Dichotomie

Exercice 1 : **Manipulation de données par dichotomie**

Le fichier `liste_triee.txt` contient des valeurs numériques entières triées par ordre croissants.

Q1 : Ecrire un programme qui importent ces données dans la mémoire vive sous la forme d'une liste d'entiers.

Q2 : Ecrire une fonction qui prennent en argument une liste de données triées par ordre croissant et un nombre x, et qui retourne les deux valeurs successives de la liste encadrant strictement le nombre x (on étudiera les différents cas particuliers que l'on peut rencontrer).

Q3 : Ecrire une fonction qui prennent en argument une liste de données triées par ordre croissant et un nombre x, et qui insère x dans la liste au bon endroit (on étudiera les différents cas particuliers que l'on peut rencontrer). Une nouvelle liste est renvoyée.

Q4 : Utiliser les fonctions ci-dessus pour trier la liste de valeurs contenus dans le fichier `liste_non_triee.txt`.

Exercice 2 : **Dichotomie sur une fonction**

On cherche à déterminer les racines du polynôme $f(x)$, avec une précision ε de 10^{-16} :

$$f(x) = x^3 - 2x^2 + 1$$

Partie 1 : Prise en main :

- Créer la fonction python qui, à l'argument x, renvoie :

$$f(x) = x^3 - 2x^2 + 1$$

- Tracer la courbe associée à cette fonction dans l'intervalle $[-1; 2]$

(pas de discrétisation : 0,01).

Partie 2 : Dichotomie

Q1 : A quelle condition la résolution par dichotomie est-elle opérationnelle ?

Q2 : Rappeler les arguments nécessaires à la méthode de dichotomie

Q3 : A partir du tracé de la courbe de f, préciser trois intervalles $[a_i, b_i]$ ($i \in \{1,2,3\}$) incluant une racine et où la fonction est monotone.

En utilisant l'algorithme de dichotomie vue en cours, programmer une fonction qui aux trois arguments : $fonction$, $[a, b]$, ε, renvoie la racine trouvée.

Q4 : Quelles valeurs de racine trouvez-vous ?

Codage des entiers et des flottants

Le module numpy permet d'imposer le type de certaines variables. Pour cela, il faut importer les types nécessaires :

☞ Taper les instructions :

```
from numpy import uint8
from numpy import float64
from numpy import float16
```

Partie 1 : Codage des nombres entiers naturels sur 8 bits.

✍ Justifier le résultat de l'instruction : `print(uint8(260))`

On pose `a = uint8(280)` et `b = uint8(240)`.

✍ Que valent `a`, `b`, `a+b`, `a-b`, `a//b` et `a/b` ? (justifier)

Partie 2 : Codage des nombres à virgule flottante

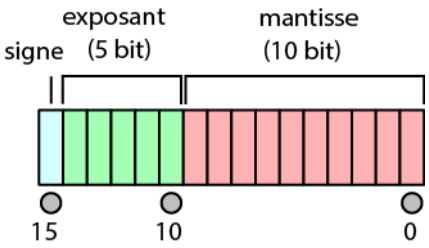
☞ Taper les instructions :

```
x=0.00005
x16=float16(x)
x64=float64(x)
print(1/x16)
print(1/x64)
print(format(1/x16,'.50f'))
print(format(1/x64,'.50f'))
```

✍ Justifier chaque affichage

Partie 3 : Décodage des nombres à virgule flottante sur 16 bits

Un nombre mystère est codé en machine en respectant la convention `float16`, c'est-à-dire en virgule flottante sur 16 bits. On rappelle ci-dessous la répartition des bits qui le compose, et les relations permettant de décoder sa valeur :



$$x = (-1)^s \cdot m \cdot 2^e$$
$$m = 1 + \sum m_i . 2^{-i}$$
$$e = \sum e_i . 2^i - 15$$

L'expression du nombre en machine est la suivante :

0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0

Q1. Quelle est la valeur décimale de ce nombre ?

La syntaxe ci-dessous permet d'afficher le code binaire d'un nombre `float`.

```
a=float16(???)
print(bin(a.view('H')))
```

Q2. En utilisant cette syntaxe, vérifier la valeur décimale trouvée.

