Equations non linéaires
Recherche de racines

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

1. Déterminer si un intervalle contient une racine au moins
2. Définir ce qu'est une méthode itérative
3. Ecrire les algorithmes des méthodes de substitution, de dichotomie, de la corde, de la fausse position et de Newton
4. Donner les conditions de convergence des méthodes étudiées
5. Définir et calculer théoriquement l'ordre de convergence d'une méthode itérative,
6. Mesurer numériquement l'ordre de convergence d'une méthode itérative
7. Enrichir une méthode itérative par une technique de sous ou sur-relaxation
8. Définir la méthode de Newton-Raphson en plusieurs dimensions

Motivation

- Dans de nombreuses situations pratiques, la solution d'un problème d'ingénieur apparaît, après modélisation, comme la solution d'une équation algébrique non-linéaire

- Par exemple:
  - les fréquences de résonnance ω d'un système mécanique sont solutions de $\det(M(\omega))=0$, où M est une matrice dépendant du système mécanique considéré
  - Mettre en œuvre une méthode implicite de résolution d'une équation différentielle nécessite de résoudre à chaque itération une équation du type $y^{n+1} = y^n + \Delta t\, Y(t^{n+1}, y^{n+1})$ pour l'inconnue y^{n+1}

Problème que l'on veut résoudre

- De manière générale, on considère donc une fonction réelle f d'une variable réelle x et on cherche à approximer les racines de f définies par

$$f(r) = 0$$

- Par exemple dans le cas une méthode implicite de résolution d'une équation différentielle qui nécessite de résoudre à chaque itération une équation du type $y^{n+1} = y^n + \Delta t\, Y(t^{n+1}, y^{n+1})$ pour l'inconnue y^{n+1}, on cherche en fait les racines de la fonction

- $f(x) = x - y^n - \Delta t\, Y(t^{n+1}, x)$ où y^n et t^{n+1} sont connus et où Y est la fonction définissant l'équation différentielle.

Difficultés - I

- Les cas où l'équation $f(r) = 0$ peut être résolue analytiquement sont rarissimes; ce sont des cas d'école, utiles académiquement (apprentissage, validation), mais très peu courants en pratique

  ex: $f(x) = x^2 + 4x - 2$

- En général la fonction est trop non-linéaire pour que l'équation admette des solutions analytiques

  ex: $f(x) = x^2 + \ln(4x-2)$

- Très souvent, la fonction n'admet même pas de représentation analytique simple

  ex: $f(x) = \det\big(M(x)\big)$ où M est une matrice de grande taille dont les éléments dépendent de x

- Chaque évaluation de f peut représenter un effort de calcul non négligeable et/ou il peut être nécessaire de résoudre un grand nombre d'équations.

Difficultés - II

- Les racines peuvent ne pas exister, être unique, multiples ou infiniment nombreuses
- Le passage du monde analytique au monde numérique a plusieurs conséquences:

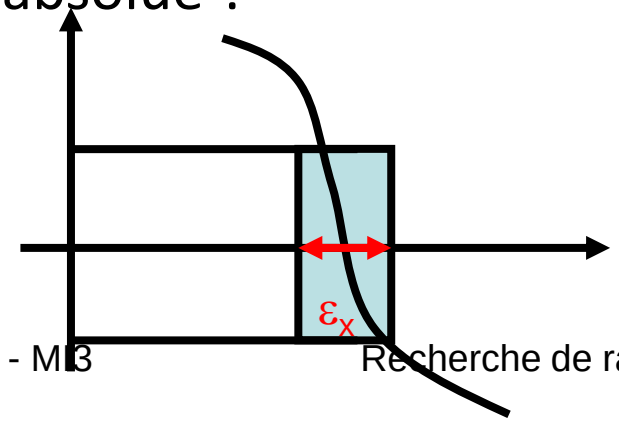
1. Les calculs se font en arithmétique finie et nous ne pouvons espérer qu'obtenir une valeur approchée de la racine. Quelle tolérance désirée/obtenue ?
2. La fonction f n'est plus un objet mathématique continu mais un opérateur numérique qui transforme un nombre « réel » en un autre nombre « réel ». Quelle précision dans le calcul de $f(x)$?
3. Le graphe de f n'est plus une courbe mais un nombre fini de points reliés par des segments de droite. Quel pas de discrétisation nécessaire ?

