Centre Universitaire de Mila
Département Mathématiques et Informatique

Chapitre 3: Résolution des équations non linéaires

Dr. Aissa Boulmerka
aissa.boulmerka@gmail.com

1- INTRODUCTION

Introduction

- L'objet essentiel de ce chapitre est **l'approximation des racines** d'une fonction réelle d'une variable réelle, c'est-`a-dire la résolution approchée du problème suivant :

  Étant donné $f: I =]a, b[\subseteq \mathbb{R} \to \mathbb{R}$, trouver $\alpha \in \mathbb{C}$ tel que $f(\alpha) = 0$ **(1)**

- Il est important de noter que, bien que f soit à valeurs réelles, ses zéros peuvent être complexes.

- Les méthodes pour approcher une racine α de f sont en général **itératives** : elles consistent à construire une suite $\left\{x^{(k)}\right\}$ telle que

$$\lim_{k \to \infty} x^{(k)} = \alpha$$

Ordre de convergence

La convergence des itérations est caractérisée par la définition suivante :

Définition 1: On dit qu'une suite $\{x^{(k)}\}$ construite par une méthode numérique converge vers α avec un ordre $p \geq 1$ si

$$\exists C > 0: \quad \frac{\left|x^{(k+1)} - \alpha\right|}{\left|x^{(k)} - \alpha\right|^p} \leq C, \qquad \forall k \geq k_0, \qquad \textbf{(2)}$$

où $k_0 \geq 0$ est un entier.

Dans ce cas, on dit que **la méthode est d'ordre p**.

Remarquer que si p est égal à 1, il est nécessaire que $C < 1$ dans **(2)** pour que $x^{(k)}$ converge vers α.

On appelle alors la constante C **facteur de convergence** de la méthode.

2- UNE APPROCHE GÉOMÉTRIQUE DE LA DÉTERMINATION DES RACINES

Introduction

- Nous introduisons dans cette section les méthodes de dichotomie (ou de bissection), de la corde, de la sécante, et de Newton.

- Nous les présentons dans l'ordre de complexité croissante des algorithmes.

- Dans le cas de la méthode de **dichotomie**, la seule information utilisée est **le signe de la fonction f** aux extrémités de sous-intervalles.

- Pour les autres algorithmes on prend aussi en compte les **valeurs de la fonction et/ou de ses dérivées**.

Méthode de dichotomie (bissection)

La méthode de dichotomie est fondée sur la propriété suivante :

Propriété (théorème des zéros d'une fonction continue) Soit une fonction continue $f : [a, b] \rightarrow R$, si $f(a)f(b) < 0$, alors $\exists\, \alpha \in]a, b[$ tel que $f(\alpha) = 0$.

En partant de $I_0 = [a, b]$, la méthode de dichotomie produit une suite de sous-intervalles $I_k = [a^{(k)}, b^{(k)}], k \geq 0$, avec $I_k \subset I_{k-1}, k \geq 1$, et tels que $f(a^{(k)})f(b^{(k)}) < 0$. Plus précisément, on pose $a^{(0)} = a$, $b^{(0)} = b$ et $x^{(0)} = (a^{(0)} + b^{(0)})/2$; alors, pour $k \geq 0$:

On pose $a^{(k+1)} = a^{(k)}, b^{(k+1)} = x^{(k)}$ $si\ f\left(x^{(k)}\right)f\left(a^{(k)}\right) < 0$
ou $a^{(k+1)} = x^{(k)}, b^{(k+1)} = b^{(k)}$ $si\ f\left(x^{(k)}\right)f\left(b^{(k)}\right) < 0$;
et $x^{(k+1)} = (a^{(k+1)} + b^{(k+1)})/2$.

