# MNI-TP1

**TP 1 – Méthodes Numériques pour l'Ingénieur CM3**

## 1 Résolution des équations non linéaires

Le texte de cette session de travaux pratiques est également disponible ici
http://nbviewer.ipython.org/github/ecalzavarini/numerical-methods-at-polytech-lille/blob/master/MNI-TP1.ipynb

#### 1.0.1 Instructions pour ce TP

Pendant ce TP vous aurez à écrire plusieurs scripts (nous vous suggérons de les nommer script1.py , script2.py ,...)

Les scripts doivent être accompagnés par un document descriptif unique ( README.txt ). Dans ce fichier, vous devrez décrire le mode de fonctionnement des scripts et, si besoin, mettre vos commentaires. Merci d'y écrires aussi vos nomes et prénoms complets.

Tous les fichiers doivent etre mis dans un dossier appelé TP1-nom1-nom2 et ensuite être compressés dans un fichier TP1-nom1-nom2.tgz .

Enfin vous allez envoyer ce fichier par email à l'enseignant :
soit Enrico (enrico.calzavarini@polytech-lille.fr) ou Stefano (stefano.berti@polytech-lille.fr)

Vous avez une semaine de temps pour compléter le TP, c'est-à-dire que la date limite pour envoyer vos travaux est 7 jours après la date du TP courant.

### 1.1 Objectif

Ecrire un script Python permettant la recherche des racines d'une équation quelconque $f(x) = 0$ par la méthode :

a) de bissection (recherche dichotomique),

b) de la tangente (de Newton-Raphson) ,

en utilisant le critère d'arrêt $|x_{n+1} - x_n| < \varepsilon$ où la valeur de $\varepsilon$ sera précisée par l'utilisateur.
La fonction $f(x)$ sera définie à l'aide d'une *function* dans le script.

### 1.2 Programmation et validation

Déterminer la racine de l'équation

$$f(x) = x^4 - 2x^3 - 11x^2 + 12x$$

par la méthode de bissection (dans l'intervalle $x \in [3.2, 8.2]$) et de la tangente (en commençant les itérations par $x_0 = 8.2$).

On effectuera les calculs de la racine avec les précisions $\varepsilon = 10^{-k}$ avec $k = 1, \ldots, 6$.

Tracer au préalable le graphique de la fonction $f(x)$ dans l'intervalle considéré.

Pour chaque calcul préciser : la valeur de la racine $x$ trouvée, de la fonction $f(x)$, de l'erreur absolue $e = |x - x^*|$ (avec $x^*$ le valeur exacte, égale ici à 4.0) ainsi que le nombre d'itérations effectuées.

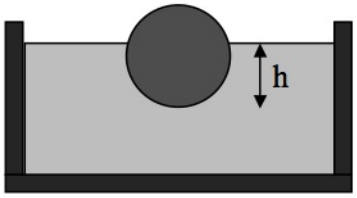
Les valeurs obtenues doivent être affichées de façon claire sur l'écran par le script et dans le compte-rendu dans un tableau.

Formuler une conclusion sur les résultats obtenus.

### 1.3 Application

Une boule de rayon R et de masse volumique $\rho_l$ est placée dans un réservoir rempli d'un liquide au repos de masse volumique $\rho_f = 1000Kg/m^3$ (eau). La boule s'enfonce alors d'une hauteur $h$ (voir la figure ci-dessous).

Le but de cet exercice est de déterminer, à l'aide des méthodes numériques vues auparavant, cette hauteur $h$ en fonction de la masse volumique de la boule $\rho_l$ .




**Question 1 :** En considérant l'équilibre des forces en présence, donner la relation permettant de déterminer $h = f(\rho_l, \rho_f, R)$.