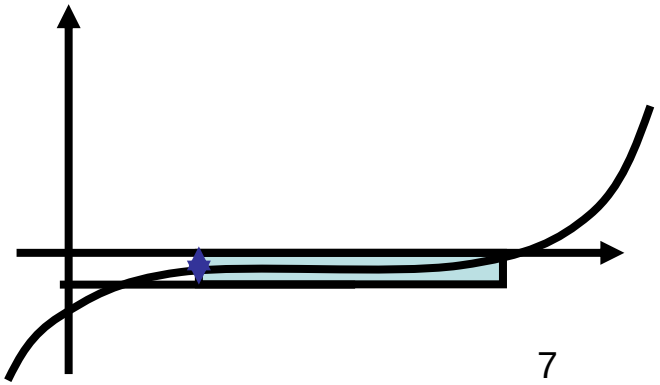
Méthodes itératives

- Il n'existe pas de méthode analytique générale pour résoudre $f(r) = 0$
- Numériquement, le principe de base est de construire une suite de nombres x_k pour $k = 0, 1, 2, \ldots$ qui approxime de mieux en mieux la racine recherchée lorsque k augmente
- La formule de récurrence utilisée pour créer la valeur de x_k à partir des itérés $x_0, x_1, \ldots, x_{k-1}$ dépend de la méthode itérative
- Après avoir choisi une valeur initiale x_0, on applique la formule de récurrence jusqu'à ce qu'un critère d'arrêt soit vérifié
- Les méthodes les plus communes pour les fonctions réelles à une variable réelle sont les suivantes: substitution, dichotomie, corde, Newton, fausse position, Brent

Critère d'arrêt

- Définir un bon critère d'arrêt n'est pas toujours facile. Il n'y a pas de choix universel
- Le critère d'arrêt doit pouvoir:
  - garantir un bon niveau de précision quelque soit le comportement de la fonction au voisinage de la racine. Ex: $|x_k - x_{k-1}| < \varepsilon_x \ \ \& \ \ |f(x_k)| < \varepsilon_f$
  - Éviter que le calcul ne s'arrête jamais: $k < iter\_max$
- Quelle valeur de tolérance choisir ? Se baser sur une erreur relative ou absolue ?





Convergence

- Une méthode itérative est dite convergente pour la racine r d'une fonction f si

$$\forall \varepsilon > 0, \exists k_0 : \forall k > k_0, |x_k - r| < \varepsilon$$

- Dit autrement, la méthode est convergente si

$$\lim_{k \to \infty} x_k = r$$

- La convergence d'une méthode dépend évidemment de la fonction f mais aussi de la racine visée r et de la valeur initiale x_0

Ordre de convergence

- Une méthode convergente est d'ordre $p \geq 1$ si p est le plus grand nombre réel tel que

$$\lim_{k \to \infty} \frac{|x_{k+1} - r|}{|x_k - r|^p} = C > 0 \ \& < \infty$$

- Dit autrement, la méthode est d'ordre p si

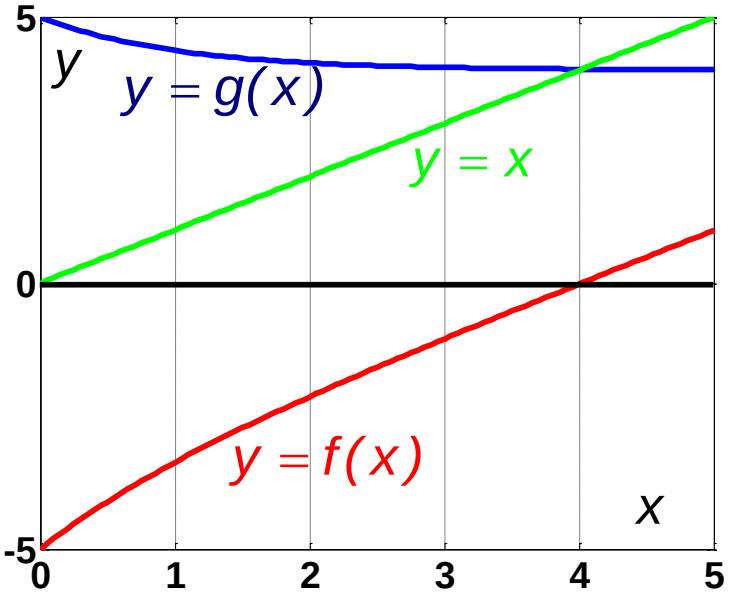
$$|x_{k+1} - r| \approx C|x_k - r|^p \text{ lorsque } k \longrightarrow \infty$$