Méthode de dichotomie (bissection)

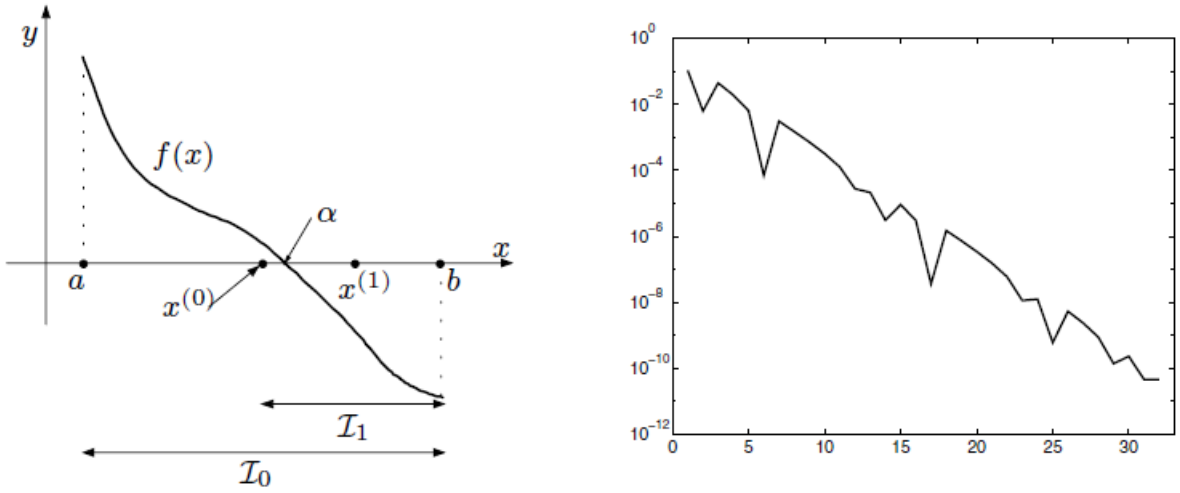


Fig. 1. Les deux premiers pas de la méthode de dichotomie (à gauche). Historique de la convergence pour l'exemple montré en programme Matlab (à droite) ; le nombre d'itérations est reporté sur l'axe des x et l'erreur absolue sur l'axe des y

Convergence de la méthode de dichotomie

- Cet algorithme converge donc à coup sur mais lentement. De plus, notons que la méthode de dichotomie ne garantit pas la réduction monotone de l'erreur absolue d'une itération à l'autre. Autrement dit, on ne peut pas assurer a priori que

$$\left|e^{(k+1)}\right| \leq M_k \left|e^{(k)}\right| \quad \text{pour tout } k \geq 0,$$
$$\text{avec } M_k < 1.$$

- Comme cette propriété n'est pas satisfaite, la méthode de dichotomie n'est pas une méthode **d'ordre $\boldsymbol{p = 1}$** au sens de la **Définition 1**.

Méthode de dichotomie

```
function
[xvect,xdif,fx,iter]=bisect(a,b,tol,nmax,fun)
% BISECT Méthode de dichotomie tente de
trouver un zéro de la fonction continue FUN
sur l'intervalle [A,B] en utilisant la méthode
de dichotomie.
% FUN accepte une variable réelle scalaire
x et renvoie une valeur réelle scalaire.
% XVECT est le vecteur des itérées, XDIF
est le vecteur des différences entre itérées
consécutives, FX est le résidu. TOL est la
tolérance de la méthode.
err=tol+1;
iter=0;
xvect=[]; fx=[]; xdif=[];
```

```
while iter<nmax & err>tol
    iter=iter+1;
    c=(a+b)/2; x=c; fc=eval(fun);
    xvect=[xvect;x];
    fx=[fx;fc]; x=a;
    if fc*eval(fun)>0
        a=c;
    else
        b=c;
    end
    err=0.5*abs(b-a); xdif=[xdif;err];
End

return
```

