Universite Batna 2
Faculté de Technologie
Département D'hydraulique

2ème Année ST_ Hydraulique
TP Méthodes numériques S4
Année 2019/2020

TP N° 1

A.Ameziane 2019/2020

Résolution d'équations non linéaires

(1-Méthode de la bissection 2- Méthode des points fixes 3- Méthode de Newton-Raphson)

1. Le But du TP

Ce TP a pour objectif de vous faire découvrir les méthodes numériques, en particulier les méthodes de la résolution d'équation non linéaires de la forme F(x)=0 :

- La méthode de la bissection
- méthode des points fixes
- Et La La méthode de newton- raphson

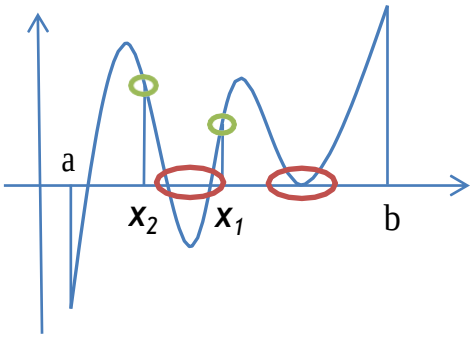
En vue de l'implémentation du code de chaque méthode sous un langage de programmation scientifique aduéquat d'un point de vue numérique commme fortran ,matlab,cilab ,maple et autres et leurs exploitation dans le domaine de l'hydralique.

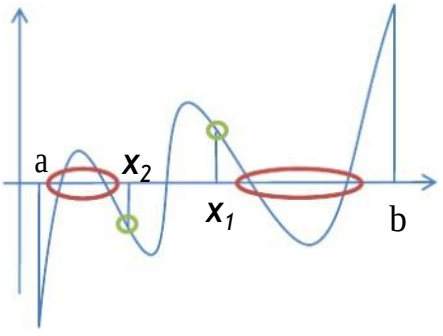
2. La méthode de la bissection (Dichotomie)

La **méthode** de la **bissection** ou **méthode** de dichotomie est, en mathématiques, un **algorithme** de recherche d'un zéro d'une fonction qui consiste à répéter des partages d'un intervalle en deux parties puis à sélectionner le sous-intervalle dans lequel existe un zéro de la fonction comme expliquer dans le principe ci-dessous.

2.1. Principe de la méthode (Recherche d'un point unique d'abscisse x dans un intervalle? qui satisfait f(x)=0)

Cette méthode repose sur le théorème des valeurs intermédiaires, basée sur le principe que pour toute f une fonction monotone sur l'intervalle [a, b].





1- Supposons que f(a) * f(b) <0.
2- On calcule $M = f\left(\frac{a+b}{2}\right)$.

Si M est du même signe que f(a) alors la racine se trouve dans l'intervalle $\left[\frac{a+b}{2}, b\right]$ autrement elle se trouve dans l'intervalle $\left[a, \frac{a+b}{2}\right]$.

On répète le processus sur ce nouvel intervalle jusqu'à ce que la précision soit suffisante

(le processus converge vers une solution après n étapes).

- **La convergence et le critère d'arrêt**

  Si le processus de dichotomie arrive jusqu'à l'étape n alors on a l'estimation:
  $\alpha_{solution} - ((a+b)/2)^n | \leq \frac{b-a}{2^{n+1}}$. Avec la suite $C_n = ((a+b)/2)^n$

  Par conséquent, la suite (C_n) converge vers $\alpha_{solution}$.

  C'est aussi vrai si $(C_n) = \alpha_{solution}$.

a. Décrire l'algorithme détaillé à la méthode de la bissection
b. Traduire l'algorithme dans un langage de programmation (FORTRAN, MATLAB, MAPLE …..) pour avoir un code de la méthode à exécuté
c. Résoudre l'équation $(x) = x + 2^{\log(x)} = 0$ dans l'intervalle itilisable par dichotomie à cinquième décimale prés.
d. Afficher le nombre d'itérations ´effectuées pour obtenir le résultat souhaité.