- Remarques:
  - La constante C est le facteur de convergence de la méthode
  - si $p = 1$, alors il est nécessaire d'avoir $C < 1$ pour que la méthode converge (suite géométrique de raison C)
  - Contrairement à C, $\boldsymbol{p}$ **ne dépend que de la méthode**

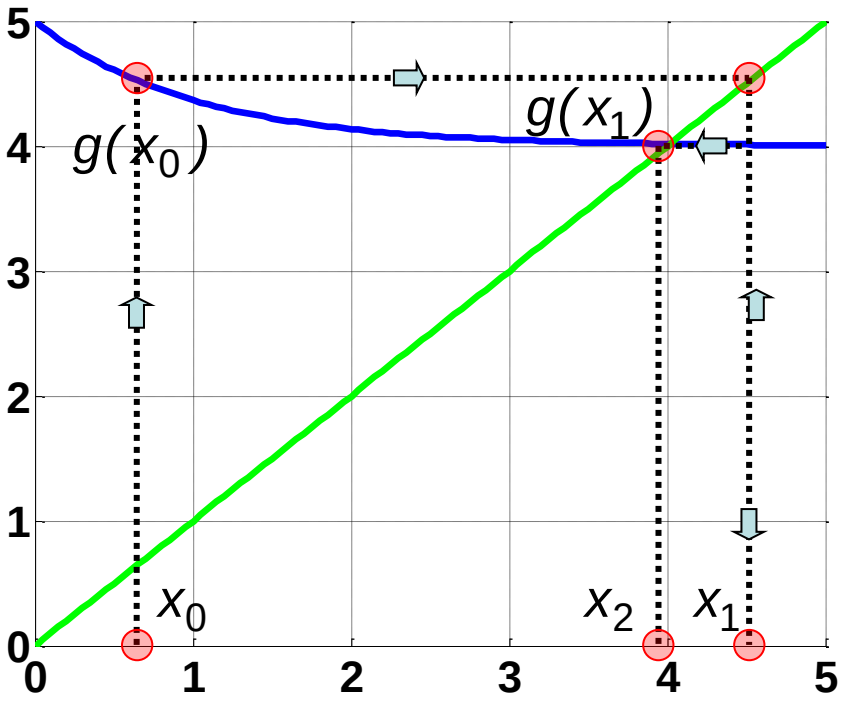
Méthode de substitution

- Partant de l'équation $f(x) = 0$, on introduit la fonction auxiliaire $g(x) = x - f(x)$.
- L'équation à résoudre est alors équivalente à $x = g(x)$
- L'algorithme de substitution s'écrit alors simplement $x_k = g(x_{k-1})$
- Si partant d'une valeur initiale x_0, la suite x_0, x_1, x_2 … converge vers une limite r, alors cette limite vérifie bien $f(r) = 0$