Exemple: Méthode de dichotomie

```
clear; clc;
a = 0.6;
b = 1;
tol = 10^-10;
nmax = 100;
fun = 'x*(63*x^4-70*x^2+15)/8';
[xvect,xdif,fx,iter]=bisect(a,b,tol,nmax,fun);
 fprintf('xvect(%d) = %1.10f\n',numel(xvect), xvect(end));
fprintf('xdif(%d) = %1.10f\n',numel(xdif),xdif(end));
fprintf('fx(%d) = %1.10f\n',numel(fx),fx(end));
fprintf('Nombre d''itérations = %d\n',iter);
```

xvect(31) = 0.9061798459
xdif(31) = 0.0000000001
fx(31) = -0.0000000003
Nombre d'itérations = 31

Notes:

- Les vecteurs **xvect**, **xdif** et **fx** contiennent respectivement les suites $\{x^{(k)}\}$, $\{|x^{(k+1)} - x^{(k)}|\}$ et $\{f(x^{(k)})\}$, pour $k \geq 0$, tandis que **iter** désigne le nombre d'itérations nécessaire à satisfaire le critère d'arrêt.
- Dans le cas de la méthode de dichotomie, le code s'arrête dès que la demi-longueur de l'intervalle est inférieure à tol.

Méthodes de la corde, de la sécante et de Newton

- Afin de mettre au point des algorithmes possédant de meilleures propriétés de convergence que la méthode de dichotomie, il est nécessaire de prendre en compte les informations données par les valeurs de f et, éventuellement, par sa dérivée f' (si f est différentiable) ou par une approximation convenable de celle-ci.
- Ecrivons pour cela le développement de Taylor de f en α au premier ordre.
- On obtient alors la version linéarisée du problème **(1)**

$$f(\alpha) = 0 = f(x) + (\alpha - x)f'(\xi), \qquad \textbf{(3)}$$

où ξ est entre α et x.

Méthodes de la corde, de la sécante et de Newton

L'équation **(3)** conduit à la méthode itérative suivante :

- pour tout $k \geq 0$,
- étant donné $x^{(k)}$, déterminer $x^{(k+1)}$ en résolvant l'équation

$$f\left(x^{(k)}\right) + \left(x^{(k+1)} - x^{(k)}\right) q_k = 0,$$

où q_k est une approximation de $f'\left(x^{(k)}\right)$.

- La méthode qu'on vient de décrire revient à chercher l'intersection entre l'axe des x et la droite de pente q_k passant par le point $\left(x^{(k)}, f\left(x^{(k)}\right)\right)$, ce qui s'écrit

$$x^{(k+1)} = x^{(k)} - {q_k}^{-1} f\left(x^{(k)}\right) \qquad \forall k \geq 0.$$

- Considérons maintenant trois choix particuliers de q_k.

Méthode de la corde

On pose

$$q_k = q = \frac{f(b) - f(a)}{b - a} \qquad \forall k \geq 0,$$

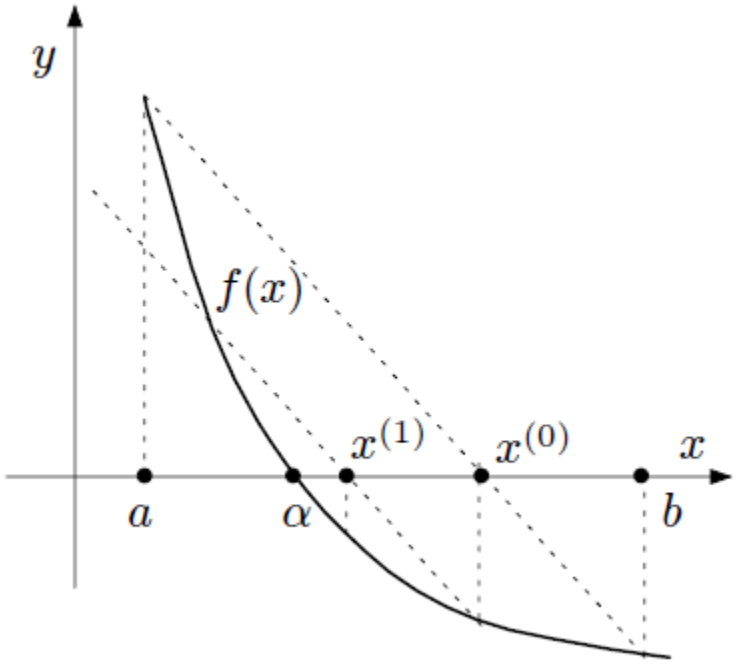
d'où on déduit la relation de récurrence suivante (pour une valeur $x^{(0)}$ donnée) :

$$x^{(k+1)} = x^{(k)} - \frac{b - a}{f(b) - f(a)} f\left(x^{(k)}\right) \qquad \forall k \geq 0. \qquad \textbf{(4)}$$

