

# Devoir maison n°4 : Méthode de Newton

Jules Charlier, Thomas Diot, Pierre Gallois, Jim Garnier  
TE1

## Partie A - Description de la méthode de Newton

1)

D'une part on sait que la fonction  $f$  est dérivable donc continue sur  $[a, b]$  et qu'elle y est strictement monotone car  $f'$  strictement négative. D'autre part, on dispose de  $f(a) > 0$  et de  $f(b) < 0$ .

Ainsi, d'après le corollaire du Théorème des Valeurs Intermédiaires, il existe un unique  $\alpha \in [a, b]$  tel que  $f(\alpha) = 0$ .

---

2)

a)

Soit  $u \in [a, b]$ . On note  $\tau_u$  la tangente à la courbe représentative de  $f$  au point d'abscisse  $u$ .

Ainsi, l'équation de  $\tau_u$  est donnée par :  $y = f'(u)(x - u) + f(u)$

Or  $y = 0 \Leftrightarrow x = u - \frac{f(u)}{f'(u)}$ .

Par conséquent,  $\tau_u$  coupe donc l'axe des abscisses au point d'abscisse  $u - \frac{f(u)}{f'(u)}$ .

b)

## Partie B - Algorithmes

1)

---

2)

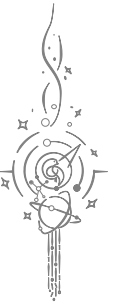
---

3)

Tentons maintenant de simplifier et d'optimiser ce code :

```
f = lambda x: x**3 - 2

def newton(f, x, h=1e-4, epsilon=1e-6):
    while abs(y := f(x)) > epsilon:
        derivee = (f(x + h) - f(x - h)) / (2 * h)
```



```
x -= (y / derivee)
return x
```

Pour aller encore plus loin dans la simplification, changeons de langage pour Haskell :

```
f :: (Num r) => r -> r
f x = x^3 - 2

derivee f x h = (f (x + h) - f (x - h)) / 2*h

newton f h e x =
  if (abs . f) x > e
  then newton f h e (x - (f(x) / (derivee f x h)))
  else x

main = do
  let result = newton f 1e-4 1e-6 1
  (putStrLn . show) result
```