Université de Genève Département d'informatique

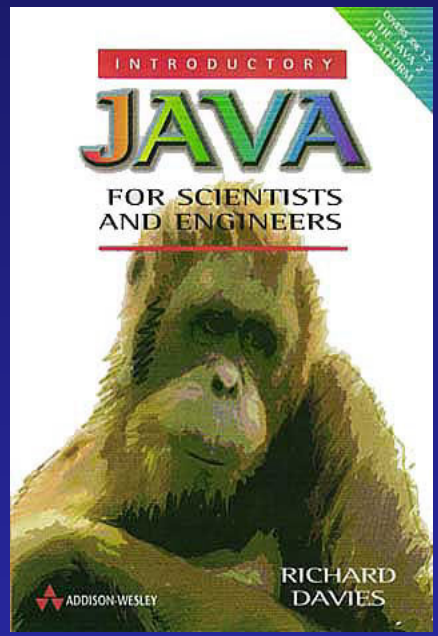
Programmer en

8. La recherche de « zéros »

« Introduction à l'informatique »
semestre d'été 2002

Olivier Schaad

Bibliographie



- Introductory JAVA for Scientists and Engineers Richard Davies, ed Addison-Wesley, 1999, ISBN 0-201-39813-3, http://cseng.aw.com/
- http://www.jscieng.co.uk/
- http://www.jscieng.co.uk/Code/NumComp/

Recherche de « zéros »

- La recherche de « zéros » d'une fonction permet de manière générale de trouver une solution à l'équation :

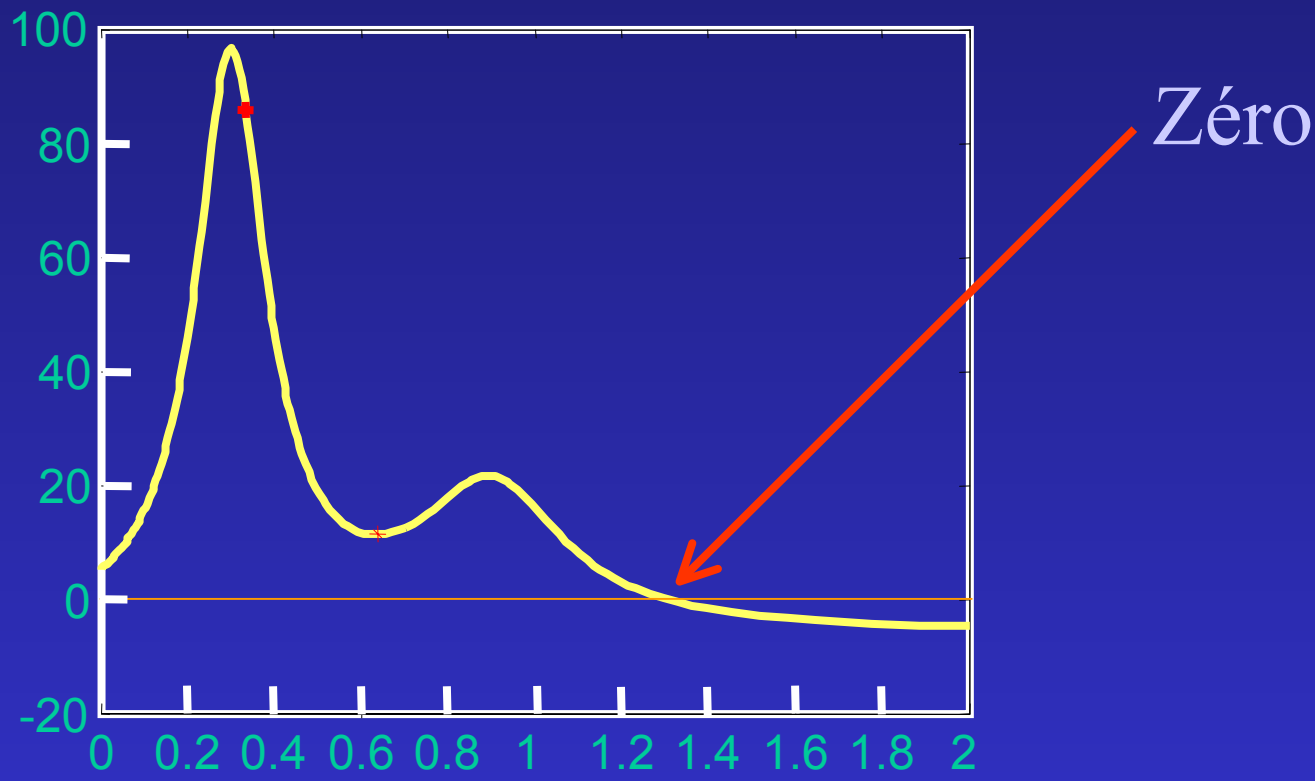
  $f(x,y,....) = z \Leftrightarrow f(x,y,...) - z = 0$