Nous pouvons démontrer que la suite $\{x^{(k)}\}$ définie par **(4)** converge vers la racine α avec un **ordre de convergence** $\boldsymbol{p = 1}$.

Méthode de la corde

Les deux premières étapes de la méthode de la corde



Méthode de la corde

```
function [xvect,xdif,fx,iter] = chord(a,b,x0,
tol, nmax, fun)
%CHORD Méthode de la corde tente de
trouver un zéro de la fonction continue FUN
sur l'intervalle [A,B] en utilisant la méthode
de la corde.
% FUN accepte une variable réelle scalaire
x et renvoie une valeur réelle scalaire.
% XVECT est le vecteur des itérées,
% XDIF est le vecteur des différences entre
itérées consécutives, FX est le résidu.
% TOL est la tolérance de la méthode.
```

```
x=a; fa=eval(fun);
x=b; fb=eval(fun);
r=(fb-fa)/(b-a);
err=tol+1; iter=0; xvect=x0; x=x0;
fx=eval(fun); xdif=[];
while iter<nmax && err>tol
    iter=iter+1;
    x=xvect(iter);
    xn=x-fx(iter)/r;
    err=abs(xn-x);
    xdif=[xdif; err];
    x=xn;
    xvect=[xvect;x]; fx=[fx;eval(fun)];
end
return
```

Exemple: Méthode de la corde

```
a = 0; b = 1.5; x0 = 0.75;
tol = 10^-10; nmax = 100; nmax = 1000;
fun = 'cos(2*x)^2-x^2';
[xvect,xdif,fx,iter]=chord(a,b,x0,tol,nmax,fun);
fprintf('xvect(%d) = %1.10f\n',numel(xvect),xvect(end));
fprintf('xdif(%d) = %1.10f\n',numel(xdif),xdif(end));
fprintf('fx(%d) = %1.10f\n',numel(fx),fx(end));
fprintf('Nombre d''itérations = %d\n',iter);
```

xvect(133) = 0.5149332647
xdif(132) = 0.0000000001
fx(133) = -0.0000000001
Nombre d'itérations = 132

Méthode de la sécante

- On pose

$$q_k = \frac{f\left(x^{(k)}\right) - f\left(x^{(k-1)}\right)}{x^{(k)} - x^{(k-1)}} \qquad \forall k \geq 0, \qquad \textbf{(5)}$$

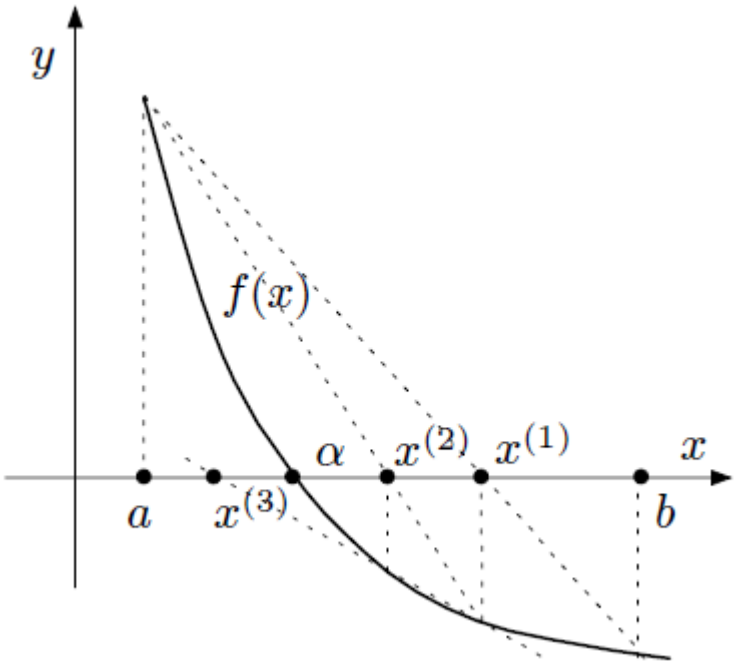
- d'où on déduit, en se donnant **deux valeurs initiales** $x^{(-1)}$ et $x^{(0)}$, la relation suivante :

