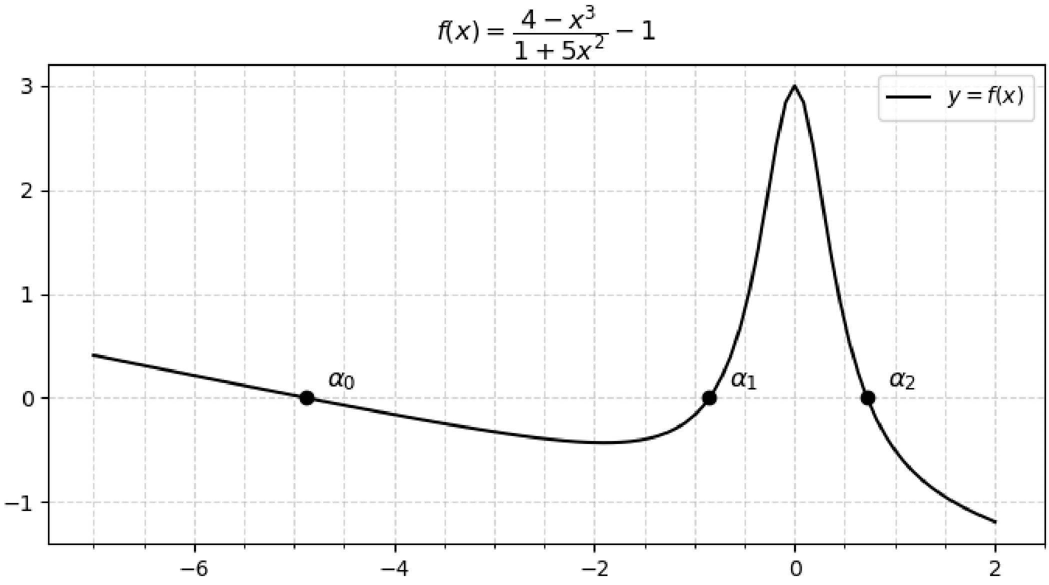
**TP 14 - Révisions**

## Résolution d'équation non-linéaire

Soit la fonction réelle d'une variable réelle $f : \mathbb{R} \to \mathbb{R}$ définie par

$$f(x) = \frac{4 - x^3}{1 + 5x^2} - 1$$

dont la représentation graphique est donnée ci-dessous sur l'intervalle $[-7, 2]$. On cherche à approcher numériquement les trois racines $\alpha_i$ de la fonction, représentées sur la figure suivante.




a) **Recherche de $\alpha_0$ :**
Pour approcher $\alpha_0$, on propose d'utiliser la méthode de la bissection *à la main*, c'est-à-dire en appliquant les différentes étapes sans écrire de code.

En partant de l'intervalle $I_1 = [-7, -3]$, quels sont les intervalles $I_k$ ($k = 1, 2, 3, 4$) considérés lors des quatre premières itérations de la méthode ainsi que les approximations correspondantes $x_j$ de la racine ?

b) **Recherche de $\alpha_1$** :
Complétez le code ci-dessous aux endroits manquants de manière à ce qu'il permette d'approcher numériquement $\alpha_1$ à l'aide de la méthode de Newton pour la fonction définie dans l'énoncé.

```
def f(x):
    return ______________________________

def f_prime ( x ) :
    return -(40*x+3*x**2+5*x**4)/(1+5*x**2)**2

def solve_Newton(f, x_0, fprime, eps, k_max):
    '''
    Resume :        Approximation d'un zero de la fonction f(x) a l'aide
                de la methode de Newton
    Parametres :
        f :         la fonction dont on recherche la racine
        x_0 :       une estimation initiale du zero de f
        fprime :    une fonction qui est la derivee de f(x)
        eps :       la tolerance souhaitee
        k_max :     le nombre maximum d'iterations autorisees
    Retourne :
        x :             un nombre approchant un zero de la fonction f
        evolution :     une liste contenant les valeurs x et f(x) a chaque
            iteration
    '''

#1. Initialisation des variables
    n_iter = 0

    x = ______________________________

    f_val = ______________________________

    evolution = [(x,f_val)]

#2. Methode de Newton
    while abs(f_val) > ______________ and n_iter < ______________:

        fprime_val = fprime(x)

        if ___________________________ < 1e-10:

            raise ValueError("Probable probleme de division par zero !")

        x = ______________________________

        n_iter += 1

        f_val = ______________________________

        evolution.______________________________((x,f_val))

    return ______________, ______________
```

c) Est-ce que la méthode de Newton donnera le même résultat en partant de $x_0 = -0.5$ qu'en partant de $x'_0 = 0.5$ ? Justifiez votre réponse.