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

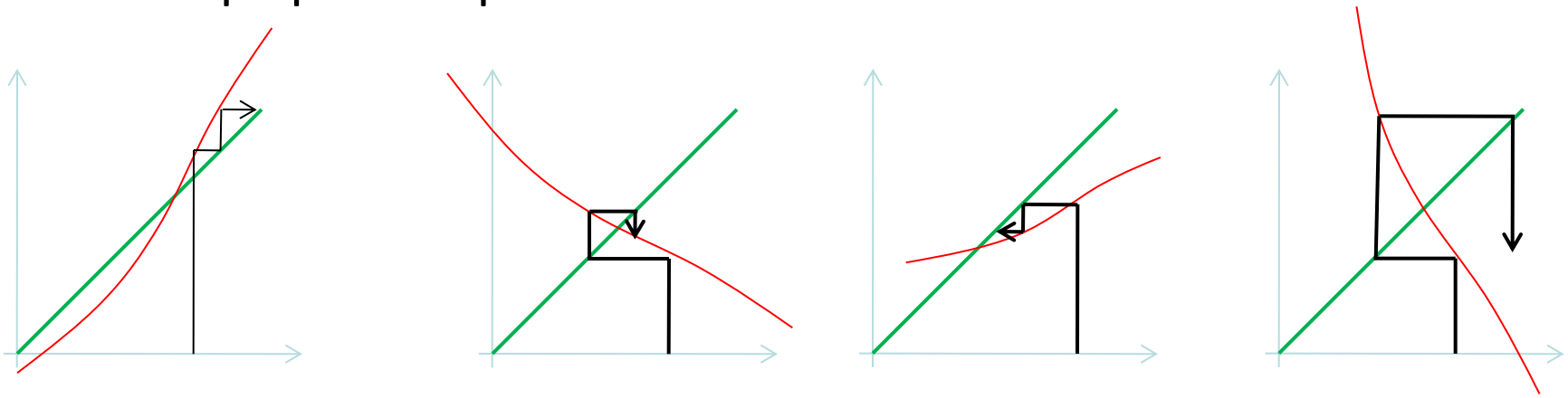


Méthode de substitution



Condition suffisante de convergence

- Ce qui peut se passer …



INSTABLE **STABLE** **STABLE** **INSTABLE**

- La méthode de substitution converge dès lors que la fonction g est contractante au voisinage de la racine :

$$\exists M < 1 : \forall (x, y), \left| g(x) - g(y) \right| \leq M \left| x - y \right|$$

Condition suffisante de convergence

- Lien avec le théorème des accroissements finis: si g est dérivable sur un intervalle I, alors
  $$\forall\,(x,y)\in I, \exists t\in I: f(x)-f(y)=g'(t)(x-y)$$

- On en déduit que la méthode de substitution converge dès lors que la fonction g :
  - est dérivable sur un voisinage I de r,
  - Est telle que $\exists \alpha<1: \forall t\in I, |g'(t)|\leq \alpha$ --> |g'(t)| < 1

- Par exemple, la méthode de substitution
  - converge pour $f(x)=\sin x\ -\sin 1$
    - 1 racine stable
  - ne converge pas pour $f(x)=\sin x-\sin 2$
    - 2 racine instable mais $\pi-2$ racine stable

Ordre de convergence de la méthode de substitution

$x_n = r + \varepsilon_n$; ε_n est l'erreur à l'itération n ;

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

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

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

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

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

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

- La méthode de substitution a donc une convergence d'ordre 1; on dit que c'est une méthode linéaire

Méthode de dichotomie

- Condition d'utilisation:
  Pas de contrainte sur la fonction; il faut juste que la racine soit séparée sur l'intervalle initial [a, b]

- 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

Mise en œuvre de la méthode de dichotomie

Définir $f(x), \varepsilon, iter\_max$

Initialiser a_0 et b_0; $n = 0$

Si $f(a_0)f(b_0) > 0$, STOP

Tant que $erreur > \varepsilon$ et $n < iter\_max$

c $= (a_n + b_n)/2$

si $f(a_n)f(c) > 0$, $a_{n+1} = c; b_{n+1} = b_n$

si $f(a_n)f(c) < 0$, $a_{n+1} = a_n; b_{n+1} = c$

$n = n + 1$

Fin

La racine est $(a_n + b_n)/2$ à ε près

Commentaires sur la méthode de dichotomie

- Méthode conceptuellement très simple et facile à programmer
- certitude de convergence, mais pas forcément vers une racine si la fonction n'est pas continue. Ex: $f(x) = 1/x$
- 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 de raison géométrique ½
- Séparation des racines: pas forcément simple à garantir car $f(a)f(b) < 0$ est nécessaire mais pas suffisant !!

