Sommaire

2 Optimisation sans contraintes

Problème non linéaire sans contraintes

$\min_{x \in R^n} f(x)$ → problème noté (PO)

Méthodes globales

- Capacité à localiser plusieurs minima locaux (éventuellement le minimum global)
- Algorithmes non déterministes (déplacements aléatoires « organisés »)
- Métaheuristiques : algorithmes génétiques, recuit simulé, essaims, colonies de fourmis, recherche tabou,…
- Convergence généralement lente, peu précise

Méthodes locales

- Recherche d'un **minimum local** à partir d'un **point initial** fourni par l'utilisateur
- Méthodes d'ordre 0 : sans dérivées → Nelder-Mead, Direct
  d'ordre 1 : avec dérivées premières → plus forte pente
  d'ordre 2 : avec dérivées premières et secondes → Newton
- Critères d'efficacité : **rapidité** de convergence (nombre d'appels de la fonction)
  précision de convergence
  robustesse à l'initialisation

Sommaire

2.1.1 Méthodes de descente

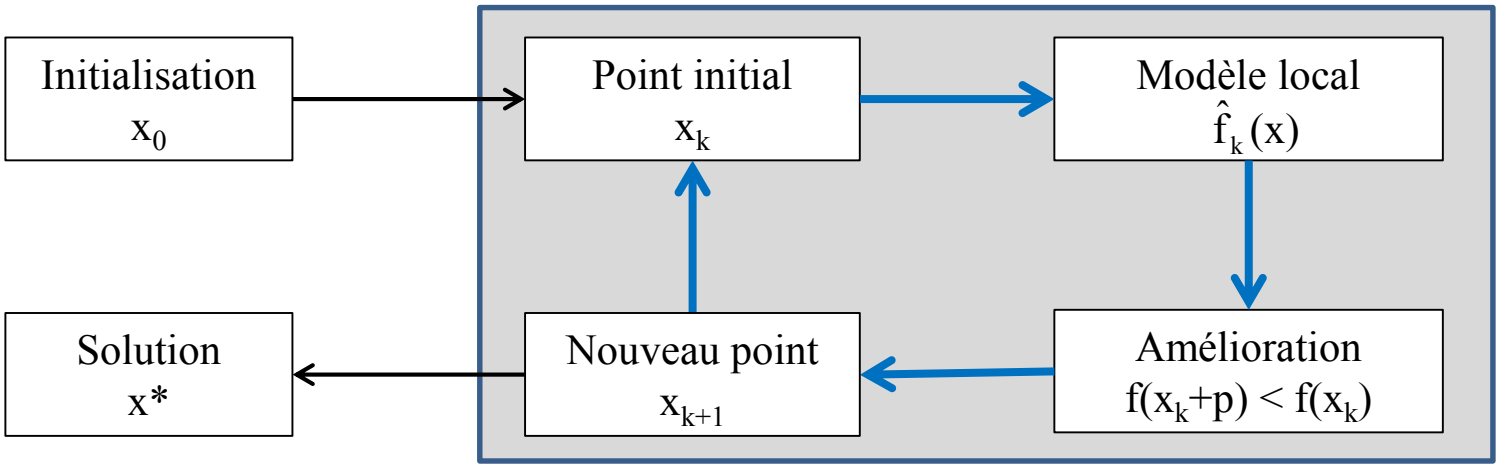
Problème sans contrainte

$$\min_{x \in R^n} f(x) \qquad x^* \text{ minimum local} \quad \Rightarrow \quad \begin{cases} \nabla f(x^*) = 0 \\ \nabla^2 f(x^*) \geq 0 \end{cases}$$

On ne sait pas trouver le minimum global dans le cas général (f quelconque).

Méthode locale

- Initialisation x_0 → recherche d'un minimum local au voisinage de x_0
- Itérations → passage du point x_k au point x_{k+1} meilleur
- Arrêt → solution x* ou blocage



2.1.2 Itérations

Modèle local : prédiction

- Point courant x_k , $f_k = f(x_k)$
- Evaluation de $g_k = \nabla f(x_k)$ ou approximation (différences finies)
  $H_k = \nabla^2 f(x_k)$ ou approximation (quasi Newton)

- Modèle quadratique : $\min_p \hat{f}_k(x_k + p) = f_k + p^t g_k + \frac{1}{2} p^t H_k p \quad \rightarrow \quad \hat{x}_{k+1} = x_k + \hat{p}$ (prédiction)

  $\rightarrow$ **Méthodes de Newton ou quasi-Newton**

Amélioration : correction

- Nouveau point $x_{k+1} = x_k + p$ tel que $f(x_k+p) < f(x_k)$

- Déplacement p à partir de x_k
  par recherche linéaire suivant $d_k = \hat{x}_{k+1} - x_k$
  par région de confiance dans $\|x - x_k\| < r$

  $\rightarrow$ **Méthodes de globalisation**

- La méthode de Newton appliquée directement ne converge pas systématiquement.
  La globalisation est nécessaire pour contrôler la convergence.

2.1.3 Initialisation et arrêt

Initialisation

- Les **méthodes locales** recherchent le minimum au voisinage du point de départ.

- Objectifs : rapidité de convergence
  précision de convergence
  → **méthodes à base de dérivées**

- Le minimum local trouvé est le plus proche du point initial x_0.
  → initialisation à modifier pour trouver un autre minimum local

- Les **méthodes « globales »** explorent « aléatoirement » les solutions
  → localisation possible de plusieurs minima locaux

Conditions d'arrêt

- Déplacement insuffisant : $\|x_{k+1} - x_k\| < \varepsilon_x$
- Amélioration insuffisante : $f_k - f_{k+1} < \varepsilon_f$
- Condition d'ordre 1 vérifiée : $\|g_k\| < \varepsilon_g$
- Nombre maximal d'itérations ou d'appels fonction : N_{iter} , N_{fonc}

Sommaire

2.3.1 Recherche linéaire

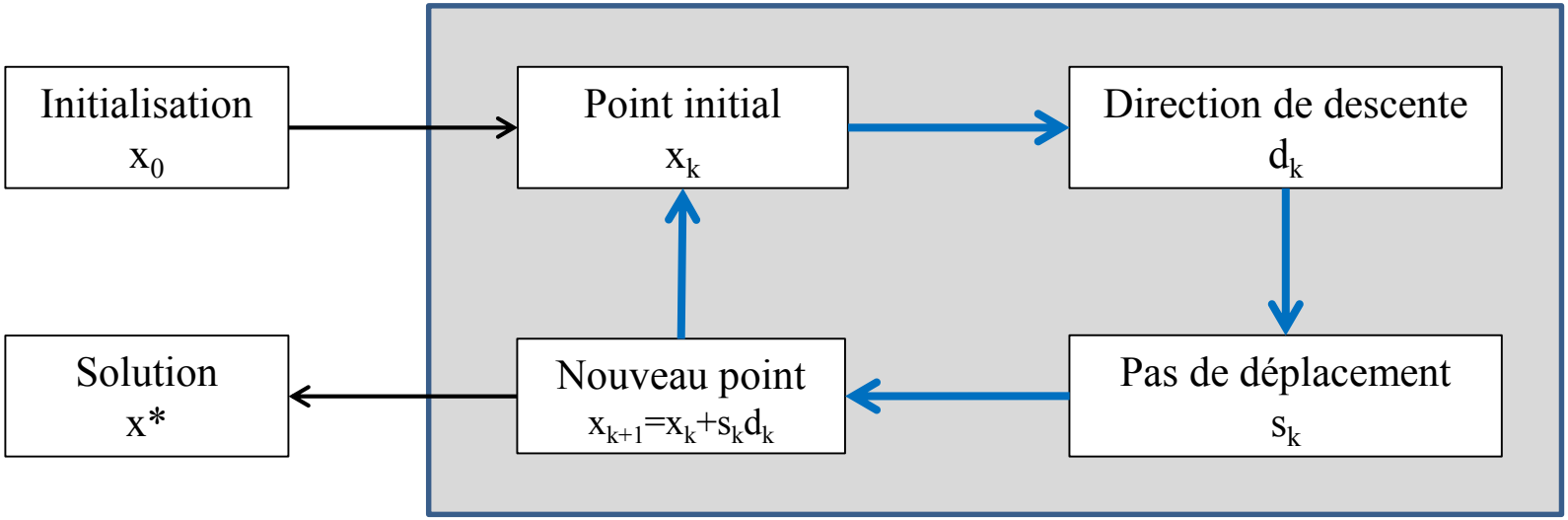
Problème sans contrainte

$$\min_{x \in R^n} f(x)$$

Etapes principales

A chaque itération

- Construction d'une direction de descente d_k à partir du point x_k
- Réglage du pas de déplacement s_k suivant d_k



2.3.1 Recherche linéaire

Etapes principales

A chaque itération

- Construction d'une direction de descente d_k à partir du point x_k
- Réglage du pas de déplacement s_k suivant d_k

Direction de descente

d_k est une direction de descente en x_k si $\nabla f(x_k)^T d_k < 0$

La direction de descente est construite à partir du gradient et du hessien.

- **Plus forte pente** → gradient (méthode d'ordre 1)
- **Préconditionnement** → hessien (méthode d'ordre 2)

Pas de déplacement

Le pas de déplacement s_k suivant d_k doit vérifier $f(x_k+s_k d_k) < f(x_k)$

L'algorithme de recherche linéaire résout un problème de minimisation à une variable s

- Minimisation **exacte** → dichotomie (Fibonacci, nombre d'or, interpolation)
- Minimisation **approchée** → règles de pas acceptable (Armijo, Goldstein, Wolfe)

2.3.2 Direction de descente

- Plus forte pente
- Gradient accéléré
- Méthode de Nesterov
- Préconditionnement

2.3.2 Plus forte pente

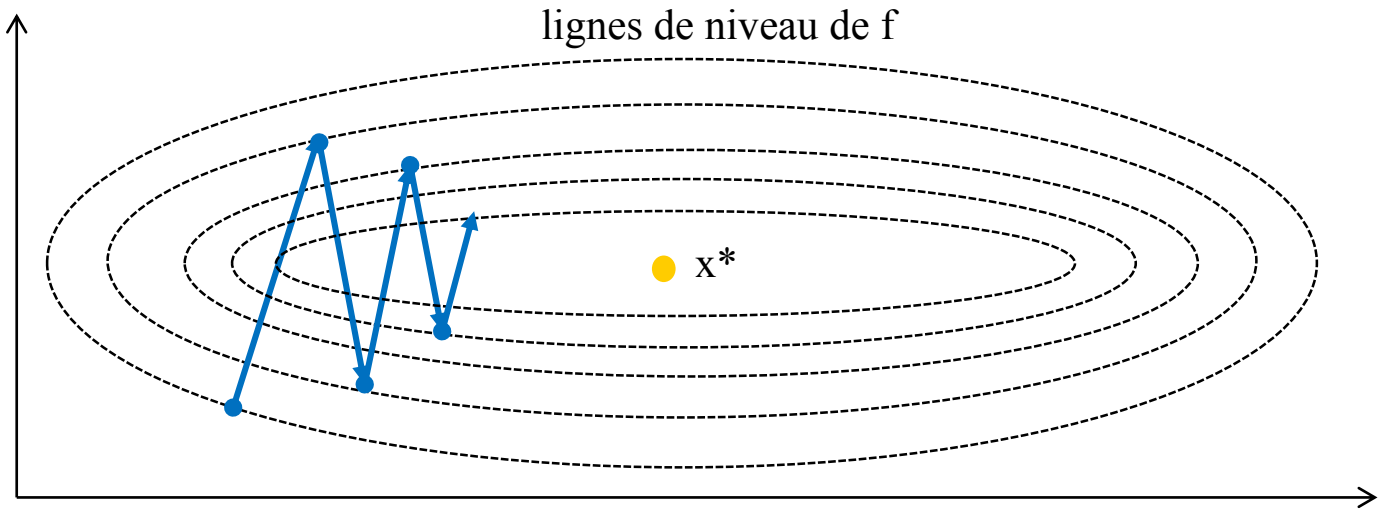
Plus forte pente

La direction de descente « naturelle » est celle du gradient = plus forte dérivée directionnelle

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

- Comportement caractéristique en zigzag
- **Convergence très lente** → méthode inefficace en général
- Améliorations possibles → gradient accéléré, méthode de Nesterov, préconditionnement

Illustration en dimension 2



2.3.2 Plus forte pente

Plus forte pente

Les directions successives de plus forte pente avec pas optimal sont **orthogonales**.