d) **Recherche de $\alpha_2$ :**
Montrez que la fonction d'itération

$$g(x) = \sqrt{\frac{3 - x^3}{5}}$$

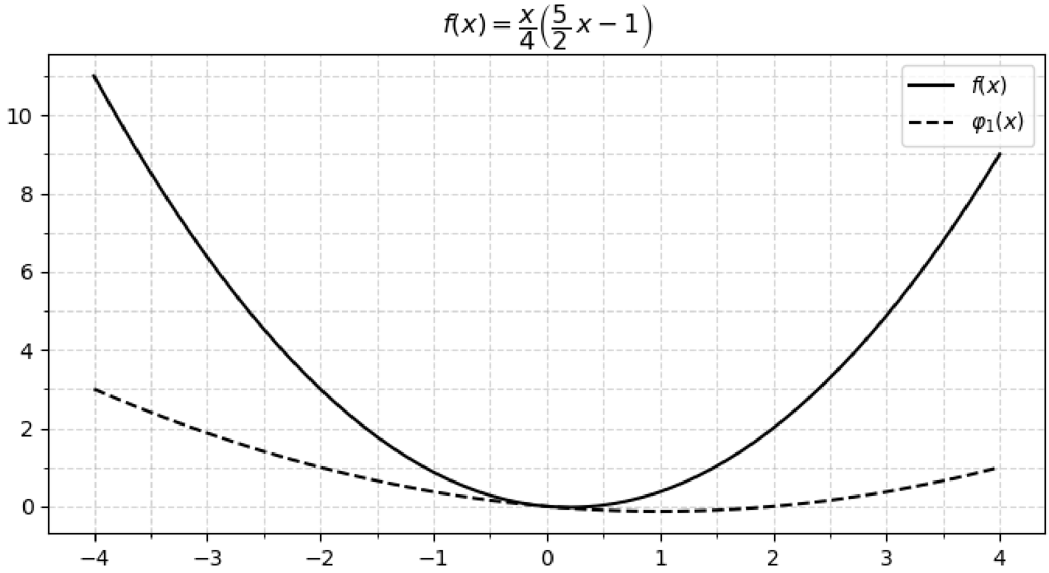
est une fonction acceptable pour une méthode de point fixe (méthode de Picard) destinée à trouver une valeur approchée d'un zéro de $f(x)$, et converge vers la racine souhaitée en partant de $x_0 = 1$.

## Intégration numérique

Soit la fonction

$$f(x) = \frac{x}{4}\left(\frac{5}{2}x - 1\right)$$

dont on cherche à approcher numériquement l'intégrale entre $a = -3$ et $b = 3$.




a) Donnez l'expression du premier élément $\varphi_1(x)$ de la base de Lagrange associée aux points $(t_1 = -2, t_2 = 0, t_3 = 2)$.

b) En utilisant la méthode du point milieu composite et en considérant une partition régulière du domaine d'intégration en trois sous-intervalles, donnez une approximation numérique de l'intégrale $\int_{-3}^{3} f(x)\,\mathrm{dx}$.

c) Nous cherchons maintenant à améliorer la précision en considérant une partition deux fois plus fine, c'est-à-dire $h' = \frac{h}{2}$. Que peut-on dire sur l'erreur $e_{h'}$ par rapport à $e_h$ ?

d) Sans effectuer de calcul, que peut-on dire sur l'erreur obtenue lorsqu'on approche l'intégrale par la méthode de Simpson sur le même intervalle ?

## Algorithmique

On donne ci-dessous l'implémentation d'un algorithme de tri, à laquelle nous avons rajouté l'instruction
`print(f"{i}e element: {L}")`.

```
def mafonction(L):
    n = len(L)
    for i in range(n-1,-1,-1):
        j = i
        while j < n-1 and L[j] > L[j+1]:
            L[j], L[j+1] = L[j+1], L[j]
            j += 1

        print(f"{i}e element: {L}")
```

a) Qu'affichent les instructions suivantes ?

```
L = [1, 3, 0, -6, 100, -1]
mafonction(L)
```

b) Quel algorithme vu en cours fonctionne selon le même principe que celui-ci ?

c) On dénote par $T(n)$ le temps de parcours de l'algorithme **`mafonction()`** lorsqu'il prend en entrée une liste de taille $n$, dans le pire des cas. Donner l'ordre de croissance de $T(n)$ en notation $\Theta(\cdot)$, en justifiant brièvement votre réponse.

d) Quelle propriété partagent les instances pour lequel le temps de parcours est minimal ? Donner l'ordre de croissance du temps de parcours **dans le meilleur des cas** en notation $\Theta(\cdot)$, en justifiant brièvement votre réponse.