---

3. La méthode des points fixes (Itératives)

3.1. Principe de la méthode

Le principe de la méthode du point fixe consiste à transformer, la fonction *f(x)* = 0,*f* : [*a b*] → *R*, en une fonction *g*(x) =x . La fonction *g* : [a *b*] → *R*, est construite de façon à ce que *g*(∝) = ∝ quand *f*(∝) = 0. Trouver la racine de f (x), se résume donc à déterminer un ∝ ∈ [*a b*] tel que :

$$\propto = g(\propto)$$

Dans le cas où un tel point existe, il sera qualifié de point fixe de *g* et cette dernière est ditefonction d'itération. Le schéma numérique de cette méthode est donné par :

$$X^{k+1} = g(x^k) \quad pour \quad k \geq 0$$

Le vecteur erreur e_n est calculé à partir de : $e_n = |x_n - x_{app}|$ Avec, x_{app} est la solution approchée, de la valeur exacte, déterminée avec une tolérance fixée préalablement. , étant le nombre d'itérations. Par ailleurs, l'estimation de l'erreur servira, entre autre, à comparer la vitesse de convergence pour des méthodes numériques différentes. Sur le plan pratique, l'erreur est représentée graphiquement en traçant e_{n+1} en fonction de e_n avec une échelle logarithmique. Ainsi, l'ordre noté , d'une méthode numérique s'obtient à partir de :

$$|e_{n+1}| \approx A|e_n|^p \Rightarrow log|e_{n+1}| \approx P\log|e_n| + \log(A)$$

Ainsi l'ordre P est quantifié via la pente de l'équation ci-dessus. On en déduira que:

1. Si P = 1 ⇒ $x(n)$ converge linéairementvers la solution approchée. Dans ce cas on gagne la même quantité de précision à chaque itération.
2. Si = 2 ⇒ $x(n)$ converge quadratiquementvers la solution approchée. Dans ce cas on gagne le double de précision à chaque itération.
3. Si = 3 ⇒ $x(n)$ converge cubiquementvers la solution approchée. Dans ce cœs on gagne le triple de précision à chaque itération.

a. Écrire un algorithme en spécifiant en spécifiant les paramètres de la fonction solution du schéma numérique de cette méthode.

b. Écrire un programme dans un langage numérique permettant l'implémentation de l'algorithme de cette méthode.

c. Valider votre programme par l'exemple f(x)= x - cos(x) dans l'intervalle [-1/2, 3] , x0=0.8, tol= e^{-5} et nombre itérations=30.

---

4. La méthode de Newton Raphson

4.1. Principe de la méthode

Comme il a été montré précédemment, la méthode de dichotomie exploite uniquement le signe de la fonction f aux extrémités des sous-intervalles. Lorsque cette fonction est différentiable, on peut établir une méthode plus efficace en exploitant les valeurs de la fonction f et de ses dérivées.

Dans ce contexte La méthode de Newton-Raphson consiste à trouver la valeur x qui annulera la fonction f(x).

La méthode de Newton-Raphson permet d'approcher par itérations la valeur x au moyen de la relation suivante : $f(x_{n+1})=f(x_n) - \frac{f(x_n)}{f'(x_n)}$

le principe de la méthode de trouver la valeur x qui annulera la fonction f(x).

Si $|X_n - X_{n-1}| < \xi$ alors X_n est le résultat de l'estimation de la racine. Où ξ représentent des erreurs d'approximations caractérisant la qualité de la solution numérique. Ce critère d'arrêt a l'avantage d'éviter une possible division par 0. Dans toutes les méthodes itératives, il est nécessaire pour éviter une divergence de la solution, de bien choisir la valeur initiale x_0. Celle-ci peut être obtenue graphiquement.

a. Développer l'algorithme pour la méthode de newton raphson

b. Ecrire le code du programme de la méthode raphson dans un langage de programmation scientifique

c. Traiter un exemple suivant le code implémenté.

