Rapport : calcul des zéros d'une fonctions

Thibault Gimenez et Allan Sforza : binome 09

24 mars 2014

Table des matières

A Introduction **3**

B Méthode mathématique **4**
B - 1 Calcul des zéros par la méthode de Newton-Raphson 4
B - 2 Conditions suffissantes 4

C Réalisation informatique **5**
C - 1 Outil utilisé 5
C - 2 Algorithme de la méthode de Newton-Raphson par la formule analytique et approchee de la derivée 5
C - 3 Test sur trois fonctions afin de trouver leurs racines 5
C - 4 Test sur deux fonctions après amélioration de l'algorithme afin de trouver l'extrémun d'une fonction 8
C - 5 Conclusion 10

D Annexe **11**
D - 1 Annexe 1 : fonction scilab 11
D - 2 Annexe 2 : fonction scilab amélioré 11
D - 3 Annexe 3 : comment utiliser le programme 12

A Introduction

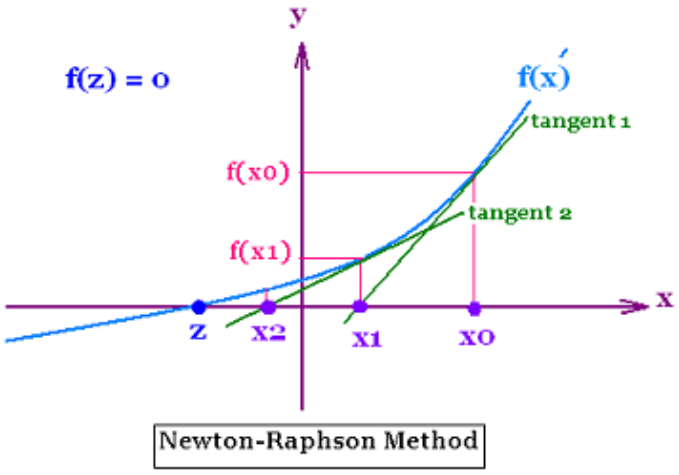
Dans ce rapport nous allons montrer comment calculer le minimal d'une fonction à une variable, non linéaire, réelle, en utilisant la méthode de Newton-Raphson pour le calcul des racines des fonctions. En analyse numérique, cette méthode est un algorithme efficace pour trouver numériquement une approximation précise d'une racine d'une fonction réelle.

B Méthode mathématique

B - 1 Calcul des zéros par la méthode de Newton-Raphson

Les méthodes de bissection et d'interpolation linéaire propose chacune une manière respective d'obtenir, étape par étape, une valeur approchée d'un zéro réel à localiser. La méthode de Newton-Raphson propose aussi, sous certaines conditions, une manière très différente d'obtenir, étape par étape, une valeur approchée d'un zéro réel.

La méthode de Newton-Raphson est basée sur l'utilisation de la tangente en un point de la courbe d'une fonction f. Plus précisément, le choix d'une première valeur x_0 approchée d'un zéro réel à localiser détermine un premier point (x_0, f(x_0)) sur la courbe qui sera considéré comme un premier point de tangence. Ce nombre x_0 est appelé « amorce » du procédé itératif de Newton-Raphson. L'abscisse x_1 du point d'intersection de la première tangente avec l'axe des x sera considéré comme une deuxième valeur approchée du zéro à localiser. À son tour, cette valeur permettra de considérer un deuxième point de tangence (x_1, f(x_1)). À nouveau, l'abscisse x_2 du point d'intersection de la deuxième tangente avec l'axe des x sera considéré comme une troisième valeur approchée du zéro. En poursuivant ce procédé itérativement, on obtiendra, sous certaines conditions, une séquence de différentes valeurs x_0 , x_1 , x_2 , x_3 , ... qui vont se rapprocher de plus en plus d'un zéro réel de la fonction f.



B - 2 Conditions suffissantes

La méthode de Newton-Raphson fonctionne sous plusieurs conditions qui permettent de justifier qu'il existe un zéro sur un intervalle [a,b] :

- $f(a).f(b) < 0$
- Sur l'intervalle de définition $f'(x) \neq 0$ et $f''(x) \neq 0$
- $f(x_0).f''(x_0) > 0$.

