

Fonctions élémentaires autour de la dérivation

But de l'activité : Ecrire des fonctions Python permettant le calcul de taux de variation, de nombres dérivés, du coefficient directeur et de l'ordonnée à l'origine d'une tangente à une courbe.

On considère la fonction f définie sur \mathbb{R} par $f(x) = \frac{1}{4}x^3 + x - 3$.

- 1) Ecrire une fonction Python **f** qui :
 - reçoit en argument une valeur x
 - renvoie son image par la fonction f .

- 2) Ecrire une fonction Python **coeff_dir** qui :
 - reçoit en arguments les coordonnées de deux points $A(x_A; y_A)$ et $B(x_B; y_B)$ (avec $x_A \neq x_B$)
 - renvoie le coefficient directeur de la droite (AB) .

- 3) A l'aide de la fonction précédente, écrire une fonction Python **taux_variation** qui :
 - reçoit en arguments une fonction f et deux valeurs a et h
 - renvoie le taux de variation de la fonction f entre a et $a + h$.

- 4) A l'aide de cette fonction, calculer le taux de variation de f entre 3 et 3,000001.
Conjecturer la valeur du nombre dérivé $f'(3)$, puis effectuer un calcul pour vérifier.

- 5) L'import « from scipy import misc » permet d'utiliser la fonction **misc.derivative** qui :
 - reçoit en arguments une fonction f et une valeur a
 - renvoie le nombre dérivé de f en a .Tester cette fonction pour calculer $f'(3)$ avec la saisie `misc.derivative(f,3,10**-9)`.

- 6) Ecrire une fonction Python **coeff_tang** qui :
 - reçoit en arguments une fonction f et une valeur a
 - renvoie le coefficient directeur et l'ordonnée à l'origine de la tangente à f en a .Tester cette fonction pour déterminer l'équation de la tangente à la courbe de f en 2.

- 7) La fonction **tab_val** ci-contre permet d'obtenir une liste de valeurs de la fonction f :
 - a) Quelle est la valeur initiale de cette liste ? le pas ? le nombre de valeurs obtenues ?
 - b) Adapter cette fonction pour qu'elle reçoive en argument la valeur initiale x_0 , le pas p et le nombre de valeurs n .

- 8) Ecrire une fonction Python **cdir_secantes** qui :
 - reçoit en arguments une fonction f , une valeur x_0 , un pas p et un entier n .
 - renvoie la liste des n coefficients directeurs des sécantes à la courbe de f à partir de x_0 avec un pas en abscisse p .

```
def tab_val(f):  
    t=[]  
    x=0  
    for k in range(10):  
        t.append(f(x))  
        x=x+2  
    return t
```

Méthode de Newton

Prérequis : Fonctions Pythons réalisées dans l'activité « Fonctions élémentaires autour de la dérivation »

But de l'activité : Approcher la solution d'une équation à l'aide de la méthode de Newton.

On considère la fonction f définie sur \mathbb{R} par $f(x) = \frac{1}{4}x^3 + x - 3$.

- 1) Démontrer que f est croissante sur \mathbb{R} .
On admettra pour la suite que l'équation $f(x) = 0$ a une unique solution sur \mathbb{R} , notée α .
- 2) Justifier que pour toute abscisse a , la tangente T_a à la courbe de f en a coupe l'axe des abscisses en un point P .
Déterminer l'expression de l'abscisse de P en fonction de a , $f'(a)$ et $f(a)$.
Ecrire une fonction Python **etap_Newton** qui :
 - reçoit en argument une fonction f et une valeur a
 - renvoie l'abscisse du point P correspondant

