Alves Vincent
Jarret Guillaume
Analyse Numérique - Groupe B10

Calcul des Zéros des Fonctions

7/01/04

Table des matières

1 Introduction

L'objectif de ce TP d'Analyse Numérique est d'écrire l'algorithme de la méthode de Newton-Raphson pour le calcul des racines des fonctions, et de l'appliquer pour calculer la valeur du minimum d'une fonction $f(x)$ avec $x \in \mathbb{R}$.

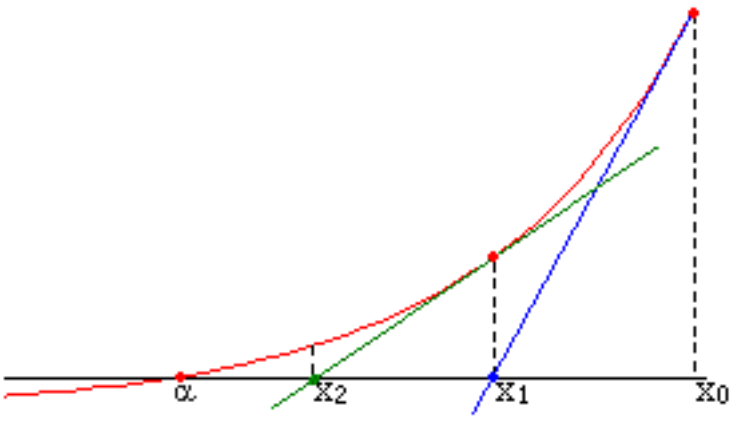
2 Présentation mathématique du problème

2.1 Méthode de Newton-Raphson

La méthode de Newton-Raphson est un algorithme permettant de trouver numériquement la racine d'une fonction $f(x)$ avec $x \in \mathbb{R}$.

Soit $f : D \to \mathbb{R}$ une fonction avec D inclus dans $\mathbb{R}$. Si nous voulons calculer la racine α de cette fonction, l'algorithme à utiliser suit les étapes suivantes :

- Choisir un point $x_0 \in D$ pas trop éloigné de la racine que l'on recherche.
- Calculer $f(x_0)$ et $f'(x_0)$.
- Trouver le point d'intersection de la tangente de f en x_0 avec l'axe des ordonnées. On nommera ce point x_1.
  Pour ce faire, résoudre l'équation affine $f(x_0) + f'(x_0)(x_1 - x_0) = 0$
- Le point x_1 obtenu est plus proche de la racine α. En répétant depuis l'étape 2, on peut se rapprocher peu à peu de la véritable racine α.



Exemple d'utilisation de la méthode de Newton-Raphson.

Cet algorithme est assez simple à créer informatiquement, mis à part pour le calcul de la fonction dérivée de f, f'.

2.2 Obtention de la dérivée de f

2.2.1 Méthode analytique

La première méthode que nous pouvons employer est de calculer nous-mêmes la fonction dérivée de f et de l'utiliser telle quelle dans l'algorithme.

2.2.2 Méthode par l'approximation de Taylor

Pour calculer la dérivée de f, nous pouvons également utiliser l'approximation de Taylor au premier ordre :

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

Avec h très petit(de l'ordre de 0.1).

2.3 Application de Newton-Raphson au calcul du minimum d'une fonction

Si l'on cherche le minimum $x^* \in \mathbb{R}$ de $f(x)$, alors on a :

$$f'(x^*) = 0$$

Donc, pour chercher le minimum de $f(x)$, il nous suffit de calculer sa dérivée $f'(x)$, puis d'y appliquer l'algorithme de Newton-Raphson tel que nous l'avons vu ci-dessus. //La racine de $f'(x)$ alors obtenue sera le minimum de la fonction $f(x)$.

2.4 Conditions suffisantes à l'application de la méthode

Pour s'assurer que la méthode de Newton-Raphson converge correctement vers la racine de la fonction sur un intervalle $[a, b]$, plusieurs conditions doivent être réunies :

- f doit être C_1 sur $[a, b]$
- $f'(x) \neq 0$ et $f''(x) \neq 0$ sur $[a, b]$

Vu que pour trouver le minimum d'une fonction sur $[a, b]$ on applique Newton-Raphson à la dérivée de cette fonction, cette dernière doit donc être C_2.

