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 τ_u la tangente à la courbe représentative de f au point d'abscisse u.

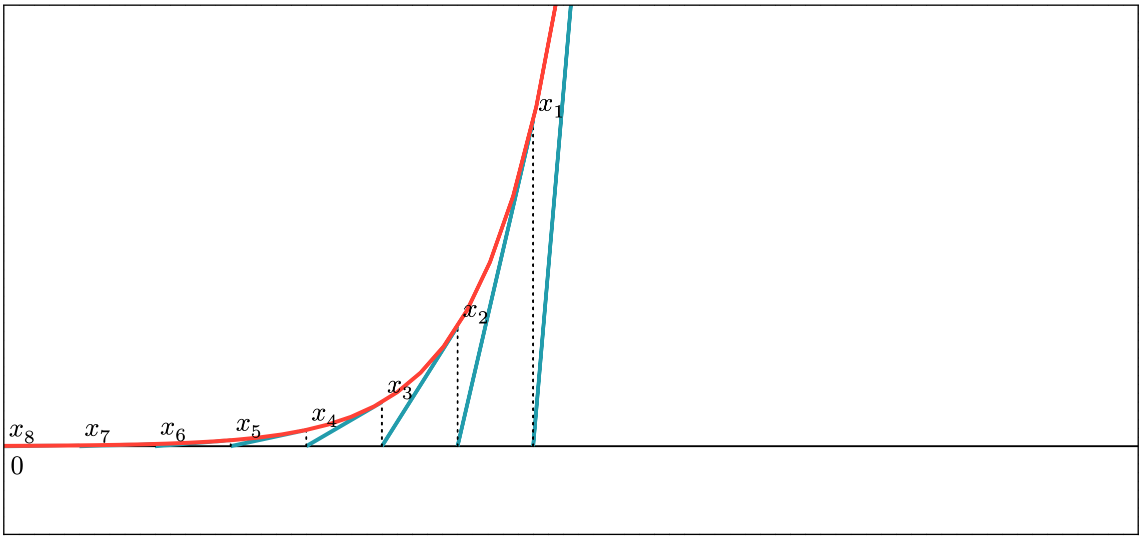
Ainsi, l'équation de τ_u est donnée par : $y = f'(u)(x - u) + f(u)$

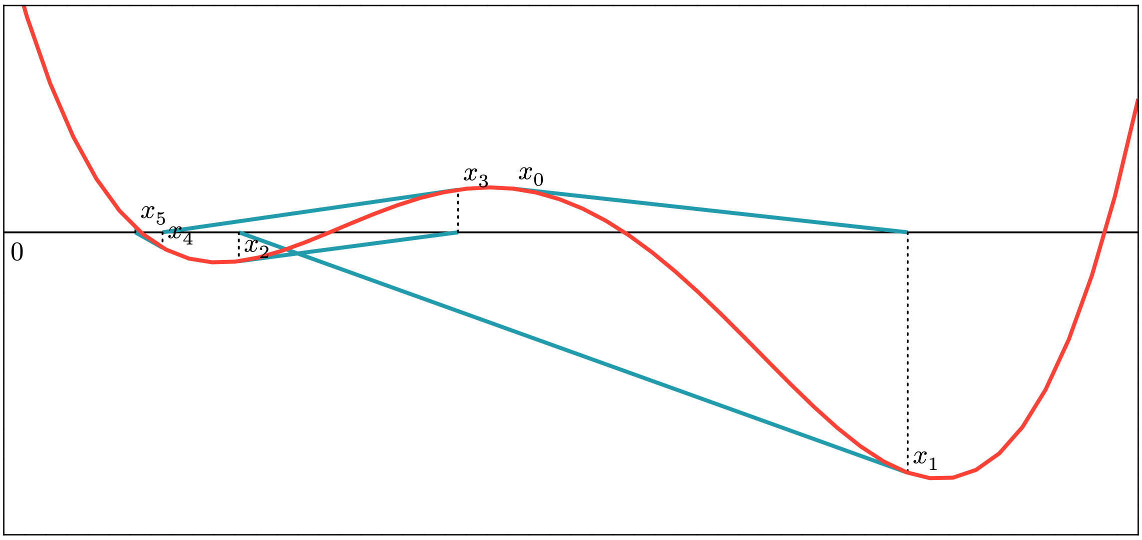
Or $y = 0 \Leftrightarrow x = u - \frac{f(u)}{f'(u)}$.

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

b) Considérons maintenant la fonction g définie sur $[a, b]$ par $g : x \longmapsto x - \frac{f(x)}{f'(x)}$ et la suite $(x_n)_{n \in N}$ par $x_0 = a$ et $x_{n+1} = g(x_n)$.

Cette suite se construit donc de la manière suivante : on part du point d'abscisse x_n sur la courbe représentative de f, on trace la tangente à cette courbe en ce point, puis on reporte l'intersection de cette tangente avec l'axe des abscisses pour obtenir le point d'abscisse x_{n+1}.





Partie B - Algorithmes

1)

---

2)

---

3)

Tentons maintenant de simplifier et d'optimiser ce code :

```python
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 language pour Haskell :

```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
```