$$x^{(k+1)} = x^{(k)} - \frac{x^{(k)} - x^{(k-1)}}{f(x^{(k)}) - f(x^{(k-1)})} f\left(x^{(k)}\right) \qquad \forall k \geq 0, \qquad \textbf{(6)}$$

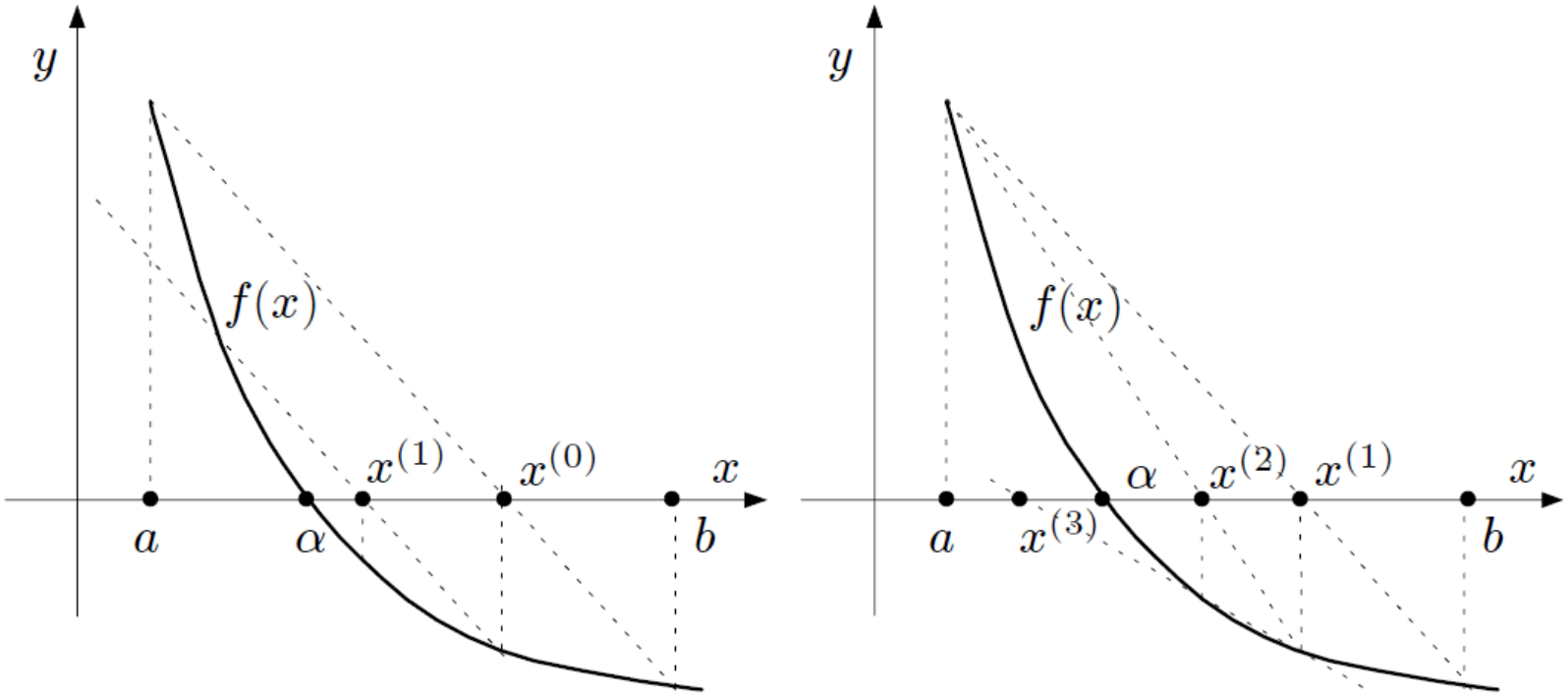
- Nous pouvons démontrer que la suite $\{x^{(k)}\}$ définie par **(6)** converge vers la racine α avec un **ordre** $\boldsymbol{p = (1 + \sqrt{5})/2 \simeq 1.63}$.