3 Réalisation informatique

Nous avons utilisé le programme Scilab pour réaliser le code suivant.

3.1 Fonction NewtonRaphson

```
//Implémentation de Newton-Raphson. Retourne la racine de f.
//fct=f(x), dfct = f'(x);
function eps=NewtonRaphson(fct,x0,ctr,dfct)
    derx0=dfct(x0); //Nous utilisons la fonction dérivée de f.
    x1= 1/derx0 * (x0*derx0 - fct(x0)); //Calcule x1
    if fct(x1) < 1*10^(-7) then
        eps = x1; //Si x1 est assez proche de 0, nous le renvoyons.
        disp("Nombre d''itérations");
        disp(ctr); //Affichage du nombre d'itérations de l'algorithme.
        disp("Résultat");
        disp(eps); //Affichage du résultat.
    else
        eps=NewtonRaphson(fct,x1,ctr+1,dfct); //Sinon, récursion sur x1.
    end
endfunction

//Implémentation de Newton-Raphson. Retourne la racine de f. Version non analytique.
//fct=f(x).
function eps=NewtonRaphsonTaylor(fct,x0,ctr)
    derx0=taylorApprox(fct,x0,0.1); //Nous utilisons Taylor avec h=0.1
    x1= 1/derx0 * (x0*derx0 - fct(x0)); //Calcule x1
    if fct(x1) < 1*10^(-7) then
        eps = x1; //Si x1 est assez proche de 0, nous le renvoyons.
        disp("Nombre d''itérations");
        disp(ctr); //Affichage du nombre d'itérations de l'algorithme.
        disp("Résultat");
        disp(eps); //Affichage du résultat.
    else
        eps=NewtonRaphsonTaylor(fct,x1,ctr+1); //Sinon, récursion sur x1.
    end
endfunction
```

Avec la fonction que nous utilisons pour l'approximation de Taylor :

```
function y=taylorApprox(fct, x, h)
  //Calcule l'approximation de Taylor au premier ordre de la fonction fct.
  //Renvoie f'(x).
  y = [fct(x+h)-fct(x)]/h;
endfunction
```

La fonction NewtonRaphson, dans ses deux versions, est une fonction récursive.
Elle calcule x_1 en appliquant la formule $x_1 = 1/f'(x_0) * (x_0 * f'(x_0) - f(x_0))$.
Ensuite, elle vérifie si $f(x_1)$ est assez proche de 0, e.g si x_1 est assez proche de la racine α.
Si oui, nous retournons x_1.
Sinon, nous rappelons NewtonRaphson, cette-fois-ci avec x_1 en paramètre au lieu de x_0.

3.2 Adaptation de NewtonRaphson au calcul du minimum

Pour calculer le minimum de f, il suffit de calculer la racine de f'.
D'ou la fonction Scilab suivante :

```
//Application de Newton-Raphson au calcul du minimum d'une fonction.
//dfct = f'(x), ddfct= f''(x).
function min=MinimumNR(dfct,x0,ddfct)
    //Le minimum de f(x) est la racine de f'(x).
    min=NewtonRaphson(dfct,x0,0,ddfct);
endfunction

//Application de Newton-Raphson au calcul du minimum d'une fonction.
//dfct = f'. Version non analytique.
function min=MinimumNR(dfct,x0)
    //Le minimum de f(x) est la racine de f'(x).
    min=NewtonRaphsonTaylor(dfct,x0,0);
endfunction
```

3.3 Programme principal

Une fois ces fonctions codées, nous pouvons tester Newton-Raphson et la recherche de minimum sur les fonctions de test données. Pour ce faire, nous définissons les fonctions de test :

```
function y = fa(x) //Fonction de test 1
    y = x^2 - 5;
endfunction

function y = fad(x) //Dérivée de la fonction de test 1 (et 2)
    y = 2*x;
endfunction

function y = fb(x) //Fonction de test 2
    y = x^2 + 17;
```

```
endfunction

function y = fc(x) //Fonction de test 3
    y = x-2*sin(x);
endfunction

function y = fcd(x) //Dérivée de la fonction de test 3
    y = 1-2*cos(x);
endfunction

function y = fd(x) //Fonction de test 4
    y = sin(3*x)+ x^2 - 2;
endfunction

function y = fdd(x) //Dérivée de la fonction de test 4
    y = 3*cos(3*x)+2*x;
endfunction

function y = fddd(x) //Dérivée seconde de la fonction de test 4
    y = 2-9*sin(3*x);
endfunction

function y = fe(x) //Fonction de test 5
    y = x*cos(x);
endfunction

function y = fed(x) //Dérivée de la fonction de test 5
    y = cos(x)-x*sin(x);
endfunction

function y = fedd(x) //Dérivée seconde de la fonction de test 5
    y = -2*sin(x)-x*cos(x);
endfunction
```