- Itération k

  On cherche le minimum de f à partir de x_k suivant la direction $d_k = -\nabla f(x_k)$

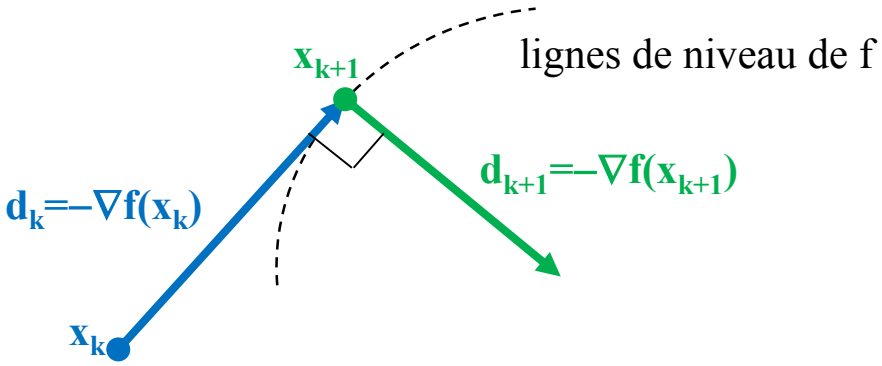
  Le nouveau point est $x_{k+1} = x_k + s d_k$ avec le pas $s>0$ solution de :

$$\min_{s\in R} f(x_k + s d_k) \Rightarrow \frac{d}{ds} f(x_k + s d_k) = 0 \Rightarrow d_k^T \nabla f(x_k + s d_k) = 0 \Rightarrow d_k^T \nabla f(x_{k+1}) = 0$$

- Itération k+1

  La direction de plus forte pente en x_{k+1} est $d_{k+1} = -\nabla f(x_{k+1})$ $\Rightarrow$ $\boxed{d_k^T d_{k+1} = 0}$

Illustration



2.3.2 Exemple

Plus forte pente

- Fonction

$$f(x)=\frac{1}{2}x_1^2+\frac{9}{2}x_2^2$$

- Direction

$$d=-\nabla f(x)=\begin{pmatrix}-x_1\\-9x_2\end{pmatrix}$$

- Pas

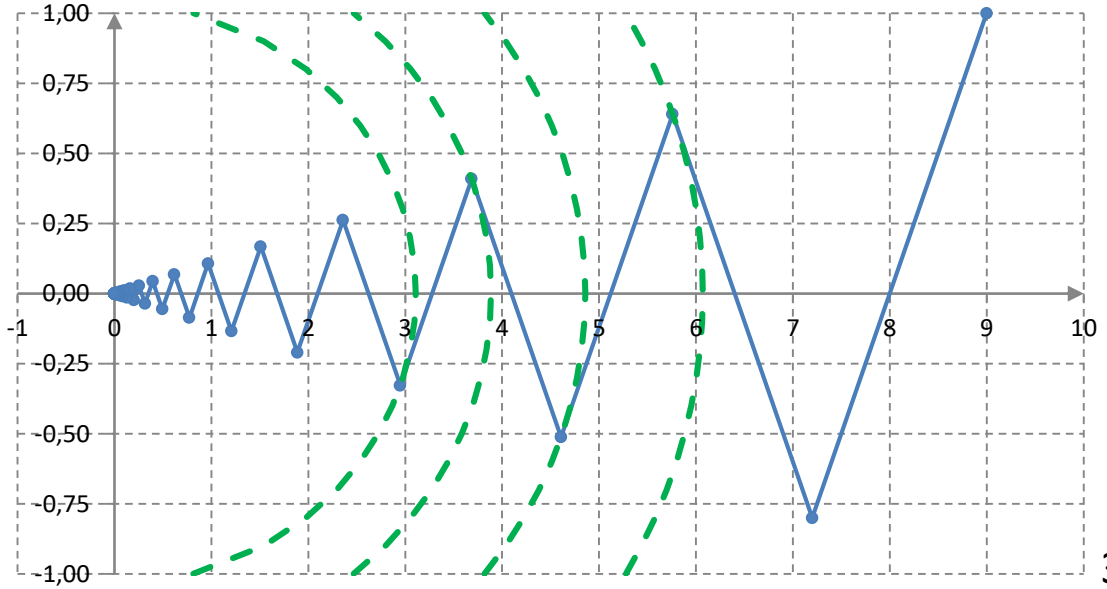
$$\min_s f(x+sd)$$

$$\rightarrow s=\frac{x_1^2+81x_2^2}{x_1^2+729x_2^2}$$

- Itération

$$x_{k+1}=x_k+s_k d_k$$

Iteration	x1	x2	f(x)	d1	d2	s	Erreur
0	9,000	1,000	45,000	-9,000	-9,000	0,2	9,055
1	7,200	-0,800	28,800	-7,200	7,200	0,2	7,244
2	5,760	0,640	18,432	-5,760	-5,760	0,2	5,795
3	4,608	-0,512	11,796	-4,608	4,608	0,2	4,636
4	3,686	0,410	7,550	-3,686	-3,686	0,2	3,709
5	2,949	-0,328	4,832	-2,949	2,949	0,2	2,967
10	0,966	0,107	0,519	-0,966	-0,966	0,2	0,972
20	0,104	0,012	0,006	-0,104	-0,104	0,2	0,104
30	1,11E-02	1,24E-03	6,90E-05	-1,11E-02	-1,11E-02	0,2	1,12E-02
40	1,20E-03	1,33E-04	7,95E-07	-1,20E-03	-1,20E-03	0,2	1,20E-03
50	1,28E-04	1,43E-05	9,17E-09	-1,28E-04	-1,28E-04	0,2	1,29E-04



2.3.2 Gradient accéléré

Gradient accéléré

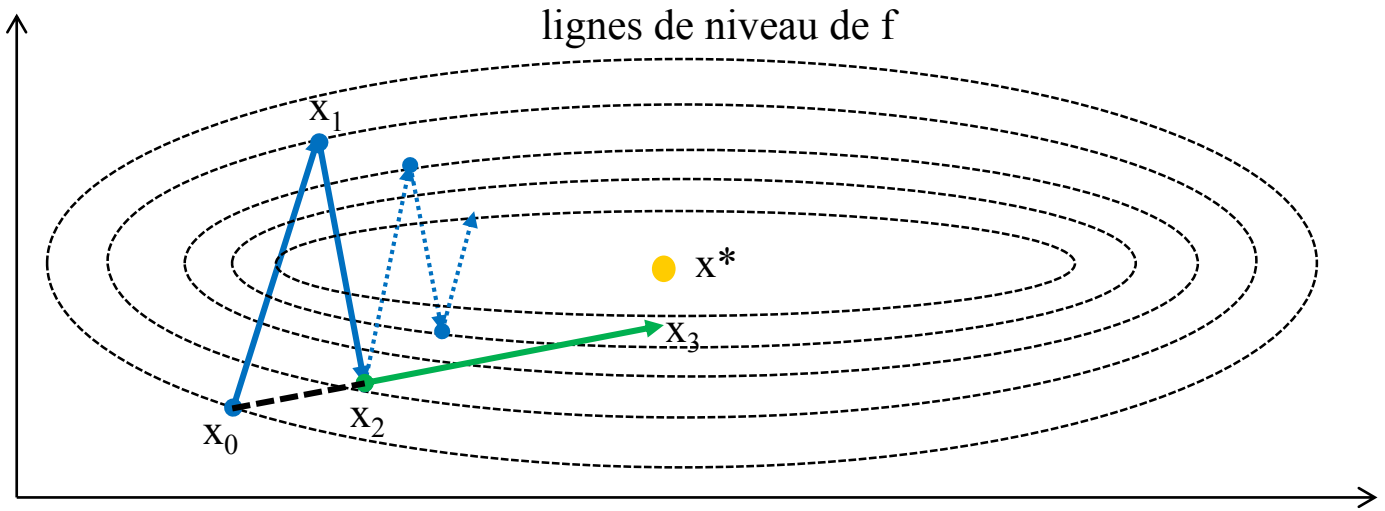
La méthode du gradient accéléré consiste à introduire des déplacements supplémentaires.

- On réalise p déplacements suivant la plus forte pente : $d_k \,//\, -g(x_k)$

$$x_0 \xrightarrow{d_0} x_1 \xrightarrow{d_1} x_2 \xrightarrow{d_2} \cdots\cdots \xrightarrow{d_{p-1}} x_p$$

- On réalise ensuite un déplacement suivant la direction : $d_p = x_p - x_0$.
  d_p = **direction moyenne** des p derniers déplacements (en général p = n , dimension de x)

$$x_p \xrightarrow{d_p} x_{p+1}$$



2.3.2 Gradient accéléré

Justification (en dimension 2)

- On suppose que la fonction à minimiser est une **fonction quadratique de 2 variables**.

$$f(x)=\frac{1}{2}x^TQx+c^Tx\ ,\ x\in R^2 \quad\rightarrow\quad g(x)=\nabla f(x)=Qx+c \quad\rightarrow\quad x^*=-Q^{-1}c$$

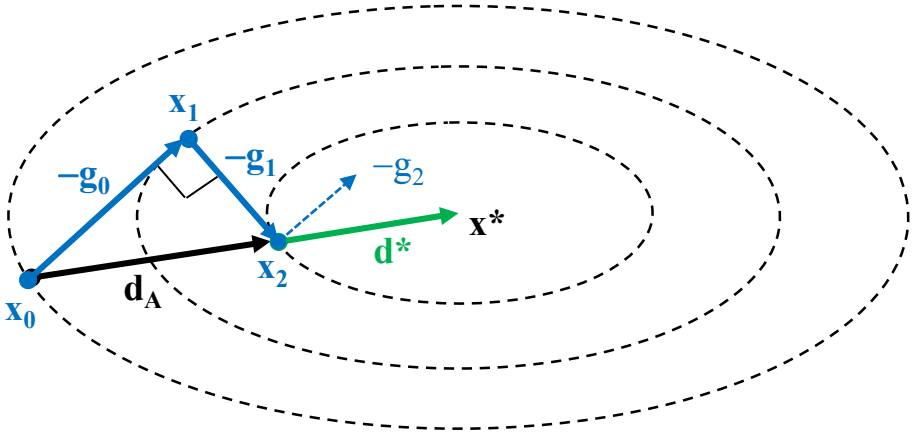
On a les relations : $\begin{cases} g_0 = Qx_0 + c \\ g_2 = Qx_2 + c \\ 0 = Qx^* + c \end{cases} \Rightarrow \begin{cases} x_2 - x_0 \underset{\text{noté}}{=} \\ x^* - x_2 \underset{\text{noté}}{=} \end{cases} \boxed{\begin{array}{l} d_A = Q^{-1}(g_2 - g_0) \\ d^* = -Q^{-1}g_2 \end{array}}$

- On considère des itérations de plus forte pente à partir du point x_0 : $d_k \ // \ g(x_k) \underset{\text{noté}}{=} g_k$
  Les directions successives sont orthogonales :

$$\begin{cases} g_1 \perp g_0 \\ g_2 \perp g_1 \end{cases} \Rightarrow g_2 \ // \ g_0 \Rightarrow g_2 - g_0 = \alpha g_2$$

$$\Rightarrow \boxed{d_A = \alpha Q^{-1} g_2}$$

Vrai seulement en dimension 2



- La direction d_A (de x_0 vers x_2) est parallèle à la direction d* (de x_2 vers le minimum x*).
  Le minimum se trouve suivant la direction d_A et peut être atteint dès la troisième itération.

2.3.2 Exemple

Gradient accéléré

- Fonction $f(x)=\frac{1}{2}x_1^2+\frac{9}{2}x_2^2 \rightarrow g(x)=\nabla f(x)=\begin{pmatrix} x_1 \\ 9x_2 \end{pmatrix} \rightarrow H(x)=\nabla^2 f(x)=\begin{pmatrix} 1 & 0 \\ 0 & 9 \end{pmatrix}$

- Minimisation suivant $d=\begin{pmatrix} d_1 \\ d_2 \end{pmatrix}$: $\min_s f(x+sd)=f(x)+sd^T g(x)+\frac{1}{2}sd^T H(x)d$

$$\rightarrow s=-\frac{d^T g}{d^T H g}=-\frac{d_1 x_1+9d_2 x_2}{d_1^2+9d_2^2}$$

- Point initial : $x^{(0)}=\begin{pmatrix} 9 \\ 1 \end{pmatrix}$

Iteration	x1	x2	f(x)	d1	d2	s	Erreur
0	9,0000	1,0000	45,0000	-9,0000	-9,0000	0,2	9,0554
1	7,2000	-0,8000	28,8000	-7,2000	7,2000	0,2	7,2443
2	5,7600	0,6400	18,4320	-3,2400	-0,3600	1,8	5,7954
3	0,0000	0,0000	0,0000				0,0000

- Déplacements 1 et 2
  Direction de plus forte pente : $d^{(0)}=-g(x^{(0)}) \longrightarrow x^{(1)}$
  $d^{(1)}=-g(x^{(1)}) \longrightarrow x^{(2)}$

- Déplacement 3 suivant : $d=x^{(2)}-x^{(0)} \longrightarrow x^{(3)}=x^*$

→ **convergence en 3 itérations**

2.3.2 Méthode de Nesterov

Méthode de Nesterov

La méthode de Nesterov consiste à ajouter un déplacement intermédiaire à chaque itération.

