TP : Numérique

I) Tracés de courbes

Exercice 1 (Épicycloïde)
Une épicycloïde est une courbe plane trajectoire d'un point fixé à un cercle (de rayon r) qui roule sans glisser à l'extérieur un autre cercle (de rayon R) dit directeur. Avec $a = \dfrac{R+r}{r}$, en divisant par r, l'équation de la courbe est

$$\begin{cases} x(t) = a\cos(t) - \cos(at) \\ y(t) = a\sin(t) - \sin(at) \end{cases}$$

Pour $a = 2$ on obtient une cardioïde, pour $a = 3$ une néphroïde, pour $a = 4$, 3 pétales, etc.
Dans tout l'exercice, on s'intéresse au cas $a = 2$. On pourra tester les tracés pour les autres valeurs de a en fin d'exercice.
Toutes les fonctions doivent être testée au fur et à mesure, évidemment.

1) Définir une fonction f qui, à un flottant t associe le couple $(x(t), y(t))$.

2) Définir, à l'aide d'une fonction NumPy ad hoc, un tableau `t` de 100 valeurs régulièrement espacées entre 0 et 2π.

3) Appliquer f à ce tableau, en prédisant ce que va faire Python.
Ici, Python le fait spontanément car toutes les fonctions sont de la bibliothèque NumPy. Sinon, il faut écrire `fv = np.vectorize(f)` puis utiliser `fv`.

4) Comment récupérer $x = (x(t_0), \dots, x(t_N))$? Après avoir récupéré un tableau de valeurs de x et de y, tracer la courbe.
Pour un repère orthonormé : `plt.axis('equal')`.

5) Rappeler la définition de la normale (on pourra se contenter de donner un point, un vecteur).

6) Python ne sait tracer que des segments (il approche la courbe par une ligne brisée). Donc pour tracer une droite, il suffit de donner deux points. Tracer une normale en un point au choix.
Pour la dérivée, vous pouvez soit l'approcher numériquement, soit plus simplement la calculer formellement.

7) Tracer une famille de normale, pour $t \in [0, 2\pi]$. Que remarque-t-on ? Rappeler la définition de la courbe qui apparaît.

Exercice 2 (Hypocycloïde)
Une hypocycloïde est une courbe plane trajectoire d'un point fixé à un cercle (de rayon r) qui roule sans glisser à l'intérieur d'un autre cercle (de rayon R) dit directeur. Avec $a = \dfrac{R-r}{r}$, en divisant par r, l'équation de la courbe est

$$\begin{cases} x = a\cos(t) + \cos(at) \\ y = a\sin(t) - \sin(at) \end{cases}$$

Pour $a = 2$ on obtient une deltoïde, pour $a = 3$ une astroïde, etc..
Adapter les fonctions de l'exercice précédent pour obtenir quelques tracés de courbes, ainsi que les développées comme enveloppe des normales.

Exercice 3 (Exercices 0, banque PT)

1) Deux paramètre b et w valant respectivement 0.5 et 6.0, définir trois fonctions d'une variable t renvoyant les couples :

$$\begin{cases} p : t \mapsto (\ \cos(t) + b\cos(wt) & , \sin(t) + b\sin(wt)\ ) \\ v : t \mapsto (\ -\sin(t) - bw\sin(wt) & , \cos(t) + bw\cos(wt)\ ) \\ a : t \mapsto (\ -\cos(t) - bw^2\cos(wt) & , -\sin(t) - bw^2\sin(wt)\ ) \end{cases}$$

Vérifier ces fonctions sur un exemple.
$p(t) = (x(t), y(t))$ désigne la position dans le plan d'une masse ponctuelle mobile au cours du temps, $v(t) = (x(t), y(t))$, sa vitesse, et $a(t) = (x(t), y(t))$, son accélération.

2) Construire la liste `L` des points $p(t)$, pour t variant de $-\pi$ à π avec un pas de discrétisation δt vérifiant $\delta t = 0.01\pi$.

3) Faire tracer dans le plan muni d'un repère orthonormal la ligne polygonale reliant les points $p(t)$ de la liste `L`.

4) Définir puis tester la fonction `centre_courbure` d'une variable t qui renvoie le couple des coordonnées du centre de courbure donné par :

$$C(t) = (x(t), y(t)) + d(-y'(t), x'(t)) \qquad \text{où} \qquad d = \frac{x'^2 + y'^2}{x'y'' - y'x''}$$

5) Rajouter sur le graphique précédent la ligne décrite par les centres de courbure, avec la même discrétisation en temps.

6) Calculer la longueur de la ligne polygonale reliant les points $p(t)$, pour différents pas de discrétisation δt. Observer l'évolution de cette longueur lorsque δt diminue.

II) Matrices et valeurs propres

Le but de cette partie est de construire des matrices symétriques à partir de la formule $A = PDP^{-1}$, où $P \in \mathscr{O}_3(\mathbb{R})$, puis de retrouver les valeurs propres de A par une méthode numérique.

Exercice 4 (Tirer une base orthonormée au hasard)

1) Définir une fonction `pscal(x,y)` qui prend deux vecteurs de $\mathbb{R}^n$ codés par des tableaux NumPy de flottants et qui retourne un flottant représentant le produit scalaire (canonique) de x et y.

2) Finir ou reprendre votre fonction d'orthonormalisation de Gramm-Schmidt. La tester pour ce produit scalaire (on pourra reprendre l'exercice 7 de la feuille sur les euclidiens).

3) Écrire une fonction qui prends une base orthonormée (donnée comme famille de vecteurs de coordonnées dans la base canonique) et retourne la matrice de passage P de la base canonique vers cette base orthonormée.

4) Écrire une fonction qui retourne une base orthonormée aléatoire. On pourra commencer par choisir une famille de vecteurs au hasard (`np.random.rand`, lire la documentation et faire des tests). Pourquoi cette famille est libre, en première approximation ?

On ne s'inquiète pas, a priori, de la loi de probabilité mise sur $\mathscr{O}_3(\mathbb{R})$

Exercice 5 (Construire une matrice symétrique au hasard)

1) Construire une fonction qui tire une matrice diagonale au hasard, avec des valeurs entre $-N$ et N. On pourra prendre $N = 10$.

2) En déduire une fonction qui tire au hasard une matrice symétrique A. Pour faire des opérations d'algèbre linéaire (transposition, produit matriciel, ...) regardez le mémento Python.

3) À l'aide des fonctions de la bibliothèque NumPy, retrouver les valeurs propres de A.

Exercice 6

On se donne une matrice symétrique A. On va essayer de retrouver les valeurs propres de A sans passer par le polynôme caractéristique (mauvaise complexité).

1) Pour $X_0 \in \mathbb{R}^n$, on construit la suite (X_n) :

$$X_{n+1} = \frac{AX_n}{\|AX_n\|}$$

Numériquement, « montrer » que cette suite converge vers un vecteur propre pour une certaine valeur propre que l'on déterminera.

2) On se donne une famille de k vecteurs, orthonormée, et l'on applique le procédé précédent (en orthonormalisant la famille à chaque étape). Que trouve-t-on ?