

# Equations non linéaires

## Recherche de racines



A la fin du chapitre, l'étudiant doit **être capable de** :

1. Définir l'algorithme et donner la condition de convergence des méthodes de substitution, de dichotomie, de la corde, de la fausse position et de Newton
2. Définir la méthode de Newton-Raphson en plusieurs dimensions
3. Calculer théoriquement et mesurer numériquement l'ordre de convergence d'une méthode itérative,
4. Reconnaître dans les scripts Matlab fournis les différentes étapes des méthodes étudiées
5. Déterminer si un intervalle contient une racine au moins
6. Coder dans Matlab un programme permettant de visualiser une fonction quelconque d'une variable

# Equations non linéaires

## Recherche de racines



### I - INTRODUCTION

Linéaire/non linéaire : *équation* :  $ax + b = 0$  ;  $r = -b/a$  ;

*système* :  $[A][X] + [B] = [0]$  ;  $[X] = -[A^{-1}][B]$

#### Approche analytique

Ex:  $ax^2 + bx + c = 0$

$$\Delta = b^2 - 4ac$$

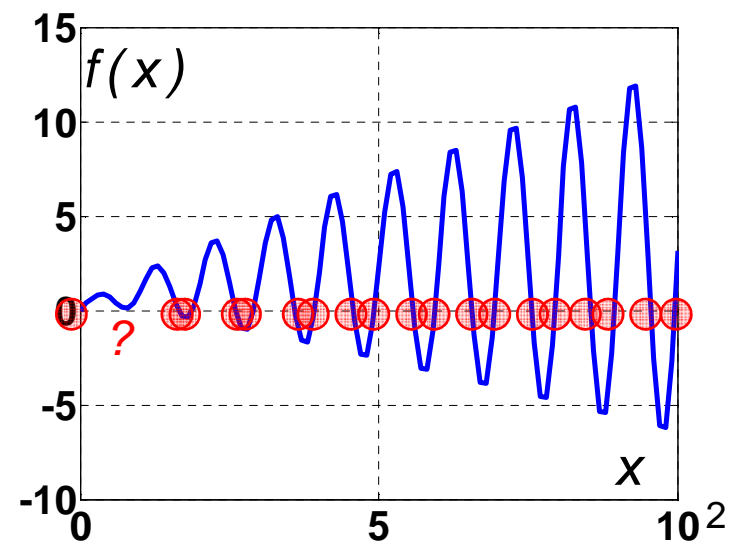
$\Delta < 0$  pas de solution réelle

$\Delta = 0$  racine double  $r = -b/2a$

$\Delta > 0$  2 solutions,  $r = \frac{-b \pm \sqrt{\Delta}}{2a}$

#### Approche numérique

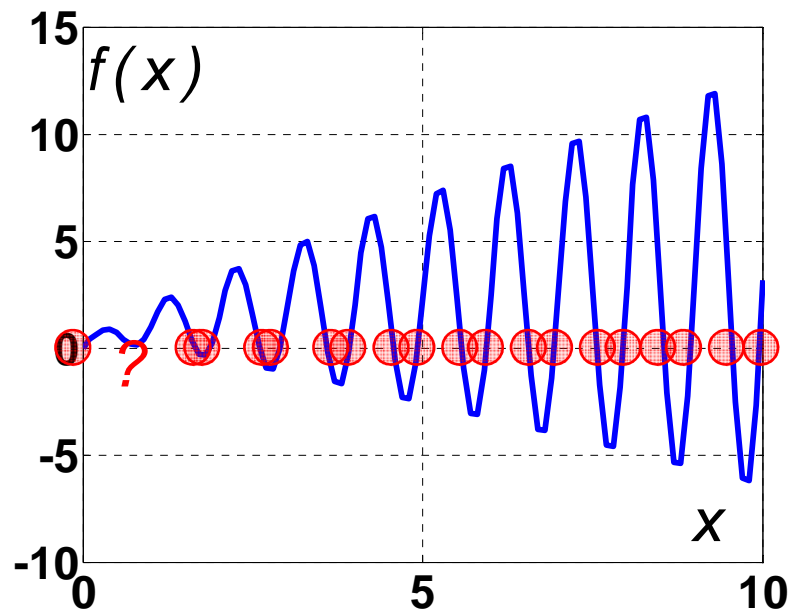
Ex:  $\sqrt{x} + x \sin(2\pi x) = 0$  ;  $x \geq 0$



## Programme MATLAB®



```
%Chapitre 1 : exemple_1
clear all
close all
x=0:0.1:10;
f=x.^0.5+x.*sin(2*pi*x);
plot(x,f)
grid on
```



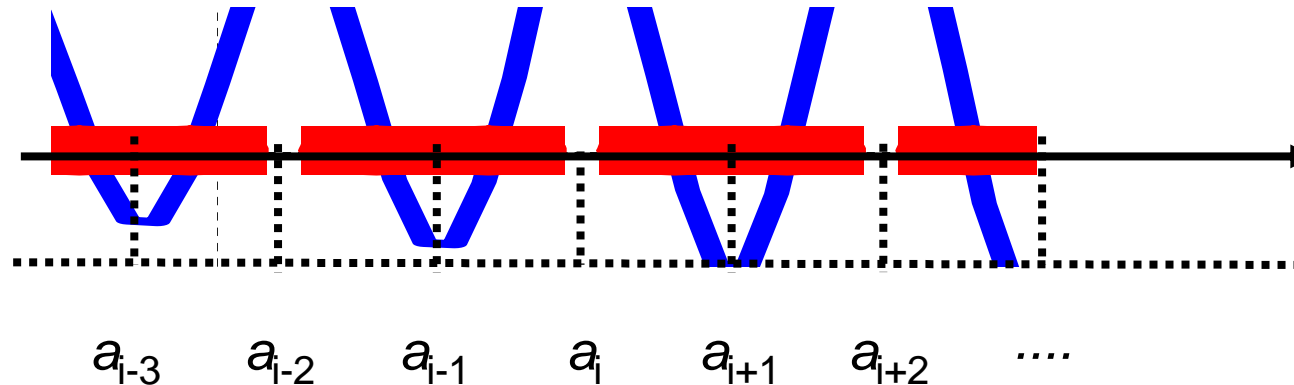
1- Racines multiples

2- Min. positif / Max. négatif

3- Pentes fortes / faibles



## I.1- Séparation des racines



Séparer la racine  $r_i$  : trouver un intervalle  $]a_i, a_{i+1}[$  où la racine est unique.