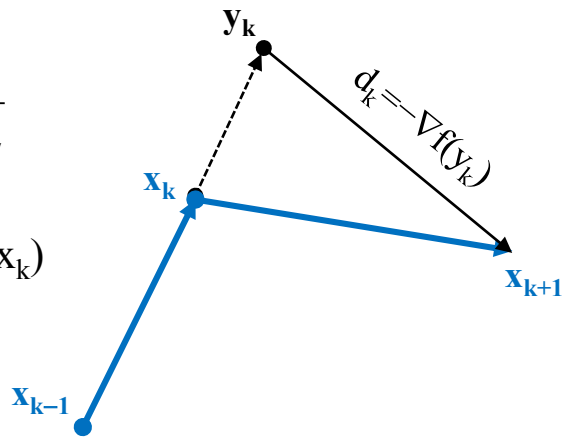
- Les 2 premières itérations sont identiques à la méthode de plus forte pente.
  A partir de l'itération 3, chaque itération se compose de 2 déplacements.

  - un déplacement à partir de x_k **suivant la direction $x_k - x_{k-1}$**

    → point y_k : $y_k = x_k + \theta_k (x_k - x_{k-1})$ avec $\theta_k = \dfrac{k-1}{k+2}$

  - un déplacement de plus forte pente **à partir de y_k** (au lieu de x_k)

    → point x_{k+1} : $x_{k+1} = y_k + s_k d_k$ avec $d_k = -g(y_k)$



- Le déplacement complémentaire décale le point suivant la dernière direction de déplacement.
  Le nouveau point y_k est plus favorable pour une itération de plus forte pente,
  en particulier si la fonction présente une vallée étroite et allongée.

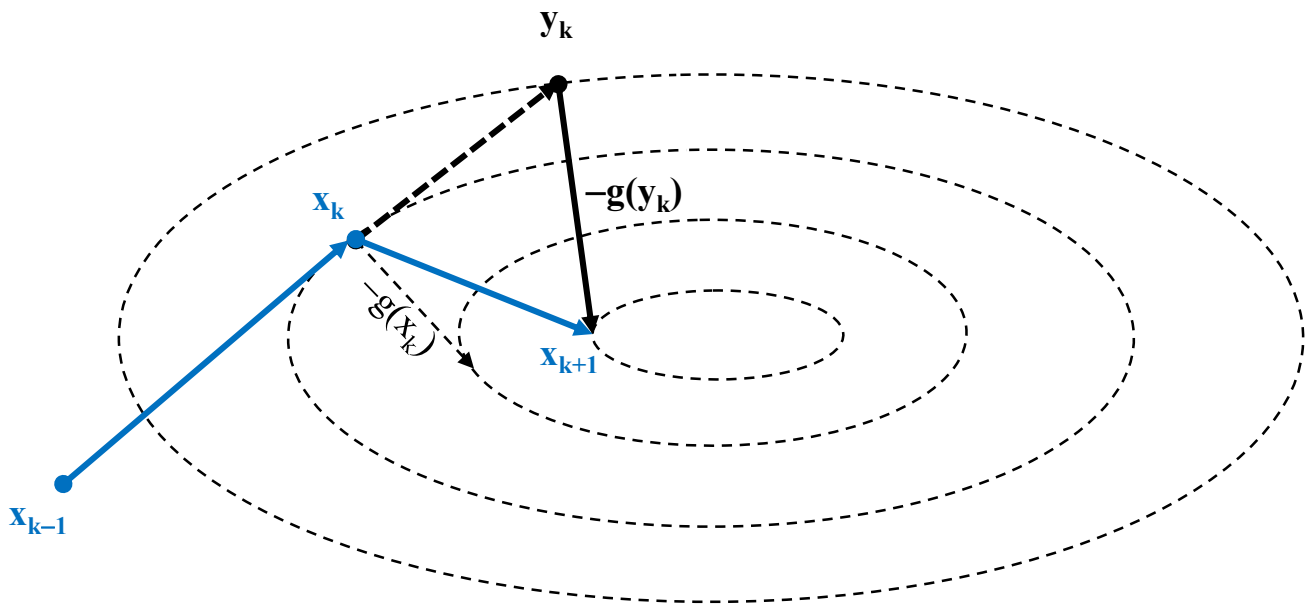
2.3.2 Méthode de Nesterov

Méthode de Nesterov

- Itération composée de 2 déplacements

$$y_k = x_k + \theta_k (x_k - x_{k-1}) \quad \text{avec} \quad \theta_k = \frac{k-1}{k+2} \qquad \text{à partir de } k = 2$$

$$x_{k+1} = y_k + s_k d_k \qquad \text{avec} \quad d_k = -g(y_k) \qquad \text{et minimisation suivant } d_k$$



2.3.2 Exemple

Méthode de Nesterov

- Fonction : $f(x)=\frac{1}{2}x_1^2+\frac{9}{2}x_2^2$
- Itération :

$$\begin{cases} y_k = x_k+\theta_k(x_k-x_{k-1}) & \text{avec } \theta_k=\dfrac{k-1}{k+2} \\ x_{k+1}=y_k+s_k d_k & \text{avec } d_k=-\nabla f(y_k)=\begin{pmatrix}-y_1\\-9y_2\end{pmatrix} \end{cases}$$

$$\text{et } \min_s f(y_k+sd_k) \rightarrow s=-\frac{d_1y_1+9d_2y_2}{d_1^2+9d_2^2}$$

- Comparaison à la méthode de plus forte pente

	Plus forte pente			Nesterov		
Iteration	x1	x2	f	x1	x2	f
0	9,000	1,000	45,0000	9,0000	1,0000	45,0000
1	7,200	-0,800	28,8000	7,2000	-0,8000	28,8000
2	5,760	0,640	18,4320	5,7600	0,6400	18,4320
3	4,608	-0,512	11,7965	4,6154	-0,3077	11,0769
4	3,686	0,410	7,5497	3,5187	0,2630	6,5018
5	2,949	-0,328	4,8318	2,5381	-0,1697	3,3507
6	2,359	0,262	3,0924	1,7054	0,0999	1,4991
7	1,887	-0,210	1,9791	1,0259	-0,0559	0,5403
8	1,510	0,168	1,2666	0,5004	0,0221	0,1274
9	1,208	-0,134	0,8106	0,1173	-0,0025	0,0069
10	0,966	0,107	0,5188	-0,1320	0,0129	0,0095

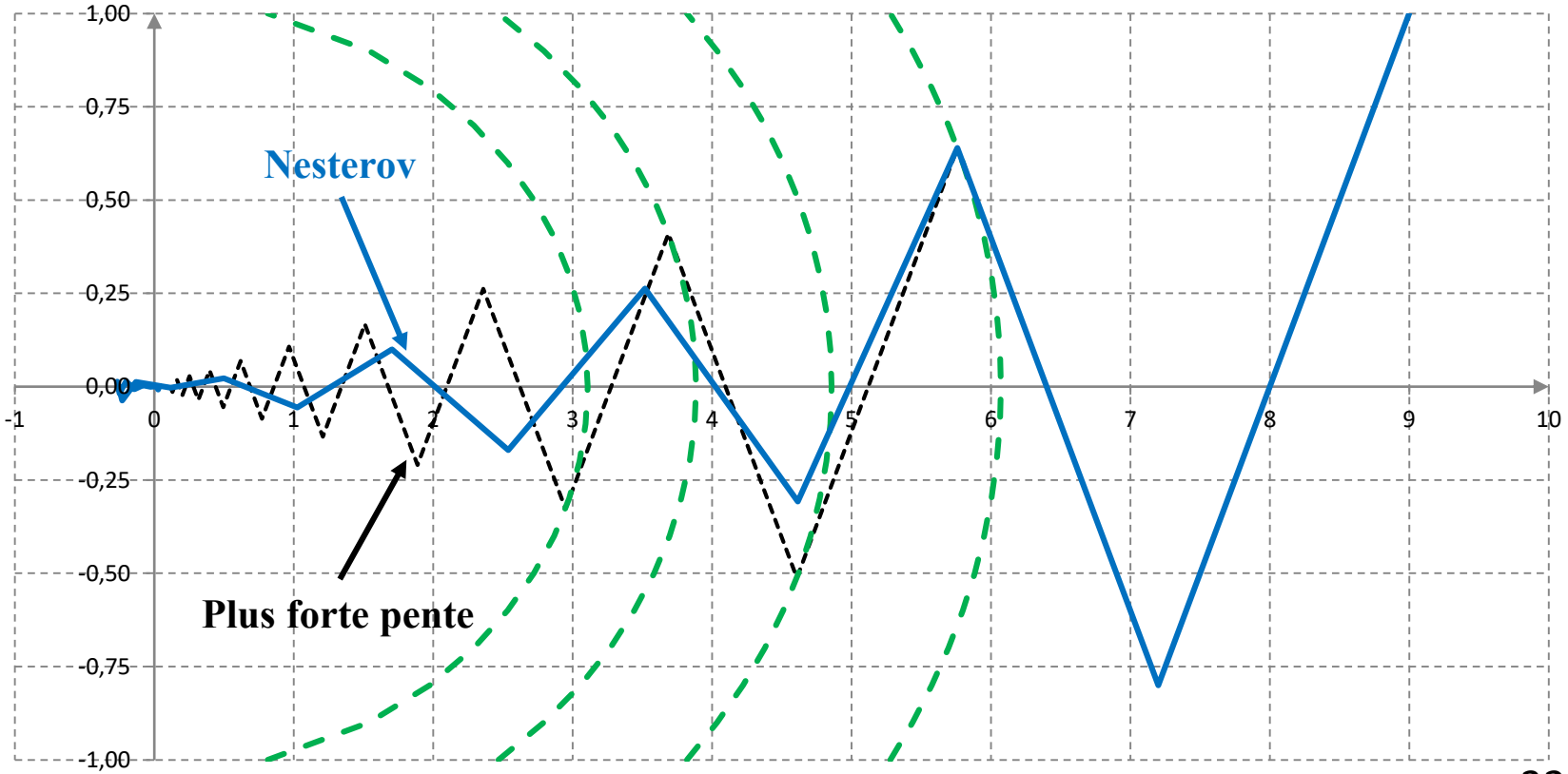
→ 2 itérations initiales identiques

→ après 10 itérations

2.3.2 Exemple

Méthode de Nesterov

- Fonction : $f(x)=\frac{1}{2}x_1^2+\frac{9}{2}x_2^2$



2.3.2 Préconditionnement

Préconditionnement

- On se donne une matrice H_k **symétrique définie positive**.
  $\rightarrow$ factorisation de Cholesky de H_k : $H_k = L_k L_k^T$
- Changement de variable : $\boxed{\tilde{x}_k = L_k^T x_k}$
- Direction de plus forte pente pour : $\tilde{f}(\tilde{x}_k) = f(x_k) = f(L_k^{-T}\tilde{x}_k)$

  $\tilde{d}_k = -\nabla\tilde{f}(\tilde{x}_k) = -\nabla f(L_k^{-T}\tilde{x}_k) = -L_k^{-1}\nabla f(L_k^{-T}\tilde{x}_k) = -L_k^{-1}\nabla f(x_k) = -L_k^{-1}d_k$
- Itération en $\tilde{x}_k$: $\tilde{x}_{k+1} = \tilde{x}_k - s_k\nabla\tilde{f}(\tilde{x}_k)$
- Itération en x_k : $x_{k+1} = L_k^{-T}\tilde{x}_{k+1} = L_k^{-T}\left(\tilde{x}_k - s_k\nabla\tilde{f}(\tilde{x}_k)\right) = L_k^{-T}\left(L_k^T x_k - s_k L_k^{-1}d_k\right)$

  $\Rightarrow x_{k+1} = x_k - s_k L_k^{-T}L_k^{-1}d_k = x_k - s_k H_k^{-1}d_k$
- Le **préconditionnement par la matrice H_k** revient à prendre comme direction de descente

  $\boxed{d_k = -H_k^{-1}\nabla f(x_k)}$
- On vérifie que d_k est une direction de descente

  $d_k^T\nabla f(x_k) = -\nabla f(x_k)^T H_k^{-1}\nabla f(x_k) < 0$ car H_k est définie positive

2.3.2 Préconditionnement

Choix du préconditionnement

Toute matrice H_k **symétrique définie positive** convient.

Cas d'un hessien défini positif

- Si le hessien $\nabla^2 f(x_k)$ est défini positif, on peut prendre $\boxed{H_k = \nabla^2 f(x_k)}$.

  $d_k = -H_k^{-1}\nabla f(x_k) = -\nabla^2 f(x_k)^{-1}\nabla f(x_k) \quad \Rightarrow \; x_{k+1} = x_k - s_k \nabla^2 f(x_k)^{-1}\nabla f(x_k)$

  On obtient l'itération de Newton si le pas s_k vaut 1.

- On peut prendre pour H_k **l'approximation du hessien** donnée par une méthode quasi-Newton.