- La recherche de zéros de la dérivée d'une fonction permet de trouver des valeurs singulières telles que minima et maxima

Méthodes numériques

- La résolution d'une équation $f(x_i) = 0$ n'est pas toujours possible ou réalisable par une voie analytique (la plus précise, la plus complète). Dans de nombreux cas en sciences, il est nécessaire de recourir à des méthodes numériques qui nécessitent l'écriture de programmes ou au minimum l'emploi de programmes informatiques.

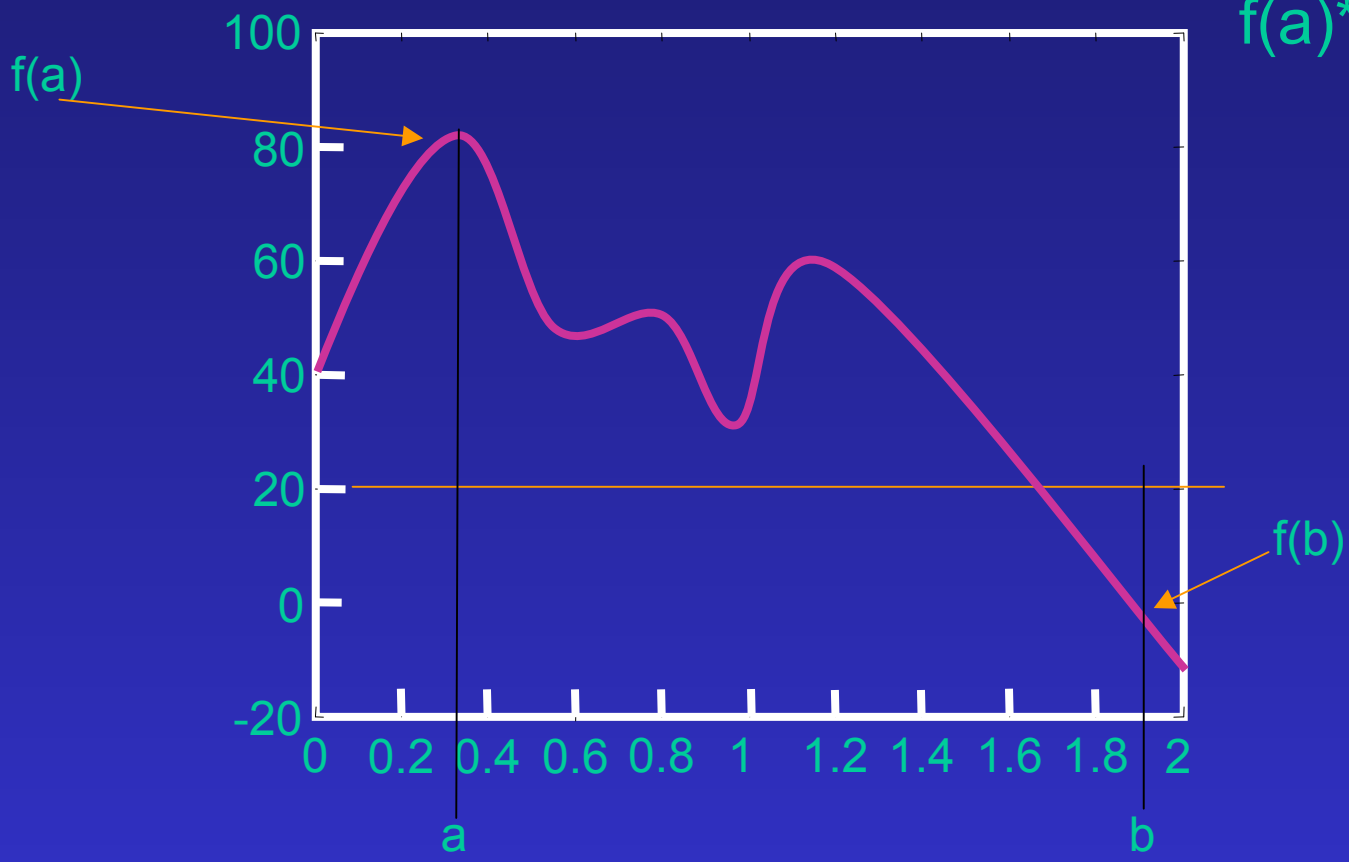
Exemple



Théorème fondamental

- La recherche numérique de « zéros » utilise les conséquences directes d'un théorème fondamental d'algèbre :
  - Si une fonction continue et monotone dans l'intervalle [a,b] possède la propriété que f(a) * f(b) < 0 alors cette fonction possède une valeur x telle f(x) = 0

Théorème fondamental



f(a)*f(b) < 0.0

Zéro

Méthode de recherche de zéros :
-1- recherche exhaustive

- Méthode exhaustive : à partir de a on calcule la valeur de f(a), puis f(a+Δx) et on évalue la valeur de la fonction. Si la valeur de f(a + n Δx) est suffisamment proche de zéro on conserve cette valeur de a + nΔx. et on continue
  - Cette méthode a le mérite de la simplicité de son concept et de sa mise en oeuvre.
