# 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 $u$.

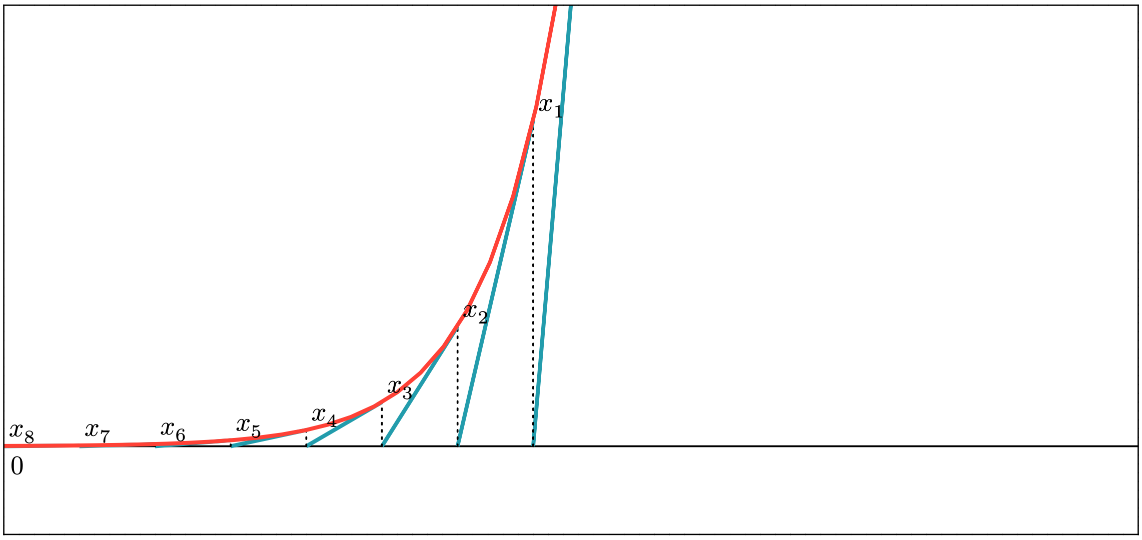
Ainsi, l'équation de $\tau_u$ est donnée par : $y = f'(u)(x - u) + f(u)$

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

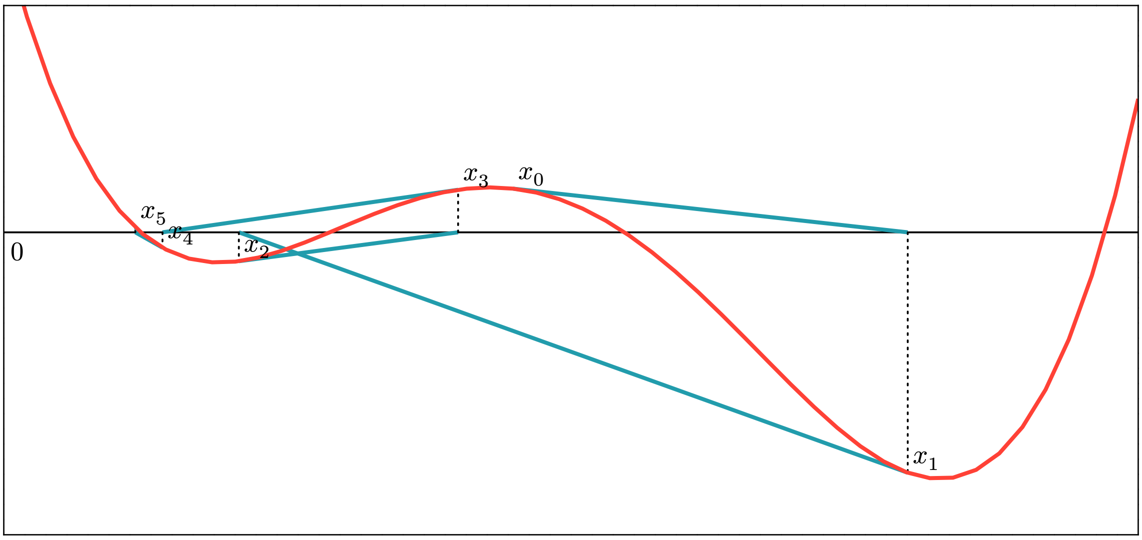
Par conséquent, $\tau_u$ coupe donc l'axe des abscisses au point d'abscisse $u - \frac{f(u)}{f'(u)}$.

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




[1]Schémas générés automatiquement pour n'importe quelle fonction. (programme dans le code source du DM, cf Github).