Cas d'un hessien non défini positif

- On peut effectuer la factorisation de Cholesky modifiée du hessien $\nabla^2 f(x_k)$ (ou de son approximation quasi Newton) pour obtenir une matrice définie positive.

- On peut ajouter un multiple de l'identité : $\boxed{H_k = \nabla^2 f(x_k) + \tau I}$ avec $\tau > 0$ assez grand.

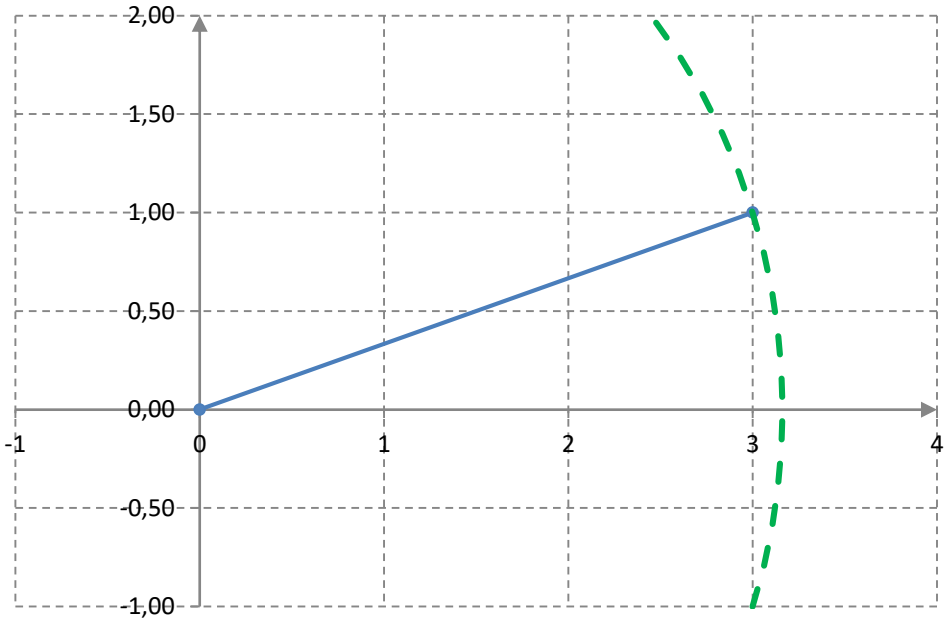
- On peut également prendre H_k diagonale : $(H_k)_{ii} = \max\left(\varepsilon, \dfrac{\partial^2 f}{\partial x_i^2}(x_k)\right)$, $\varepsilon > 0$
  à partir des dérivées secondes diagonales de f.

2.3.2 Exemple

Préconditionnement

- Fonction : $f(x)=\frac{1}{2}x_1^2+\frac{9}{2}x_2^2 \Rightarrow \nabla f(x)=\begin{pmatrix} x_1 \\ 9x_2 \end{pmatrix} \Rightarrow \nabla^2 f(x)=\begin{pmatrix} 1 & 0 \\ 0 & 9 \end{pmatrix}=\begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix}^T$
- Préconditionnement : $L=\begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix} \Rightarrow \begin{cases} \tilde{x}_1 = x_1 \\ \tilde{x}_2 = 3x_2 \end{cases} \Rightarrow \tilde{f}(\tilde{x})=\frac{1}{2}\tilde{x}_1^2+\frac{9}{2}\left(\frac{1}{3}x_2\right)^2=\frac{1}{2}\tilde{x}_1^2+\frac{1}{2}\tilde{x}_2^2$
- Direction : $\tilde{d}=-\nabla\tilde{f}(\tilde{x})=\begin{pmatrix} -\tilde{x}_1 \\ -\tilde{x}_2 \end{pmatrix}$
- Pas : $\min_s \tilde{f}(\tilde{x}+s\tilde{d}) \rightarrow s=1$
- Itération : $\tilde{x}_{k+1}=\tilde{x}_k+s_k\tilde{d}_k=0$

  $\rightarrow$ **convergence en 1 itération**



2.3.3 Pas de déplacement

- Minimisation unidimensionnelle

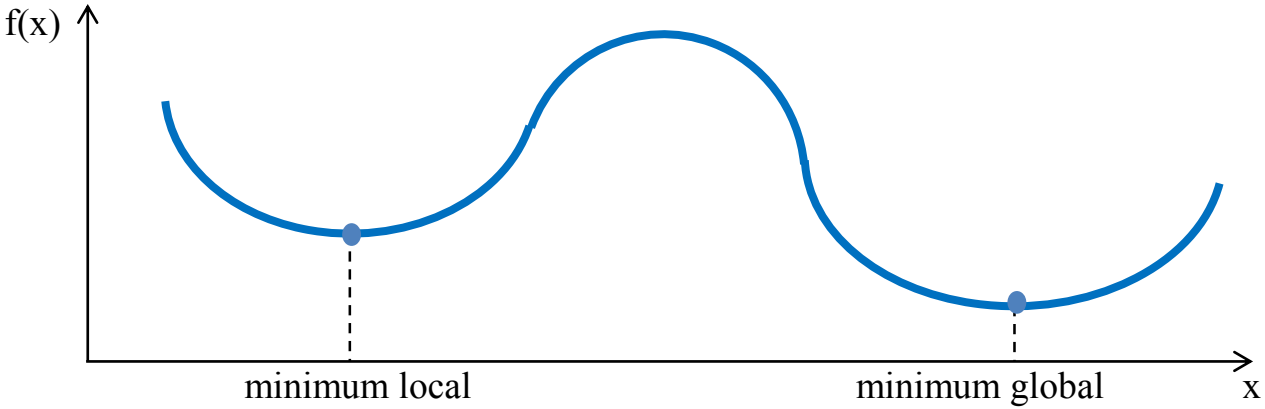
- Minimisation exacte
  - Méthode de dichotomie
  - Méthode de Fibonacci
  - Méthode du nombre d'or
  - Interpolation quadratique

- Minimisation approchée
  - Règle d'Armijo
  - Règle de Goldstein
  - Règle de Wolfe

2.3.3 Minimisation unidimensionnelle

Problème à une variable

$$\min_{x \in R} f(x)$$



Méthodes

- **Minimisation exacte**

  On cherche à trouver un minimum local x* avec une précision donnée
  → réduction itérative de l'intervalle de recherche : dichotomie
  → réduction optimale : Fibonacci, nombre d'or

- **Minimisation approchée**

  On cherche une réduction suffisante de la fonction sans déterminer précisément le minimum x*
  → règles d'acceptation (Armijo, Goldstein, Wolfe)
  → limitation du nombre d'évaluations de la fonction

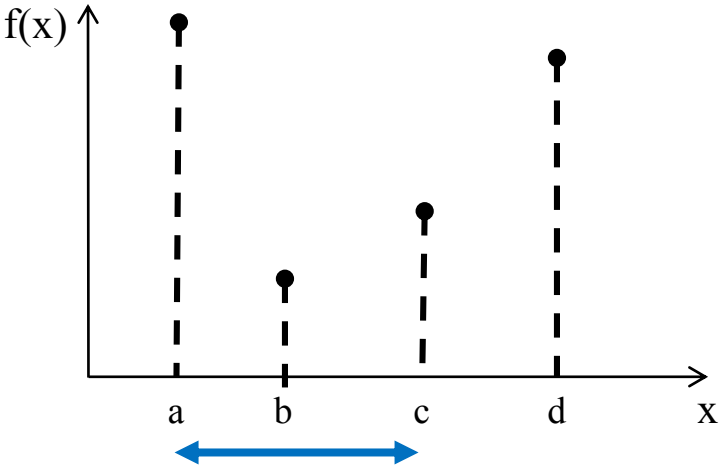
2.3.3 Minimisation exacte

Recherche par dichotomie

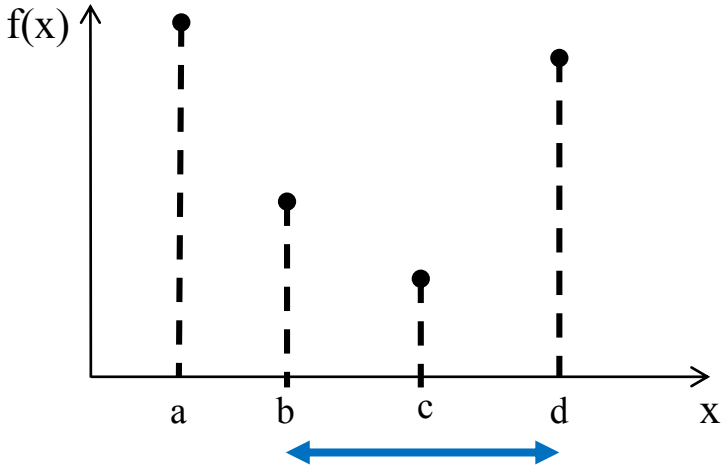
L'allure de la fonction f n'est pas connue.

On suppose qu'il existe un minimum unique dans l'intervalle de recherche [a,d]

- On évalue la fonction aux extrémités a et d : → f(a) , f(d)
- On évalue la fonction en 2 points b et c : a < b < c < d → f(b) , f(c)
- On conserve soit l'intervalle [a,c], soit l'intervalle [b,d]



On conserve [a,c]

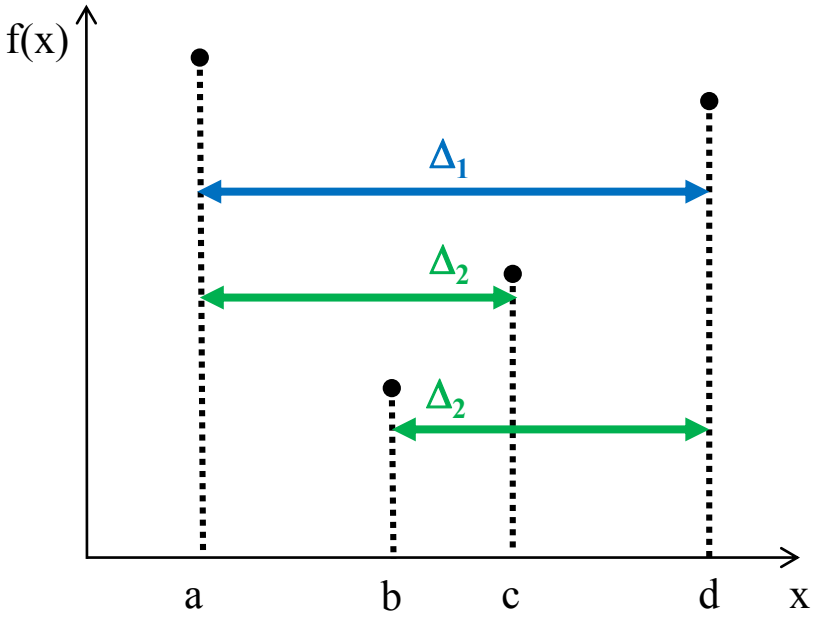


On conserve [b,d]

2.3.3 Minimisation exacte

Réduction optimale de l'intervalle

- On cherche b et c pour que l'intervalle restant soit le plus petit possible.
  La taille de l'intervalle restant ne doit pas dépendre du choix de [a,c] ou [b,d].
- On note : Δ_1 la longueur de l'intervalle initial → $\Delta_1 = d - a$
  Δ_2 la longueur de l'intervalle restant → $\Delta_2 = c - a$ si on garde [a,c]
  ou $\Delta_2 = d - b$ si on garde [b,d]



$$\boxed{\Delta_2 = d - b = c - a}$$

$$\Rightarrow a + d = b + c \Rightarrow \frac{a+d}{2} = \frac{b+c}{2}$$

Les points b et c doivent être symétriques par rapport au milieu de l'intervalle [a,d].

2.3.3 Minimisation exacte

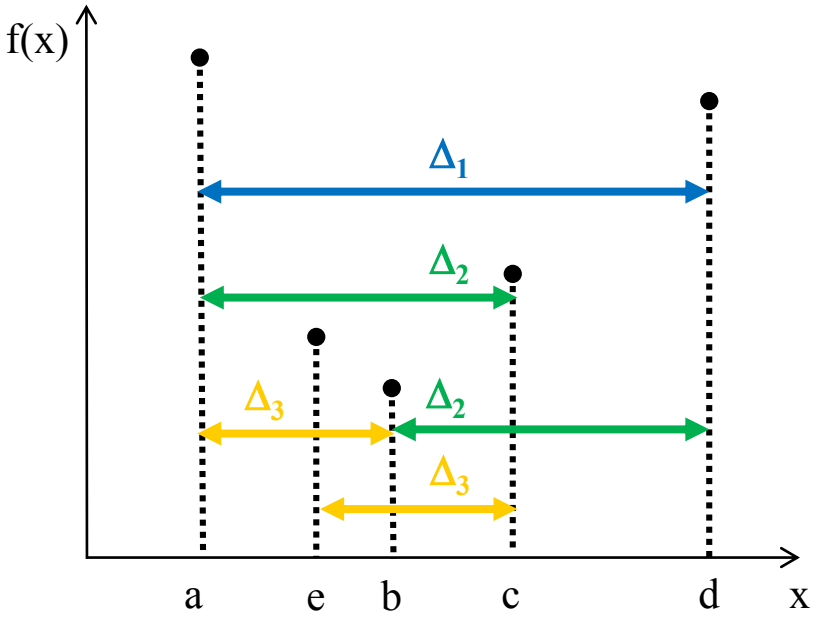
Réduction optimale de l'intervalle

- On suppose que l'intervalle restant est [a,c].
- Pour **réutiliser le point b** à l'itération suivante, on choisit le nouveau point e symétrique de b par rapport au milieu de l'intervalle [a,c].