  - C'est une méthode robuste qui permet l'obtention de plusieurs solutions.
  - C'est une méthode qui n'est pas pas optimale du point de vue du temps de calcul.

-1- recherche exhaustive
Pseudo-code

1 x = a (borne inférieur)

2 calcul de $x = x + \Delta x$

3 calcul de f(x)

4 tester si f(x) < epsilon (valeur acceptable)

4’ si f(x) < Epsilon conserver x

4’’ si f(x) > Epsilon ne rien faire

5 tester si x est supérieur à b (borne supérieure)

5’ si oui alors on arrête

5’’ si non alors on continue la boucle en 2

Méthode de recherche de zéros : -2- recherche aléatoire

- Cette méthode consiste à choisir un nombre aléatoire dans l'intervalle [a,b] puis d'évaluer sa similitude avec la valeur zéro.
  - Méthode plus rapide
  - Méthode robuste, exhaustive si le nombre de tirage est grand
  - Méthode lente dans certains cas
  - Ne présente pas d'effet de mémoire

-1- Recherche aléatoire
Pseudo-code

1 calcul de x = a + rand [0 1] * (b-a)

3 calcul de f(x)

4 tester si f(x) < epsilon (valeur acceptable)

4’ si f(x) < Epsilon conserver x

4’’ si f(x) > Epsilon ne rien faire

5 tester si N le nombre de tirages est suffisant

5’ si oui alors on arrête

5’’ si non alors on continue la boucle en 1

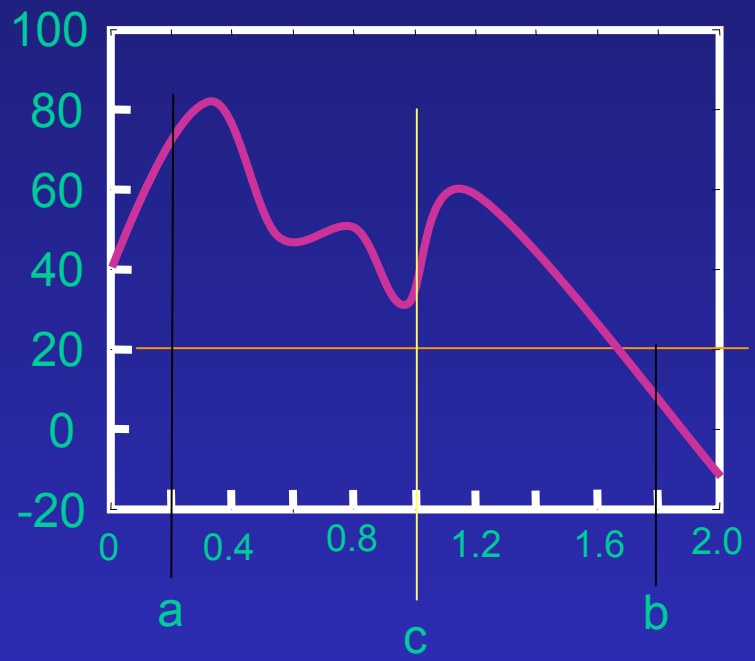
Méthode de recherche de zéros :
-3- recherche par bissection

