# Dichotomie

Informatique Appliquée au Calcul Scientifique 1 Séance 4-5

*Comment approcher une solution d'une équation lorsque nous ne savons pas la calculer directement ? La dichotomie consiste à encadrer une racine, puis à réduire cet encadrement de moitié à chaque étape. Nous allons comprendre le choix des nouvelles bornes, suivre les calculs à la main et dans un tableur, puis traduire les mêmes décisions en Python. La largeur de l'intervalle nous permettra de contrôler la précision obtenue.*

## I. Retour sur la dichotomie

### I.1. Théorème des Valeurs Intermédiaires

L'algorithme de dichotomie peut être mis en œuvre pour résoudre les équations de type $f(x) = 0$, où $f$ est une fonction continue de $\mathbb{R}$ dans $\mathbb{R}$ grâce au Théorème des Valeurs Intermédiaire (TVI) :

**Théorème des Valeurs Intermédiaire / Bolzano :**

Soient $a$ et $b$ deux nombres réels, $f : [a;\ b] \longrightarrow \mathbb{R}$ une fonction continue telle que $f(a).f(b) < 0$, alors il existe $c \in ]a; b[\ tel\ que\ f(c) = 0$.

Remarques :

- La notion de continuité est ici essentielle.
- $f(a).f(b) < 0$ permet d'avoir $f(a)$ et $f(b)$ de signe contraire.
- Si la fonction est strictement monotone sur $[a; b]$, la racine est unique. Sans cette hypothèse, la dichotomie reste applicable lorsque les valeurs aux bornes sont de signes contraires, mais elle ne permet pas de rechercher toutes les racines à la fois.

## I.2. Algorithme de dichotomie

Si $f$ est une fonction continue de $[\alpha, \beta]$ dans $\mathbb{R}$, telle que $f(\alpha) < 0$ et $f(\beta) > 0$ (ou l'inverse) alors, $\exists\, \gamma \in ]\alpha, \beta[, f(\gamma) = 0$.

L'algorithme de dichotomie pose $\alpha_0 = \alpha$ et $\beta_0 = \beta$ ; puis si $[\alpha_k, \beta_k]$ est donné ($k \in \mathbb{N}$), calcule $\gamma_k$ tel que $\gamma_k = \frac{\alpha_k + \beta_k}{2}$.

Alors :

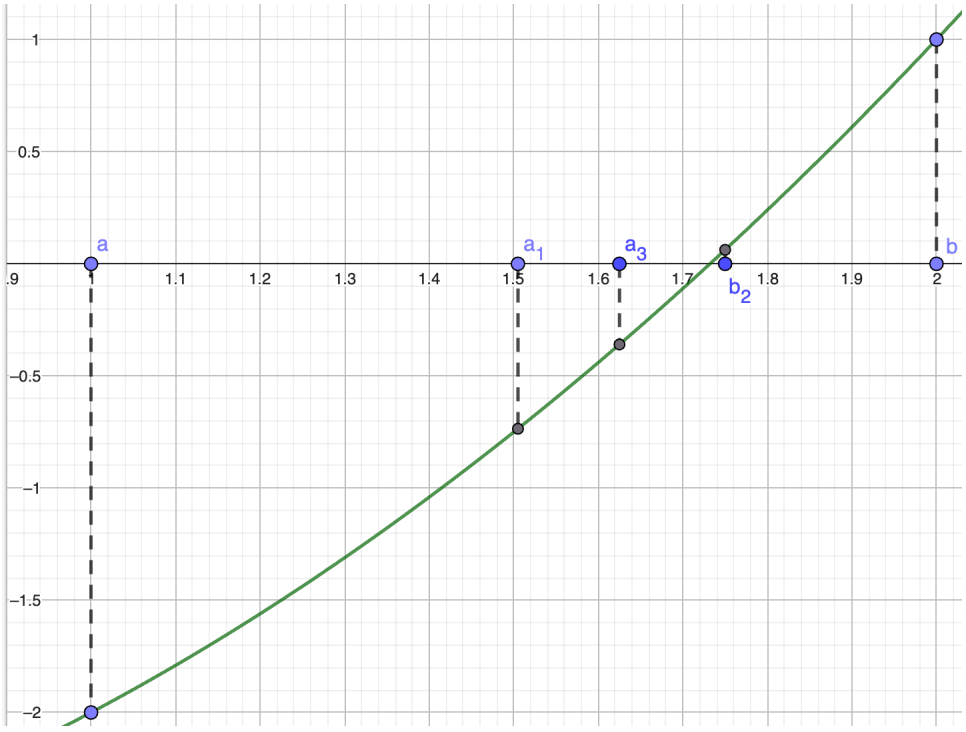
- soit $f(\alpha_k)f(\gamma_k) < 0$ : nous conservons $[\alpha_k; \gamma_k]$, en posant $\alpha_{k+1} = \alpha_k$ et $\beta_{k+1} = \gamma_k$ ;
- soit $f(\beta_k)f(\gamma_k) < 0$ : nous conservons $[\gamma_k; \beta_k]$, en posant $\alpha_{k+1} = \gamma_k$ et $\beta_{k+1} = \beta_k$ ;
- soit $f(\gamma_k) = 0$, alors $\gamma_k$ est notre racine.