$\rightarrow \quad \Delta_1 = d - a$

$\rightarrow \quad \Delta_2 = c - a = d - b$

$\rightarrow \quad \Delta_3 = b - a = c - e$



$$d - a = (d - b) + (b - a)$$

$$\Rightarrow \Delta_1 = \Delta_2 + \Delta_3$$

Les longueurs Δ_k des intervalles successifs vérifient :

$$\boxed{\Delta_k = \Delta_{k+1} + \Delta_{k+2}}$$

2.3.3 Minimisation exacte

Réduction optimale de l'intervalle

- Les intervalles successifs sont de longueur Δ_k vérifiant : $\Delta_k = \Delta_{k+1} + \Delta_{k+2}$
- Après un nombre N d'itérations, on obtient un intervalle de longueur Δ_{N-1} .
- On définit la suite de nombres F_k par : $\boxed{\Delta_k = F_{N-k}\,\Delta_{N-1}}$ pour $k = 1,\ldots,N-1$
- La suite $(F_n)_{n=1,\ldots,N-1}$ vérifie

$$\Delta_k = \Delta_{k+1} + \Delta_{k+2} \Rightarrow \frac{\Delta_k}{\Delta_{N-1}} = \frac{\Delta_{k+1}}{\Delta_{N-1}} + \frac{\Delta_{k+2}}{\Delta_{N-1}} \Rightarrow F_{N-k} = F_{N-k-1} + F_{N-k-2}$$

$$\Rightarrow F_n = F_{n-1} + F_{n-2} \quad \text{pour } n = N-k,\ n = 3,4,\ldots,N-1$$

$$\Delta_{N-1} = F_1 \Delta_{N-1} \text{ pour } k = N-1 \Rightarrow F_1 = 1$$

- La suite est complètement déterminée par la valeur de F_2

$$\begin{cases} F_1 = 1 \\ F_2 \quad \text{à choisir} \\ F_n = F_{n-1} + F_{n-2} \quad \text{pour } n = 3,4,\ldots,N-1 \end{cases}$$

2.3.3 Minimisation exacte

Méthode de Fibonacci

- Pour un nombre N d'itérations fixé, on choisit F_2 pour que Δ_{N-1} soit minimal.

$$\Delta_{N-1} = \frac{\Delta_1}{F_{N-1}} \text{ minimal } \Rightarrow F_{N-1} \text{ maximal } \Rightarrow F_2 \text{ maximal}$$

- Valeur maximale de F_2 : $\Delta_{N-1} \geq \frac{1}{2}\Delta_{N-2}$ $\left(\text{car } \Delta_k \geq \frac{1}{2}\Delta_{k-1}\right)$ $\Rightarrow F_1 \geq \frac{1}{2}F_2 \Rightarrow F_{2\max} = 2$

- On obtient la **suite de Fibonacci** : 1, 2, 3, 5, 8, 13, 21, …

Nombre d'itérations

La méthode de Fibonacci est optimale si l'on fixe à l'avance la précision requise sur la solution.

- **Précision requise** : $\Delta_{N-1} = p$
- Intervalle initial [a,b] : $\Delta_1 = b-a$

$$\Rightarrow \Delta_{N-1} = \frac{\Delta_1}{F_{N-1}} \Rightarrow F_{N-1} = \frac{b-a}{p} \rightarrow \text{valeur de N}$$

La méthode de Fibonacci donne le nombre minimal d'itérations N pour obtenir la solution **avec une précision donnée**.

- La disposition des premiers points b et c dépend de N : $\frac{\Delta_1}{\Delta_2} = \frac{F_{N-1}}{F_{N-2}} \rightarrow \Delta_2$

2.3.3 Minimisation exacte

Méthode du nombre d'or

La méthode de Fibonacci nécessite de changer la disposition des points à chaque itération.
La disposition des points n'est optimale que **pour une précision donnée**.
La méthode du nombre d'or est plus générale et plus simple à mettre en oeuvre.

- On impose un **rapport de réduction fixe** de l'intervalle à chaque itération.

$$\frac{\Delta_1}{\Delta_2}=\frac{\Delta_2}{\Delta_3}=\ldots\frac{\Delta_k}{\Delta_{k+1}}=\frac{\Delta_{k+1}}{\Delta_{k+2}}=\ldots=\gamma \quad \text{avec} \quad \Delta_k=\Delta_{k+1}+\Delta_{k+2}$$

$$\Rightarrow \frac{\Delta_k}{\Delta_{k+1}}=1+\frac{\Delta_{k+2}}{\Delta_{k+1}} \Rightarrow \gamma=1+\frac{1}{\gamma} \Rightarrow \gamma^2-\gamma-1=0$$

- On obtient pour le rapport γ le nombre d'or :
  → **Méthode du nombre d'or** (ou de la **section dorée**)

$$\boxed{\gamma=\frac{1+\sqrt{5}}{2}\approx 1.618034}$$

Optimalité

- La méthode du nombre d'or n'est pas optimale pour un nombre d'itérations donné N.
- Pour un nombre d'itérations grand : $\lim_{N\to\infty}\frac{F_N}{F_{N-1}}=\gamma$

  → La disposition des points de Fibonacci tend vers celle du nombre d'or.

2.3.3 Minimisation exacte

Comparaison Fibonacci - Nombre d'or

Le rapport de réduction de l'intervalle après n itérations vaut :

- Pour la méthode de Fibonacci : $\frac{\Delta_1}{\Delta_n} = F_n$
- Pour la méthode du nombre d'or : $\frac{\Delta_1}{\Delta_n} = \gamma^{n-1}$

Nombre d'itérations

Une itération de la méthode de Fibonacci ou du nombre d'or correspond à une évaluation de f.

Nombre d'évaluations

Rapport de réduction	Fibonacci	Nombre d'or
10^{-2}	11	13
10^{-3}	15	18
10^{-4}	20	22
10^{-5}	25	27
10^{-6}	30	31

2.3.3 Minimisation exacte

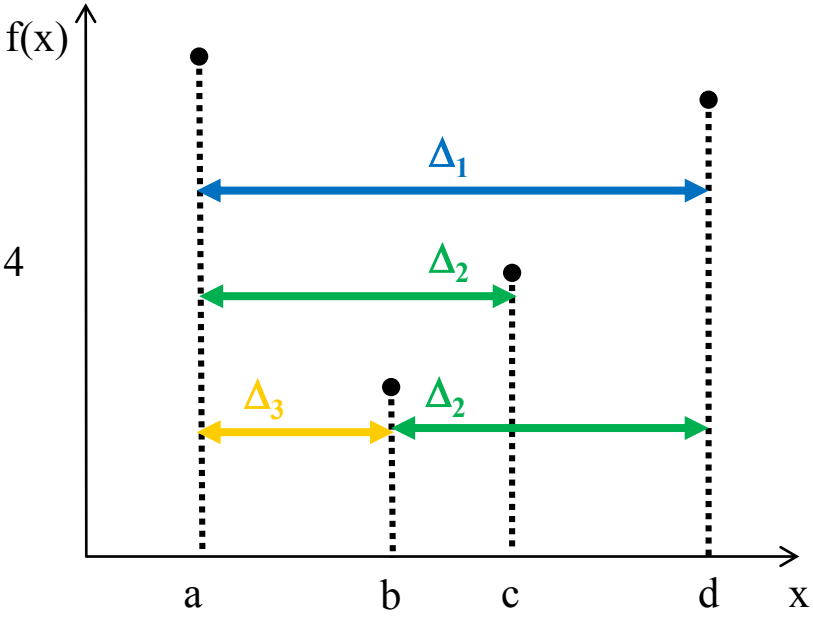
Méthode du nombre d'or

- Positionnement des points

$$\frac{\Delta_1}{\Delta_2} = \frac{\Delta_2}{\Delta_3} = \gamma = \frac{1+\sqrt{5}}{2} \approx 1.618034$$

$$\Rightarrow \begin{cases} \Delta_2 = \frac{1}{\gamma}\Delta_1 = r\Delta_1 \\ \Delta_3 = \frac{1}{\gamma^2}\Delta_1 = r^2\Delta_1 \end{cases} \quad \text{avec } r = \frac{1}{\gamma} = \frac{\sqrt{5}-1}{2} \approx 0.618034$$

$$\Rightarrow \begin{cases} b = a + r^2\Delta_1 \\ c = a + r\Delta_1 \\ d = a + \Delta_1 \end{cases}$$



- **Itération**

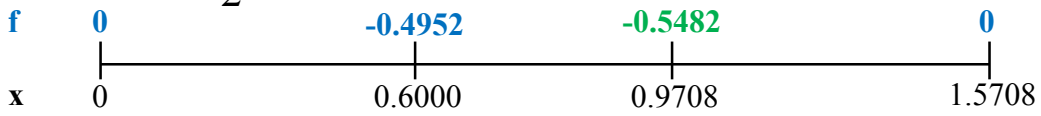
$$\text{Si } f(b) < f(c) \rightarrow \begin{cases} d \leftarrow c \\ c \leftarrow b \\ \Delta_1 \leftarrow \Delta_2 \\ b = a + r^2\Delta_1 \end{cases} \qquad \text{Si } f(b) > f(c) \rightarrow \begin{cases} a \leftarrow b \\ b \leftarrow c \\ \Delta_1 \leftarrow \Delta_2 \\ c = a + r\Delta_1 \end{cases}$$

2.3.3 Exemple

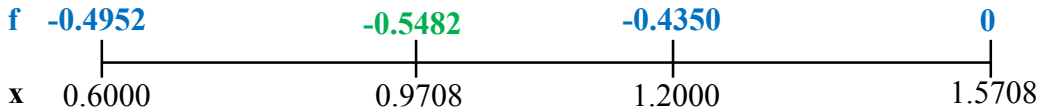
Méthode du nombre d'or

- Minimisation de $f(x) = -x\cos(x)$, $0 \le x \le \frac{\pi}{2}$

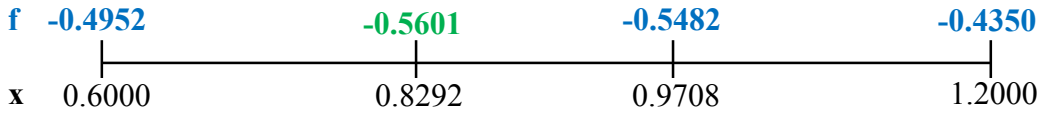
- Itération 1



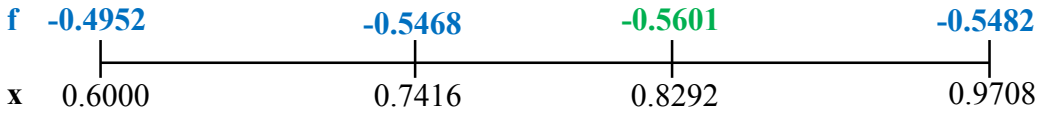
- Itération 2



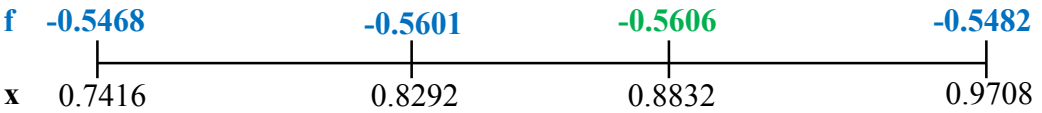
- Itération 3



- Itération 4



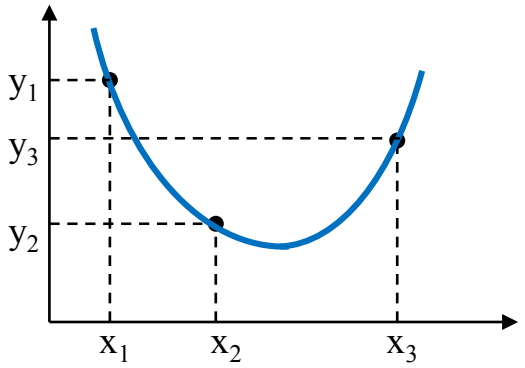
- Itération 5



- Solution : $x^* \approx 0.8832 \rightarrow f(x^*) \approx -0.5606$
  au lieu de $x^* \approx 0.8603 \rightarrow f(x^*) \approx -0.5611$

2.3.3 Minimisation exacte

Interpolation quadratique



- On connaît la valeur de la fonction f en 3 points x_1, x_2, x_3.

$$\begin{cases} y_1 = f(x_1) \\ y_2 = f(x_2) \\ y_3 = f(x_3) \end{cases}$$

- On construit le polynôme q de degré 2 passant par les 3 points.

$$q(x) = y_1 \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} + y_2 \frac{(x - x_3)(x - x_1)}{(x_2 - x_3)(x_2 - x_1)} + y_3 \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)} \rightarrow \begin{cases} q(x_1) = y_1 \\ q(x_2) = y_2 \\ q(x_3) = y_3 \end{cases}$$

- Dérivée du polynôme q

$$q'(x) = y_1 \frac{(2x - x_2 - x_3)}{(x_1 - x_2)(x_1 - x_3)} + y_2 \frac{(2x - x_3 - x_1)}{(x_2 - x_3)(x_2 - x_1)} + y_3 \frac{(2x - x_1 - x_2)}{(x_3 - x_1)(x_3 - x_2)}$$

On obtient une approximation du minimum de f en minimisant q.