- La méthode par bissection est une méthode rapide et simple permettant de trouver la solution d'une équation
  - La fonction doit être continue dans l'intervalle [a,b].
  - La fonction doit avoir un signe différent à la borne a de la borne b.
  - Cette méthode est plus rapide que la recherche exhaustive.
  - Cette méthode permet l'obtention d'une solution, en cas de solutions multiples dans l'intervalle [a,b] la méthode par bissection va procurer qu'une seule solution.
  - Afin d'obtenir d'autres solutions, il convient de modifier les bornes [a,b].

-3- Recherche par bissection
Principe

- Afin de trouver une solution on calcule le point milieu de l'intervalle [a,b], le point c puis on évalue si la fonction change de signe entre [a c] ou [c,b].
- Le sous intervalle choisi, on l'utilise pour l'itération suivante.
- L'arrêt est déterminé par comparaison de f(x) avec une valeur Epsilon proche de zéro choisie par l'utilisateur.

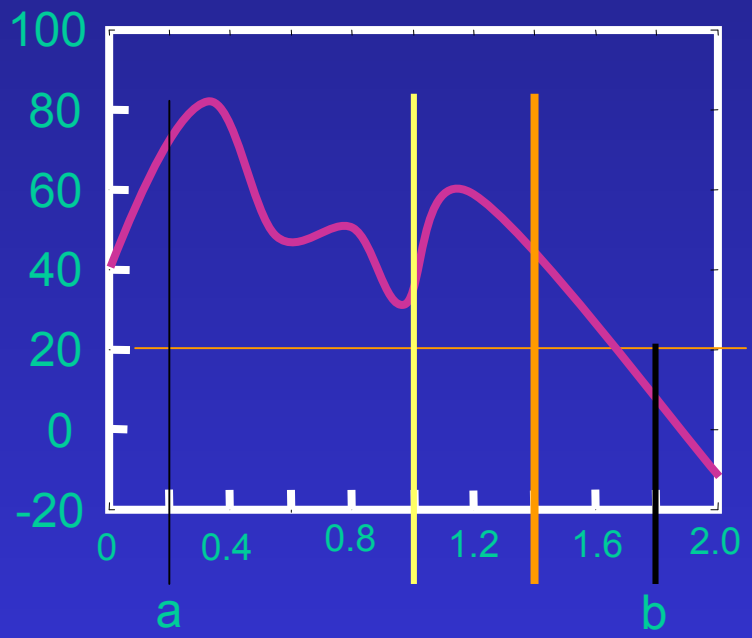
Recherche par bissection
Principe



$$f(c) * f(a) > 0$$

$$\Rightarrow \quad c = \frac{b - c}{2}$$

$$f(c) * f(b) < 0$$



-3- Recherche par bissection
Pseudo-code

1 milieu = moyenne des deux bornes [a,b]

2 calcul de f(a),f(b),f(milieu)

3 tester la valeur de f(milieu)

si la valeur est acceptable arrêter

4 tester si f(milieu)*f(b) < 0

4’ si oui le milieu devient la nouvelle gauche (a)