Pour cette phase, nous allons utiliser un branchement ou if :

On choisit $a$ et $b$ tels que $f(a)f(b) < 0$, puis :

```
m = (a + b)/2 ; c = f(m)
si c = 0 alors : retourner m
sinon si f(a)*c < 0 alors : b = m
sinon : a = m
fin si
```

On a ainsi réduit l'intervalle de moitié.




Sur l'exemple ci-dessus, nous voyons l'étude pour trouver une approximation de $\sqrt{3}$ grâce à la méthode par dichotomie sur la fonction $x \mapsto x^2 - 3$.

Les conditions initiales se font avec $a = 1$ et $b = 2$.

Ainsi, l'intervalle de départ est $[a, b]$, à la fin de l'étape 1, on obtient l'intervalle $[a_1, b]$, puis $[a_1, b_2]$, $[a_3, b_2]$, …

Pour le cas du calcul de la valeur approchée de $\sqrt{2}$, cela donne le pseudocode suivant :

```
a = 1 ; b = 2
fonction f(x)
    retourner x^2 - 2
fin fonction
Pour i allant de 1 à 60 faire
    m = (a + b)/2
    Si f(m) = 0 alors
        retourner m
    Sinon si f(a)*f(m) < 0 alors
        b = m
    Sinon
        a = m
    Fin si
Fin pour
retourner (a + b)/2
```

Le premier algorithme impose un nombre de répétitions. Une boucle while permet plutôt de continuer tant que la largeur de l'intervalle dépasse la tolérance demandée. Attention : en machine, multiplier les itérations sans limite ne donne pas une précision infinie ; les arrondis peuvent finir par empêcher le milieu de se distinguer des bornes.

On peut ainsi modifier notre algorithme comme suit :

```
a = 1 ; b = 2 ; t = tolerance
fonction f(x)
    retourner x^2 - 2
fin fonction
Tant que (b - a) > t faire
    m = (a + b)/2
    Si f(m) = 0 alors
        retourner m
    Sinon si f(a)*f(m) < 0 alors
        b = m
    Sinon
        a = m
    Fin si
Fin tant que
retourner (a + b)/2
```

# II. TP2 Dichotomie

*Objectif : approcher une solution de f(x) = 0 par dichotomie. Vous étudierez d'abord la méthode à la main, puis dans un tableur, avant de construire l'algorithme et de le programmer en Python.*

## 1. Comprendre et réaliser une étape à la main

**Théorème des valeurs intermédiaires.**

Si $f$ est continue sur [a ; b] et si $f(a).f(b) < 0$, alors f possède au moins une racine dans $]a\,;\,b[$. Si $f$ est strictement monotone sur cet intervalle, la racine est unique.

On prend le milieu $m = \frac{a+b}{2}$. Si $f(m) = 0$, on a trouvé une racine. Sinon, on conserve la moitié dont les valeurs de la fonction aux extrémités sont de signes contraires. La largeur de l'intervalle est ainsi divisée par deux.

1. On considère $f(x) = x^2 - 2$ sur $[1\,;\,2]$. Calculer $f(1)$ et $f(2)$. Justifier qu'il existe une unique racine sur cet intervalle. Quelle valeur cherche-t-on à approcher ?