La fonction f est C_1 sur [a,b]. Toutes ces conditions doivent être réunis pour assurer la convergence de la méthode de Newton-Raphson vers la racine de la fonction. Pour trouver le minimum local à une fonction, on applique la méthode

de Newton-Raphson à la dérivé de la fonction. Pour cela la fonction doit donc être C_2.

C Réalisation informatique

C - 1 Outil utilisé

Pour la réalisation informatique, on se sert de Scilab. Le problème de cet outil est que l'on ne peut pas utiliser toutes les fonctions mathématiques existantes. Nous devons donc les créer à l'aide de Scilab. Scilab est un outil de calcul numérique, il ne contient pas d'outil de différentiation formelle. Ceci nuît à l'tilisation de fonctions C_2 et aux conditions de convergence.

C - 2 Algorithme de la méthode de Newton-Raphson par la formule analytique et approchee de la derivée

Fonction NewtonRaphson(f : fonction, x_0 : reel, p : reel, h :reel)
Si abs(f(x_0))>p faire
r=derivee(f,x_0,h)
$x_0 = x_0 - \frac{f(x_0)}{r}$
NewtonRaphson(f,(x_0,p,h) sinon retourner (x_0)
finsi
finfonction
fonction derivee(f :fonction,(x_0 :reel,h :reel))
j=f(x+h)-f(x)
j=$\frac{j}{h}$
retourner j
finfonction

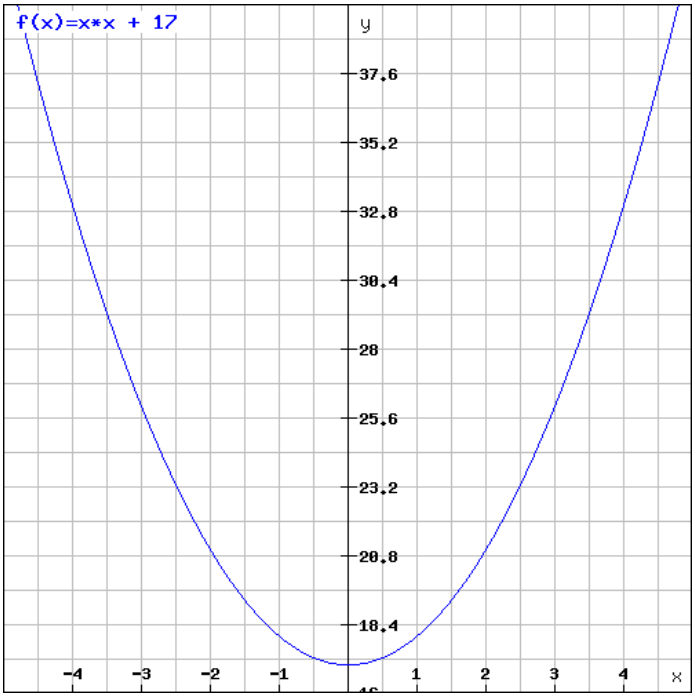
La fonction "dérivée" calcul la dérivé d'une fonction au point x_0 avec la précision h (formule approchée)

C - 3 Test sur trois fonctions afin de trouver leurs racines

$$f(x) = x^2 - a \quad a > 0 \tag{1}$$

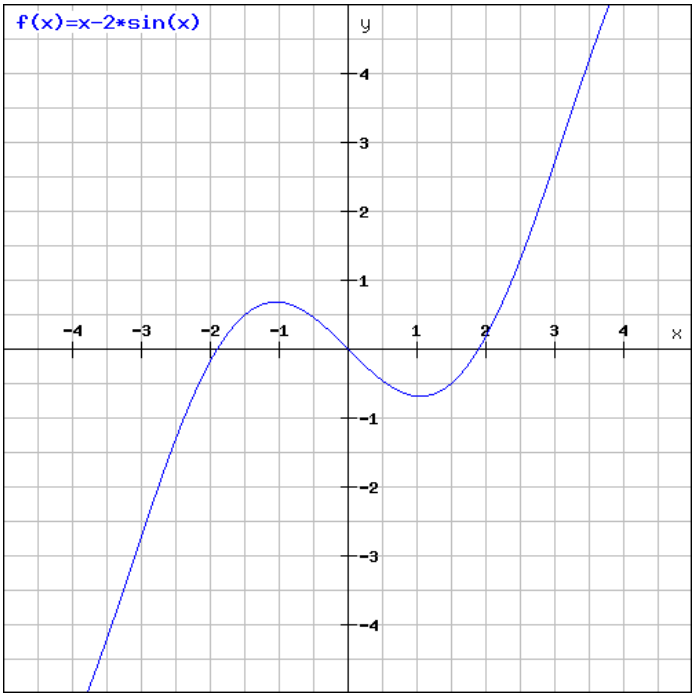
C'est fonction représentative d'une parabole est C_1. Par conséquence on obtient facilement une de ces deux racines. Pour en cibler une, il faut prendre un point de départ proche de celle-ci.
L'algorithme fonctionne pour tout a>0.