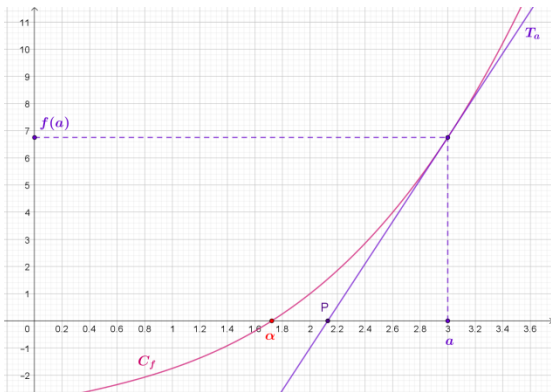


Figure pour la question 2

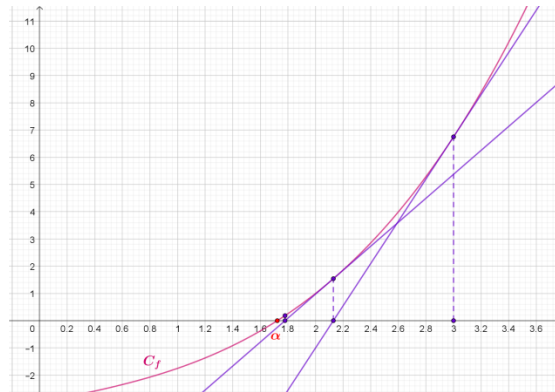


Figure pour la question 3

- 3) A partir d'un point de l'axe des abscisses, on peut donc construire une suite de points.
On admettra ici que la suite des abscisses de ces points a pour limite α .

- a) La fonction Python **appl_Newton** donnée ci-contre :
 - reçoit en arguments une fonction f , une valeur a et un entier n
 - renvoie une liste de valeurs.Expliquer ce que représentent les termes de la liste renvoyée.

```
def appl_Newton(f, a, n):  
    t=[a]  
    for k in range(n):  
        a=etap_Newton(f, a)  
        t.append(a)  
    return t
```

- b) Coder cette fonction et tester pour la fonction f de l'énoncé avec $a = 3$ et $n = 10$.

- 4) a) Proposer et coder en Python des fonctions g et h s'annulant respectivement en $\sqrt{5}$ et $\sqrt[3]{7}$.
b) A l'aide des fonctions Python précédentes, proposer des valeurs approchées de ces deux nombres.

Algorithme de dichotomie

Prérequis : Aucun, mais les question 1)a)b) peuvent être supprimées si l'activité « Méthode de Newton » a été traitée.

But de l'activité : Approcher la solution d'une équation à l'aide d'un algorithme de dichotomie (méthode plus lente que la méthode de Newton, mais pour laquelle la précision du résultat est connue).

On considère la fonction f définie sur \mathbb{R} par $f(x) = \frac{1}{4}x^3 + x - 3$.

- 1) a) Démontrer que f est croissante sur \mathbb{R}
On admettra pour la suite que l'équation $f(x) = 0$ a une unique solution sur \mathbb{R} , notée α
b) Ecrire une fonction Python f qui :
 - reçoit en argument une valeur x
 - renvoie son image par la fonction f .c) Déterminer les images de 0 et 3 par f , et en déduire que $\alpha \in [0; 3]$.
- 2) a) On considère un intervalle $[a; b]$ contenant α et on pose $m = \frac{a+b}{2}$.
Justifier que : (*) si $f(a) \times f(m) < 0$ alors $\alpha \in [a; m]$, et sinon $\alpha \in [m; b]$
b) En utilisant (*), écrire une fonction Python **etap_dichoto** qui :
 - reçoit en arguments une fonction f et les bornes a et b d'un intervalle contenant α
 - renvoie les bornes a et b d'un nouvel intervalle contenant α .c) A partir de l'intervalle $[a; b] = [0; 3]$, obtenir successivement 3 nouveaux intervalles contenant α .
d) Que peut-on dire de la longueur de chaque intervalle obtenu par rapport à la précédente ?
- 3) a) Ecrire une fonction Python **dichoto_iter** qui :
 - reçoit en arguments une fonction f , les bornes a et b d'un intervalle contenant α et un entier n
 - renvoie les bornes d'un nouvel intervalle contenant α obtenu en répétant n fois la fonction précédente.b) Tester avec la fonction f de l'énoncé en partant de l'intervalle $[0; 3]$ et en répétant 10 fois la méthode.
- 4) a) Ecrire une fonction Python **dichoto_test** qui :
 - reçoit en arguments la fonction f , les bornes a et b d'un intervalle contenant α et une valeur h
 - renvoie les bornes du premier intervalle de longueur inférieure à h obtenu avec la méthode décrite précédemment.b) Tester avec la fonction f de l'énoncé pour obtenir un encadrement de α à 10^{-5} près.
- 5) a) Proposer et coder en Python des fonctions g et h s'annulant respectivement en $\sqrt{5}$ et $\sqrt[3]{7}$.
b) A l'aide des fonctions Python précédentes, proposer des encadrements de ces deux nombres à 10^{-7} près.