Q3. Quelle est la valeur du double de ce nombre ? Quelle est son expression codée en `float16` ?

Q4. Quelle est la précision sur ce nombre ?

Partie 4 : Calculs corrects et incorrects en nombres flottants.

La syntaxe ci-dessous permet d'afficher les valeurs de la mantisse et de l'exposant d'un nombre flottant.

```
from math import frexp
print("mantisse : %1.26f , exposant : %2d" % frexp(???))
```

(remplacer les ??? par la valeur étudiée)

Q1. En utilisant le cde de la partie précédente, comparer les valeurs des mantisses et exposants des nombres suivants : 0.1 0.2 et 0.3

Q2. Justifier que le test 0.1+0.1==0.2 soit vrai, et que 0.1+0.1+0.1==0.3 soit faux

Exercice 3 : **Simulation géométrique d'un système bielle manivelle**

On se place dans le cas d'un compresseur à air. Le mouvement d'entrée est la rotation de la manivelle, le mouvement de sortie est la translation alternative du piston. (cf. Annexe - modélisation et paramétrage).

L'objectif est de décrire les mouvements en fonction du temps. Cela n'est pas possible dans une approche numérique basée sur des listes. Le temps sera donc défini uniquement pour certains instants suffisamment proches pour assimiler la liste à un intervalle continu.
On souhaite donc définir des instants "t" toutes les millisecondes pendant une période de 50 ms.

☞ Créer une liste *temps* qui contiendra les valeurs successives du temps.

Partie 1 : Définition de l'angle Alpha α: fonction à deux arguments

La vitesse de rotation d'entrée *omega* égale à 1500 tr/min.

✍ Définir α en fonction d'*omega* et de *temps*.

☞ Créer une liste *Alpha* contenant les valeurs de α aux instants étudiés.

Partie 2 : Calcul de Theta θ et de Lambda λ

☞ définir les longueurs *L1* (50mm) et *L2* (80mm)

☞ Créer deux listes, à partir des équations donnés en annexe, *Theta* et *Lambda* en fonction du temps.

Remarques :

- Les fonctions trigonométriques sont fonctions d'angle en radians.
- Les fonctions `acos()` et `asin()` ne sont pas chargées par défaut. Pour les charger, taper l'instruction :

```
from math import acos,asin
```

Partie 3 : Affichage du graphe Lambda en fonction de Alpha

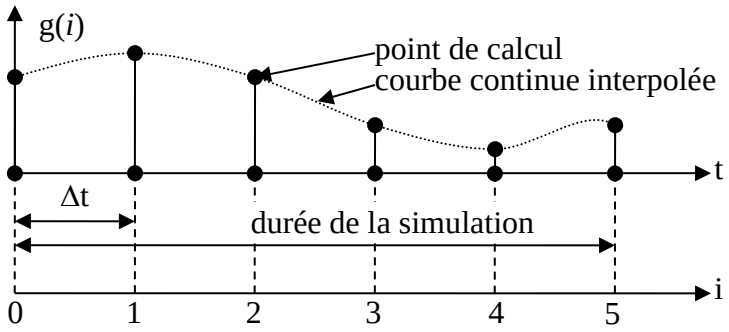
☞ Afficher la courbe de *Lambda* en fonction de *Alpha*

✍ Le mouvement est-il sinusoïdal ? Relancer le programme avec *L1*=60 et *L1*=20. Commenter.

Remarque 1 : échantillonnage

En méthode numérique, les calculs ne se font pas en continu, mais à des instants réguliers, appelés **point de calcul**, référencés par un **index** (*i*). La grandeur calculée g(t) n'est donc calculée qu'à ces instants. On la note aussi g(*i*).

L'intervalle de temps séparant deux points (Δt) est appelé le **pas** ou la **période d'échantillonnage**. Un point de calcul d'index *i* est alors défini à un instant t=*i*.Δt



Remarque 2 : Paramètres de simulation

Pour toute simulation numérique, il est nécessaire de définir :

- le pas (période d'échantillonnage)
- la durée de la simulation (le nombre total d'index)
- un mouvement d'entrée, par sa nature et ses valeurs à chaque instant
- Les conditions initiales (positions initiales)

Dans notre exemple, les paramètres de simulation utilisés sont :

- Une période d'échantillonnage : 1ms,
- la durée : 50ms,
- le mouvement d'entrée : la rotation définie par α,
- la valeur d'initiale du mouvement d'entrée : α_0=0.