### 3)

**a)** La fonction $g$ est dérivable sur $[a, b]$ par composition de fonctions dérivables, dont $f'$ qui ne s'y annule pas, et pour tout $x \in [a, b]$, on a : $g'(x) = \frac{f(x)f''(x)}{(f'(x))^2}$

Ainsi, $g'$ de même signe que $f$ sur $[a, b]$ et donc $g$ est strictement croissante sur $[a, \alpha]$ et strictement décroissante sur $[\alpha, b]$.

**b)** Montrons que pour tout $n \in \mathbb{N}$, $a \leqslant x_n \leqslant b$ en procédant par récurrence :

- Initialisation : $x_0 = a$ donc $a \leqslant x_0 \leqslant b$.
- Hérédité : Soit $n \in \mathbb{N}$, supposons que $a \leqslant x_n \leqslant b$.
  - Si $x_n = \alpha$, alors $x_{n+1} = g(x_n) = g(\alpha) = \alpha$ donc $a \leqslant x_{n+1} \leqslant b$.
  - Si $x_n < \alpha$, alors par croissance de $g$ sur $[a, \alpha]$, on a $g(a) \leqslant g(x_n) \leqslant g(\alpha) = \alpha$. Or $a \leqslant g(a) \leqslant \alpha$ par l'hypothèse de récurrence, donc $a \leqslant x_{n+1} \leqslant b$.
  - Si $x_n > \alpha$, alors par décroissance de $g$ sur $[\alpha, b]$, on a $g(b) \geqslant g(x_n) \geqslant g(\alpha) = \alpha$. Or $b \geqslant g(b) \geqslant \alpha$ par l'hypothèse de récurrence, donc $a \leqslant x_{n+1} \leqslant b$.

Par conséquent, on en déduit que pour tout $n \in \mathbb{N}$, $a \leqslant x_n \leqslant b$.

### 4)

**a)**

**b)**

## Partie B - Vitesse de convergence

**1)** $\varphi : x \mapsto f(b) - f(x) - (b - x)f'(x) - \frac{(b-x)^2}{(b-a)^2}(f(b) - f(a) - (b - a)f'(a))$, définie sur $[a; b]$, est dérivable sur $[a; b]$ par opérations. Sa dérivée est pour tout $x \in [a; b]$ :

$$\varphi'(x) = (b - x)\left(\frac{2((a - b)f'(a) - f(a) + f(b))}{(b - a)^2} - f''(x)\right)$$

De plus, on vérifie que $\varphi(a) = \varphi(b) = 0$. Le théorème de Rolle donne l'existence de $c \in ]a : b[$ tel que $\varphi'(c) = 0$, c'est à dire :

$$f''(c) = \frac{2(f(b) - f(a) - (b-a)f'(a))}{(b-a)^2}$$
$$\Leftrightarrow f(b) - f(a) = (b-a)f'(a) + \frac{(b-a)^2}{2} f''(c)$$

Ce qu'il fallait démontrer.

---

**2)** $f'$ est dérivable donc continue sur $[a; b]$. Le théorème des bornes atteintes donne donc un $m \in \mathbb{R}$ tel que $f' \leqslant m$. De plus, $f'$ atteint $m$ : comme $f' < 0$, $m < 0$ et pour tout $x \in [a; b]$, $|f'(x)| \geqslant |m| > 0$.

De même, $f''$ est continue sur $[a; b]$, donc bornée sur $[a; b]$ : il existe donc $M \geqslant 0$ tel que $|f''(x)| \leqslant M$. Comme $f'' > 0$, $M > 0$ et on a ce que l'on voulait démontrer.

---

**3)** La formule de Taylor-Lagrange appliquée à $f$ sur $[x; \alpha]$ donne $c \in ]x; \alpha[$ tel que :

$$f(\alpha) - f(x) - (\alpha - x)f'(x) = \frac{(\alpha - x)^2}{2} f''(c)$$

En passant à la valeur absolue et en utilisant $f'' \leqslant M$, on obtient :

$$|f(\alpha) - f(x) - (\alpha - x)f'(x)| \leqslant \frac{(\alpha - x)^2}{2} M$$

## Partie C - 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 :: IO ()
main = do
    let initialGuess = 1.0  -- Initial guess for the root
        h = 1e-4            -- Small step for derivative approximation
        e = 1e-6            -- Tolerance level for convergence
        root = newton f h e initialGuess
    putStrLn $ "Root found: " ++ show root
```