2.3.3 Minimisation exacte

Interpolation quadratique

- La dérivée du polynôme q s'écrit :

$$q'(x) = y_1 \frac{(2x - x_2 - x_3)}{(x_1 - x_2)(x_1 - x_3)} + y_2 \frac{(2x - x_3 - x_1)}{(x_2 - x_3)(x_2 - x_1)} + y_3 \frac{(2x - x_1 - x_2)}{(x_3 - x_1)(x_3 - x_2)}$$

$$\Rightarrow \quad q'(x) = \frac{-2x\left[y_1(x_2 - x_3) + y_2(x_3 - x_1) + y_3(x_1 - x_2)\right] + \left[y_1(x_2^2 - x_3^2) + y_2(x_3^2 - x_1^2) + y_3(x_1^2 - x_2^2)\right]}{(x_1 - x_2)(x_2 - x_3)(x_3 - x_1)}$$

$$\Rightarrow \quad q'(x) = \frac{-2x\left(y_1 s_{23} + y_2 s_{31} + y_3 s_{12}\right) + \left(y_1 r_{23} + y_2 r_{31} + y_3 r_{12}\right)}{(x_1 - x_2)(x_2 - x_3)(x_3 - x_1)} \quad \text{avec} \quad \begin{cases} s_{ij} = x_i - x_j \\ r_{ij} = x_i^2 - x_j^2 \end{cases}$$

- On cherche la valeur x_m qui annule la dérivée du polynôme q.

$$q'(x_m) = 0 \quad \Rightarrow \quad 2x_m\left[y_1 s_{23} + y_2 s_{31} + y_3 s_{12}\right] + \left[y_1 r_{23} + y_2 r_{31} + y_3 r_{12}\right] = 0$$

$$\Rightarrow \boxed{x_m = \frac{1}{2} \frac{y_1 r_{23} + y_2 r_{31} + y_3 r_{12}}{y_1 s_{23} + y_2 s_{31} + y_3 s_{12}} \quad \text{avec} \quad \begin{cases} s_{ij} = x_i - x_j \\ r_{ij} = x_i^2 - x_j^2 \end{cases}}$$

2.3.3 Minimisation exacte

Interpolation quadratique

- On évalue la valeur de la fonction en $x_m \rightarrow y_m = f(x_m)$
  → On dispose de 4 points x_1, x_2, x_3, x_m avec les valeurs respectives de f : y_1, y_2, y_3, y_m
  → On conserve 3 points choisis parmi x_1, x_2, x_3, x_m selon :
  - la position de x_m par rapport à x_1, x_2, x_3
  - le minimum parmi y_m, y_1, y_2, y_3

Points retenus	$x_m < x_1$	$x_1 < x_m < x_2$	$x_2 < x_m < x_3$	$x_3 < x_m$
Minimum = y_m	(x_m,x_1,x_2)	(x_1,x_m,x_2)	(x_2,x_m,x_3)	(x_2,x_3,x_m)
Minimum = y_1	(x_m,x_1,x_2)	(x_1,x_m,x_2)	(x_1,x_2,x_m)	divergence
Minimum = y_2	divergence	(x_m,x_2,x_3)	(x_1,x_2,x_m)	divergence
Minimum = y_3	divergence	(x_m,x_2,x_3)	(x_2,x_m,x_3)	(x_2,x_3,x_m)

Cas de divergence : Le polynôme est trop éloigné de la fonction
Il faut un balayage plus fin avant l'interpolation quadratique.

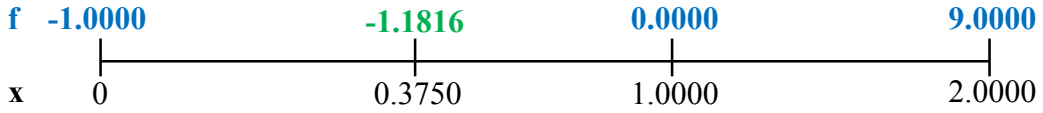
- On réitère l'interpolation quadratique avec les 3 nouveaux points jusqu'à réduire la taille de l'intervalle à la précision souhaitée.

2.3.3 Exemple

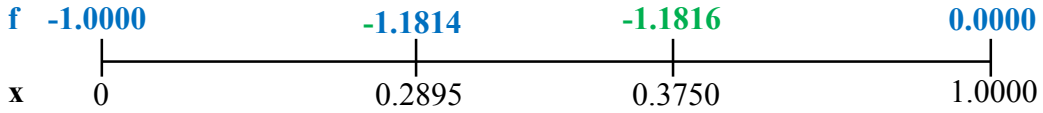
Interpolation quadratique

- Minimisation de $f(x) = (x-1)(x+1)^2$, $0 \le x \le 2$

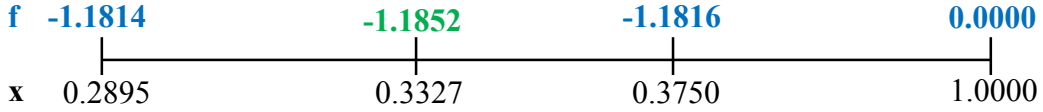
- Itération 1



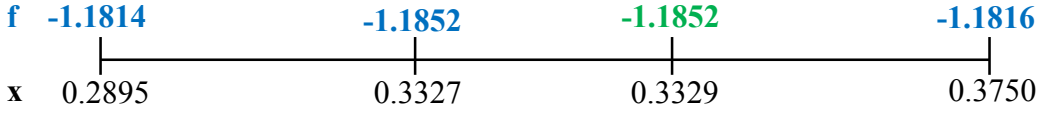
- Itération 2



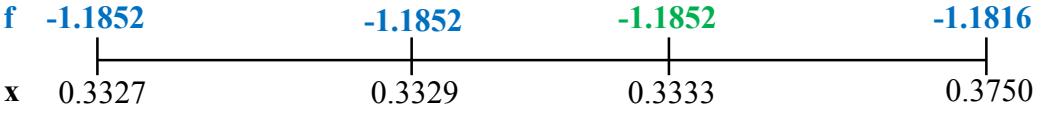
- Itération 3



- Itération 4



- Itération 5



- Solution : $x^* \approx 0.3333 \rightarrow f(x^*) \approx -1.1852$

2.3.3 Minimisation approchée

Principes

- Il n'est pas utile de réaliser une minimisation exacte suivant la direction de descente :
  → nécessite un grand nombre d'évaluations de la fonction
  → n'apporte pas une amélioration significative loin de la solution
- On peut se contenter d'une minimisation approchée
  → 2 **règles d'acceptation** d'un pas de déplacement

Notations

- x_k = point courant → $f(x_k)$
- d_k = direction de descente → $\nabla f(x_k)^T d_k < 0$

- Variation de la fonction f dans la direction d_k : $\varphi(s) = f(x_k + s d_k),\ s \geq 0$

Règles d'acceptation du pas

- **Diminution suffisante** de la valeur de la fonction → condition d'Armijo
  1ère condition de Wolfe

- **Déplacement suffisant** par rapport au point initial → condition de Goldstein
  2ème condition de Wolfe

2.3.3 Minimisation approchée

Diminution suffisante

La fonction f doit décroître suffisamment pour accepter le pas.

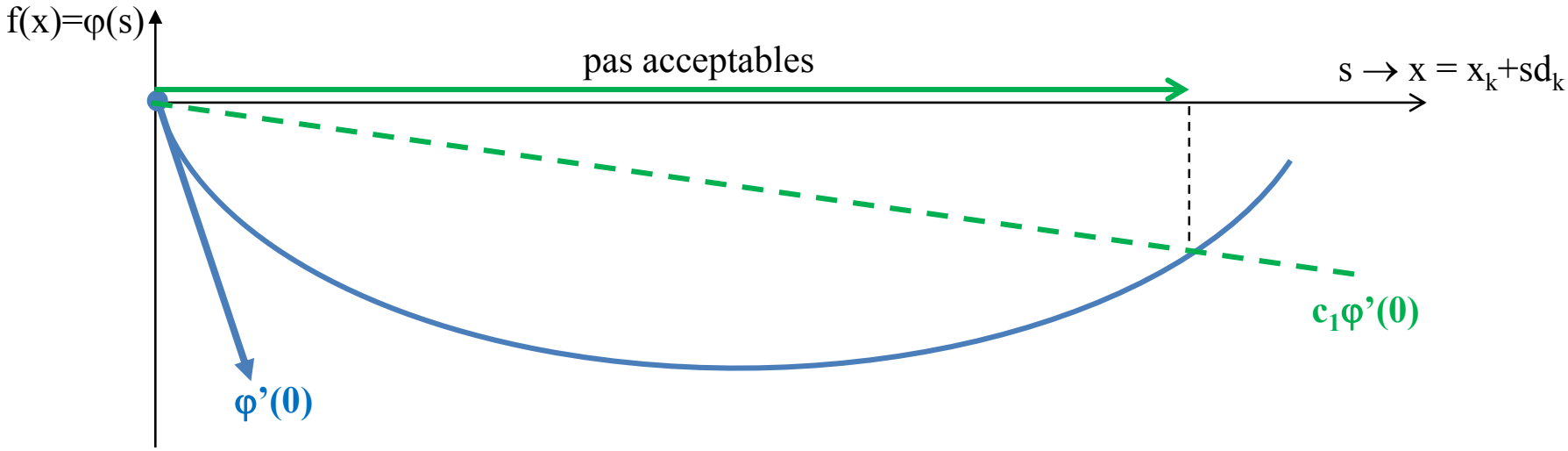
- Dérivée directionnelle de f suivant la direction d_k

$$\varphi(s) = f(x_k + sd_k),\ s \geq 0 \quad \Rightarrow \quad \varphi'(0) = \nabla f(x_k)^T d_k < 0$$

- On impose une diminution proportionnelle à la dérivée directionnelle, avec $0 < \varepsilon < c_1 < 0.5$

$$\varphi(s) < \varphi(0) + c_1 s\varphi'(0) \quad \Leftrightarrow \quad \boxed{f(x_k + sd_k) < f(x_k) + c_1 s\nabla f(x_k)^T d_k} \quad \rightarrow \text{ valeur typique } c_1 = 0.1$$

→ **Condition d'Armijo** ou 1ère condition de Wolfe ou 1ère condition de Goldstein



2.3.3 Minimisation approchée

Déplacement suffisant

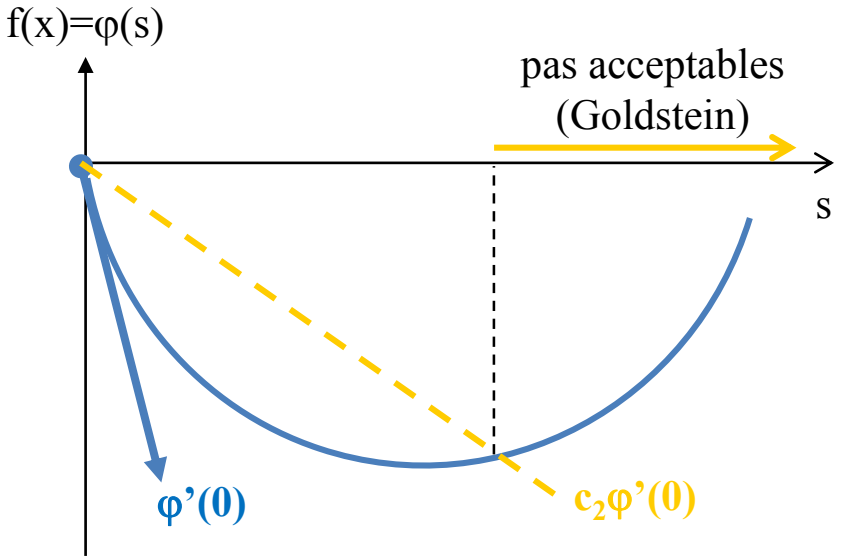
Le déplacement à partir du point initial doit être suffisant pour accepter le pas.

