x_0$  et poser  $d_0 = -\nabla f(x_0)$  Poser  $k = 0$ .

Étape 1. Poser

$$x_{k+1} = x_k + \alpha_k d_k$$

où

$$\alpha_k = \arg \min_{\alpha} f(x_k + \alpha d_k)$$

Étape 2. Arrêt si  $\|\nabla f(x_{k+1})\| < \epsilon$ .

Étape 3. Calculer

$$d_{k+1} = -\nabla f(x_{k+1}) + \beta_{k+1} d_k$$

où

$$\beta_{k+1} = \frac{\|\nabla f(x_{k+1})\|^2}{\|\nabla f(x_k)\|^2}$$

Incrémenter  $k$  de 1. Retour à l'étape 1.