$$f(x) = x^2 + 17 \quad \textit{avec pour point initial } x_0 = 1 \tag{2}$$



L'algorithme nous donne un résultat faux pour cette fonction. En effet la fonction ne posséde pas de racine, c'est pourquoi après un certain nombre d'itération fixé l'algorithme donne un résultat aux hasard (sinon l'algorithme tournerais indéfiniment). Le résultat obtenue est de -5,27 pour un nombre d'itérations de 5.

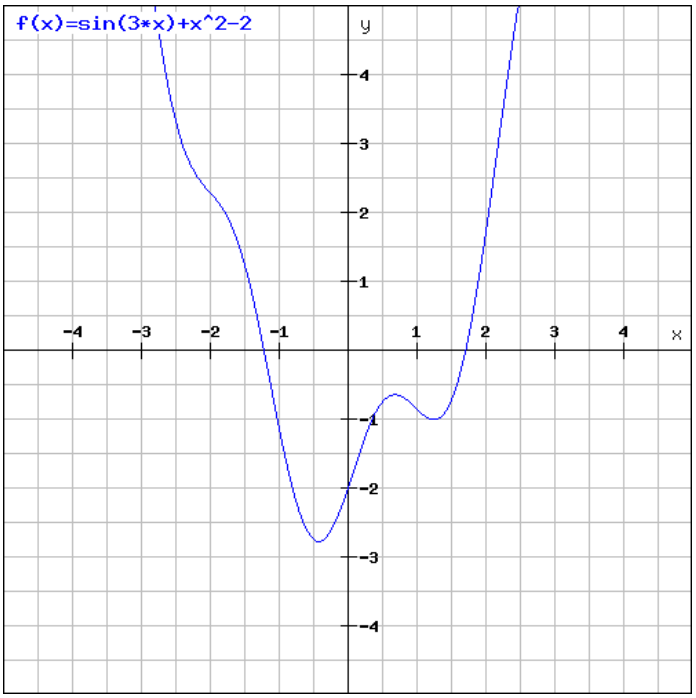
$$f(x) = x - 2sin(x) \quad avec\, pour\, point\, initial\, x_0 = 1,1 \tag{3}$$



L'algorithme ne fonctionne pas du fait d'un message d'erreur indiquant une division par zéro. En effet la fonction ne respecte pas toutes les conditions de la méthode de Newton-Raphston.

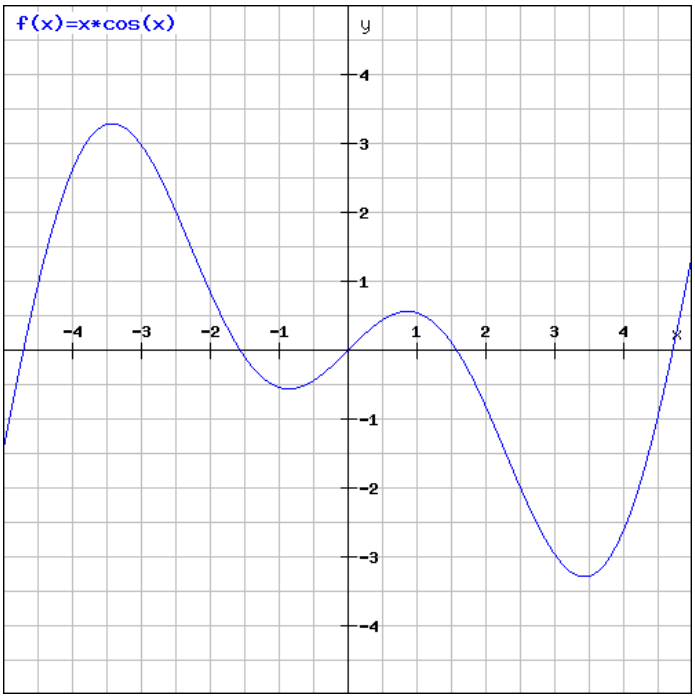
C - 4 Test sur deux fonctions après amélioration de l'algorithme afin de trouver l'extrémun d'une fonction