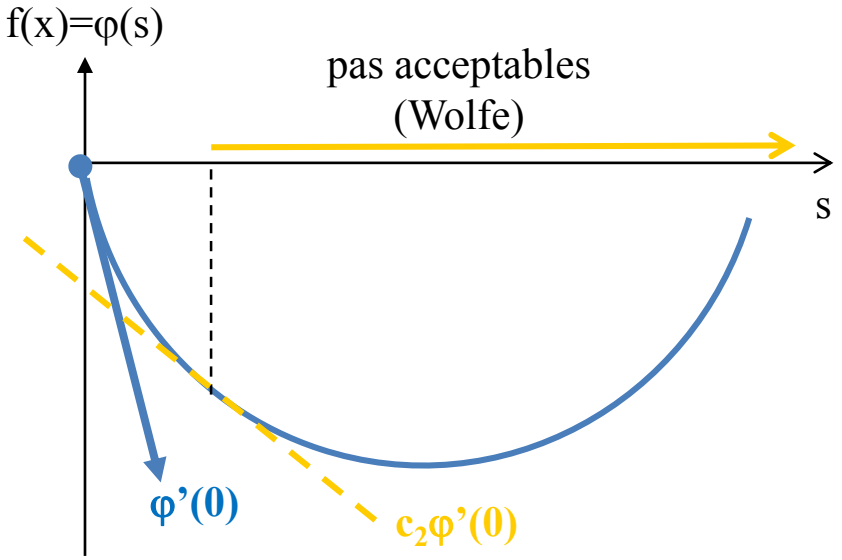
- Condition de Goldstein

$$\varphi(s) > \varphi(0) + c_2 s\varphi'(0) \quad \Leftrightarrow \quad \boxed{f(x_k + sd_k) > f(x_k) + c_2 s\nabla f(x_k)^T d_k} \quad \rightarrow \text{ valeur typique } c_2=0.9$$

- Condition de Wolfe : réduction de la dérivée

$$\varphi'(s) > c_2\varphi'(0) \quad \Leftrightarrow \quad \boxed{\nabla f(x_k + sd_k)^T d_k > c_2 \nabla f(x_k)^T d_k} \quad \rightarrow \text{ valeur typique } c_2=0.9$$





2.3.3 Minimisation approchée

Récapitulatif

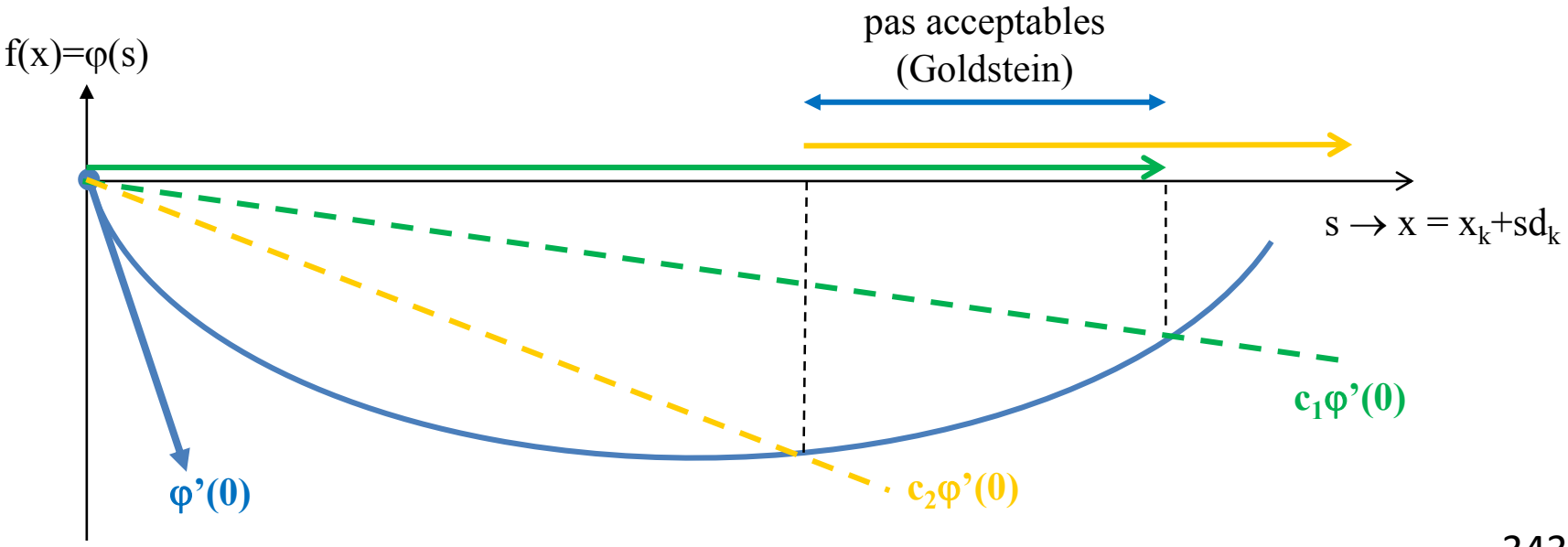
- **Conditions de Goldstein**
  (si la dérivée de f est coûteuse)

$$\begin{cases} \varphi(s) < \varphi(0) + c_1 s\varphi'(0) & \rightarrow \ c_1 \approx 0.1 \\ \varphi(s) > \varphi(0) + c_2 s\varphi'(0) & \rightarrow \ c_2 \approx 0.9 \end{cases}$$

- **Conditions de Wolfe**
  (si la dérivée de f est disponible)

$$\begin{cases} \varphi(s) < \varphi(0) + c_1 s\varphi'(0) & \rightarrow \ c_1 \approx 0.1 \\ \varphi'(s) > c_2\varphi'(0) & \rightarrow \ c_2 \approx 0.9 \end{cases}$$

ou $|\varphi'(s)| < c_2|\varphi'(0)|$ → assure théoriquement la convergence



2.3.3 Réglage du pas

Méthode de dichotomie

On cherche un pas s vérifiant les conditions de Goldstein : $\begin{cases} \varphi(s) < \varphi(0) + c_1 s\varphi'(0) \\ \varphi(s) > \varphi(0) + c_2 s\varphi'(0) \end{cases}$

Initialisation

- Intervalle initial : $[s_{min}, s_{max}]$ avec $s_{min} = 0$
  s_{max} assez grand
- Valeur initiale : $s=1$ (pas de Newton)

Itérations

Evaluation de $\varphi(s)$ et comparaison aux droites de Goldstein

- Si s ne respecte pas la condition 1 → amélioration insuffisante, pas trop grand : $s_{max}=s$
- Si s ne respecte pas la condition 2 → déplacement insuffisant, pas trop petit: $s_{min}=s$

→ Réduction de l'intervalle $[s_{min}, s_{max}]$

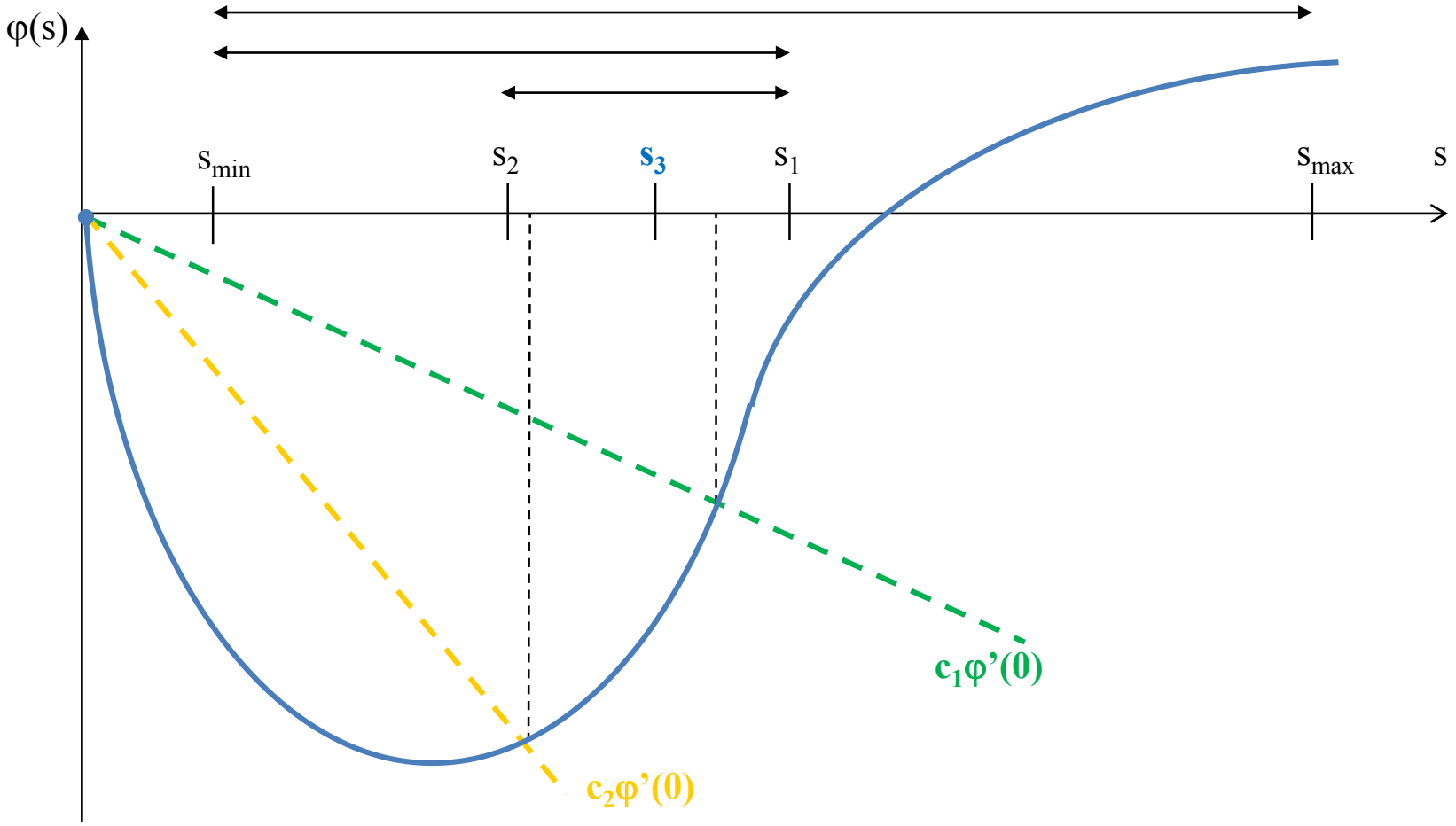
→ Essai suivant : $s = \frac{1}{2}(s_{min} + s_{max})$

Arrêt

- Si s respecte les 2 conditions → pas acceptable
- Si l'intervalle $[s_{min}, s_{max}]$ devient inférieur à un seuil donné

2.3.3 Réglage du pas

Illustration

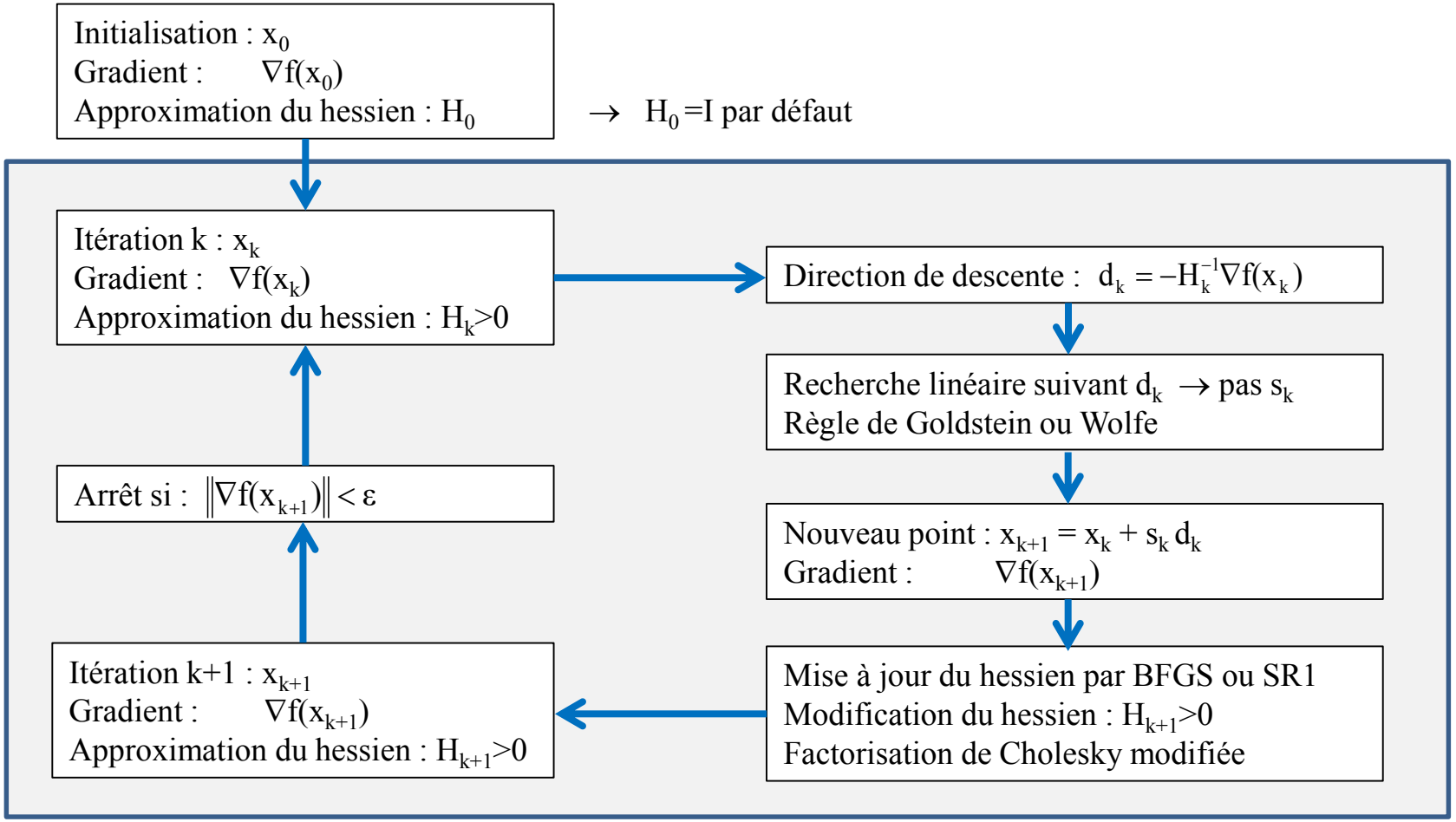


2.3.4 Algorithme

- Algorithme de recherche linéaire
- Convergence
- Exemple
- Descente non monotone