.....

.....

2. Compléter le tableau à la main. Pour chaque ligne, calculer $m$, puis le signe de $f(a).f(m)$, et choisir les bornes de la ligne suivante.

| Étape n | a | b | m | Signe du produit | Intervalle suivant |
|---|---|---|---|---|---|
| 0 | 1 | 2 | | | |
| 1 | | | | | |
| 2 | | | | | |

3. Expliquer en une phrase comment choisir la nouvelle valeur de $a$ puis celle de $b$. Quelle propriété l'intervalle doit-il conserver ?

.....

.....

.....

4. Sans poursuivre tous les calculs, donner la largeur après 1, 2, puis $n$ réductions. Quel critère permettrait de s'arrêter ?

.....

.....

**Travail demandé : enregistrer un classeur TP2_Dichotomie et un fichier Python. Rédiger les réponses sur cette fiche ou dans une feuille Réponses du classeur.**

## 2. Construire la dichotomie dans un tableur

Le tableur rend visibles les données d'une étape et celles de la suivante. L'étape 0 désigne l'intervalle initial ; elle est placée en ligne 2 du classeur.

| A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|
| Étape n | a | b | m | f(a) | f(m) | Nouveau a | Nouveau b | b − a |
| 0 | 1 | 2 | | | | | | |

## 3. Calculer et tester

5. Saisir les en-têtes en A1:I1, puis les valeurs initiales en A2:C2.

Écrire en D2 la formule =(B2+C2)/2. Proposer les formules de E2, F2 et I2. Toute formule commence par = ; la puissance s'écrit avec ^.

.....

.....

6. Calculer le produit E2*F2. Quelle moitié faut-il conserver ? Prévoir manuellement les valeurs de G2 et H2 avant d'écrire une formule.

.....

.....

## 4. Automatiser le choix avec SI

La fonction **=SI(test;valeur_si_vrai;valeur_si_faux)** renvoie une valeur selon un test. Dans Excel anglais, elle se nomme IF ; les arguments sont généralement séparés par des virgules.

7. Écrire en G2 : =SI(E2*F2<0;B2;D2). Expliquer le test et les deux valeurs possibles. Écrire vous-même la formule de H2. Vérifier que les résultats correspondent à votre choix manuel.

.....

.....

Pour cette première construction avec $x^2 - 2$, on considère f(a) et f(m) non nuls. Le cas d'une racine exacte sera repris dans le défi.

## 5. Passer à l'étape suivante

8. Quelles cellules de la ligne 2 doivent alimenter B3 et C3 ? Écrire ces deux formules ainsi que celle de A3. Recopier D2:I2 en D3:I3. Vérifier les références et les résultats.

.....

.....

9. Recopier A3:I3 vers le bas jusqu'à l'étape 15. Pourquoi recopier les nombres 1 et 2 en B et C ne permettrait-il pas de faire progresser le calcul ?

.....

.....

## 6. Observer l'arrêt souhaité

On note $r = \sqrt{2}$.

Si la racine $r$ appartient à $[a; b]$, le milieu $m$ vérifie $|m - r| \leq \frac{b-a}{2}$. Nous choisissons ici le critère $b - a \leq \varepsilon$ : il garantit une erreur sur le milieu inférieure ou égale à $\frac{\varepsilon}{2}$, donc une approximation à $\varepsilon$ près.

10. Observer la colonne I. Donner la première étape n pour laquelle la largeur est ≤ $10^{-3}$. Relever $a$, $b$, $m$ et la borne garantie de l'erreur. Afficher assez de décimales et distinguer étape $n$ et numéro de ligne.

.....

.....

.....

11. Prolonger jusqu'à l'étape 25. Refaire l'étude pour $\varepsilon = 10^{-6}$. Justifier le nombre de réductions grâce à la formule de la largeur. Le nombre de décimales affichées prouve-t-il la précision ?

.....

.....

.....

12. Ajouter J1 = Précision atteinte. Écrire une formule SI qui indique « oui » lorsque la largeur est ≤ $10^{-6}$ et « non » sinon. La première ligne marquée « oui » suffit-elle à empêcher le calcul des lignes suivantes ?

.....

.....