Beaucoup de méthodes ne « marchent » que sur un intervalle où la racine est unique (séparée)



Procédure automatique de séparation :  $p_i = f(a_i) \cdot f(a_{i+1})$

Si  $p_i < 0$ , il existe  $2p+1$  racines dans  $]a_i, a_{i+1}[$  (\*)

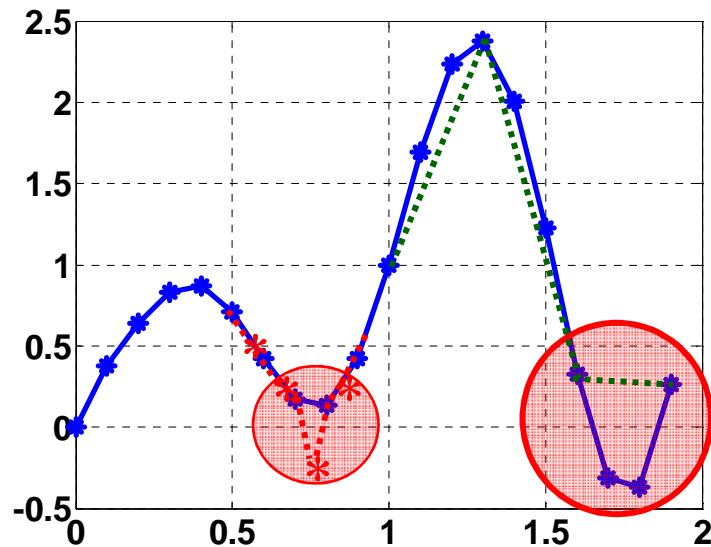
Si  $p_i > 0$ , il existe  $2p$  racines dans  $]a_i, a_{i+1}[$  (\*)

Si  $p_i = 0$ , bingo!  $a_i$  ou  $a_{i+1}$  racine.

(\*) en tenant compte l'ordre de multiplicité des racines



## I.2- Minimum positif (/max<0)



- Pas de discrétisation  
maillage initial  
maillage grossier  
maillage fin

- tracé graphique : option par défaut  
« interpolation linéaire »

**Rappel :**  $f(x_0 + h) = f(x_0) + f'(x_0).h + f''(x_0).\frac{h^2}{2!} + \dots + f^{(p)}(x_0).\frac{h^p}{p!} + \dots$

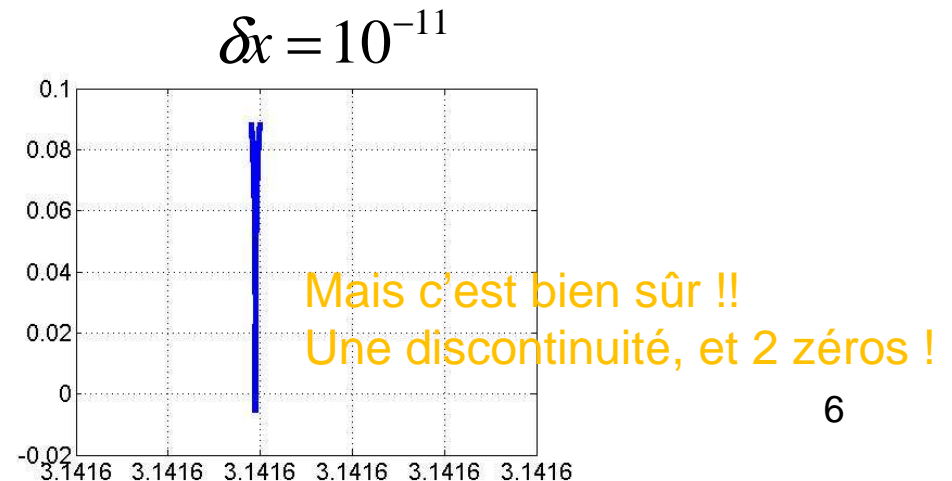
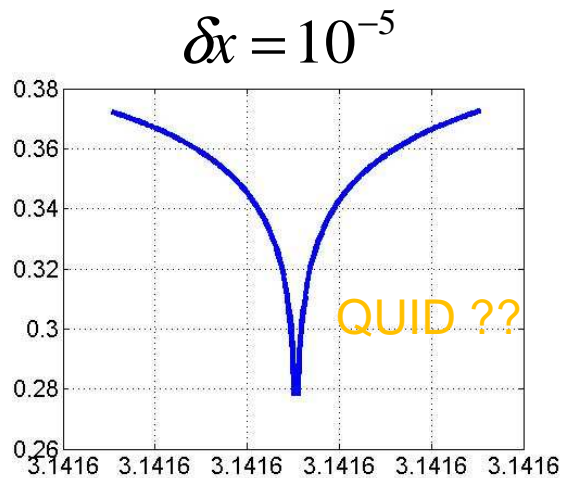
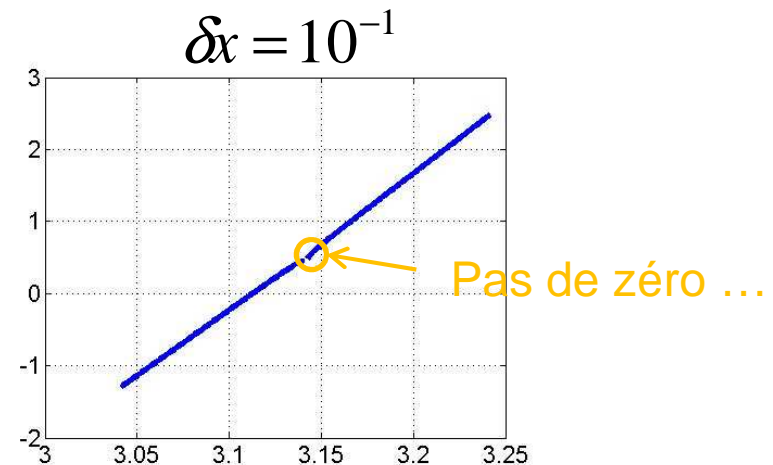
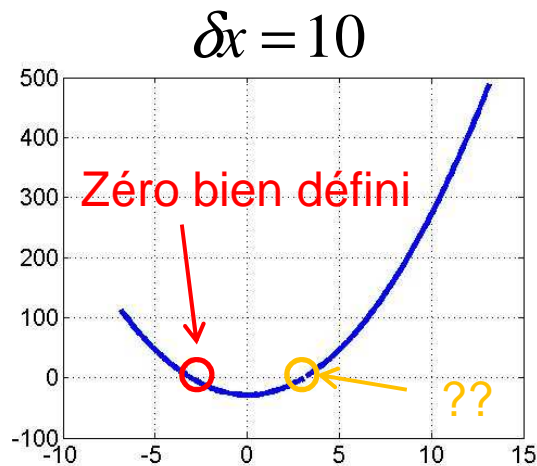
pour:  $x_i \leq x \leq x_{i+1}$ ,  $f(x) \approx \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i) + f(x_i)$

*Interpolation linéaire d'autant moins valable que la courbe est « chahutée », i.e. que les dérivées d'ordre supérieur sont importantes.*

### I.3- Exemple



$f(x) = 3x^2 + \frac{1}{\pi^4} \ln[(\pi - x)^2] - 29$  tracée sur  $[\pi - \delta x, \pi + \delta x]$  avec 100 points



