TP N°1
Méthodes numériques (Résolution de l'équation $f(x)=0$)
La Méthode de bissection (de dichotomie)

La méthode de bissection **(**de dichotomie**)** est, en mathématiques, un algorithme de recherche d'un zéro d'une fonction f continue sur l'intervalle $[a, b]$ qui consiste à répéter des partages d'un intervalle en deux parties ($[a, m]$ et $[m, b]$ óu $m = (a+b)/2$) puis à sélectionner le sous-intervalle dans lequel existe un zéro de la fonction.

I. L'algorithme de la méthode de bissection

Cette méthode est utilisée pour calculer **les zéros** d'une fonction continue f ($\Re \Leftrightarrow \Re$) dans un intervalle **[*a*,*b*]** :

1. ***Si*** $f(a).f(b) > 0$, alors la méthode termine **(pas de zéros)**.
2. ***Sinon si*** $f(a).f(b) < 0$, on sait alors qu'il existe au moins un **zéro** $\bar{x}$ de ***f*** dans l'intervalle **[*a*,*b*].**
3. ***Tant que*** $|b - a| > tol$:
   - **a)** Soit $m = \frac{(a+b)}{2}$;
   - **b)** ***Si*** $f(m) = 0$, alors $\bar{x} = m$ et la méthode **termine**.
   - **c)** ***Sinon si*** $f(a) \,.\, f(m) < 0$ on pose $b = c$;
   - **d)** ***Sinon si*** $f(m) \,.\, f(b) < 0$, on pose $a = m$;

   fin si

   affichez : (***$\bar{x}$ et f($\bar{x}$) pour chaque itération*** **)**

   fin tant que

fin si

II. Application

1. Implémenter la méthode de **la bissection** sous forme d'une fonction de **Matlab**:

   function [*xzero*, *iter*, *err*, *fzero*] = *bissection*(*a*, *b*, *tol*);

Avec

les arguments d'entrée	**les arguments de sortie**
a*, *b : les limites de l'intervalle $[a, b]$. ***tol*** : L'erreur tolérée par le résultat.	***xzero*** : La racine trouvée par la méthode. ***iter*** : Le nombre d'itérations effectuées ***err*** : Valeur absolue de l'erreur finale (***abs*(*a-b*)** finale). ***fzero*** : : l'image de zéro à trouver dans la fonction

2. Appliquer la fonction produite pour trouver le zéro de la fonction suivante :
   $f(x) = x^3 + x^2 - 3x - 3$ dans l'intervalle $[1, 2]$ à implémenter dans une fonction séparée
3. Produire le tableau des valeurs progressives (x,$f(x)$) jusqu'au zero.
4. Tracer cette fonction en indiquant dans le graphe le zéro trouvé.

RM

Pour définir une fonction courte il est possible d'utiliser les fonctions ***inline*** et les fonctions anonymes.
La création d'une fonction inline suit la syntaxe suivante :

nom_fonction = inline('expression mathematique de la fonction')

Par exemple pour définir la fonction : $2x^3-4x^2+x-3$, on écrit :

>> **f = inline ('2*x^3-4*x^2+x-3','x')** Ou >> **f = inline ('2*x^3-4*x^2+x-3')**

Pour calculer f(3) on écrit :

>> **f(3)**

ans = 18

```
function [xzero,iter,err,fzero]=bissection(a,b,tol);
%la fonction de dichotomie
%-------------------------
f=inline('x^3+x^2-3*x-3');% définition la fonction f(x):
a0=a;b0=b;% enregistrement les limites initiale pour la partie graphique
iter=0;
fa=f(a);
fb=f(b);
if fa*fb>0
    disp('pas de zero dans cet intervalle');
elseif fa*fb<0
    while abs(b-a)>tol
        m=(a+b)/2;
        fzero=f(m);
        if fzero ==0
            a=m; b=m;
            err=(a+b)/2;
        elseif f(a)*fzero<0
            b=m;
        elseif f(b)*fzero<0
            a=m;
        end
        str=['zero=',num2str(m),'fzero=',num2str(fzero)];
        disp(str)
        iter=iter+1;
    end
end
%le zero et leur image:
xzero=(a+b)/2;
fzero=f(xzero);
err=abs(b-a);
%la partie graphique:
fplot(f,[a0 b0]);
xlabel('x')
ylabel('la fonction f(x)')
text(xzero,fzero,'le zero calculé');
hold on
plot(xzero,fzero,'o')
grid on
```

Exécution :

```
>>[zero,fzero,err]=bisect(1,2,1e-5);
```

Les questions :

Considérant l'équation : $f(x) = x^3+x^2-3x-3$

1. Dessinez la courbe de $f(x)$ sur l'intervalle [-2, 2], puis trouvez des intervalles convenables pour appliquer la méthode de bissection.
2. Pour chaque intervalle (un pour chaque racine), appliquez la fonction Matlab 'bissection.m' sur $f(x)$, en considérant : $tol = 0.001$.