Méthode de la sécante

Les deux premières étapes de la méthode de la sécante



Comparaison graphique entre la méthode de la corde et méthode de la sécante



Méthode de la sécante

```
function [xvect,xdif,fx,iter] = secant(xm1,x0,
tol,nmax,fun)
%SECANT Méthode de la sécante tente de
trouver un zéro de la fonction continue FUN
en utilisant la méthode de la sécante.
% FUN accepte une variable réelle scalaire
x et renvoie une valeur réelle scalaire.
% XVECT est le vecteur des itérées,
% XDIF est le vecteur des différences entre
% itérées consécutives,
% FX est le résidu.
%TOL est la tolérance de la méthode.
```

```
x=xm1; fxm1=eval(fun);
xvect=[x]; fx=[fxm1];
x=x0; fx0=eval(fun);
xvect=[xvect;x]; fx=[fx;fx0];
err=tol+1; iter=0; xdif=[];
while iter<nmax && err>tol
    iter=iter+1;
    x=x0-fx0*(x0-xm1)/(fx0-fxm1);
    xvect=[xvect;x];
    fnew=eval(fun); fx=[fx;fnew];
    err=abs(x0-x);
    xdif=[xdif;err];
    xm1=x0; fxm1=fx0;
    x0=x; fx0=fnew;
end
return
```

Exemple: Méthode de la sécante

```
clear; clc;
% la solution exacte alpha = 0.5149
a = 0; b = 1.5; x0 = 0.75; xm1 = 0; tol = 10^-10; nmax = 100;
fun = 'cos(2*x)^2-x^2';
[xvect,xdif,fx,iter] = secant(xm1,x0,tol,nmax,fun);
fprintf('xvect(%d) = %1.10f\n',numel(xvect),xvect(end));
fprintf('xdif(%d) = %1.10f\n',numel(xdif),xdif(end));
fprintf('fx(%d) = %1.10f\n',numel(fx),fx(end));
fprintf('Nombre d''itérations = %d\n',iter);
```

xvect(8) = 0.5149332647
xdif(6) = 0.0000000000
fx(8) = 0.0000000000
Nombre d'itérations = 6

Méthode de Newton

- Supposons $f \in C^1(I)$ et $f(\alpha) = 0$ (i.e. α est une racine simple de f). En posant

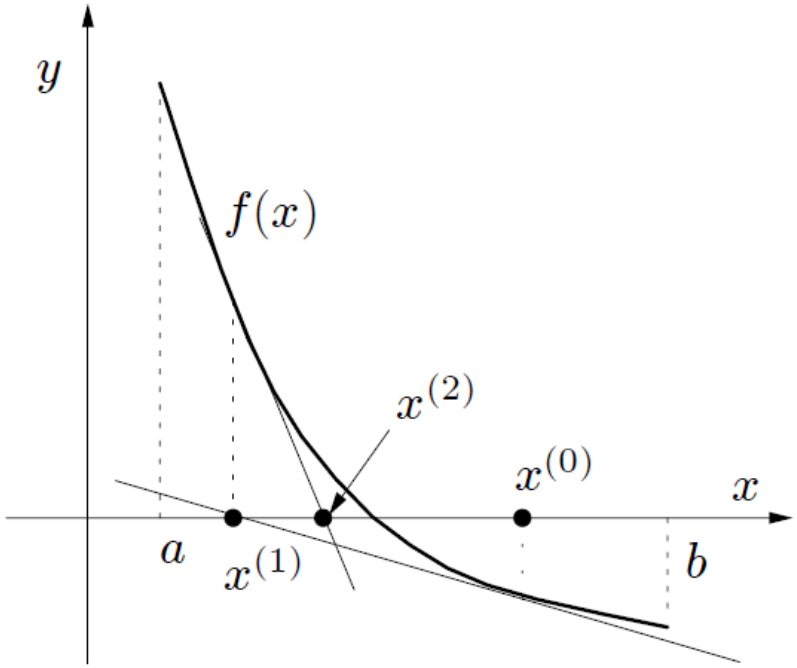
$$q_k = f'\left(x^{(k)}\right) \qquad \forall k \geq 0$$

