Cours

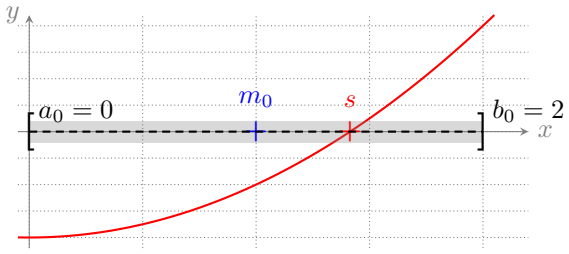
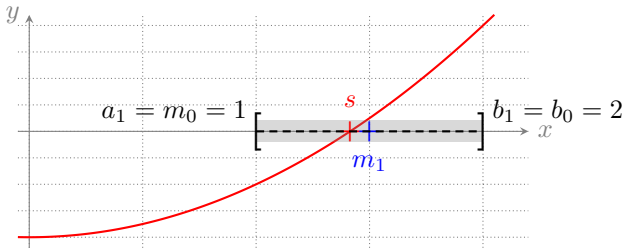
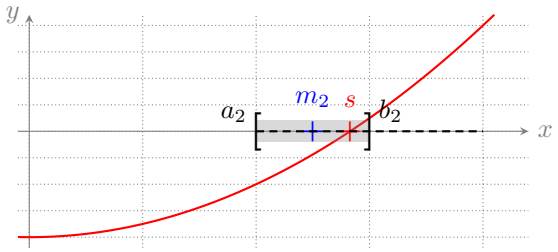
5 Méthode dichotomique

Capacités exigibles

- Déterminer, en s'appuyant sur une représentation graphique, un intervalle adapté à la recherche numérique d'une racine par une méthode dichotomique
- Mettre en œuvre une méthode dichotomique afin de résoudre une équation avec une précision donnée
- Utiliser la fonction **bisect** de la bibliothèque **scipy.optimize** (sa spécification étant fournie)

A Principe de la méthode dichotomique

OBJECTIF : résoudre une équation de la forme $f(x) = 0$.

PRINCIPE	Exemple : résoudre $f(x) = 0$, où $f(x) = x^2 - 2$, de solution $s = \sqrt{2} = 1,4142...$
Déterminer un 1er intervalle $[a_0, b_0]$ dans lequel on est sûr que la solution se situe. **Le milieu $m_0 = \dfrac{a_0 + b_0}{2}$ de $[a_0, b_0]$ est un 1er estimateur de la solution s.**	
Recommencer l'opération précédente avec un intervalle $[a_1, b_1]$ plus petit que $[a_0, b_0]$, de sorte qu'on soit sûr que la solution s recherchée s'y trouve. Comment en être sûr ? La fonction f change nécessairement de signe dans l'intervalle $[a_1, b_1]$. Donc, on choisit : ➤ $[a_1, b_1] = [a_0, m_0]$ si $f(a_0)$ et $f(m_0)$ sont de signes opposés, ➤ $[a_1, b_1] = [m_0, b_0]$ sinon. **Le milieu $m_1 = \dfrac{a_1 + b_1}{2}$ de $[a_1, b_1]$ est un 2ème estimateur de la solution s.**	
On réitère à nouveau. Ci-contre, le rang $n = 2$. **On arrête l'itération lorsque la précision ϵ souhaitée a été atteinte.** Concrètement : il faut : $\lvert b_n - a_n \rvert \leqslant \epsilon$ Ainsi, l'estimateur m_n est écarté de la valeur s d'au plus $\dfrac{\epsilon}{2}$.	

R *Lien entre précision et nombre d'itérations ?*

La précision obtenue au rang n est $\epsilon_n = \dfrac{b_0 - a_0}{2^n}$, puisqu'à chaque itération, on ne garde que la moitié de l'intervalle de l'itération précédente.

Ainsi, pour une précision finale ϵ souhaitée, nécessitant N d'itérations, on veut $\epsilon_N = \epsilon \Leftrightarrow N = \dfrac{\ln\left(\frac{b_0 - a_0}{\epsilon}\right)}{\ln 2}$.

On perçoit clairement que plus ϵ est faible, plus N est grand. Cela paraît plutôt logique ...

N. Gaudouen

B Implémentation sous Python

Voici la structure de l'algorithme :

- Indiquer des valeurs initiales de a et b telles qu'on soit sûr que la solution recherchée se trouve dans $[a,b]$
- Préciser une valeur de ϵ, taille de l'intervalle final souhaité, pour obtenir une solution suffisamment précise.
- Tant que $b-a>\epsilon$:
  - évaluer $m=\dfrac{a+b}{2}$
  - si $f(a)$ et $f(m)$ sont de signes opposés, alors la solution est dans $[a,m]$. Les bornes du nouvel intervalle sont donc $a=a$ et $b=m$
  - si $f(a)$ et $f(m)$ sont de même signe, alors la solution est dans $[m,b]$. Les bornes du nouvel intervalle sont donc $a=m$ et $b=b$
- Dès que $b-a\leqslant\epsilon$: l'estimation suffisamment précise de la solution est la dernière valeur de m obtenue.

On peut alors implémenter l'algorithme ainsi sous Python :

```
def solution_dichotomie(f,a,b,epsilon):
    while b-a > epsilon :
        m = (a+b)/2
        if f(a)*f(m)<0 : # càd si f(a) et f(m) sont de signes opposés
            b = m
        else: # sinon, càd si f(a) et f(b) sont de même signe
            a = m
    return m
```

Exemple

Reprenons l'exemple étudié dans l'encadré précédent en définissant préalablement la fonction f :

```
def f(x):
    return x**2-2

print('La racine carré de 2 vaut', solution_dichotomie(f,0,2,1E-10))
```

⇛ `La racine carré de 2 vaut 1.4142135623260401`

À comparer à la valeur «plus exacte» fournie par une calculatrice : `1.4142135623730950488016`.
L'écart entre les deux valeurs est $4,7.10^{-11}$ *qui est bien inférieur à* $\epsilon/2=5.10^{-11}$.

C Fonction bisect

La fonction **`bisect`** de la bibliothèque **`scipy.optimize`** exploite tout simplement la méthode dichotomique. Il faut indiquer les mêmes arguments que notre fonction **`solution_dichotomie`** mais sans nécessairement indiquer de précision ϵ particulière, qui vaut **`xtol = 2E-12`** par défaut.

Exemple

Choisissons le même exemple et la même précision que précédemment. Le résultat affiché est alors exactement le même que précédemment.

```
import scipy.optimize as sp
print(sp.bisect(f,0,2,xtol=1E-10))
```

⇛ `1.4142135623260401`

R *À l'avenir, il sera déraisonnable de coder nous-même une méthode de dichotomie quand un code existe déjà et a probablement été plus testé et optimisé que le nôtre. Par exemple, la fonction* **`bisect`** *peut renvoyer un message d'erreur spécifique lorsque l'intervalle $[a,b]$ initial a été mal choisi (lorsque la solution ne s'y trouve pas notamment).*