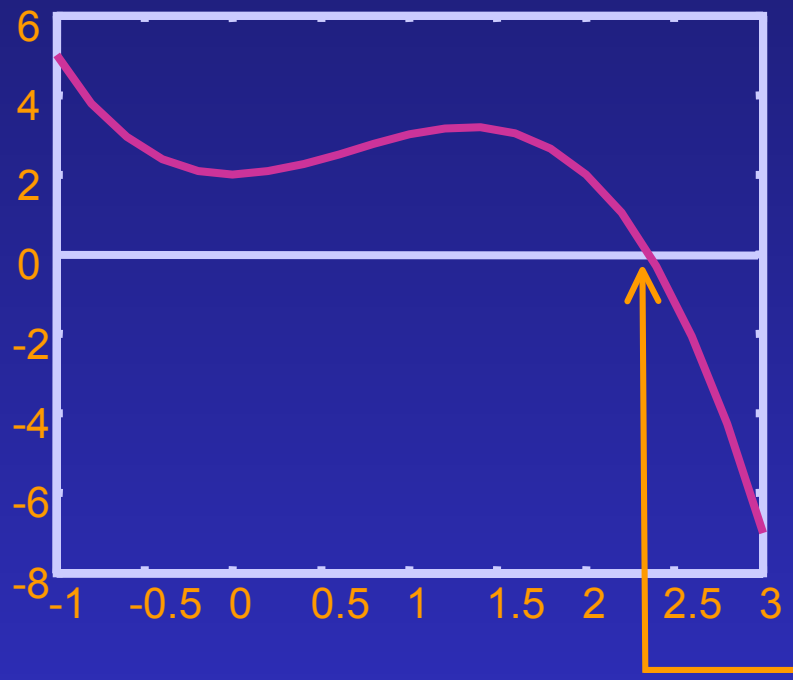
4’’ si non le milieu devient la nouvelle droite (4)

5 on boucle via 1

-3- Recherche par bissection code java

```
    // repetition tant que la valeur > epsilon et Niter< maxIter
epsilon = epsilon*epsilon ;
milieu=(a+b)/2.0;
y=f(milieu) ;
gauche=a;droite=b;
Niter=0;
 while ((y*y>epsilon) && (Niter<MaxIteration) ) {
     Niter=Niter+1;
     milieu=(gauche+droite)/2.0;
     y=f(milieu) ;
      // change de maniere a etre dans l'intervalle qui change de signe
     if (y*f(droite) <=0) {
             gauche = milieu }
        else {
             droite = milieu;}
}
```

-3- Recherche par bissection applet java



AppletViewer : DemoBissection.class

Applet

Calcul

limite inférieur a -1 limite supérieur b 3

Epsilon 1e-5 Maximum Iteration 100

Résultats:
on affiche les Coefficients a,b
a= -1.000 b= 3.000
epsilon= 0.00000000
MaxIteration= 100

f(a) = 5.000000
f(b) = -7.000000

le zero de la fonction se situe autour de la valeur d
x= 2.359303 f(x) = 0.000004
Après 21 iterations

Applet démarré.

Exemple d'applet :

Recherche de « zéros »
-4- Méthode de Newton

- La méthode de Newton ou son application plus générale : Newton-Raphson sont des algorithmes plus rapides que la méthode de bissection.
  - Méthode rapide
  - Moins stable
  - Dépend plus des conditions de départ
  - L'idée de base est une approximation par une série de Taylor

-4- Méthode de Newton

Série de Taylor

- Une expansion en série de Taylor d'une fonction consiste à calculer les différentes dérivées de la fonction étudiée, puis de calculer une somme :

$$f(x+h) \approx f(x) + hf'(x) + \frac{h^2}{2} f'' + \frac{h^2}{6} f''' + ...$$

- Si h est petit et que la fonction est monotone, les termes au delà de la première dérivée peuvent être négligés

$$f(x+h) \approx f(x) + hf'(x)$$

-4- Méthode de Newton

Série de Taylor

- On recherche que f(x+h) soit égale à zéro

$$f(x+h) \approx f(x) + hf'(x)$$

$$f(x+h) = 0 \approx f(x) + hf'(x) \Leftrightarrow h \approx -\frac{f(x)}{f'(x)}$$

- Ce qui nous procure la formule itérative suivante :