## II –METHODE DE SUBSTITUTION



### II.1 –Condition d'utilisation

Méthode applicable si on peut mettre l'équation  $f(x)=0$  sous la forme

$$x = g(x)$$

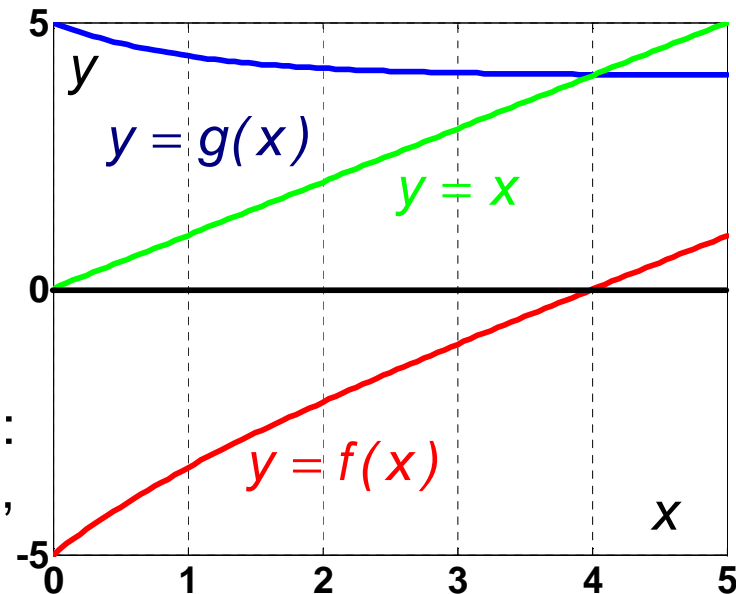
avec  $g$  fonction continue.

Exemple :

$$\begin{cases} f(x) = x - e^{-x} - 4 \\ g(x) = e^{-x} + 4 \end{cases}$$

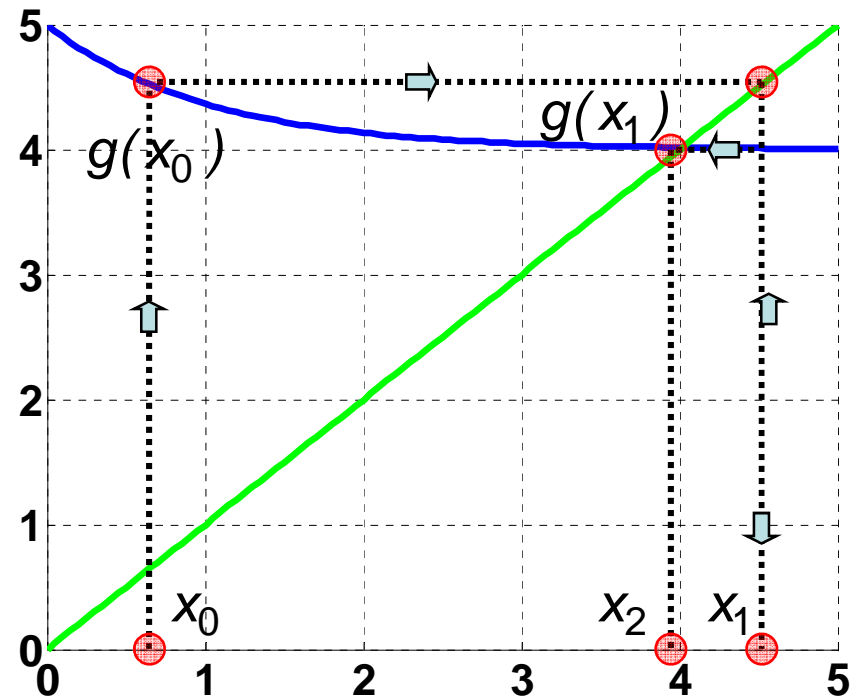
### II.2 – Principe

On construit une suite récurrente du type :  $x_{i+1} = g(x_i)$ . Si, partant d'une valeur initiale, la suite de  $x_0, x_1, \dots, x_i, \dots$ , converge vers une limite  $r$ , alors  $r$  vérifie  $g(r) = r$  et  $r$  s'identifie à la racine de  $f$ .



## Programme MATLAB

```
%Chapitre 1 : exemple fig2
clear all
close all
x=0:0.05:5
f=x-exp(-x)-4;
g=exp(-x)+4;
%plot(x,f,'r-')
hold on
plot(x,g,'b-')
plot(x,x,'g')
grid on
```



### II.3 – Condition suffisante de convergence

Si, sur un intervalle  $[a,b]$ , entourant la racine  $r$ , on peut trouver un majorant  $M < 1$  de la valeur absolue de la dérivée  $|g'(x)|$ , la suite des  $x_n$  est convergente.



## II.5 – Etude de la C. S. de convergence

$$x_{n+1} = g(x_n)$$

À la limite  $r = g(r)$   $r$  : racine, **point fixe**

**C.S. : S'il existe un intervalle autour de  $r$  tel que  $|g'| \leq M < 1$ , alors la série  $x_n$  converge vers  $r$ .**

*théorème des accroissements finis :*

$\exists t, t \in ]x, r[$  telque  $g(x) - g(r) = g'(t)(x - r)$

Comme  $g(r) = r$  et  $x_n = g(x_{n-1})$ ,

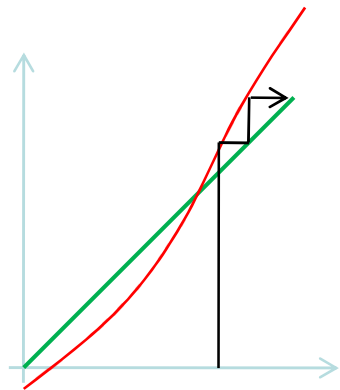
$$\begin{aligned} |x_n - r| &= |g(x_{n-1}) - g(r)| = |g'(t)| |x_{n-1} - r| \leq M |x_{n-1} - r| \\ &\leq M^2 |x_{n-2} - r| \leq \dots \leq M^n |x_0 - r| \end{aligned}$$

Comme  $M < 1$ ,  $\lim_{n \rightarrow \infty} |x_n - r| = 0$  ■

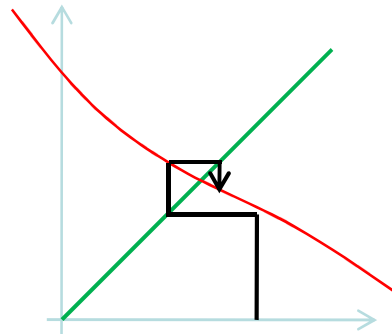


## II.6 – Remarques

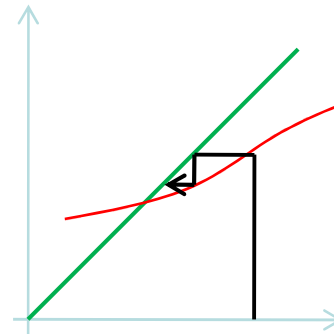
1- Ce qui peut se passer ...