4.2. Proposition de solution du TP pour la Méthode de newton Raphson

4.2.1. Algorithme de newton Raphson

Debut Newton_raphson

Paramètres d'entrées :p0, Epsilon , N
p0 : approximation initiale
Epsilon : la précision désirée
N0 le nombre maximum d'itérations
Paramètres de sorties : p
P valeur approchée où un message d'échec
Trouve← false
N ← 1
Tant que (N ≤ N0) and trouve= false faire ´

P← $p0 - \frac{f(p0)}{f'(p_0)}$

Si | p − p0 | > Epsilon Alors
N ← N+1
P0=p
sinon
Trouve=true
Imprimer ' approximation est ' , lp
Finsi

Fin Tant que

Si trouve= false Alors
Imprimer ' la méthode échouée après N itérations
Fin Si

Fin Newton_Raphson

4.2.2. Le code matlab de la méthode de Newton Raphson

On se propose d'appliquer cette méthode pour la recherche des racines de la fonction non linéaire suivante : $f(x)=e^x -2\cos(x)$

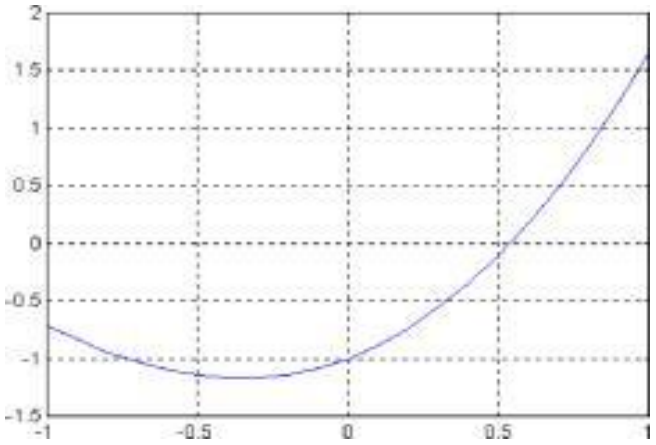
Dans un premier temps, on se propose de tracer la courbe représentative de cette fonction en utilisant le programme ci-dessous :

```
% Etude de la fonction :

% f (x) =exp (x) -2 . cos (x)
x=−1: 0.1:1;
f=exp(x)-2*cos(x);
plot(x,f); grid on;
title('Fonction : f(x)=exp(x)-2.cos(x)   ) ;
```

Après exécution du programme, on obtient la courbe Ci-dessous. D'après cette courbe, x_0 doit égale à 0 ,5 x_0 =0 ; car f(0 ,5) est proche de zéro pour avoir une convergence rapide. La fonction dérivée f (x) a pour expression : f (x) $=e^x$+2 sin (x).

Fonction : f(x)=exp(x)-2.cos(x)



Repérer la valeur de x_0 où $f(x_0)= 0$

Pour chercher la solution de f(x), on peut rajouter au programme précédent 'NewtonRaphson.m' quelques lignes :

```
Le Code Matlab de la Méthode newton raphson

% Etude de la fonction :

% f (x) =exp (x) -2 . cos (x)
cl f ;
x=-1: 0.1:1;
f=exp(x)-2*cos(x);
figure(1);
plot(x,f); gridon;
title('Fonction : f(x)=exp(x)-2.cos(x)' ) ;
clear all
ele;
x(1)=input(1 Donner la valeur initiale x(l): X n1);
e=le-10;
n=5000;
for i=2:n
     f=exp(x(i-l))-2*cos(x(i-l));
     diff=exp(x(i-l))+2*sin(x(i-l));
     x(i)=x(i-1)-f/diff;
     if abs(x(i)-x(i-1))<=e
          xp=x(i);
          fprintf('xp=% f\n',x(i) );
          break;
     End
 End
 j=l:i;
 figure(2) ;
 plot(j,x(j), '*r',j,x(j) ) ;
 xlabel('Nombre d''itérations');
 title('Convergence de la solution : Méth. de Newton.-Raphson.'
) ;
 disp('Les valeurs successives de x(i) sont :');
x'
```

4.2.3 Le Code Maple de la méthode de Newton Raphson

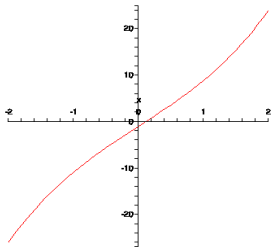
Soit $h(x) = x + 8x + \sin(x) - 1$

Calculer h'(x) et h"(x) et tracer les sur un même diagramme sur l'intervalle [-2, 2].

