TD : La simulation numérique
Numpy et Matplotlib

```
>>> import numpy as np
```

Sous Python, l'import du module *numpy* permet de réaliser des opérations pratiques sur les tableaux.
Les indices de ces tableaux commencent à 0.

Exercice 1 :

Construire un objet de type *numpy.ndarray*

- dont les termes sont les multiples de 3 compris entre 0 et 27 ;
- dont les termes sont les puissances de 2, de $2^0 = 1$ à $2^{12}=4096$;

Exercice 2 :

Pour cet exercice, on prend n = 1 000 000.
On pourra augmenter ou diminuer cette valeur en fonction de la machine utilisée.

1. Calculer $\sum_{i=0}^{n} i$ sans utiliser *numpy*.
   Chronométrer le temps nécessaire pour le calcul précédent, par exemple en utilisant **time.clock()**.
2. Utiliser un tableau *numpy* et la méthode **sum** pour calculer à nouveau la somme proposée et comparer le temps de calcul avec la méthode précédente.

NB :

Le module ***time*** : Permet de mesurer le temps écoulé et le temps CPU (le temps passé par un processus en mémoire centrale) :

- **time()** mesure le temps ordinaire, celui de l'horloge.
- **clock()** donne le temps CPU consommé par le processus Python actuel depuis le début de son exécution.

Exemples :

```
import time
e0 = time.time() # temps écoulé en sécondes depuis l'époque (01-01-1970 00:00:00)
c0 = time.clock() # temps CPU total en sécondes passé dans l'exécution du script
temps_ecoule = time.time() - e0
cpu_time = time.clock() - c0
```

Traçage des courbes : le module *matplotlib*

Exercice 3 :

Ecrire un script qui demande à l'utilisateur les coordonnées des trois points $A(x_A, y_A)$, $B(x_B, y_B)$ et $C(x_C, y_C)$, et trace le triangle ABC.

Exercice 4 :

Réaliser le graphe de la fonction $y(t) = v_0 t - \frac{1}{2} g t^2$ pour $v_0 = 10$, $g = 9.81$, et $t \in [0, 2v_0/g]$.
Le label sur l'axe des x devra être "temps (s)" et le label sur l'axe des y "hauteur (m)".

Exercice 5 :

La factorisation Cholesky, consiste, pour une matrice carrée **A** d'ordre **n** symétrique définie positive, à déterminer une matrice triangulaire inférieure **L** telle que **A=LL'** (avec L' = transposé de L). Les relations suivantes permettent de déterminer les éléments de **L** à partir de **A** :

- Initialement tous les éléments de **L** sont nuls
- $L_{00} = \sqrt{A_{00}}$
- $L_{j0} = \frac{A_{0j}}{L_{00}}$ avec j=1..n-1
- $L_{ii} = \sqrt{A_{ii} - \sum_{k=0}^{i-1} L_{ik}^2}$ avec i=1..n-1
- $L_{ji} = {(A_{ij} - \sum_{k=0}^{i-1} L_{ik}\, L_{jk})}/{L_{ii}}$ avec i=1..n-1 et j=i+1..n-1

Le déterminant de **A** est égal alors au carré du produit des éléments de la diagonale principale de **L**.

1) Ecrire une fonction **DECOMP** qui calcule la matrice **L** à partir d'une matrice **A** d'ordre **n** en utilisant les formules précédentes.
2) Ecrire une fonction **DETER** qui retourne le déterminant d'une matrice **A** d'ordre **n** sans utilisation de la commande **det**.

L'inverse d'une matrice carrée **A** d'ordre **n** est donné par la formule suivante :
$A^{-1}=C'/det(A)$

Notation:
C = matrice des cofacteurs dont les coefficients sont obtenus en utilisant la formule suivante :

$$C_{ij}=(-1)^{i+j}*det(A_{ij})$$

Avec : **C_{ij}** : L'élément de **C** à la ligne **i** et colonne **j**
A_{ij} : La matrice **A** privée de la ligne d'indice **i** et de la colonne d'indice **j**
det : Le déterminant de la matrice
C' : transposé de **C**

3) Ecrire une fonction **MINOR** qui supprime la ligne d'indice **i** et la colonne d'indice **j** d'une matrice **A** d'ordre **n**.
4) Ecrire une fonction **COFACT** qui calcule et retourne la matrice des cofacteurs d'une matrice **A** d'ordre **n**.
5) Ecrire une fonction **INVERSE** qui retourne l'inverse d'une matrice **A** d'ordre **n**.