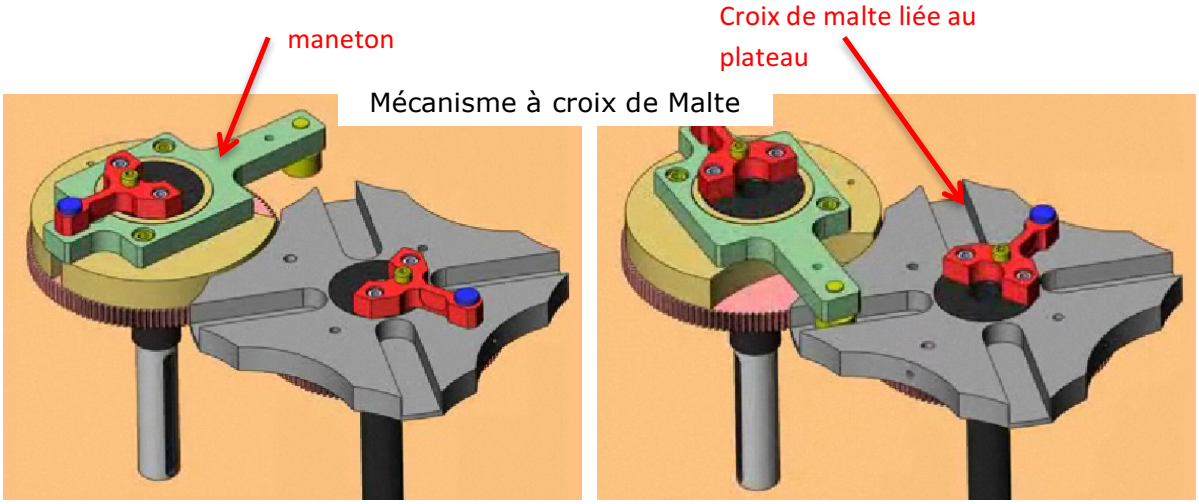
Mouvement du plateau de la capsuleuse de bocaux (labo SII)

Rappel : si ce n'est pas déjà fait, copier le dossier complet du TP sur votre compte.

1. Mise en situation

Le conditionnement de nombreux produits alimentaires est réalisé dans des bocaux en verre fermés par des capsules vissées. La société RAVOUX, spécialisée dans le conditionnement, a créé ce prototype afin d'optimiser ses machines de production. Il est donc équipé de nombreux capteurs permettant, via un ordinateur, d'optimiser les paramètres de production tels que la qualité totale, la production maximale...

Nous nous intéressons dans ce sujet uniquement au traitement informatique des données issues du capteur de vitesse de rotation du plateau (notée $\dot{\theta}$) transférant les bocaux de poste en poste (remplissage, serrage du bouchon ….). Ce mouvement de rotation est obtenu grâce au mécanisme à croix de malte. Le maneton est animé d'un mouvement de rotation continue par un moteur électrique, entrainant la croix de malte (liée au plateau) en une rotation discontinue (angle θ).



Objectif général : Déterminer l'évolution de la position du plateau au cours du temps et comparer avec le résultat théorique. Déterminer l'accélération angulaire afin de connaître le couple subi par le plateau.

Objectif informatique :

- **Lire et traiter** des données sous forme de fichiers **.txt.**
- **Tracer des courbes** à l'aide de la bibliothèque Python **matplotlib.**
- Réaliser des **intégrations numériques.**
- **Ecrire** des données dans un fichier **.txt**.

La programmation se fera dans la trame en Python proposée : *TP6_capsuleuse.py*.

Les données issues des capteurs sont stockées dans le fichier *vitesses.txt* du dossier **Mesures**. Ce fichier contient notamment le temps de chaque prise de mesure, en secondes, et la valeur de la vitesse de rotation instantanée à chaque prise de mesure, en tour/min.

2. Travail proposé

2.1. Découverte de la trame Python

Travail 1. Ouvrir la trame python *TP6_capsuleuse.py* et vérifier qu'elle contient dans l'ordre la définition des deux fonctions suivantes.