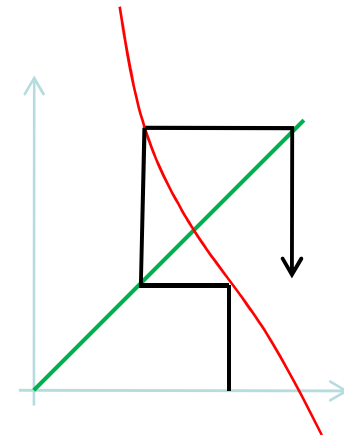
**INSTABLE**



**STABLE**



**STABLE**



**INSTABLE**

2- La méthode peut s'appliquer aux fonctions non dérivables; l'important est que  $g$  soit contractante

$$\exists M < 1 : \forall (x, y) \in [a, b]^2, |g(x) - g(y)| \leq M |x - y|$$



## II.7 – Calcul de l'ordre de la méthode de substitution

$$x_n = r + \varepsilon_n \quad \varepsilon_n : \text{erreur au pas } n$$

Un développement limité de  $x_{n+1}$  donne :

$$x_{n+1} = g(x_n) = g(r) + g'(r)(x_n - r) + \frac{g''(r)}{2!}(x_n - r)^2 + \dots$$

$$x_{n+1} - g(r) = x_{n+1} - r = g'(r)(x_n - r) + \frac{g''(r)}{2!}(x_n - r)^2 + \dots$$

$$\varepsilon_{n+1} = g'(r)\varepsilon_n + \frac{g''(r)}{2!}\varepsilon_n^2 + \dots$$

L'ordre d'une méthode itérative donne une mesure de sa vitesse de convergence.

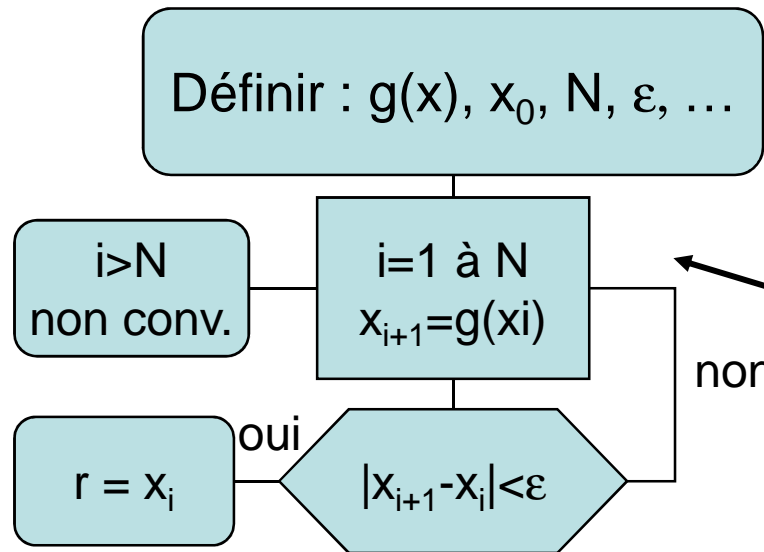
Au final, en négligeant les termes d'ordre supérieur :

$$|\varepsilon_{n+1}| = |g'(r)| \times |\varepsilon_n|^p, \text{ avec } p = 1$$

Le coefficient **p** est l'**ordre** de la méthode. Ici on montre donc que la méthode de substitution a une **convergence d'ordre 1** ou bien qu'elle a une convergence **linéaire** ...



## II.4 – Mise en œuvre de la méthode



**Méthode itérative :**

Test d'arrêt

Pente forte, Pente faible

**Programme : substi.m (cf. TD)**

$$\begin{cases} f(x) = x - e^{-x} - 4 \\ g(x) = e^{-x} + 4 \end{cases}$$

la racine vaut : 4.018

la valeur de f(racine) vaut : 6.7046e-008

le nombre d itérations : 6



### III – METHODE DE DICHOTOMIE

*Tomie* : vient de couper

*Dicho* : ...en deux

#### III.1 – Conditions d'utilisation

Equation quelconque, il faut juste être sûr que la racine  $r$  a été « séparée » sur  $[a, b]$ .

#### III.2 – Principe de la méthode

L'idée est de prendre le milieu  $c$  de l'intervalle  $[a, b]$  et de voir dans quel sous intervalle  $[a, c]$  ou  $[c, b]$  se trouve la racine cherchée. Une fois cet intervalle repéré, on a un nouvel encadrement de la racine. On recommence alors l'opération...



### III.3 – Mise en oeuvre

racine séparée sur  $[a,b]$ ,

déclarer la fonction  $f$ , le nombre d'itérations, ...

entrer :  $a_0 = a$  et  $b_0 = b$ ,

vérifier que  $f(a_0).f(b_0) < 0$  , sinon problème (racine non séparée !)

calculer  $c_0 = (a_0 + b_0) / 2$  et  $f(c_0)$

introduire  $P_0 = f(a_0) \times f(c_0)$

tester : si  $P_0 > 0$  alors la racine est dans  $[c_0, b_0]$  et on pose  $a_1 = c_0$ ,  $b_1 = b_0$

sinon la racine est dans  $[a_0, c_0]$  et on pose  $a_1 = a_0$ ,  $b_1 = c_0$

calculer  $c_1 = (a_1 + b_1) / 2$  ....et on continue jusqu'au....

test d'arrêt : taille de l'intervalle  $[a_n, b_n]$ , i.e.  $|b_n - a_n| < \epsilon$

si le test est positif, la racine vaut  $c_n$



### III.4 – Exemple

Programme : `dicho.m` (cf. TD)

$$\begin{cases} f(x) = x - e^{-x} - 4 \\ g(x) = e^{-x} + 4 \end{cases}$$

racine : 4.0696

valeur de  $f(\text{racine})$  : 2.9e-005

nombre d itérations : 16

- Comparaison des résultats obtenus avec `subst.m`

- Avec la méthode de dichotomie :

- certitude de convergence

- facile à programmer

- majoration de l'erreur sur la racine :  $|c_n - r| < \frac{|b_n - a_n|}{2} = \left(\frac{1}{2}\right)^n |b_0 - a_0|$

- méthode linéaire

- Méthode souvent lente : raison géométrique 1/2.



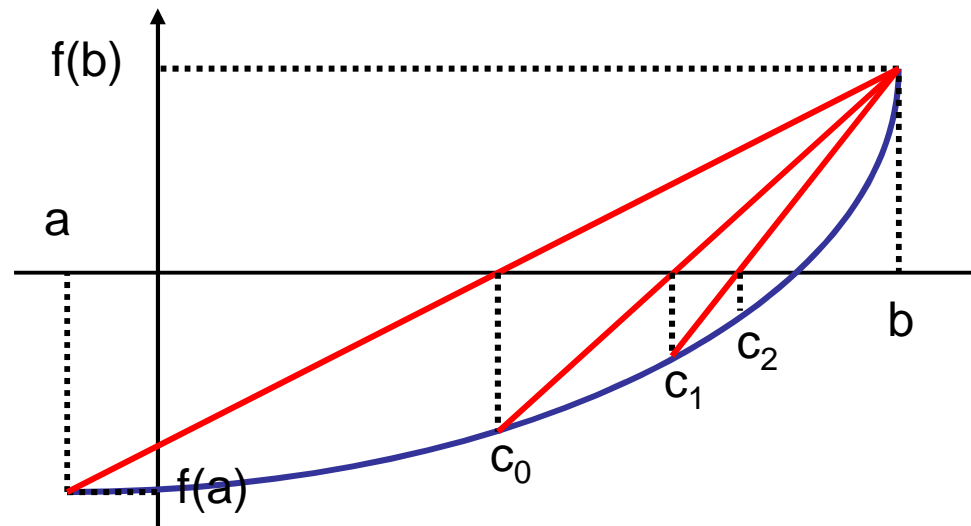
## IV – METHODE DE LA CORDE (LAGRANGE)

### IV.1 – Conditions d'utilisation

Une racine encadrée par deux valeurs  $a$  et  $b$ .