Par la suite, pour calculer la racine d'une de ces fonctions, nous appelons :

```
//Affiche la racine calculée via Taylor, x0=1
NewtonRaphsonTaylor(fa,1,0);
//Affiche la racine calculée par méthode Analytique, x0=1
NewtonRaphson(fa,1,0,fad);
```

En remplacant fa/fad par fc/fcd, fd/fdd, etc.

4 Résultats obtenus

4.1 Calcul de racines : Fonctions de test

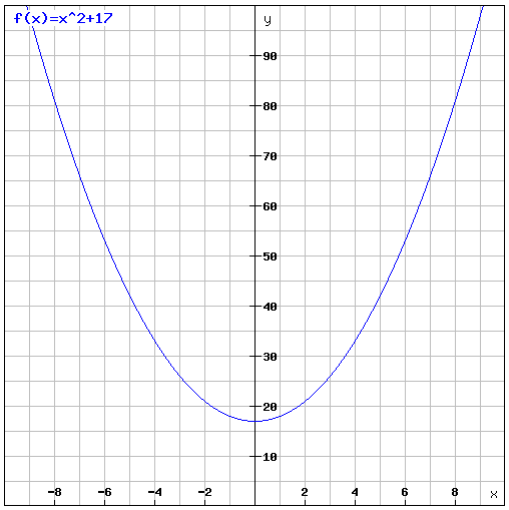
4.1.1 $f(x) = x^2 - a$

Pour a = 5.

```
-->NewtonRaphsonTaylor(fa,1,0);

 Nombre d'itérations
    6.

 Résultat
    2.236068

-->NewtonRaphson(fa,1,0,fad);

 Nombre d'itérations
    4.

 Résultat
    2.236068

-->NewtonRaphsonTaylor(fa,-1,0);

 Nombre d'itérations
    3.

 Résultat
  - 2.2360629

-->NewtonRaphson(fa,-1,0,fad);

 Nombre d'itérations
    4.

 Résultat
  - 2.236068
```

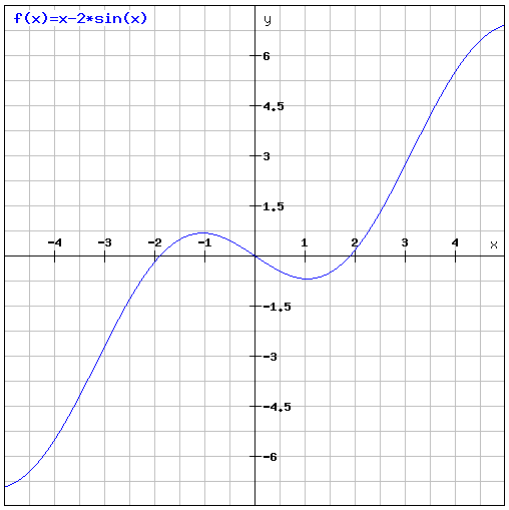
En prenant $x_0 = 1$, nous obtenons une des deux racines de la fonction.
En prenant $x_0 = -1$, nous obtenons l'autre racine.
Peu importe la méthode utilisée, nous obtenons les mêmes racines.

4.1.2 $f(x) = x^2 + 17$



Comme nous pouvons le voir sur le graphique, cette fonction n'a pas de racine.
Dans ce cas, l'algorithme de Newton-Raphson n'aboutit pas, et nous renvoie l'erreur "Récursivité trop complexe !"