- La fonction `integrale_rect()`, incomplète, devant réaliser une intégrations par la méthode des rectangles, où a et b sont les bornes d'intégration, et listX et listY les coordonnées des points définissant le profil à intégrer.
- La fonction `lecture_fichier()`.

Le programme principal sera écrit à la suite, et devra faire appel aux fonctions définies précédemment pour atteindre les objectifs du TP.

2.2. Calcul d'une intégration numérique

Travail 2. Compléter la fonction `integrale_rect(ListX,ListY,a,b)` réalisant l'intégration $\int_a^b y(x)dx$. La fonction $y(x)$ n'est pas connue, mais on connaît une liste de points dont les abscisses et les ordonnées sont stockées dans les `ListX` et `ListY` :

$$\textbf{ListX=[}\ x_0, x_1, x_2, \dots, x_{n-2}, x_{n-1}\textbf{]}$$

$$\text{et}\ \ \textbf{ListX=[}\ f(x_0), f(x_1), f(x_2), \dots, f(x_{n-2}), f(x_{n-1})\textbf{]}$$

On supposera que $a \geq x_0$ et $b \leq x_{n-1}$

2.3. Extraction des données

L'extraction des données contenues dans un fichier nécessite préalablement l'ouverture du fichier. Ci-dessous la marche générale à suivre pour ouvrir un fichier *nom.txt* :

```
fich = open('nom.txt','r') # ouvre  le fichier nom.txt en mode
lecture ('r'=read), sous le nom fich
contenu=fich.readlines() # extraction du contenus de fich, sous
forme d'une liste
fich.close()  # fermeture du fichier
```

La variable ***contenu*** contient alors l'ensemble des caractères du fichier « nom.txt » sous forme d'une liste dont chaque élément est une ligne du fichier lu.

Remarque : si le fichier n'est pas dans le même répertoire que le .py, alors il faut donner le chemin d'accès complet vers le fichier, par exemple, dans notre cas : `'Mesures/nom.txt'`

Travail 3. Observer la fonction `lecture_fichier(namefile)` et préciser à quoi sert chacune des lignes.

Travail 4. Utiliser cette fonction pour extraire le temps et la vitesse de rotation du plateau tournant mesurés. Afficher le résultat pour vérification.

Bilan : Vous venez donc d'extraire, à partir d'un fichier .txt, une liste de décimaux représentant les grandeurs mesurées par un système ☺. Vous pouvez donc exploiter ces données pour les utiliser à votre convenance.

2.4. Exploitation des données

Nous allons dans un premier temps tracer la répresentation graphique des données extaites. Pour cela, nous allons utiliser la librairie libre et gratuite : `matplotlib`. Cette librairie est déjà intégrée à Spyder.

Prendre connaissance de l'annexe matplotlib (courbe simple).

Travail 5. Tracer le graphe représentant l'évolution de la vitesse de rotation du plateau en fonction du temps. On affichera la vitesse de rotation en *°/s*, sachant qu'elle est mesurée en *tr/min* par le capteur.

On rappelle que, mathématiquement, l'angle de rotation du plateau est obtenu par l'intégration de la vitesse.

Travail 6. En exploitant la fonction `Integrale_rect()`, déterminer la liste `list_position` des valeurs de l'angle de rotation en degrés.

Travail 7. Tracer alors le graphe représentant l'évolution de l'angle de rotation du plateau en fonction du temps, sur le même graphe que celui de la vitesse de rotation. On consultera l'annexe matplotlib (plusieurs courbes avec légende). Quel est l'angle final obtenu ? Que remarquez-vous, sur la dernière partie du graphe ? Comment expliquer ce résultat ?