### IV.2 – Principe de la méthode

Le principe est identique au procédé de la dichotomie mais au lieu de prendre  $c$  milieu de  $[a, b]$ , on prend  $c$  intersection de la corde joignant les points  $A = (a, f(a))$ ,  $B = (b, f(b))$ .





### IV.3 – Mise en œuvre

programme corde.m

racine séparée sur  $[a,b]$ ,

déclarer la fonction  $f$ , le nombre d'itérations, ...

entrer :  $a_0 = a$  et  $b_0 = b$ ,

vérifier que  $f(a_0).f(b_0) < 0$  , sinon problème (racine non séparée !)

calculer  $c_0 = [a_0 f(b_0) - b_0 f(a_0)] / [f(b_0) - f(a_0)]$

introduire  $P_0 = f(a_0) \times f(c_0)$

tester : si  $P_0 > 0$  alors la racine est dans  $[c_0, b_0]$  et on pose  $a_1 = c_0$ ,  $b_1 = b_0$

sinon la racine est dans  $[a_0, c_0]$  et on pose  $a_1 = a_0$ ,  $b_1 = c_0$

calculer  $c_1 = [a_1 f(b_1) - b_1 f(a_1)] / [f(b_1) - f(a_1)]$  ....et on continue jusqu'au....

test d'arrêt : taille de l'intervalle  $[a_n, b_n]$ , *i.e.*  $|b_n - a_n| < \epsilon$

si le test est positif, la racine vaut  $c_n$



## II.7 – Calcul de l'ordre de la méthode de la corde

On part de :

$$c_n = a_n - \frac{b_n - a_n}{f(b_n) - f(a_n)} f(a_n) = b_n - \frac{b_n - a_n}{f(b_n) - f(a_n)} f(b_n), \text{ et } a_n < r < b_n$$

On en déduit :

$$c_n + \frac{b_n - a_n}{f(b_n) - f(a_n)} f(a_n) < r < c_n + \frac{b_n - a_n}{f(b_n) - f(a_n)} f(b_n)$$

Après quelques calculs:

$$|c_n - r| < \frac{\max(|f(a_n)|, |f(b_n)|)}{|f(b_n) - f(a_n)|} \times |b_n - a_n|$$

La méthode de la corde est linéaire également ...

## V – METHODES DE SCHRÖDER



### V.1 – Conditions d'utilisation

Recherche d'une racine unique dans un intervalle  $[a,b]$ .

Cette méthode itérative nécessite le calcul des  $n$  premières dérivées pour une méthode de Schröder d'ordre  $n+1$ . Il faut donc que la fonction  $f$  dont on cherche la racine soit au moins  $n$  fois dérivable dans l'intervalle  $[a,b]$ .

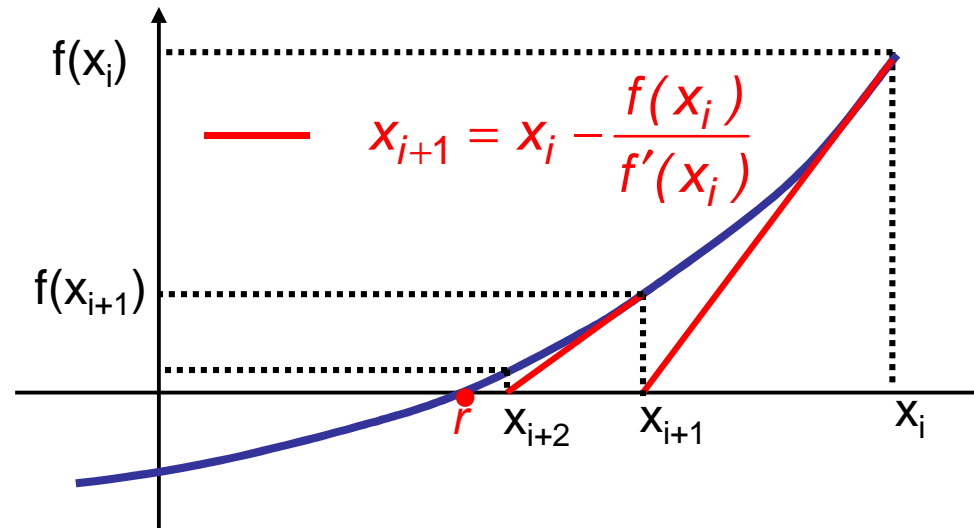
### V.2 – Principe de la méthode

On bâtit une suite récurrente  $x_n$  de la façon suivante :

- autour d'un point  $x_i$ , on effectue un développement limité de  $f(x)$  jusqu'à l'ordre  $m$ .
- on appelle  $x_{i+1}$  la racine du développement limité de  $f(x)$
- on itère ...



### V.3 – Illustration : la méthode de Newton



$x_{i+1}$  : racine du développement limité à l'ordre 1 de  $f$  au voisinage de  $x_i$ .

Méthode du 2nd ordre : on néglige les termes à partir du 2nd ordre

$$0 = f(x_{i+1}) = f(x_i) + f'(x_i) \cdot (x_{i+1} - x_i)$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$



## V.4 – Algorithmes pour l'ordre 2, 3 et 4 :

### Schröder d'ordre 2

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

...Le calcul numérique des dérivées successives peut devenir vite très coûteux (voie analytique / voie numérique)...

### Schröder d'ordre 3

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} - \frac{f''(x_i).f^2(x_i)}{2f'^3(x_i)}$$

### Schröder d'ordre 4

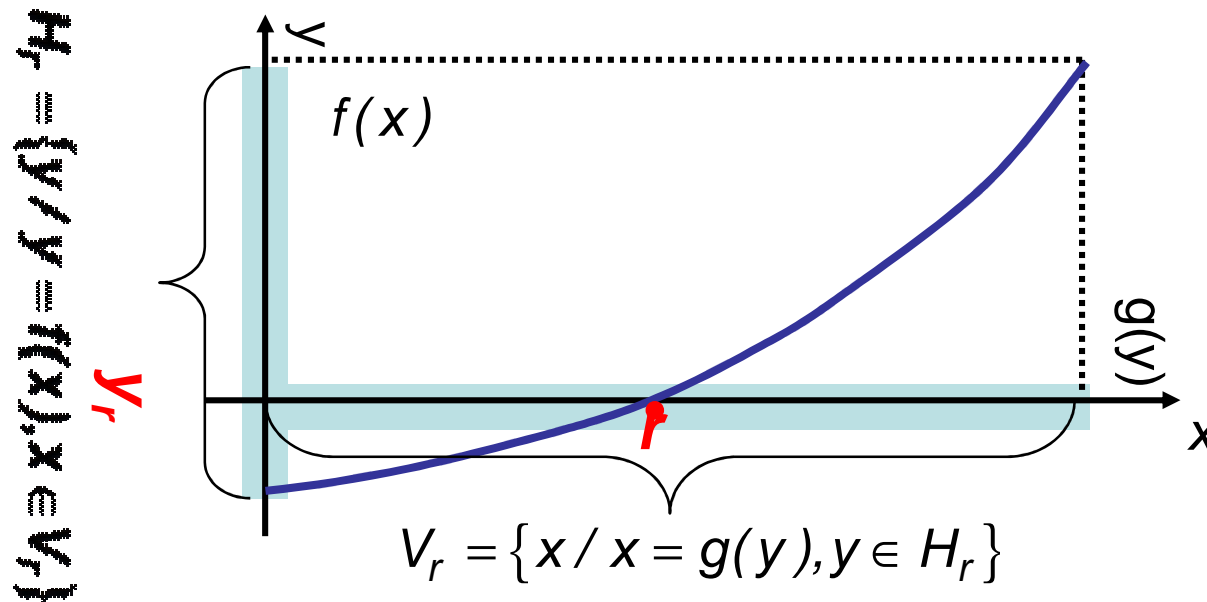
$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} - \frac{f''(x_i).f^2(x_i)}{2f'^3(x_i)} - \frac{3f''^2(x_i) - f'(x_i)f'''(x_i)}{6f'^5(x_i)}.f^3(x_i)$$