Rappel :

Le volume d'une sphère de rayon $r$ est donné par l'expression

$$V_s = 4/3\pi r^3$$

Le volume d'une calotte sphérique de rayon $r$ et de hauteur $h$ est

$$V_c = \pi h^2(3r - h)/3$$

**Question 2 :** Pour $R = 0.125m$, et pour les valeurs suivantes de $\rho_l$

a) $\rho_l = 1800Kg/m^3$ (plexiglas) ,

b) $\rho_l = 1000Kg/m^3$ (caoutchouc) ,

c) $\rho_l = 400Kg/m^3$ (pin) ,

la boule coulera-t-elle ou non? Sinon, de quelle profondeur $h$ s'enfoncera-t-elle?

Pour le trois cas tracer au préalable le graphique de la fonction $f(x)$ dans l'intervalle étudié.

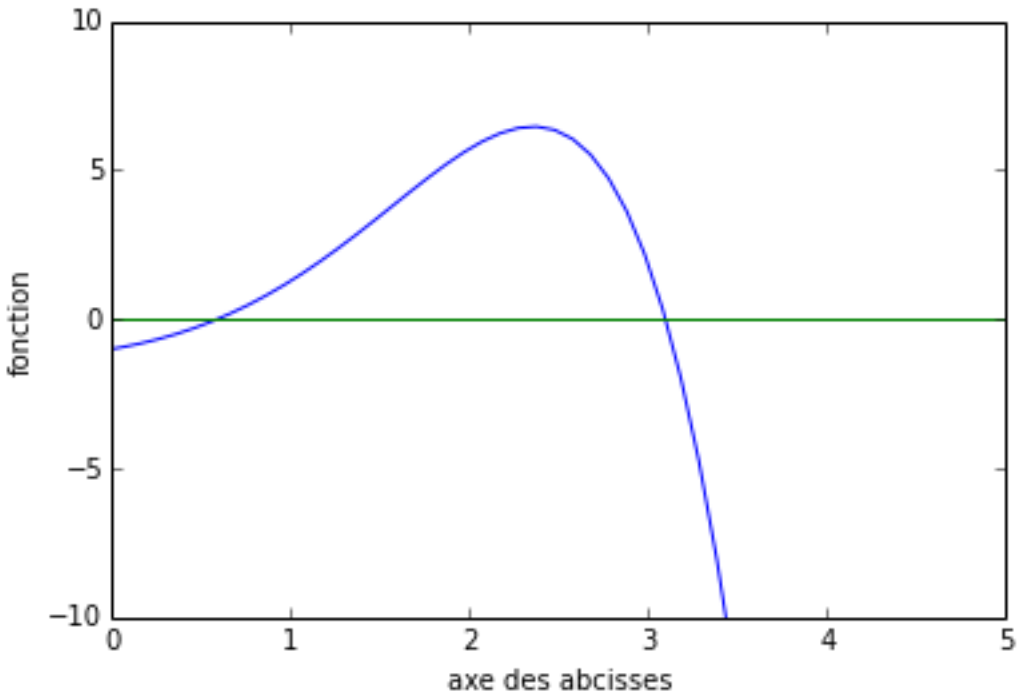
**Bonus:** Déterminer $h = f(\rho_l, \rho_f, R)$ à l'aide de la méthode de la corde (ou de la regula-falsi).

**Rappel : pour tracer un graphique en Python**

```
In [2]: import numpy as np
        import matplotlib.pyplot as plt

        %matplotlib inline
        #ignorez la ligne ci-dessus.
        #c'est juste une commande de notre éditeur de texte

        # nous définissons une liste (array) avec Numpy
        x=np.linspace(-5,5,100)
        # on utilise les fonctions sinus et exponentiel de Numpy
        plt.plot(x,np.sin(x)*np.exp(x)-1)
        plt.plot(x,np.zeros(100))
        plt.ylabel('fonction')
        plt.xlabel("axe des abcisses")
        plt.axis([0, 5, -10, 10])
        plt.show()
```




**Rappel : pour definir une function en Python**

```
In [3]: def f(x):
            return 2 * x + 1
```