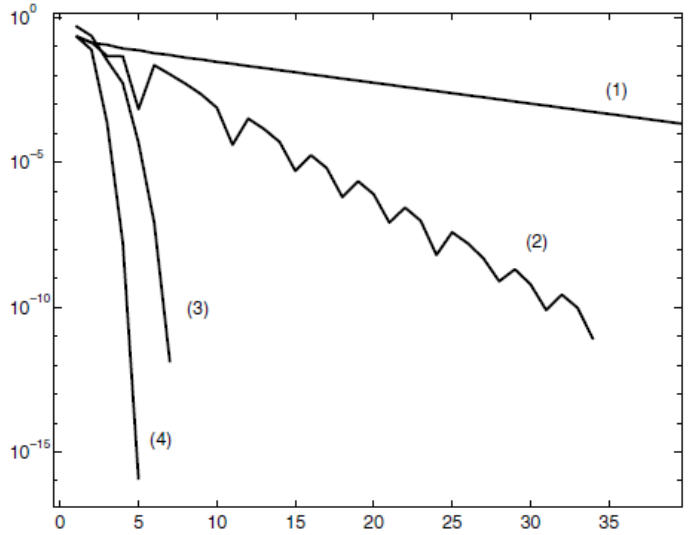
- et en se donnant la valeur initiale $x^{(0)}$, on obtient la méthode de Newton (encore appelée méthode de **Newton-Raphson** ou des **tangentes**)

$$x^{(k+1)} = x^{(k)} - \frac{f\left(x^{(k)}\right)}{f'(x^{(k)})} \qquad \forall k \ \geq \ 0. \qquad \textbf{(7)}$$

- A la k-ème itération, la méthode de Newton nécessite l'évaluation des deux fonctions f et f' au point $x^{(k)}$. Cet effort de calcul supplémentaire est compensé par une accélération de la convergence, la méthode de Newton étant **d'ordre 2**.

Méthode de Newton





Les deux premières étapes de la méthode de Newton (à gauche) ; historique des convergences de montré en programme Matlab pour les méthodes de la corde (1), de dichotomie (2), de la sécante (3) et de Newton (4) (à droite). Le nombre d'itérations est reporté sur l'axe des x et l'erreur absolue sur l'axe des y.

Méthode de Newton

```
function [xvect,xdif,fx,iter] = newton(x0, tol, nmax, fun, dfun)
%NEWTON méthode de Newton tente de trouver un zéro de la fonction continue FUN avec la méthode de Newton en partant de la donnée initiale X0.
% FUN et DFUN accepte une variable réelle scalaire x et renvoie une valeur réelle scalaire.
% XVECT est le vecteur des itérées,
% XDIF est le vecteur des différences entre itérées consécutives,
% FX est le résidu.
% TOL est la tolérance de la méthode.
err=tol+1; iter=0; xvect=x0; x=x0; fx=eval(fun);
xdif=[];
```

```
while iter<nmax && err>tol
   iter=iter+1;
   x=xvect(iter);
   dfx=eval(dfun);
   if dfx==0
      err=tol*1.e-10;
      fprintf('arrêt car dfun est nulle');
   else
      xn=x-fx(iter)/dfx; err=abs(xn-x);
      xdif=[xdif; err];
      x=xn; xvect=[xvect;x]; fx=[fx;eval(fun)];
   end
end
return
```