**Remarque :** la méthode de Schröder peut diverger si l'on part d'une valeur « trop » éloignée de la racine  $r$ .

Le nombre d'itérations est d'autant plus faible que l'ordre  $m$  de la méthode est élevé; en contrepartie, le temps de calcul d'une itération à l'autre peut être plus long.



## V.5 – Obtention de l'algorithme



La dérivée  $f'$  est non nulle dans un intervalle  $V_r$  entourant la racine.  
 La fonction  $f$  est donc monotone ; on pose  $g=f^{-1}$

$$g(y_r) = g(y) + (y_r - y) \cdot g'(y) + \frac{(y_r - y)^2}{2!} g''(y) + \dots + \frac{(y_r - y)^p}{p!} g^{(p)}(y) + \varepsilon_p(|y_r - y|^p)$$

$$\lim_{y_r \rightarrow y} \varepsilon_p(|y_r - y|^p) = 0$$



Or :  $g(y_r) = r, y_r = f(r) = 0$  et  $y = f(x)$

D'où l'expression analytique (exacte) de la racine :

$$r = x - f(x).g'(f(x)) + \frac{f^2(x)}{2!}g''(f(x)) - \frac{f^3(x)}{3!}g'''(f(x)) + \dots$$

Afin d'éliminer les dérivées successives de  $g$ , on remarque que :

$$g(f(x)) = x$$

$$g'(f(x))f'(x) = 1$$

$$g''(f(x))f'^2(x) + g'(f(x))f''(x) = 0$$

$$g'''(f(x))f'^3(x) + 3g''(f(x))f'(x)f''(x) + g'(f(x))f'''(x) = 0$$

En reportant l'expression des dérivées de  $g$  dans le DL, on obtient :

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} - \frac{f''(x_i).f^2(x_i)}{2f'^3(x_i)} - \frac{3f''^2(x_i) - f'(x_i)f'''(x_i)}{6f'^5(x_i)}.f^3(x_i) + \dots$$



## V.6 – Calcul de l'ordre de la méthode de Newton

On part de :

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \quad \text{avec} \quad \varepsilon_i = x_i - r \quad \text{et} \quad \varepsilon_{i+1} = x_{i+1} - r, \quad f(r) = 0$$

Les développements limités de  $f$  au voisinage de  $r$  donnent :

$$f(x_i) = f(r) + \varepsilon_i f'(r) + \frac{\varepsilon_i^2}{2} f''(r) + O(\varepsilon_i^3)$$

$$f'(x_i) = f'(r) + \varepsilon_i f''(r) + O(\varepsilon_i^2), \quad \text{avec} \quad f(r) = 0$$

Un simple calcul donne alors :

$$\varepsilon_{i+1} = \varepsilon_i - \frac{f(x_i)}{f'(x_i)} = \varepsilon_i^2 \frac{f''(r)}{2f'(r)} + O(\varepsilon_i^3)$$

On dit que la méthode de Newton a une **convergence d'ordre 2**  
ou convergence **quadratique** ...



## Calcul analytique des dérivées :

Parfois lourd

Parfois impossible

## Estimation numérique des dérivées :

Illustration avec la méthode de Newton (fausse position)

$$f'(x_i) \approx \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

$$x_{i+1} = \frac{x_{i-1}f(x_i) - x_i f(x_{i-1})}{f(x_i) - f(x_{i-1})}$$

Ressemble à la corde à ceci près que l'on ne cherche pas à encadrer la racine à l'aide des itérés successifs. Cela rend la méthode moins robuste

Changement de l'ordre !! Moins robuste mais plus efficace !!!

## V.7 – Calcul de l'ordre de la méthode de la fausse position



On part de :

$$x_{i+1} = x_i - \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} f(x_i)$$

A l'aide des développements limités de  $f$  au voisinage de  $r$ ,

$$f(x_i) = f(r) + \varepsilon_i f'(r) + \frac{\varepsilon_i^2}{2} f''(r) + O(\varepsilon_i^3)$$

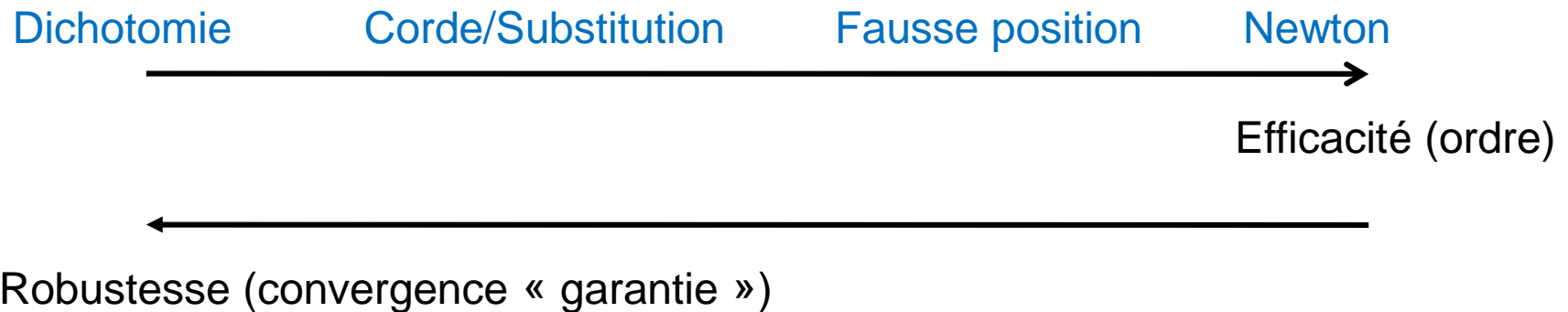
$$f(x_{i-1}) = f(r) + \varepsilon_{i-1} f'(r) + \frac{\varepsilon_{i-1}^2}{2} f''(r) + O(\varepsilon_{i-1}^3)$$

On obtient:  $\varepsilon_{i+1} \propto \varepsilon_i \varepsilon_{i-1}$  et on cherche  $p$  tel que  $\varepsilon_{i+1} \propto \varepsilon_i^p$ ,  $\varepsilon_i \propto \varepsilon_{i-1}^p$

L'ordre  $p$  vérifie alors la relation:  $\varepsilon_i^p \propto \varepsilon_i \varepsilon_i^{1/p}$ , soit  $p = 1 + 1/p$

La méthode est alors super-linéaire, d'ordre  $p = \frac{\sqrt{5} + 1}{2}$

## Petit récapitulatif ...



Peut-on allier efficacité (ordre  $> 1$ ) et robustesse ?