Solution :

Dessiner la courbe

>> f = inline ('x^3+x^2-3*x-3','x'); >> fplot (f,[-2,2]), grid on

Choisir les intervalles : (Chaque intervalle [a,b] est choisie tel que f(a).f(b)<0)

Il existe 3 racines, donc on choisit 3 intervalles

I_1 = [-2, -1.5] Pour la première racine >> [Xzero, N, E, fzero] = bissection (-2,-1.5,0.001)

I_2 = [-1.5, -0.5] Pour la deuxième racine >> [Xzero, N, E, fzero] = bissection (-1.5,-0.5,0.001)

I_3 = [1.5, 2] Pour la troisième racine >> [Xzero, N, E, fzero] = bissection (1.5,2,0.00000001)

TP N°1
Méthodes numériques (Résolution de l'équation *f*(*x*)=*0*)
La Méthode de Newton

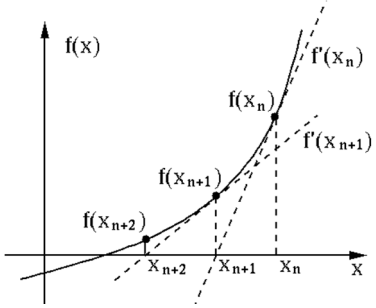
I. La méthode de Newton

Cette méthode numérique est la meilleur méthode utilisée pour résoudre une équation non linéaire du type $f(x) = 0$ ($\Re \Rightarrow \Re$) parce que elles est simple et rapide, le seule inconvénient est qu'elle utilise la fonction $f(x)$ et sa première dérivé $f'(x)$.

I. L'algorithme de la méthode de Newton

f : la fonction concernée
df : la fonction dérivée de f
x_o : le point initial
k : le nombre d'itérations effectuées.
$\in$: le critère d'arrêt (erreur tolérée).
k_{max} : le nombre maximal d'itérations
x_{zero} : la racine trouvée par la méthode

On commence par x_o ($k = 0$) ;
On calcule $\Delta x = \left|\frac{f(x_o)}{f'(x_o)}\right|$;
Tant que ($\Delta x > \in$),
    On calcule : $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$;
    Calculer de nouveau $\Delta x = |x_{k+1} - x_k|$;
    On définit $k \leftarrow k + 1$;
    Si ($k >= k_{max}$), alors la boucle "tant que" se **termine**
    fin si
    affichez : (***$\bar{x}$ et f($\bar{x}$) pour chaque itération*** **)**
fin Tant que
$x_{zéro} = x_k$;



II. Application

1. Implémenter la méthode de Newton sous forme d'une fonction de Matlab:

function *[xzero,fzero,err]=newton(f, df, kmax, Xo, e)*

Avec

les arguments d'entrée	**les arguments de sortie**
f : la fonction à trouver les zéros (à entrer comme un string - entre guillemet ' '). ***df*** : le dérivé première de la fonction '***f***' (à entrer comme un string - entre guillemet ' '). **kmax** : le nombre maximal d'itérations. ***Xo*** : la valeur initiale de la racine. ***e*** : la valeur admissible de l'erreur de la solution **x** (de l'ordre de 1e - 5).	***xzero***, ***fzero*** : le racine à trouver et sa valeur dans la fonction. ***err*** : valeur absolue de l'erreur finale ($abs(\Delta x)$).

2. Appliquer la fonction produite pour trouver le zéro de la fonction suivante :

$f(x) = x^2 + 4x + 3 + \sin(x) - x\cos(x)$; **<u>à implémenter dans une fonction séparée</u>**

avec sa dérivé :

$f'(x) = 2x + 4 + x\sin(x)$ **<u>à implémenter dans une fonction séparée</u>**

3. Produire le tableau des valeurs progressives (x,$f(x)$) jusqu'au zéro.
4. Tracer cette fonction en indiquant dans le graphe le zéro trouvé.

RM

Utiliser la command **feval()** qui exécute une fonction spécifiée par un string pour une valeur donnée :

Ex

```
>>feval('sin',pi/2)
  ans= 1
```

```
function [xzero,fzero,dx]=newton(f,df,kmax,Xo,e)
k = 1; X(1) = Xo;
dx = abs(- (feval(f,X(1)))./(feval(df,X(1))));

while (abs(dx) > e)
    X(k+1) = X(k) - (feval(f,X(k)))/(feval(df,X(k)));
    dx = abs(X(k+1) - X(k));
    xzero = X(k+1);
    fzero = feval(f,xzero);
% produit le tableau des valeur progressive
    disp(['itération N°= ' num2str(k) ', xzero = ' num2str(xzero) ', fzero =' num2str(fzero)]);
    k = k + 1;
    if (k >= kmax)
      disp('Pas de convergence avec le nombre d''itérations indiqués')
      break
    end
end

% tracer cette fonction indiquant dans le graphe le zeron
l=abs(Xo-xzero);
fplot(f,[xzero-l xzero+l]);
xlabel('x')
ylabel('la fonction f(x)')
text(xzero,fzero,'le zero calculé');
hold on
plot(xzero,fzero,'ro')
grid on
```

```
%La fonction fun
function y = f(x);
y=x.^2+4*x+3+sin(x)-x.*cos(x);
```

```
%Le dérivé de la fonction fun
function y = df(x)
y = 2*x+x.*sin(x)+4;
```

```
%Exécution
>>[xzero,fzero,dx]=newton('fun','dfun',10,0,1e-5)
```