## 7. Décrire les instructions à répéter

13. En vous appuyant sur le tableau, répondre aux quatre questions : quelles valeurs initialise-t-on ? Que calcule-t-on à chaque étape ? Quel test décide des nouvelles bornes ? Quelle condition permet de continuer ?

.....

.....

.....

14. Compléter le pseudocode suivant avec des expressions et des affectations. Prévoir aussi le résultat renvoyé une fois la précision atteinte.

```
initialiser a, b et epsilon
tant que ..........................................
    m = ..........................................
    si f(m) = 0 alors
        retourner m
    sinon si ......................................
        ..........................................
    sinon
        ..........................................
    fin si
fin tant que
retourner .........................................
```

## 8. Relier le tableau et le programme

15. Associer les colonnes B, C et D à des variables Python. Quel bloc if / else remplace les deux formules SI de G et H ? Quelles affectations remplacent les références à la ligne précédente ?

.....

.....

.....

16. Expliquer la différence entre recopier 25 lignes et exécuter une boucle while. Que gagne-t-on lorsque l'on modifie la précision demandée ?

.....

.....

.....

## 9. Écrire une fonction générale

17. Programmer une fonction dichotomie(fonction, a, b, epsilon) qui renvoie une valeur approchée d'une racine. Utiliser votre pseudocode. Les données doivent vérifier $a < b$, $epsilon > 0$ et la continuité de fonction sur $[a; b]$. Vérifier le signe des valeurs aux bornes et traiter le cas d'une borne déjà racine.

Point Python. Une fonction peut être définie avec def ou avec lambda. Pour $f(x) = x^2 - 2$ :

| **Avec def** | **Avec lambda** |
|---|---|
| `def f(x):`<br>`    return x**2 - 2` | `f = lambda x: x**2 - 2` |

18. Utiliser votre fonction pour approcher $\sqrt{2}$ à $10^{-6}$ près avec les bornes 1 et 2. Comparer le résultat à celui du tableur. Que renvoie votre programme lorsque $f(m) = 0$ ?

.....

.....

.....

19. Ajouter un compteur du nombre de réductions d'intervalle dans une version de test. Comparer ce nombre à la première étape utile du tableau pour $\varepsilon = 10^{-3}$ puis $\varepsilon = 10^{-6}$.

.....

.....

.....

20. Une valeur $|f(m)|$ très petite garantit-elle toujours que m est proche d'une racine ? Expliquer pourquoi nous utilisons la largeur de l'intervalle comme critère de précision.

.....

.....

## 10. Applications à d'autres fonctions

21. Pour $f(x) = 1 - x - x^3$ puis $g(x) = -1 + 3x - x^3$, étudier la continuité, les variations et le signe aux bornes d'intervalles adaptés. Encadrer chaque racine et utiliser votre fonction Python pour en donner une approximation à $10^{-6}$ près. Un travail à la main est nécessaire pour choisir les intervalles.

.....

.....

.....

22. Proposer une fonction et un intervalle permettant d'approcher $\pi$ à $10^{-6}$ près par dichotomie. Justifier le choix des bornes et l'unicité de la racine sur cet intervalle. Ne pas utiliser math.pi comme résultat fourni à l'algorithme.

.....

.....

.....

## 11. Défi de paramètres dans le tableur

23. On considère maintenant $f(x) = x^2 - q$. Placer a initial, b initial, ε et q en L1:L4 avec les étiquettes en K1:K4. Relier B2 et C2 aux paramètres. Adapter E2 et F2 à l'aide de $L$4 et expliquer le rôle de $. Adapter le test de précision en J à $L$3.

24. Tester q = 3 sur $[1; 2]$, puis q = 5 sur $[2; 3]$, avec $\varepsilon = 10^{-3}$ et $10^{-6}$. Vérifier les hypothèses et comparer les résultats avec Python.

.....

.....

25. Tester q = 4 sur $[1; 3]$. Observer f(m). Modifier G et H avec des SI imbriqués pour que la ligne suivante contienne [m ; m] lorsqu'une racine exacte est trouvée. Comment traiter une borne initiale déjà racine ?

.....

.....

26. Proposer des formules qui conservent les bornes dès que la précision est atteinte. Les formules des lignes suivantes cessent-elles alors d'être calculées ? Comparer ce comportement avec celui de while et de return dans Python.

.....

.....