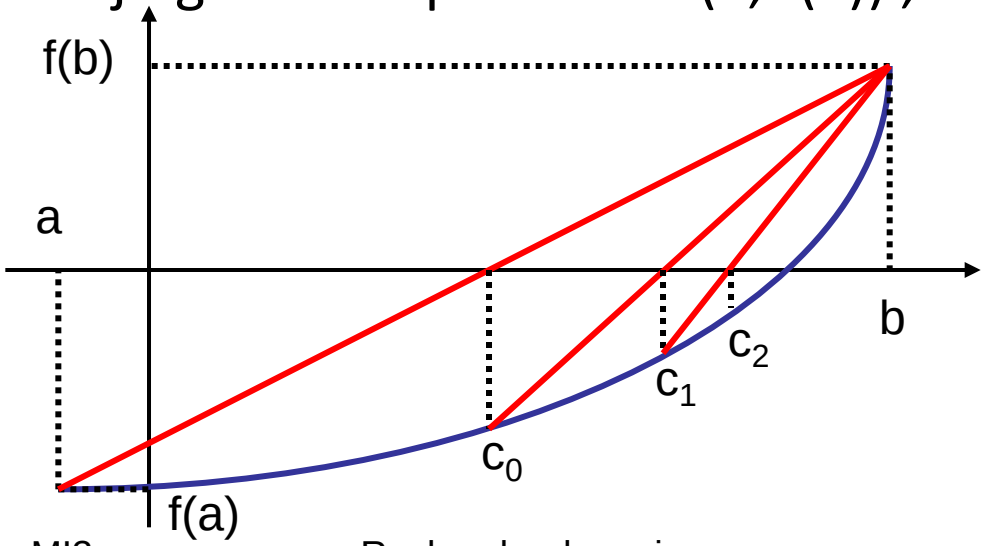
Méthode de la corde

- Condition d'utilisation:

  Pas de contrainte sur la fonction; il faut juste que la racine soit séparée sur l'intervalle initial [a, b]

- 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))



Mise en œuvre de la méthode de la corde

Définir $f(x), \varepsilon, iter\_max$

Initialiser a_0 et b_0; $n = 0$

Si $f(a_0)f(b_0) > 0$, STOP

Tant que $erreur > \varepsilon$ et $k < iter\_max$

c = $(a_n f(b_n) - b_n f(a_n))/\big(f(b_n) - f(a_n)\big)$

si $f(a_n)f(c) > 0$, $a_{n+1} = c$; $b_{n+1} = b_n$

si $f(a_n)f(c) < 0$, $a_{n+1} = a_n$; $b_{n+1} = c$

$n = n + 1$

Fin

La racine est c à ε près

Ordre de la méthode de la corde

On part de :

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

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:

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

La méthode de la corde est linéaire également …

Méthodes de Schröder

- Conditions d'utilisation:

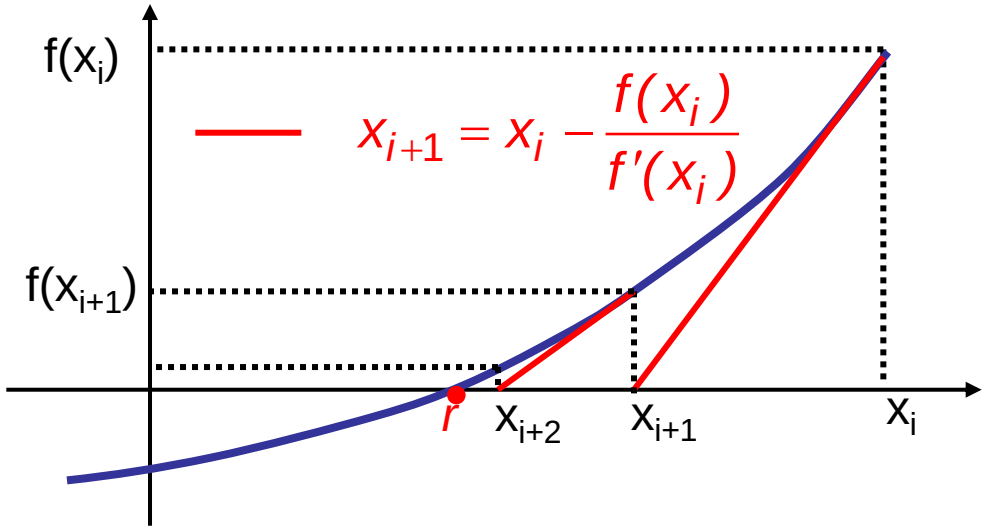
  Recherche d'une racine unique dans le voisinage I de x_0. Cette méthode itérative nécessite le calcul des m premières dérivées pour une méthode de Schröder d'ordre $\mathrm{m}+1$. Il faut donc que la fonction f dont on cherche la racine soit au moins m fois dérivable dans l'intervalle I.