Implémenter la méthode de Newton-Raphson pour déterminer la racine de h avec une précision de 10^ (-20) près.

> **h := x -> x^3+8*x+sin(x)-1;** $h := x \to x^3 + 8\,x + \sin(x) - 1$

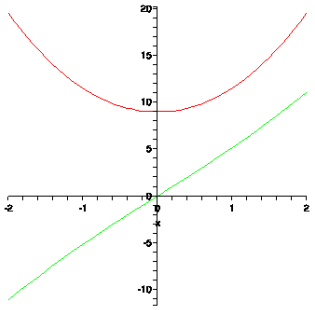
> **plot(h(x),x=-2..2);**



> **hp := D(h); hpp := D(hp);**

$hp := x \to 3\,x^2 + 8 + \cos(x)$ $hpp := x \to 6\,x - \sin(x)$

> plot([hp(x), hpp(x)],x=-2..2);



> **hp(0));evalf(hpp(2));** 9. 11.09070257

> **m1:= 9; M2:=11.1;** $m1 := 9$ $M2 := 11.1$

>

m1^2/(4*M2) ;

1.824324324

> **evalf(h(0));**

```
> Newton_Raphson := proc (f, x0, prec)
  local fp, a, b;
  fp := D(f);
  b:=x0;
  a:=b-evalf(f(b))/evalf(fp(b));
  while abs(a-b) >prec do
  b := a;
  a:=b-evalf(f(b))/evalf(fp(b));
  end do;
  return a;
  end proc:
```

> r := Newton_Raphson(h, 0, 10^(-20)); $r := 0.1109845160$

> evalf(h(r)); $-1\ 10^{-10}$

4.2.4 Le Code Fortran de la méthode de Newton raphson

Le programme Fortran d'implémentation de l'algorithme de newton raphson est donné selon le code ci_dessous ; pour déterminer les racines réelles de l'équation F(x)=cos(x)-x =0.

```
      PROGRAM NEWTON_RAPHSON_ REAL x( 10000),t,x0
c     ******** La Fonction F(x):
      F(t)=COS(t)-t
c     **********La Derivée F'(x):
      FP(t)= -SIN (t)-1.
      Print1,'DONNER a et b LE DEBUT ET LA FIN DE L
      INTERVALLE [a,b]'
      Read*, a,b
      IF(F(a)*F(b).GE.0.)then
      print*, 'LA RACINE N APPARTIENT PAS A CET INTERVALLE'
      ELSE
      print*, 'DONNER (x0) L APPROXIMATION INTIALE'
      read*, x0
      print*, 'DONNER (n) LE NOMBRE D ITERATION '
      print*, 'n: il faut egale a un nombre entier positif'
      read*, n
      print*,' _____ '
      print*,'METHODE DE NEWTON RAPHSON:'
      print*, 'k', '                    ','x(k)'
      x( 1)=x0
      DO 10 k=1, n
      x(k+1)=x(k)-F(x(k))/FP(x(k))
      print*, k,x(k)
10    IF(x(k+1).EQ.x(k)) THEN
      PRINT*, 'LA SOLUTION DE NEWTON RAPHSON ='
      PRINT*, 'iteration k=',k,' ', 'eta=',x(k+1)
      exit
      END IF
      Continue
      STOP
      End
```

En Conclusion

a. Evaluer le nombre d'opérations éffectuées par chaque algorithme.

b. Ccmparer les trois méthodes, est déduire la méthode la plus performante.