Devoir d'informatique

Consignes

- Le devoir se fera avec un éditeur python que vous aurez préalablement installé.
- Télécharger le fichier **DM2_eleve.py** sur l'espace `https://ptsilamartin.github.io/info/DM/DM2_eleve.py`.
- Renommer le fichier sous la forme : **DM2_NOM_prénom.py** en remplaçant **Nom** et **prénom** par vos noms et prénoms.
- Compléter le code en répondant aux questions. Vos tests, remarques et réponses textuelles, apparaîtront sous forme de commentaires (#)
- **Pour les PTSI 1**, votre fichier **DM2_NOM_prénom.py** sera a envoyé par mail à Mme Cotta.
- **Pour les PTSI 2**, votre fichier **DM2_NOM_prénom.py** sera déposé sur la plateforme `http://envoi.lamartin.fr/` en choisissant M. Pessoles dans le champ « Pour qui? ».

A rendre avant le 4 janvier 2024 minuit.

1 Tracé d'une came

On étudie l'usinage de la came d'un moteur à pistons axiaux dont l'architecture est donnée en figure 1. Les ondulations de la came permettent d'obtenir un mouvement de rotation de l'arbre moteur, à partir du mouvement de translation des pistons induit par une pression hydraulique.

L'usinage de cette came se fait par enlèvement de matière par une fraise dont le centre est placé au point C. Un système de pignon (3) roulant sans glisser en I sur une couronne (1) permet d'obtenir la trajectoire souhaitée du point C.

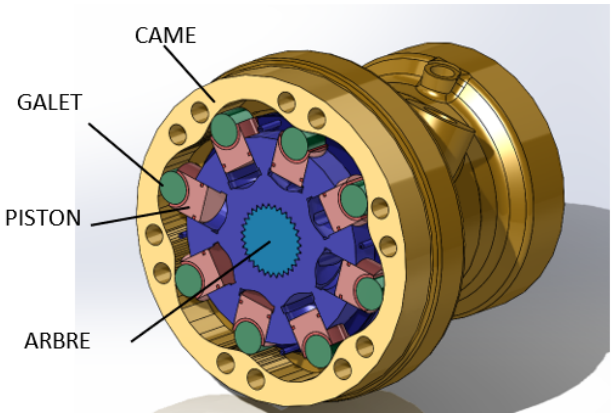

FIGURE 1 – Architecture d'un moteur à pistons radiaux

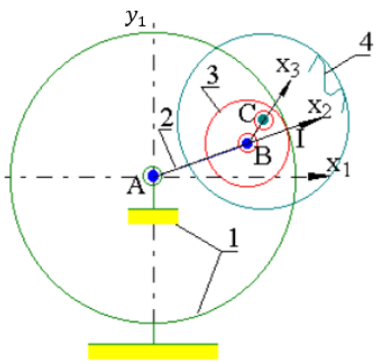

FIGURE 2 – Principe du procédé d'usinage

Les équations de la courbe paramétrée de la trajectoire du point C dans le repère $(A, \vec{x}_1, \vec{y}_1)$ sont données ci-dessous :

$$\begin{cases} x_C(\alpha) = R\cos(\alpha) + e\cos(\alpha(1-k)) \\ y_C(\alpha) = R\sin(\alpha) + e\sin(\alpha(1-k)) \end{cases}$$

Le paramètre α est un angle qui prend des valeurs allant de 0 à 2π.
On définit les grandeurs suivantes : $R_1 = 70\,\text{mm}$; $k = 6$, $e = 6\,\text{mm}$; $R_3 = R_1/k$; $R = R_1 - R_3$.
Le rayon de la fraise est $R_4 = 30\,\text{mm}$.

Ces grandeurs sont déjà définies dans le script qui vous a été fourni. Il s'agit de variables globales qui pourront être utilisées dans vos fonctions.
Les lignes de code permettant d'importer les bibliothèques utiles pour ce DM sont déjà présentes.

Tracé de la trajectoire du point C

Question 1 *Ecrire une fonction* `liste_angle(n: int)->[]` *contenant* `n` *valeurs, réparties uniformément, dont la première valeur est* 0 *et la dernière est* 2π. *On rappelle que l'on obtient la valeur de* π *avec l'instruction* `np.pi`. *Appeler cette fonction pour créer une liste* `les_alpha` *contenant 500 valeurs équitablement réparties de l'angle* α.

Question 2 *Écrire une fonction* `calcule_point_traj(a:float)-> (float,float)` *permettant de calculer les coordonnées d'un point de la trajectoire de C pour un angle* α *donné (argument* `a`*). Cette fonction retourne donc le couple* (x_C, y_C).

Question 3 *Tracer cette trajectoire pour* α *allant de* 0 *à* 2π.

Indication : il faut créer 2 listes `les_x` et `les_y` contenant chacune des coordonnées prises par le point C pour chaque valeur de α. Veillez aussi à ce que les axes soient à la même échelle : `plt.axis('equal')`.

Tracé de la trajectoire d'un point de la fraise

Pour chaque position du point C, on considère que la fraise enlève de la matière dans un cercle de centre C et de rayon R_4. On rappelle l'équation paramétrée d'un cercle de centre $C(x_C, y_C)$ et de rayon R_C :

$$\begin{cases} x(\theta) = x_C + R_C\cos(\theta) \\ y(\theta) = y_C + R_C\sin(\theta) \end{cases}$$

Question 4 *Écrire une fonction* `trace_cercle(xc : float , yc : float , Rc : float)-> None` *permettant de tracer sur la figure courante, un cercle de centre C (*`xc`*,* `yc`*) et de rayon* `Rc`. *On peut utiliser 200 points par cercle par exemple.*

Pour ne pas trop charger le dessin, on propose de ne tracer le cercle empreinte de la fraise que toutes les 10 positions du point C. (tracé de 50 cercles au lieu de 500).