- Principe de la méthode:

  On bâtit une suite récurrente de la façon suivante :

  - autour du point x_n , on effectue un développement limité de $f(x_n)$ jusqu'à l'ordre m
  - on définit x_{n+1} comme la racine du développement limité de $f(x_n)$
  - on itère …

Méthode de Newton (Schröder d'ordre 2)



- Méthode du 2nd ordre : on néglige les termes à partir du 2nd ordre: $f(x_{n+1}) = 0 = f(x_n) + (x_{n+1} - x_n)f'(x_n)$

- x_{n+1} : racine du développement limité à l'ordre 1 de f au voisinage de x_n

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Ordre de la méthode de Newton

On part de :

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

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

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

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

Un simple calcul donne alors :

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

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

Méthode de la fausse position

- Le calcul analytique des dérivées apparaissant dans les méthodes de Schröder est parfois lourd, parfois impossible
- Il est alors avantageux d'estimer ces dérivées par une formule au différences finies
- Appliquée à la méthode de Newton, cette approche conduit à la méthode de la 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 mais aussi plus efficace (ordre>1)

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* ,

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

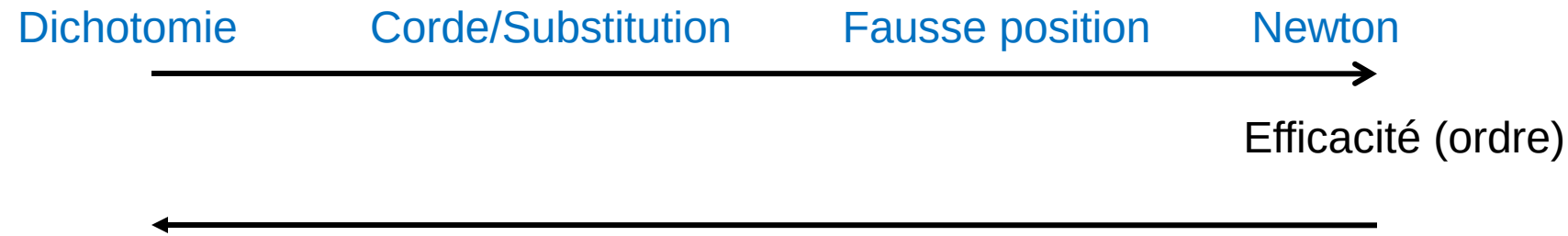
$$f(x_{i-1}) = f(r) + \varepsilon_{i-1} \; f'(r) + \frac{\varepsilon_{i-1}^2}{2} f''(r) + \mathrm{O}(\varepsilon_{i-1}^3)$$

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 ?