$$f(x) = sin(3x) + x^2 - 2 \quad avec\, x_0 = 1\, et\, x = [-3, 3] \tag{4}$$



On trouve comme minimun -0,42 avec un nombre d'itération de 50.

$$f(x) = xcos(x) \quad avec\, x_0 = 0.1\, et\, x_0 = 2.5\, et\, x = [-2 * pi, 2 * pi] \qquad (5)$$



On trouve avec un point de départ valant 2,5 et un nombre d'itération de 50, un résultat de 3,4255.
On trouve avec un point de départ valant 0 et un nombre d'itération de 50, un résultat de -0,86.
L'extremun dépend donc du point de départ.

C - 5 Conclusion

Si la forme analytique de $f'(x)$ n'est pas connue, il faut calculer numériquement cette quantité par

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

on doit donc évaluer $f(x)$ 2 fois à chaque itération, l'ordre de convergence va donc passer de 2 à $\sqrt{2}$. De plus, si l'intervalle dx est trop petit, il y a un risque que les erreurs d'arrondi fassent diverger le calcul. A l'opposé, si l'intervalle dx est grand, la convergence ne sera que linéaire.

En conclusion on peut donc dire que la méthode de Newton-Raphson n'est pas une méthode intéressante si l'on ne peut déterminer de forme analytique de $f'(x)$.

D Annexe

D - 1 Annexe 1 : fonction scilab

function y = newtonRaphsonBIS(f,x_0, nbIte)
eps = 0.00001 ;
h = 0.0001 ;
i = 0 ;
while (abs(f(x_0))> eps and i < nbIte) ;
i = i+1 ;
df = $\frac{f(x_0+h)-f(x_0)}{h}$;
ddf = Deriv2(x_0)
x0 = x_0 - $\frac{df}{ddf}$;
end
y = x_0 ;
endfunction

nbIte = 100//input("Veuillez saisir nombre d iteration") ;
x_0 = input("Veuillez saisir nombre de depart") ;
esultat=newtonRaphsonBIS(f,x_0, nbIte) ;
disp(resultat) ;

D - 2 Annexe 2 : fonction scilab amélioré

function y = newtonRaphson(f,fd,x_0, nbIte) //Trouver une racine de fd
eps = 0.0001 ;
h = 0.0001 ;
i = 0 ;
while (abs(fd(x_0))> eps and i < nbIte)
i = i+1 ;
$df = (f(x0 + h) - \frac{f(x_0)}{h}$;
$ddf = (fd(x_0 + h) - \frac{fd(x_0)}{h}$;
x_0 = x_0 - $\frac{df}{ddf}$;
end
disp('le nombre d iteration est : ') ;
disp(i) ;
y = x_0 ;
endfunction

function racine = minimum(f,fd,a, b, x_0) //Trouver un minimum à la fonction f
racine = newtonRaphson(f,fd,x_0,nbIte)
if (abs(fd(racine))<0.01 and f(racine)<f(racine-0.01) and f(racine)<f(racine+0.01) and a<racine and racine<b)
return racine ;

else
$x80$ = b - rand()*(b-a) ;

```
disp ('changement de x0 : x0 = ') ;
disp(x80) ;
racine = minimum(f,fd,a, b, x0) ;
end
endfunction


nbIte = input("Veuillez saisir nombre d iteration") ;
a = input("Veuillez saisir la borne inférieure") ;
b = input("Veuillez saisir la borne supérieure") ;
x0 = input("Veuillez saisir nombre de depart") ;

resultat=minimum(f,fd, a, b, x0) ;
disp('la racine vaut : ') ;
disp(resultat) ;
```

D - 3 Annexe 3 : comment utiliser le programme