Question 5 *Tracer sur la même figure que précédemment les 50 cercles représentant la trajectoire prise par un point de la fraise. Conclure quant à l'obtention du profil désiré.*

2 Repérage des alignements dans un jeu de puissance 4

Le but du jeu de **puissance 4** est d'aligner une suite de 4 pions de même couleur sur une grille comptant 6 lignes et 7 colonnes. Chaque joueur dispose de 21 pions d'une couleur (par convention, en général jaune ou rouge). Tour à tour, les deux joueurs placent un pion dans la colonne de leur choix, le pion coulisse alors jusqu'à la position la plus basse possible dans ladite colonne à la suite de quoi c'est à l'adversaire de jouer. Le vainqueur est le joueur qui réalise le premier un alignement (horizontal, vertical ou diagonal) consécutif d'au moins quatre pions de sa couleur.*[Wikipedia]*

On propose ici d'écrire les fonctions permettant de tester si un alignement (horizontal, vertical ou diagonal) est obtenu par l'un des joueurs.

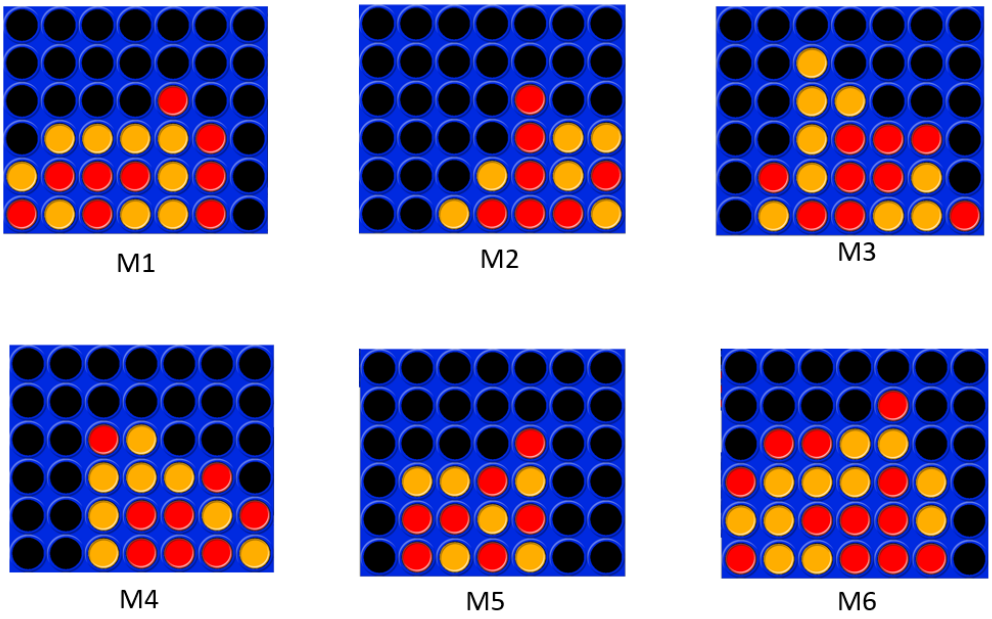

FIGURE 3 – Présentation des grilles de tests

Grilles de tests

Dans votre script sont définies 6 listes de listes (variables `M1` à `M6`) associées aux 6 grilles de la figure 3.

Question 6 *Indiquer sous forme de commentaire dans le script, quel entier est utilisé pour coder la couleur rouge ? la couleur jaune ?. On définit la variable* `p=M[i][j]`. *La variable* `i` *correspond-il à la ligne ou à la colonne sur la grille ? Où se situe le pion pour* `i=0` *? reprendre les 2 questions précédentes pour la variable* `j`.

Fonctions de tests des alignements

La fonction `quatre_a_la_suite_horizontal` est donnée. Cette fonction permet de tester si la grille contient un alignement horizontal de 4 poins d'une couleur `c`.

Le principe de cette fonction est illustré sur la figure 4 : pour chaque ligne de la grille on teste les 4 alignements possibles.

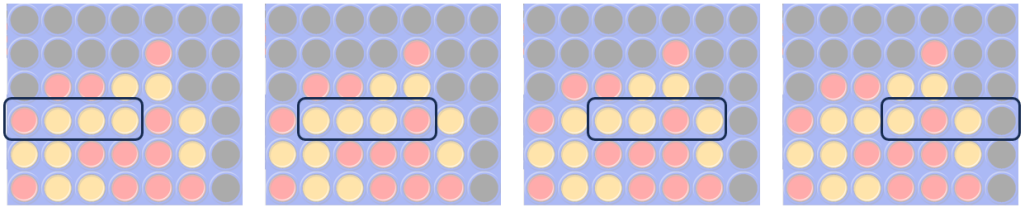

FIGURE 4 – Illustration du principe de détection d'un alignement horizontal

Question 7 *Tester cette fonction à l'aide des grilles de tests fournies. Les résultats de vos tests seront donnés sous forme de commentaires dans le script.*

Question 8 *Écrire une fonction* `quatre_a_la_suite_vertical(grille : [[int]], c: int)-> bool` *permettant de détecter un alignement vertical. Tester cette fonction à l'aide des grilles de tests fournies.*

Question 9 *Écrire une fonction* `quatre_a_la_suite_oblique(grille : [[int]], c: int)-> bool` *permettant de détecter un alignement diagonal. Tester cette fonction à l'aide des grilles de tests fournies.*