2.3.4 Algorithme

Algorithme de recherche linéaire



2.3.4 Algorithme

Principaux résultats de convergence

- Si d est une direction de descente, et f est bornée inférieurement suivant d, alors il existe un pas s suivant d vérifiant les conditions de Wolfe

- Si les directions de descente ne deviennent pas « trop » orthogonales au gradient, l'algorithme de recherche linéaire avec les **conditions de Wolfe** est **globalement convergent**.
  $\lim_{k\to\infty} \|\nabla f(x_k)\| = 0$ $\rightarrow$ convergence vers un point stationnaire

En pratique

- Les directions de descente peuvent être construites avec BFGS ou SR1 (si matrices $H_k > 0$)

- La valeur des coefficients de Goldstein ou Wolfe c_1 et c_2 n'est pas critique pour la convergence.

- Le pas de Newton (s=1) est systématiquement testé, car il donne la solution si la fonction est proche de son modèle quadratique.

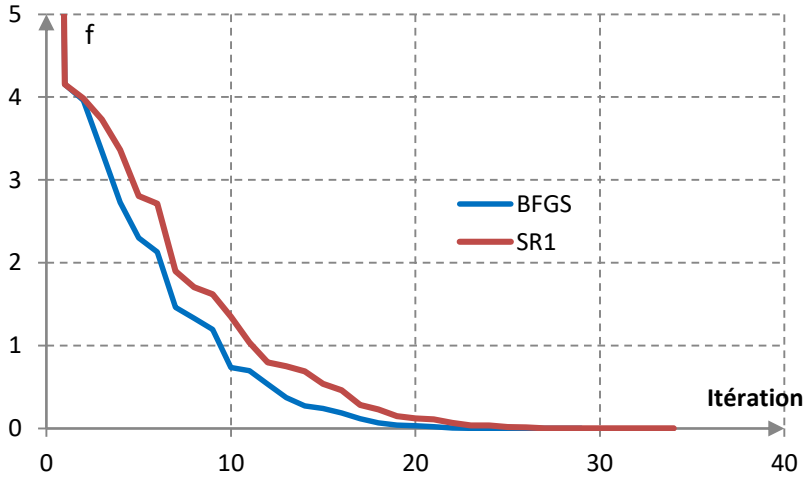
- La méthode de plus forte pente a la propriété de convergence globale, mais peut être très lente. On peut assurer la convergence globale d'un algorithme de recherche linéaire utilisant d'autres directions **en effectuant périodiquement une itération de plus forte pente**.

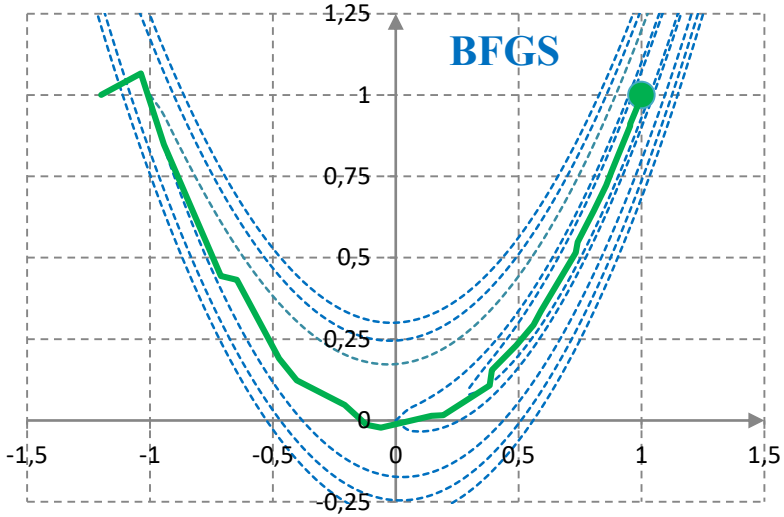
2.3.4 Exemple

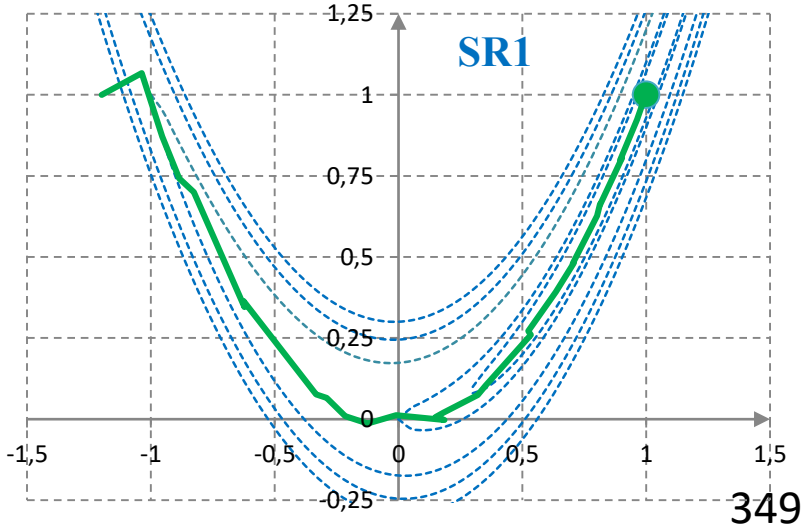
Fonction de Rosenbrock

$$f(x_1, x_2) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2$$

- Point initial : $\begin{pmatrix} -1.2 \\ 1 \end{pmatrix}$
- Recherche linéaire avec BFGS ou SR1







2.3.4 Descente non monotone

Condition de décroissance

- La condition d'Armijo impose une décroissance de la fonction **à chaque itération**.

$$f(x_k + s d_k) < f(x_k) + c_1 s \nabla f(x_k)^T d_k$$

Le pas de déplacement s est réduit jusqu'à vérifier la condition de décroissance.
→ **décroissance monotone**

$$f(x_0) > f(x_1) > f(x_2) > \cdots > f(x_k) > f(x_{k+1}) > \cdots$$

La convergence peut devenir très lente sur une fonction mal conditionnée (« vallée étroite »).

- On peut accélérer la convergence en acceptant une croissance temporaire de la fonction.
La condition de décroissance est imposée **entre le point x_{k-m} et le point x_{k+1}**.

$$f(x_k + s d_k) < f(x_{k-m}) + c_1 s \nabla f(x_k)^T d_k$$

La fonction peut croître sur m itérations, mais les sous-suites $x_{j+k(m+1)}$ restent décroissantes.
→ **décroissance non monotone** (méthode « watchdog »)

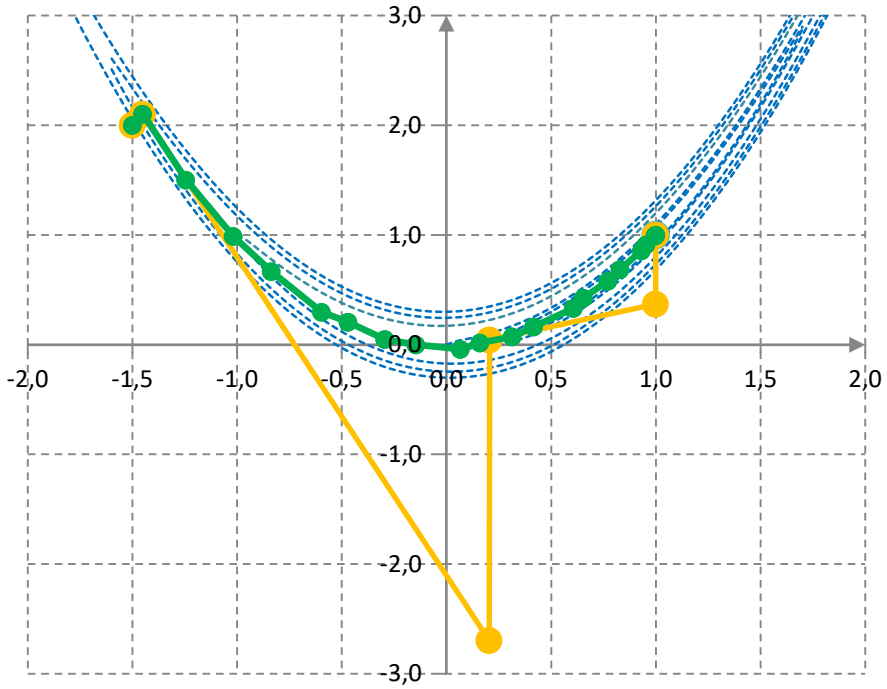
$$f(x_0) > f(x_{m+1}) > f(x_{2(m+1)}) > \cdots > f(x_{k(m+1)})$$

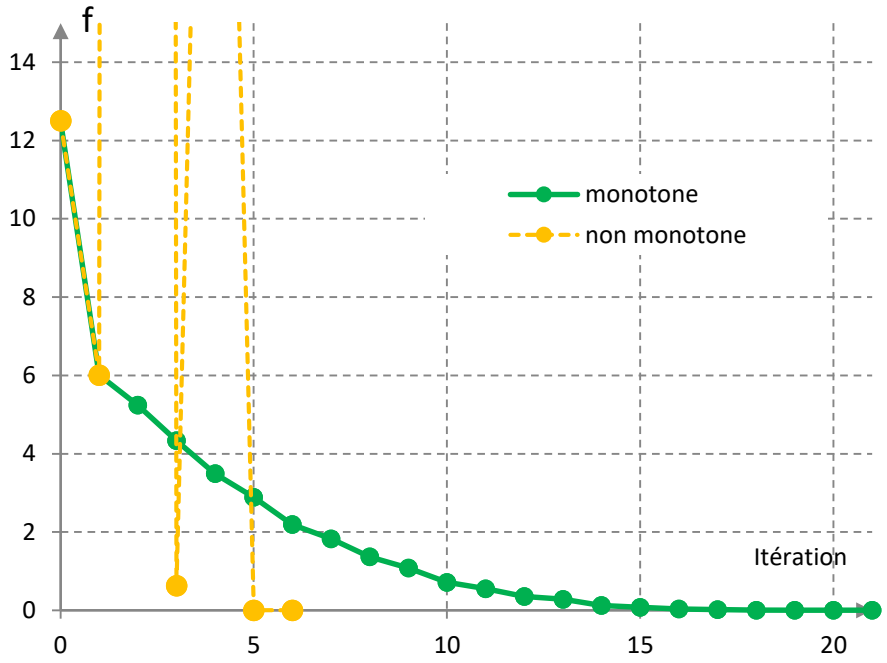
$$f(x_1) > f(x_{1+m+1}) > f(x_{1+2(m+1)}) > \cdots > f(x_{1+k(m+1)}) \quad \text{sans imposer} \quad f(x_1) < f(x_0)$$

2.3.4 Descente non monotone

Exemple

Fonction de Rosenbrock à deux variables : $f(x_1, x_2) = 100\left(x_2 - x_1^2\right)^2 + \left(1 - x_1\right)^2$





Méthode de Newton avec pas s

- Descente monotone (s → s/2) : convergence en 21 itérations
- Descente non monotone (s=1) : convergence en 6 itérations (2 itérations avec dégradation)