Ce TP est constitué d'exercices dont le but est de vous familiariser avec la manipulation de tableaux et leur utilisation dans des fonctions.
Ce TP constitue aussi le premier de 3 TPs sur l'implémentation objet d'un programme de recherche du zéro d'une fonction soit par dichotomie soit par la méthode de Newton. Ainsi les codes que vous produirez dans ce TP seront réutilisés lors des prochaines séances.

EXERCICE 1: POLYNÔMES

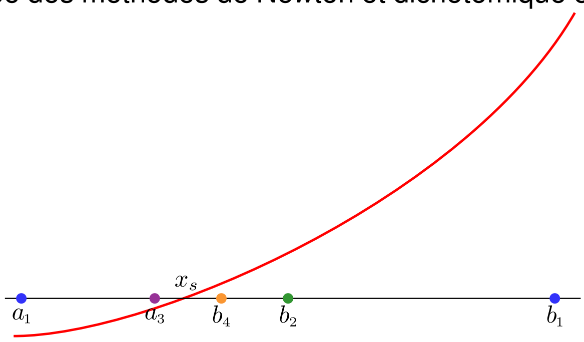
Le but de cette partie est d'implémenter des fonctions permettant le calcul d'une valeur ou de la dérivée d'un polynôme. Ces fonctions seront utilisée par la suite pour trouver une racine du polynôme.

Un polynôme $\left(p(x) = \sum_{i=0}^{n} a_i x^i\right)$ sera représenté par un tableau contenant les différents coefficients a_i par ordre croissant de puissance. Ainsi a_0 sera conservé dans la case 0 du tableau et a_n dans la dernière case.

1. Écrire une fonction `getValeur( p, x )` qui calcule et renvoie la valeur du polynôme p au point x.
2. Écrire une fonction `getDerivee( p, x )` qui calcule et renvoie la valeur de la dérivée du polynôme p au point x.
3. Écrire un programme principal pour vérifier le bon fonctionnement de ces 2 fonctions.

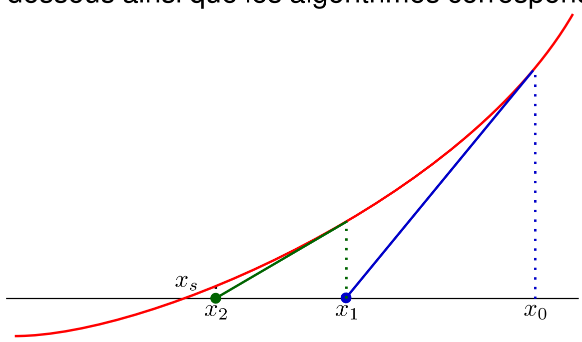
EXERCICE 2: RECHERCHE D'UNE RACINE

Les polynômes ayant été implémentés, on s'intéressera maintenant à la résolution de l'équation $p(x) = 0$. Le principe des méthodes de Newton et dichotomique est rappelé ci-dessous ainsi que les algorithmes correspondants.



fonction *dichotomie(p : tableau de réels ; a, b, ε : réel)*
  Variables : c, pA, pB, pC : réel
  pA ← getValeur(p, a) ;   pB ← getValeur(p, b)
  si *pA = 0* **alors retourner** *a*
  sinon si *pB = 0* **alors retourner** *b*
  répéter
    c ← (a + b) / 2 ;   pC ← getValeur(p, c)
    si *pA×pC < 0* **alors**
      b ← c ;   pB ← pC
    sinon
      a ← c ;   pA ← pC
  tant que *(pC ≠ 0)* ***et*** *(b-a > ε)*
  retourner *c*

Algorithme 1 : Dichotomie



fonction *newton(p : tableau de réels ; x0, ε : réel)*
  Variables : uN : réel
  uN ← precision + 1
  tant que *(getValeur(p, x0) ≠ 0)*
        et *(|uN| > ε)* **faire**
    uN ← getValeur(p, x0) / getDerivee(p, x0)
    x0 ← x0 - uN
  retourner *x0*

Algorithme 2 : Méthode de Newton

1. En vous aidant de l'algorithme 1, implémenter une fonction `dichotomie( p, xMin, xMax, eps )` qui calcule une racine d'un polynôme par dichotomie
2. En vous aidant de l'algorithme 2, implémenter une fonction `newton( p, x0, eps )` qui calcule une racine d'un polynôme par la méthode de Newton
3. Compléter le programme principal précédent pour rechercher le zéro du polynôme $p(x) = 0.0714x^4 + 0.0714x^3 - 0.9286x^2 - 0.0714x + 0.8571$ avec une précision $\varepsilon = 10^{-5}$ par la méthode de Newton ($x_0 = 2$) et dichotomique (intervalle $[2; 4]$).
   Corriger si nécessaire les erreurs détectées.

EXERCICE 3: AMÉLIORATIONS

Dans cette exercice, il vous est proposé d'implémenter quelques améliorations pour la manipulation des polynômes ou dans le calcul de leurs racines. Après chaque question, vous penserez à tester la nouvelle fonctionnalité dans votre programme principal.

1. **Polynôme**
   (a) Écrire une fonction permettant l'affichage d'un polynôme sous la forme `an*x^n + an-1*x^n-1 + ... + a0`
   (b) Écrire une fonction permettant de tester l'égalité de 2 polynômes
2. **Recherche d'une racine**
   (a) Comment gérer le cas où il n'y a pas de solution à l'équation $p(x) = 0$ dans $\mathbb{R}$?