Algorithmique PEIP2

TP3 : Polynômes

Le but de ce TP est d'implémenter les différentes opérations algébriques sur les polynômes, en utilisant diverses représentations.

1 Opérations algébriques

On représente un polynôme $P = a_d X^d + a_{d-1} X^{d-1} + \cdots + a_0$ sous la forme du tableau `t` de ses coefficients, de taille $d+1$, dont les éléments sont des flottants. On a donc `t[i]` $= a_i$ pour tout $0 \leq i \leq d$.

1. Écrire une fonction `Afficher(P)` qui affiche le polynôme représenté par le tableau `P` sous la forme $a_d X^d + a_{d-1} X^{d-1} + \cdots + a_0$.
2. Écrire une fonction `Derivee(P)` qui renvoie la dérivée du polynôme P.
3. Écrire une fonction `Primitive(P)` qui renvoie la primitive du polynôme P.
4. Écrire une fonction `Somme(P,Q)` qui renvoie la somme des polynômes P et Q.
5. Écrire une fonction `Produit(P,Q)` qui renvoie le produit des polynômes P et Q.
6. Écrire une fonction `Puissance(P,n)` qui renvoie la puissance n du polynôme P.
7. Écrire une fonction `Division(P,Q)` qui renvoie le quotient et le reste de la division Euclidienne de P par Q. On peut trouver une illustration de cette opération sur la page Wikipédia `https://fr.wikipedia.org/wiki/Division_d'un_polyn%C3%B4me`
8. (S'il y a le temps) Lorsqu'un polynôme de degré d a de nombreux coefficients $a_i = 0$ (pour $i < d$), alors il est plus intéressant d'utiliser une représentation *creuse*, qui consiste en la liste de couples (a_i, i) pour chaque coefficient non nul, par ordre de degré croissant. Redéfinir chacune des fonctions précédentes avec une implémentation creuse des polynômes.

2 Opérations analytiques

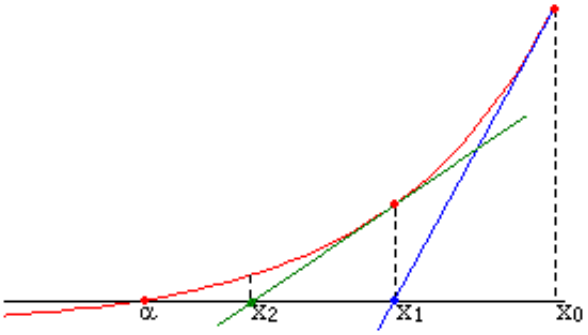

Figure 1: Les deux premières itérations de la méthode de Newton.

Ici, on s'intérèse aux opérations analytiques sur un polynôme. On choisira la représentation des polynômes de son choix (creuse ou non).

1. Écrire une fonction `Evaluer(P,x)` qui renvoie la valeur de $P(x)$. On cherchera à minimiser le nombre de multiplications effectuées (on évitera d'utiliser l'opération `x**i`).

2. Écrire une fonction `RacinesDeg2(P)` qui prend en argument un polynôme P *de degré* 2, et renvoie la liste de ses racines réelles. On pourra importer la fonction `sqrt` du module `math`.

3. Écrire une fonction `RacinesDeg3(P,`ε`)` qui prend en argument un polynôme P *de degré* 3, et renvoie la liste de ses racines réelles approchées à ε près, pour n'importe quel réel $\varepsilon > 0$. Pour cela, on devra en premier lieu calculer ses minimum et maximum locaux, en déduire son nombre de racines (1 ou 3), puis les approcher à l'aide de la méthode de Newton en partant de guess convenablement choisis.

   La méthode de Newton procède par itérations pour approximer la racine d'une fonction f continue et dérivable, en partant d'un guess initial x_0. Elle définit ensuite inductivement $x_{k+1} \coloneqq x_k - \frac{f(x_k)}{f'(x_k)}$. On prendra un x_0 qui assure la convergence de cette méthode (en particulier on doit avoir $f'(x_0) \neq 0$), et on fera autant d'itérations que nécessaire pour que l'on ait $|x_{k+1} - x_k| < \varepsilon$.

4. Généraliser la fonction `RacinesDeg3(P,`ε`)` à la fonction `Racines(P,`ε`)` qui a un fonctionnement similaire sur un polynôme P de degré arbitrairement grand.