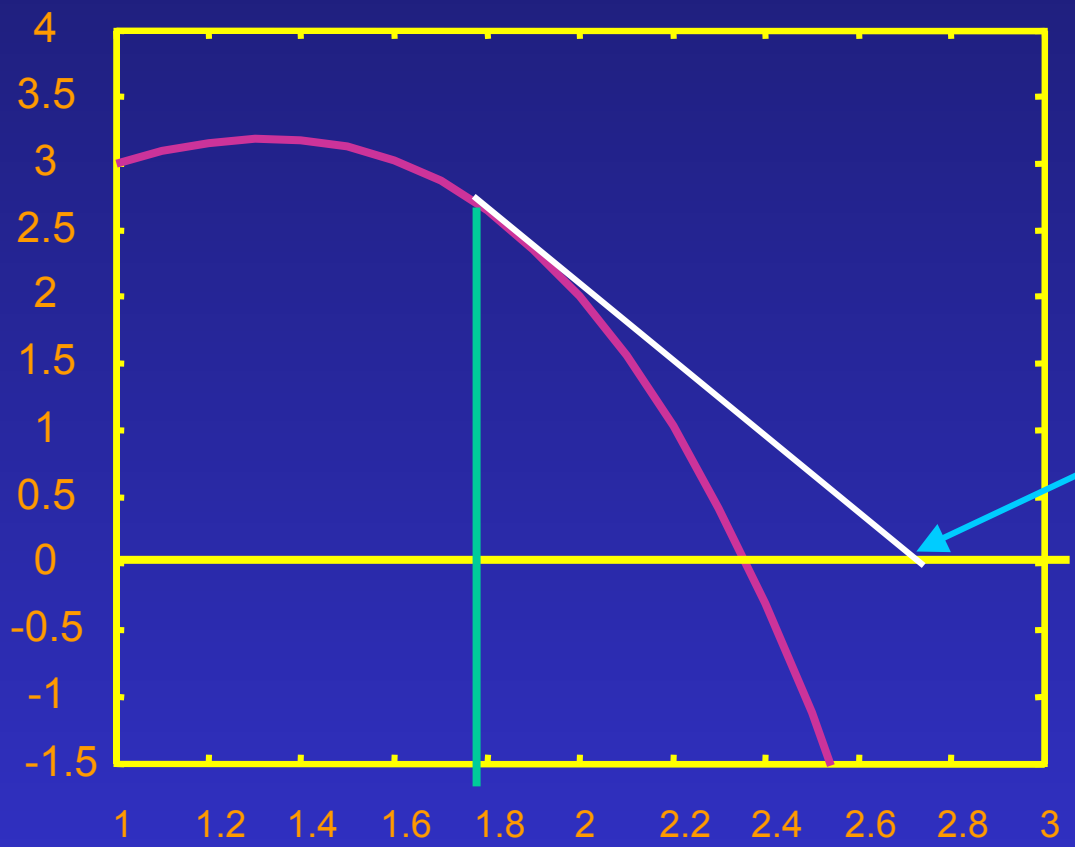
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

- Cette formule nécessite la connaisance de la dérivée, elle peut être numérique estimée :

$$f'(x_n) \cong \frac{f(x+h) - f(x-h)}{2h}$$

-4- Méthode de Newton

Interprétation géométrique



$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

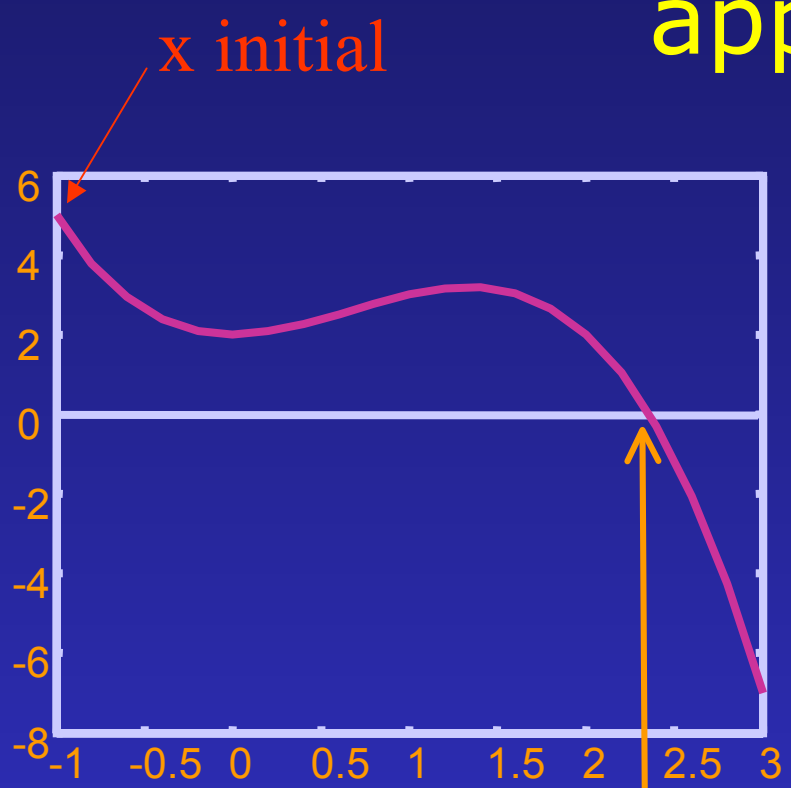
-3- Recherche par bissection
Pseudo-code