Tous les autres mouvements, dont celui de sortie, peuvent être alors calculés.

Partie 4 : Analyse de la courbe : détermination de la course.

On cherche la course du piston, c'est-à-dire l'amplitude du déplacement de celui-ci. Elle peut facilement être déterminée en calculant : $\lambda_{max} - \lambda_{min}$

- ☞ Proposer une fonction, basée sur une boucle, permettant de déterminer la valeur maximale de Lambda. Cette fonction devra aussi renvoyer l'instant de ce maximum.
- ☞ Utiliser cette fonction pour déterminer la course du piston et la durée d'un aller de celui-ci.

Partie 5 : Détermination de la position où la vitesse du piston est maximale

Il peut être démontré que la vitesse du piston est maximale si $\overrightarrow{y_1}$ et $\overrightarrow{y_2}$ sont perpendiculaires.

- ☞ Compléter le programme précédant pour afficher l'angle $(\overrightarrow{y_1}, \overrightarrow{y_2})$ en fonction du temps. Comment évolue cet angle ?

- ✍ Proposer un algorithme dichotomique qui recherche une valeur dans une liste triée décroissante. Proposer une condition d'arrêt.
- ☞ Tester et valider le programme pour déterminer la position pour laquelle $\overrightarrow{y_1}$ et $\overrightarrow{y_2}$ sont perpendiculaires

Partie 6 : Détermination d'une trajectoire

- ✍ Déterminer l'expression des coordonnées en X et Y, dans le repère lié au bâti, du point A et du point C et D, respectivement situés au tiers et au deux tiers du segment [AB]
- ☞ Faire calculer ces coordonnées des points A, C et D en fonction du temps.
- ☞ Représenter dans un nouveau graphe la trajectoire de A par rapport au bâti.

On va maintenant ajouter la trajectoire des points C, D et B.

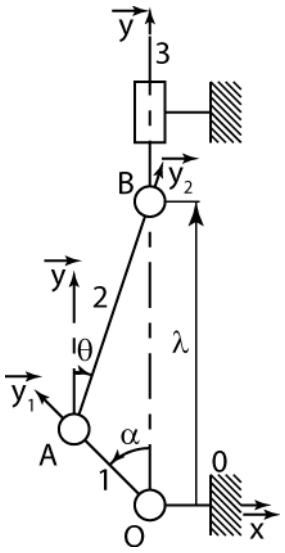
- ☞ Sans fermer cette figure, faire afficher la trajectoire des autres points.
- ☞ Enfin, sur une nouvelle figure, faire afficher le segment [AB] pour chacune des positions du temps.

Annexe : Système Bielle - Manivelle

Un système bielle – manivelle est un mécanisme de transformation du mouvement dont les applications sont très nombreuses.

La manivelle **1**, appelée aussi vilebrequin, a un mouvement de rotation par rapport au bâti **0**. La bielle **2** admet une rotation par rapport à la manivelle et par rapport au piston **3**. Ce piston a un mouvement de translation par rapport au bâti.
Le mouvement d'entrée est la rotation continue de la manivelle **1** par rapport à **0** (liée à un actionneur électrique). Le mouvement de sortie est la translation alternative du piston **3**.

On en déduit le schéma cinématique suivant :
Où :
$\alpha = (\vec{y}, \overrightarrow{y_1})$
$\theta = (\vec{y}, \overrightarrow{y_2})$
$\lambda = \overrightarrow{OB}.\vec{y}$
Et $l_1 = OA$; $l_2 = AB$



L'étude géométrique menée théoriquement donne :

$$\lambda = l_1 \cos\alpha + \sqrt{{l_2}^2 - {l_1}^2 \sin^2\alpha}$$

$$\theta = -\arcsin\left(\frac{l_1 \sin\alpha}{l_2}\right)$$

$$\dot{\lambda} = -l_1.sin\alpha.\dot{\alpha} - \frac{l_1^2.sin\alpha.cos\alpha}{\sqrt{l_2^2 - l_1^2 \sin^2\alpha}}\dot{\alpha}$$