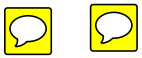
Oui, souvent, en 1D

Une méthode super-linéaire, robuste et sans dérivée – Méthode de Brent

Conditions d'utilisation

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

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,

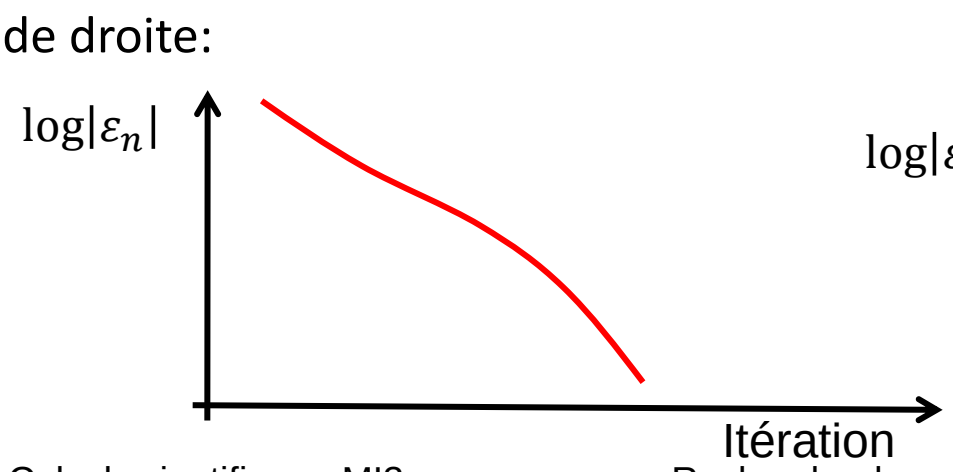
2- Dans les cas où la nouvelle estimation sort de l'intervalle initial, ou lorsque la décroissance de l'intervalle est trop lente, on utilise une simple dichotomie pour garantir la robustesse et la convergence …

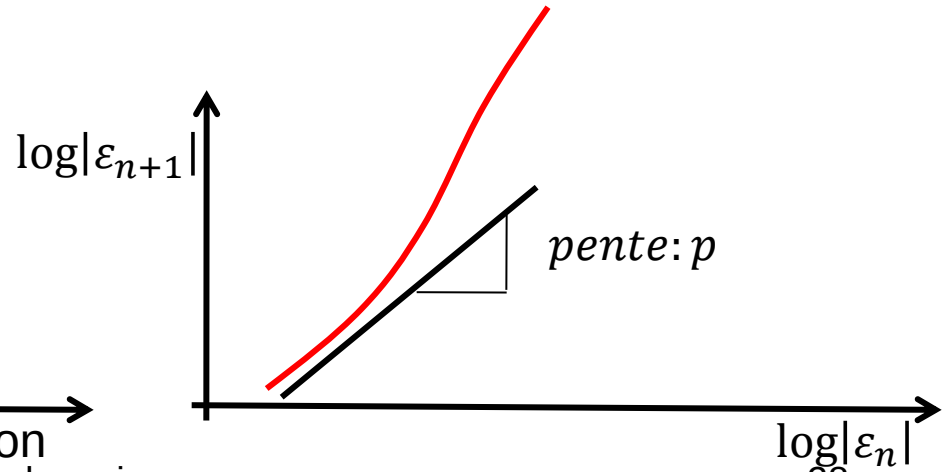
Mesure pratique de l'ordre

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, …). On trace le plus souvent $\log|\varepsilon_n|$ en cours des itérations.

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:

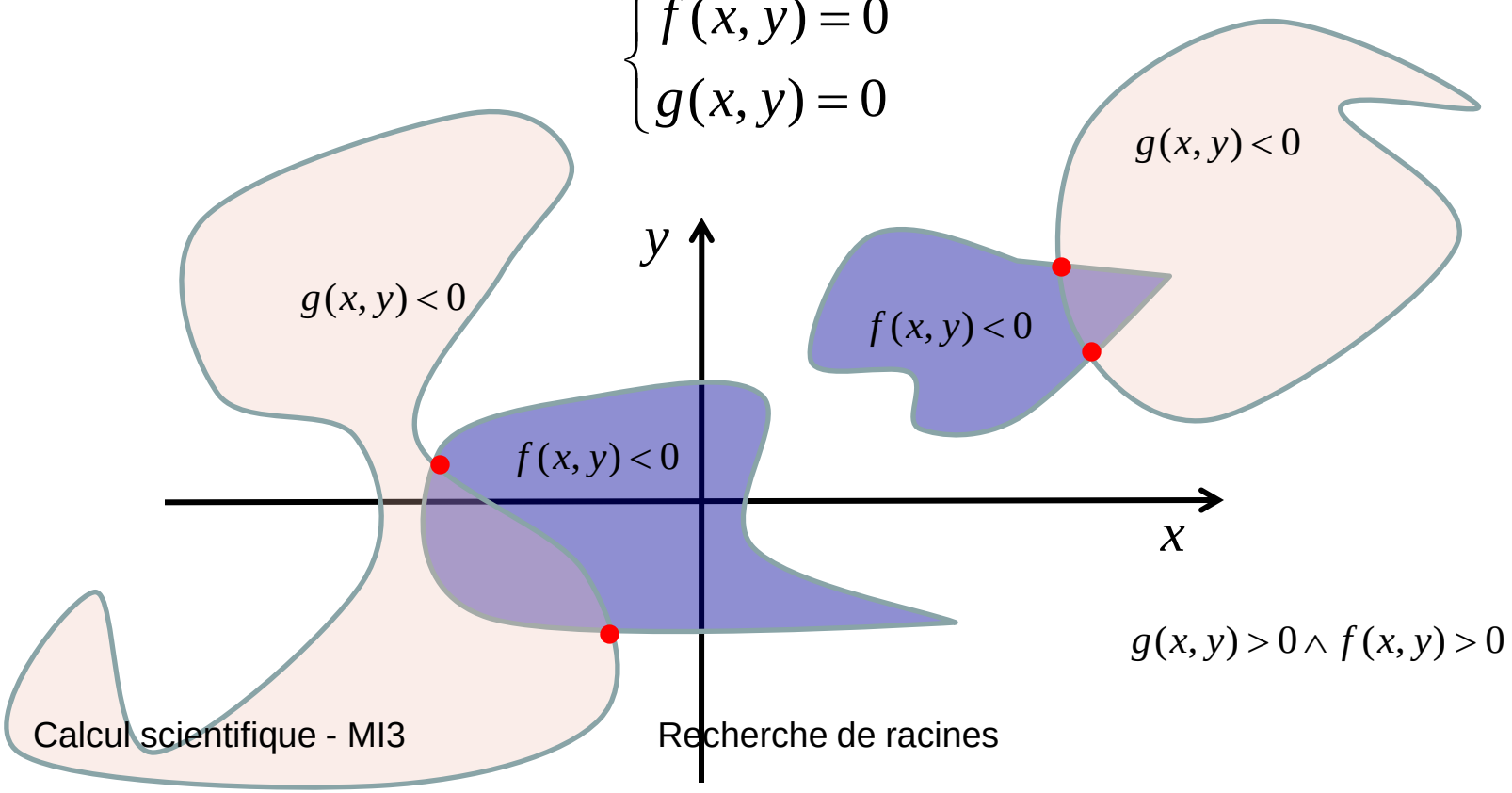




Cas des fonctions à plusieurs variables

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:

$$\begin{cases} f(x,y)=0 \\ g(x,y)=0 \end{cases}$$



Cas des fonctions à 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…

Méthode de Newton-Raphson

Le système à résoudre étant:

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

On part du développement de Taylor suivant:

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

En notation matricielle, en introduisant la Jacobienne 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 :

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

dx = J(-1).(-f(Xk))

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

Relaxation

- La technique de relaxation consiste à combiner linéairement les deux derniers itérés d'une méthode itérative

- La formule de récurrence devient:

$$x_{k+1} = \alpha\widetilde{x_{k+1}} + (1-\alpha)x_k$$

où $\widetilde{x_{k+1}}$ est la valeur issue de la méthode itérative sans relaxation

- $\alpha > 0$ est le paramètre de relaxation:
  - $\alpha = 1$: pas de relaxation
  - $\alpha < 1$: sous-relaxation: tend à stabiliser la méthode (mais augmente le nombre d'itérations nécessaire pour atteindre la convergence !!)
  - $\alpha > 1$: sur-relaxation: permet d'atteindre plus rapidement la convergence (mais diminue la stabilité !)

Réalisations sous Matlab

1. Créer un programme qui résout une équation du type f(x)=0 où f est une fonction fournie sous forme d'une fonction Matlab indépendante. On commencera par implémenter la méthode de substitution

2. Mettre en œuvre le programme précédent pour résoudre numériquement sinx –sin1=0 et sinx –sin2=0. Discuter les résultats obtenus.

3. Mesurer l'ordre de convergence de la méthode

4. Implémenter la méthode de Newton et mesurer son ordre de convergence

5. Transformer le programme précédent en une fonction Matlab et utiliser cette fonction pour terminer l'implémentation de la méthode Euler implicite pour la résolution des équations différentielles