

# Recherche de zéros : l'algorithme de dichotomie

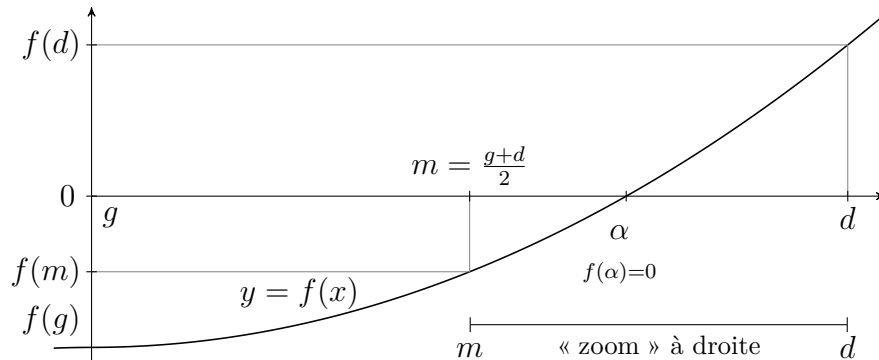
## 1 Formulation

### 1.1 Objectif

#### Proposition 1 (*Théorème des valeurs intermédiaires*)

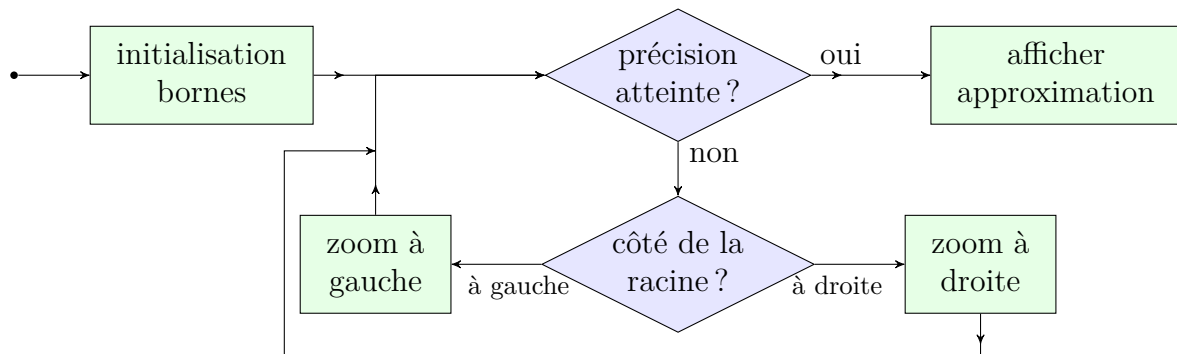
Soit  $f : [g; d] \rightarrow \mathbb{R}$  une fonction continue qui vérifie :  $f(g)f(d) < 0$  (*changement de signes*).  
Alors  $f$  s'annule sur  $[g; d]$  :  $\exists \alpha \in [g; d], f(\alpha) = 0$ .  
De plus si  $f$  est strictement monotone, alors  $\alpha$  est unique.

L'algorithme de dichotomie permet de trouver une approximation de ce  $\alpha$  (« la racine »).  
Pour se fixer les idées, on pense à  $f : x \mapsto x^2 - 2$ ,  $[g; d] = [0; 2]$ , et  $\alpha = \sqrt{2}$  est unique.



### 1.2 Organigramme de l'algorithme

À chaque étape de la boucle, on coupe le nouvel intervalle par le milieu.



### 1.3 Implémentation

Soit  $f : [g; d] \rightarrow \mathbb{R}$  une fonction continue telle que  $f(g)f(d) < 0$ .

On peut approximer un zéro  $\alpha$  de  $f$  en construisant deux suites comme suit :

#### Proposition 2 (*Convergence de la méthode de dichotomie*)

- Pour  $n \in \mathbb{N}$ , on pose  $m_n = \frac{g_n + d_n}{2}$  (le milieu du segment  $[g_n; d_n]$ ).
- Si  $f(g_n)f(m_n) > 0$ , alors :  $\begin{cases} g_{n+1} = m_n \\ d_{n+1} = d_n \end{cases}$  et sinon :  $\begin{cases} g_{n+1} = g_n \\ d_{n+1} = m_n \end{cases}$

Alors les suites  $(g_n)$  et  $(d_n)$  sont adjacentes, et leur limite commune  $\alpha$  satisfait  $f(\alpha) = 0$ .

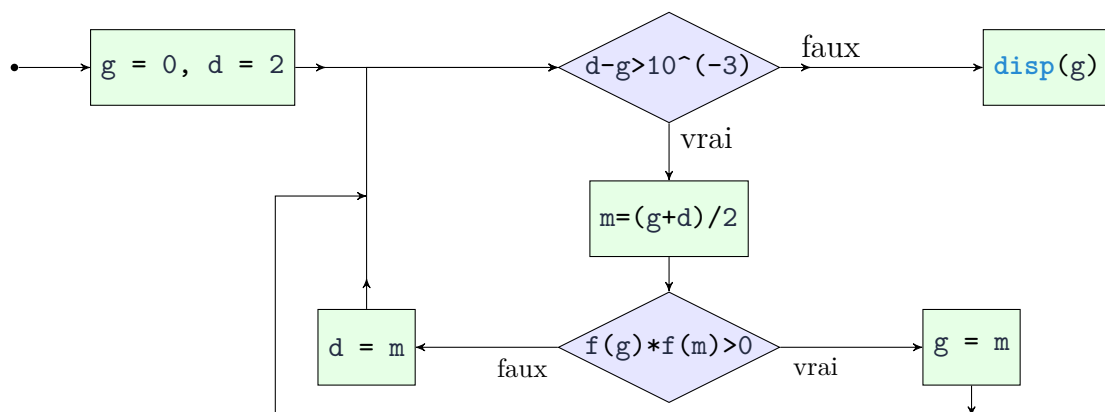
► **Situer la racine**

- Si  $f(g)f(m) > 0$ , alors  $f$  change de signes sur  $[m; d]$ , et la racine  $\alpha$  est **à droite** de  $m$ .
- Sinon  $f(g)f(m) \leq 0$ , alors  $f$  change de signes sur  $[g; m]$ , et  $\alpha$  est **à gauche** de  $m$ .

► **Zoom**

- Pour zoomer **à droite**, on remplace  $[g; d]$  par  $[m; d]$ , soit  $g$  par  $m$ .
- Pour zoomer **à gauche**, on remplace  $[g; d]$  par  $[g; m]$ , soit  $d$  par  $m$ .

## 2 En Scilab



dichotomie.sce

```

1 // Constantes : données du problème
2 GAUCHE_INIT = 0 // encadrement initial de la racine
3 DROITE_INIT = 2
4 PRECISION = 10^(-3) // précision souhaitée
5 function y = f(x) // fonction étudiée
6     y = x.^2 - 2
7 endfunction
8
9 // initialisation
10 gauche = GAUCHE_INIT
11 droite = DROITE_INIT
12
13 // la boucle de l'algorithme
14 while ( droite - gauche > PRECISION ) // jusqu'à avoir la bonne précision
15     milieu = (gauche+droite) / 2
16     if ( f(gauche) * f(milieu) > 0 ) then
17         gauche = milieu // si la racine est à droite de `milieu`
18     else
19         droite = milieu // si la racine est à gauche de `milieu`
20     end // fin du `if`
21 end // fin du `while`
22
23 // affichage
24 disp ("Approximation par défaut de sqrt(2) à 10^(-3) près :")
25 disp (gauche) // retourne 1.4140625
  
```