4.1.3 $f(x) = x - 2sin(x)$



A voir le graphique, nous pouvons conjecturer que selon la valeur de x_0, nous obtiendrons une des deux racines de la fonction. ($\approx$ -1.8 ou 1.8).
En appliquant notre algorithme dans Scilab, la sortie est la suivante :

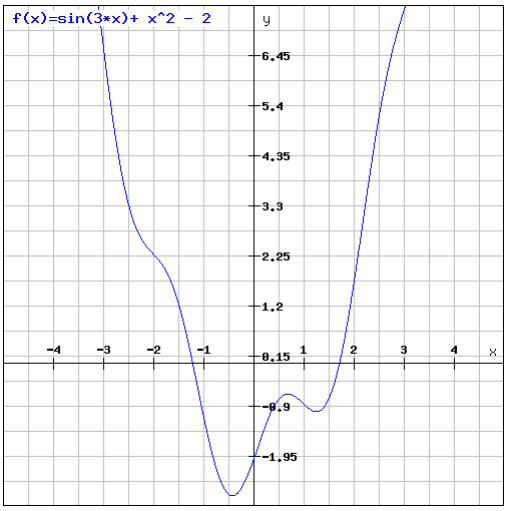
```
-->NewtonRaphsonTaylor(fc,3,0);

 Nombre d'itérations
    6.

 Résultat
    1.8954943

-->NewtonRaphson(fc,3,0,fcd);

 Nombre d'itérations
    3.

 Résultat
    1.8954943

-->NewtonRaphsonTaylor(fc,-3,0);

 Nombre d'itérations
    0.

 Résultat
  - 2.0826394

-->NewtonRaphson(fc,-3,0,fcd);

 Nombre d'itérations
    0.

 Résultat
  - 2.0879954
```

Nous obtenons des racines cohérentes avec $x_0 = 3$ et $x_0 = -3$.
Les racines obtenues via les deux méthodes sont très similaires.
(Identiques à $x_0 = 3$, différence de l'ordre de 10^{-3} à $x_0 = -3$)

A $x_0 = -3$, les racines obtenues sont assez loin de la véritable racine. Cela est dû à la manière dont notre algorithme vérifie si le chiffre obtenu est la racine. Il serait possible de se rapprocher en rendant la vérification plus restrictive, mais cela pourrait causer des erreurs avec d'autres fonctions qui demanderaient trop d'itérations pour passer cette vérification.

4.2 Calcul de minimum : Fonctions de test

4.2.1 $f(x) = sin(3x) + x^2 - 2$



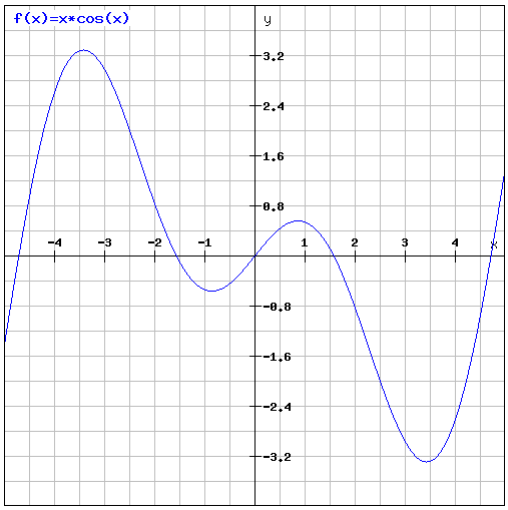
Pour calculer le minimum, nous appelons les fonctions MinimumNR et MinimumNRT. La sortie Scilab est la suivante :

```
-->MinimumNR(fdd,-0.75,fddd);

 Nombre d'itérations
    1.

 Résultat
  - 0.4287655

-->MinimumNRT(fdd,-0.75);

 Nombre d'itérations
    1.

 Résultat
  - 0.4291101

-->MinimumNR(fdd,1,fddd);

 Nombre d'itérations
    5.

 Résultat
    0.6805763
```

```
-->MinimumNRT(fdd,1);

 Nombre d'itérations
    9.

 Résultat
    1.2445903
```

On voit ici que selon le x_0 fourni (soit -0.75, soit 1 dans nos tests), le minimum renvoyé par Scilab est différent. Cependant, on peut voir qu'ils correspondent à des minimums locaux, comme 1.2445903 sur l'intervalle $[0, +\infty]$. On obtient bien le minimum de la fonction avec $x_0 = -0.75$ avec les deux méthodes, avec une légère différence.