Exemple: Méthode de Newton

```
clear; clc;
% la solution exacte alpha = 0.5149
a = 0; b = 1.5; x0 = 0.75; tol = 10^-10; nmax = 100;
fun = '(cos(2*x))^2-x^2';
dfun = '-4*cos(2*x)*sin(2*x)-2*x';
[xvect,xdif,fx,iter] = newton(x0,tol,nmax,fun,dfun);
fprintf('xvect(%d) = %1.10f\n',numel(xvect),xvect(end));
fprintf('xdif(%d) = %1.10f\n',numel(xdif),xdif(end));
fprintf('fx(%d) = %1.10f\n',numel(fx),fx(end));
fprintf('Nombre d''itérations = %d\n',iter);
```

xvect(6) = 0.5149332647
xdif(5) = 0.0000000000
fx(6) = 0.0000000000
Nombre d'itérations = 5

3- MÉTHODE DE POINT FIXE POUR LES ÉQUATIONS NON LINÉAIRES

Méthode de point fixe

- La méthode est fondée sur le fait qu'il est toujours possible, pour $f: [a, b] \to R$, de transformer le problème $f(x) = 0$ en un problème équivalent $x - \varphi(x) = 0$, où la fonction auxiliaire $\varphi : [a, b] \to R$ a été choisie de manière à ce que $\varphi(\alpha) = \alpha$ quand $f(\alpha) = 0$.
- Approcher les zéros de f se ramène donc au problème de la détermination des points fixes de φ.
- Ce qui se fait en utilisant l'algorithme itératif suivant : étant donné $x^{(0)}$, on pose

$$x^{(k+1)} = \varphi\left(x^{(k)}\right), \qquad k \geq 0. \qquad \textbf{(8)}$$

- On dit que **(8)** est une itération de point fixe et φ la fonction d'itération associée.
- Le choix de φ n'est pas unique. Par exemple, toute fonction de la forme $\varphi(x) = x + F(f(x))$, où F est une fonction continue telle que $F(0) = 0$, est une fonction d'itération possible.

Méthode de point fixe

```
function [xvect,xdif,fx,iter] = fixpoint(x0, tol, nmax, fun, phi)
%FIXPOINT Méthode de point fixe tente de trouver un zéro de la fonction continue FUN en utilisant la méthode de point fixe
% X=PHI(X), en partant de la donnée initiale X0.
% XVECT est le vecteur des itérées,
% XDIF est le vecteur des différences entre itérées consécutives,
% FX est le résidu.
% TOL est la tolérance de la méthode.
```

```
err=tol+1; iter=0;
xvect=x0; x=x0; fx=eval(fun); xdif=[];
while iter<nmax && err>tol
    iter=iter+1;
    x=xvect(iter);
    xn=eval(phi);
    err=abs(xn-x);
    xdif=[xdif; err];
    x=xn; xvect=[xvect;x]; fx=[fx;eval(fun)];
end
return
```

Exemple: Méthode de point fixe

```
clear; clc;
% la solution exacte alpha = 0.7390
a = 0; b = 1; x0 = 0.5; tol = 10^-10; nmax = 100;
fun = 'x-cos(x)';
phi = 'cos(x)';
[xvect,xdif,fx,iter] = fixpoint(x0,tol,nmax,fun,phi);
fprintf('xvect(%d) = %1.10f\n',numel(xvect),xvect(end));
fprintf('xdif(%d) = %1.10f\n',numel(xdif),xdif(end));
fprintf('fx(%d) = %1.10f\n',numel(fx),fx(end));
fprintf('Nombre d''itérations = %d\n',iter);
```

xvect(58) = 0.7390851333
xdif(57) = 0.0000000001
fx(58) = 0.0000000001
Nombre d'itérations = 57