Pour créer un fichier texte *nom.txt* et écrire dans ce fichier, on utilise les commandes suivantes :

```
fich = open(nom.txt,'w') # ouvre  le fichier nom.txt en mode
écriture ('w'=write)
fich.write('début du texte à écrire dans le fichier…) #
écriture d'une chaîne de caractères
…
fich.write('suite ou fin du texte à écrire dans le fichier…) #
suite/fin de l'écriture
fich.close()  # fermeture du fichier
```

Travail 8. Compléter la fonction `ecriture_fichier(namefile,Lx,Ly)` qui, à partir d'une liste `Lx` de flottants (abscisses), et une liste `Ly` de flottants (ordonnées), crée un fichier texte nommé *namefile* contenant, à chaque ligne, `Lx[i] Ly[i]` convertis en chaînes de caractères, séparés par une tabulation. On gardera le point comme séparateur décimal.

Travail 9. Utiliser cette fonction pour créer un fichier texte *positions.txt* à partir de la liste des positions calculées et des temps correspondants. Vérifier le résultat en ouvrant le fichier créé.

Travail supplémentaire si le temps le permet :

On donne le résultat de l'étude théorique de cinématique :

$$\dot{\theta} = \frac{R^2 + L.R.\cos(\omega(t + t_0))}{R^2 + 2.LR.\cos(\omega(t + t_0) + L^2}\omega,$$

où ω est la vitesse de rotation du maneton en entrée, et vaut 10 tr/min, et avec les paramètres géométriques *L* = 125 mm et *R* = 78 mm, et temporel $t_0 = 1s$.

Travail 10. Tracer la courbe théorique de l'évolution de $\dot{\theta}$ en fonction du temps, et comparer avec le résultat des mesures.

Annexe MATPLOTLIB

La librairie *matplotlib* permet de tracer des nuages de points dont les abscisses et ordonnées seront données sous la forme de listes.

Etape 1. Ecrire en en-tête de programme la commande permettant d'importer la librairie :

```
import matplotlib.pyplot as plt
```

L'utilisation de la librairie se fera donc grâce à l'instance de classe nommée plt.

Etape 2.

Courbe simple : Soient les abscisses stockées dans la liste notée LX1, et les ordonnées dans la liste LY1. Pour tracer la courbe, il faut alors écrire :

```
plt.plot(LX1,LY1,'g-') # déclaration des listes abscisse et ordonnée, g pour
green, - pour ligne continue
plt.xlabel('grandeur en abscisse') # définition étiquette des abscisses
plt.ylabel('grandeur en ordonnée') # définition étiquette des ordonnées
plt.show()  # affiche la courbe
```

Plusieurs courbes sur un graphe : il est possible de tracer plusieurs courbes sur un même graphe en rajoutant une ligne plt.plot(LX2,LY2,'b-') avant la ligne plt.show(), exemple :

```
plt.plot(LX1,LY1,'r-') # déclaration courbe 1, en rouge
plt.plot(LX2,LY2,'b+') # déclaration courbe 2,+ pour des croix à chaque point
plt.xlabel('grandeur en abscisse') # définition étiquette des abscisses
plt.ylabel('grandeur en ordonnée') # définition étiquette des ordonnées
plt.show()  # affiche la courbe
```

Plusieurs courbes sur un graphe avec légende: il suffit de rajouter un `label` dans la déclaration des courbes, et un appel à `plt.legend` :

```
plt.plot(LX1,LY1,'r-',label='courbe 1')  # déclaration courbe 1
plt.plot(LX2,LY2,'b+', label='courbe 2') # déclaration courbe 2
plt.xlabel('grandeur en abscisse') # définition étiquette des abscisses
legend = plt.legend(loc="upper left") # permet d'afficher la légende en haut
à gauche du graphe
plt.show()  # affiche la courbe
```

character	description
'-'	solid line style
'--'	dashed line style
'-.'	dash-dot line style
':'	dotted line style
'.'	point marker
','	pixel marker
'o'	circle marker
'v'	triangle_down marker
'^'	triangle_up marker
'<'	triangle_left marker
'>'	triangle_right marker
'1'	tri_down marker
'2'	tri_up marker
'3'	tri_left marker
'4'	tri_right marker
's'	square marker
'p'	pentagon marker
'*'	star marker
'h'	hexagon1 marker
'H'	hexagon2 marker
'+'	plus marker
'x'	x marker
'D'	diamond marker
'd'	thin_diamond marker
'|'	vline marker
'_'	hline marker

character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white