4.2.2 $f(x) = xcos(x)$



On appelle à nouveau MinimumNR et MinimumNRT, avec x_0 égal à -1 et 3, deux nombres proches de minimums locaux de la fonction. La sortie scilab est la suivante :

```
-->MinimumNR(fed,-1,fedd);

 Nombre d'itérations
    0.

 Résultat
  - 0.8645364
```

```
-->MinimumNRT(fed,-1);

 Nombre d'itérations
    0.

 Résultat
  - 0.8617125

-->MinimumNR(fed,3,fdd);

 Nombre d'itérations
    9.

 Résultat
    3.4256185

-->MinimumNRT(fed,3);

 Nombre d'itérations
    5.

 Résultat
    3.4256185
```

Comme pour la fonction précédente, selon le x_0 renseigné, nous obtenons différents minimums locaux de la fonction(-0.86 et 3.42). Nous trouvons également le minimum de la fonction avec $x_0 = 3$. Une fois de plus, il n'y a pas de grande différence entre les deux méthodes utilisées.

5 Conclusions

5.1 Méthode de Newton-Raphson pour le calcul de racines

Pour calculer les racines d'une fonction, il est nécessaire de renseigner des x_0 proches des racines que nous cherchons. Pour des fonctions possédant plusieurs racines, il faut essayer plusieurs x_0. La méthode est assez précise, qu'on utilise la méthode analytique ou l'approximation via Taylor.

5.2 Méthode de Newton-Raphson pour le calcul du minimum

Pour le calcul de minimus, Newton-Raphson ne fonctionne pas très bien, car elle tend à donner des minimus locaux de la fonction selon le x_0 fourni. On peut alors difficilement déterminer quel est le vrai minimum de la fonction. Il faut essayer des valeurs différentes et ensuite tester soi-même pour voir quel minimum local est le minimum global de la fonction.

Cependant, la méthode donne des minimums assez précisément, si l'on renseigne des bonnes valeurs de x_0.

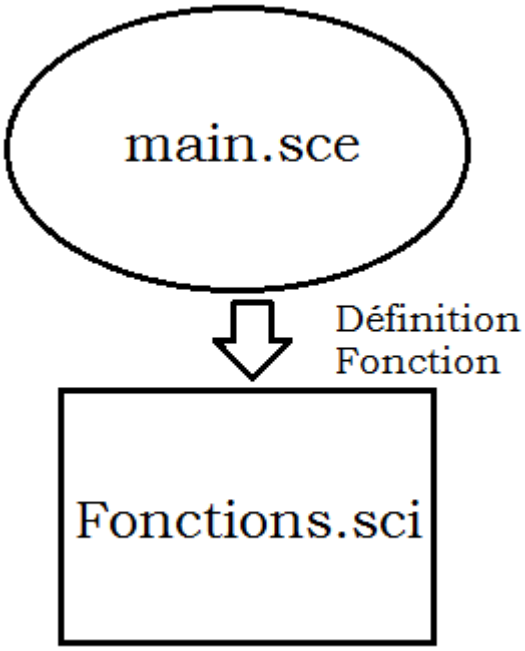
5.3 Conclusion générale

La méthode de Newton-Raphson est assez précise pour le calcul de racines, beaucoup moins pour le calcul de minimums.
De plus, la méthode en elle-même est assez limitée par la nécessité de renseigner soi-même des points de départ. Si ces points sont mal renseignés, l'algorithme a de très fortes chances de ne pas aboutir sur un résultat correct.
Cependant, la dérivation de la fonction par l'approximation de Taylor fonctionne bien, ce qui élimine la nécessité d'entrer les fonctions dérivées nous-mêmes.

6 Reférences

Support de cours Scilab, Cycle ingénieur, EISTI Première année.
Internet : `http://fr.wikipedia.org/wiki/Méthode_de_Newton`

7 Annexe A - Organigramme du programme



8 Annexe B - Guide d'utilisation du programme

- Dans Scilab, sélectionner **Fichier** → **Changer le répertoire courant**
- Sélectionner le répertoire dans lequel ont été extraits les fichiers Fonctions.sce et main.sce
- Entrer dans scilab **exec('main.sce');**
- Le programme executé affiche les racines pour les fonctions 1 et 3, et les minimum pour les fonctions 4 et 5 via les deux méthodes, Taylor et analytique.