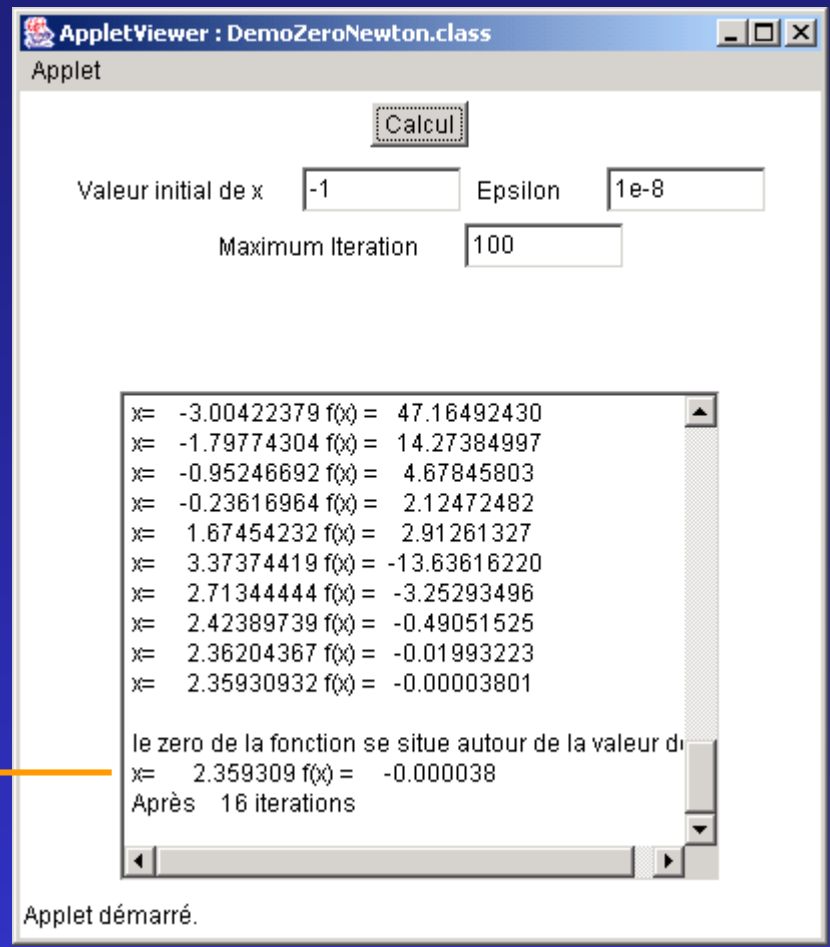
1. Choisir un x de départ
2. calculer x(n+1) = x(n) – f(x)/derivée(f(x))
3. tester la valeur de f(x)
   si la valeur est acceptable arrêter
4. on boucle via 2

-3- Recherche par bissection code java

```
Niter=0; double x=xo; y=f(x);
while ((y*y>epsilon) && (Niter<MaxIteration) ) {
     Niter++;
     x = x - f(x) / deriv(x,h);
     y=f(x) ;
}
// Estimation numérique de la dérivée d'une fonction « f »
 public static double deriv(double x, double h)
   {
      return (f(x+h)-f(x-h))/(2.0*h);
     }
```

-3- Recherche par Newton applet java





Exemple d'applet :