Oui, souvent, en 1D



## VI – UNE METHODE SUPER-LINEAIRE ROBUSTE ET SANS DERIVEE

### VI.1 – Conditions d'utilisation

Une racine encadrée par deux valeurs  $a$  et  $b$ .

### VI.2 – Méthode de Brent (73) – Ingrédients de base

1- déterminer la nouvelle estimation de la racine en réalisant une interpolation (polynôme de Lagrange) de la **fonction inverse**  $x=f^{-1}(y)$ . Nécessite la connaissance de trois couples  $A = (a, f(a))$ ,  $B = (b, f(b))$  et  $C = (c, f(c))$ :

$$f^{-1}(y) = \frac{(y - f(a))(y - f(b))}{(f(c) - f(a))(f(c) - f(b))} c + \frac{(y - f(a))(y - f(c))}{(f(b) - f(a))(f(b) - f(c))} b + \frac{(y - f(b))(y - f(c))}{(f(a) - f(b))(f(a) - f(c))} a$$

Connaissant  $a = x_{i-2}$ ,  $b = x_{i-1}$ ,  $c = x_i$ , l'itéré suivant est obtenu en fixant  $x_{i+1} = f^{-1}(0)$

On montre alors une convergence d'ordre 2,

2- Dans les cas où la nouvelle estimation sort de l'intervalle initial, ou lorsque la décroissance de l'intervalle est trop lente, on utilise une **simple dichotomie** pour garantir la robustesse et la convergence ...

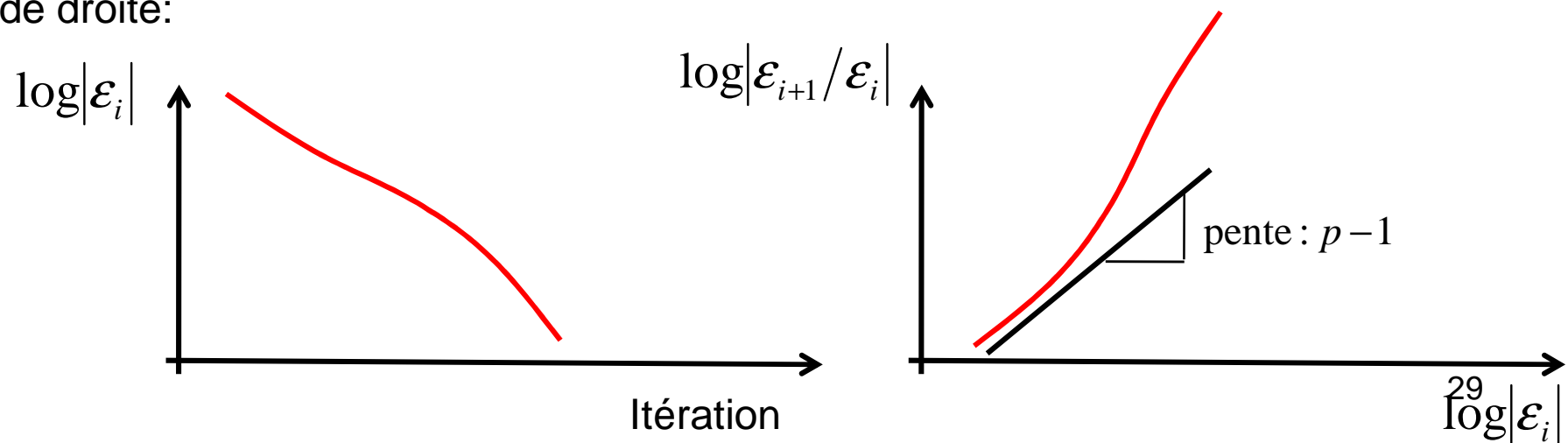
## SUIVI DE L'ERREUR - MESURE PRATIQUE DE L'ORDRE



Suivre l'évolution de l'**erreur** au cours du processus de convergence est **INDISPENSABLE** afin de détecter d'éventuels problèmes (saturation de la convergence, début de divergence, ...) ou s'assurer que l'on progresse bien vers la solution. On trace le plus souvent  $\log|\varepsilon_i|$  en cours des itérations.

L'ordre d'une méthode est obtenu de manière théorique sous certaines hypothèses (régularité de la fonction, petitesse de l'erreur,...). Il est parfois utile de mesurer l'ordre d'une méthode à partir d'une expérience numérique. Lorsque l'on utilise une méthode évoluée, cela permet notamment de **vérifier le bon codage** de celle-ci.

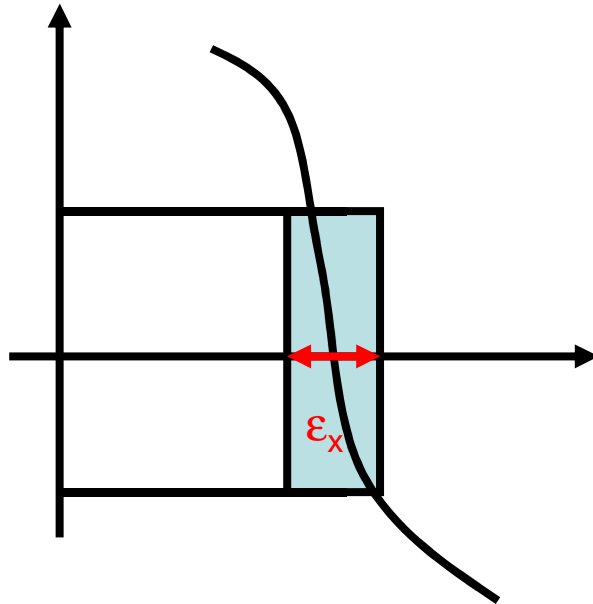
La mesure pratique de l'ordre peut se faire par exemple comme sur le graphe de droite:



## Remarques :

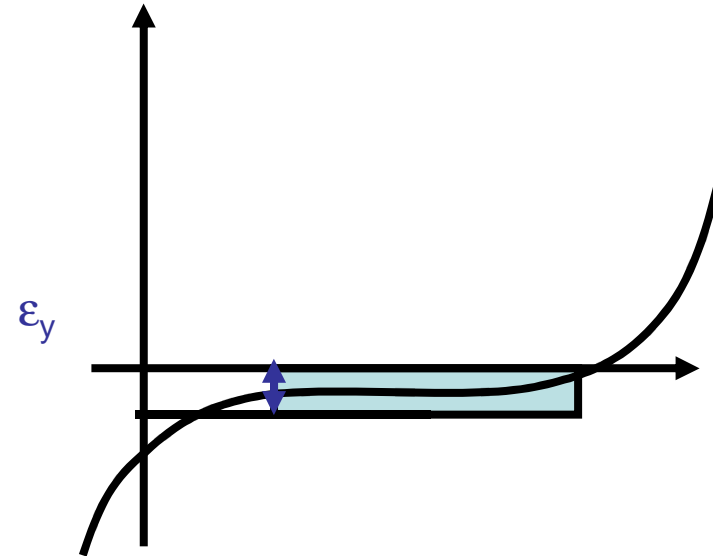


### 1- fort gradient, faible gradient



Test d'arrêt :  $\epsilon_x$

$$|x_{n+1} - x_n| \leq \epsilon_x$$



Test d'arrêt :  $\epsilon_y$

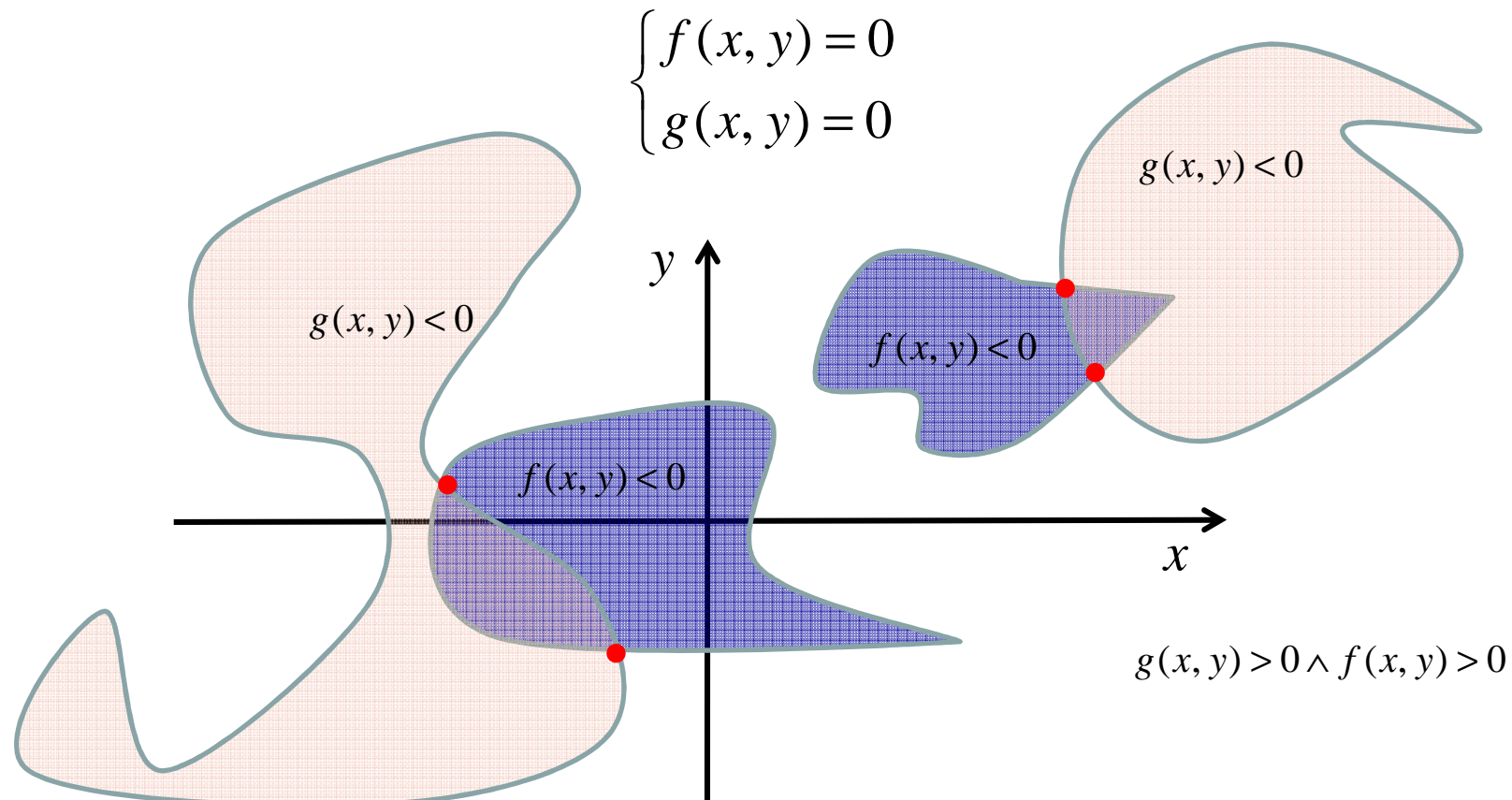
$$|f(x_n)| \leq \epsilon_y$$

### 2- tolérance relative, tolérance absolue, valeur de la tolérance ? Pas de réponse universelle ...

## VII – CAS DES FONCTIONS A PLUSIEURS VARIABLES



La modélisation mathématique des situations physiques conduit souvent à introduire des fonctions **de plusieurs variables**. Il est alors nécessaire de déterminer des solutions de système de plusieurs équations non-linéaires. Par exemple en 2D:



## VII – CAS DES FONCTIONS A PLUSIEURS VARIABLES



La recherche de racine en multi-dimensionnel est **BEAUCOUP** plus complexe qu'elle ne l'est en dimension 1. La raison fondamentale est que l'on ne peut plus **encadrer** la racine ...

L'éventail des méthodes disponibles est par conséquent beaucoup plus restreint. En fait, il n'y a **pas de bonne méthode générale**, robuste et efficace. La raison en est que les iso-f et iso-g n'ont en général aucun lien ...

Parmi les méthodes vues en 1D, la plupart ne peuvent pas être généralisée au cas multi-D. La méthode de **Newton** peut l'être...

## LA METHODE DE NEWTON-RAPHSON



Le système à résoudre étant:

$$f_i(\mathbf{x}) = f_i(x_1, x_2, \dots, x_n) = 0 \quad , \quad i = 1, 2, \dots, n$$

On part du développement de Taylor suivant:

$$f_i(\mathbf{x} + \mathbf{dx}) = f_i(\mathbf{x}) + \sum_{j=1}^n \frac{\partial f_i}{\partial x_j} dx_j + O(\mathbf{dx}^2) \quad , \quad i = 1, 2, \dots, n$$

En notation matricielle, en introduisant la Jacobienne  $\mathbf{J}$ :

$$\mathbf{f}(\mathbf{x} + \mathbf{dx}) = \mathbf{f}(\mathbf{x}) + \mathbf{J} \cdot \mathbf{dx} + O(\mathbf{dx}^2) \quad , \quad J_{ij} = \frac{\partial f_i}{\partial x_j}$$

L'algorithme s'obtient en annulant  $\mathbf{f}(\mathbf{x} + \mathbf{dx})$  et en négligeant les termes d'ordre supérieur :

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{dx} \quad \text{avec} \quad \mathbf{J} \cdot \mathbf{dx} = -\mathbf{f}(\mathbf{x}^k)$$

Nécessite la **résolution d'un système linéaire** à chaque pas ...