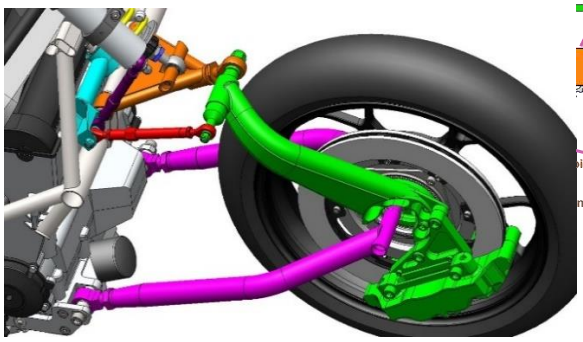
Informatique

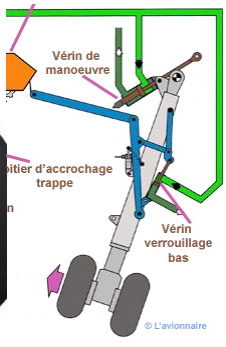
Intelligence artificielle

Réseau de neurones
Interpolation appliquée au système 4 barres

Contexte

En Sciences Industrielles de l'Ingénieur, l'analyse cinématique des mécanismes est une étape incontournable pour prédire le comportement d'un système. Le mécanisme à quatre barres, omniprésent dans l'industrie (suspensions automobiles, bras de tracteur, ponts levants, suspensions de vélos...), en est un exemple typique.





Simple en apparence, il conduit pourtant à une équation caractéristique dont la résolution analytique est particulièrement lourde.

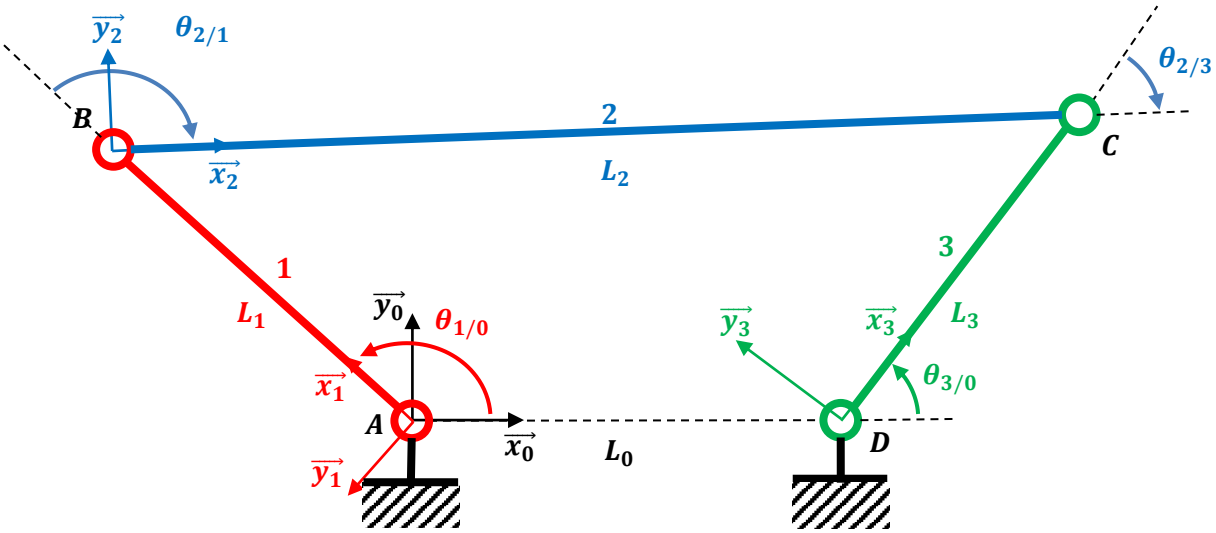
Ce sujet propose une approche moderne et complémentaire : plutôt que de chercher une solution exacte, on commence par résoudre numériquement cette équation par dichotomie, afin d'obtenir un ensemble de points représentatifs de la loi entrée-sortie du mécanisme. Ces données numériques servent ensuite de base à l'entraînement d'un réseau de neurones artificiel, qui apprend à interpoler cette relation et peut alors prédire la position de sortie pour n'importe quelle position d'entrée.

L'activité se déroule en trois temps :

- Modélisation et résolution dichotomique : comprendre et exploiter l'équation de fermeture géométrique.
- Construction et entraînement du réseau de neurones : utiliser scikit-learn, une bibliothèque de machine learning, pour apprendre la loi entrée-sortie.
- Validation et analyse : comparer les deux approches et évaluer la qualité de l'interpolation en fonction du nombre de points d'entraînement.

Modèle cinématique

On propose le modèle cinématique général suivant :



Données

Nous étudierons un système dont les dimensions sont les suivantes :

$$L_0 = 100\,mm \quad ; \quad L_1 = 100\,mm \quad ; \quad L_2 = 150\,mm \quad ; \quad L_3 = 100\,mm$$

On note :

$$x = \theta_{10} \quad ; \quad y = \theta_{30}$$

Mise en équation

Par une fermeture géométrique, on obtient l'équation suivante liant x et y :

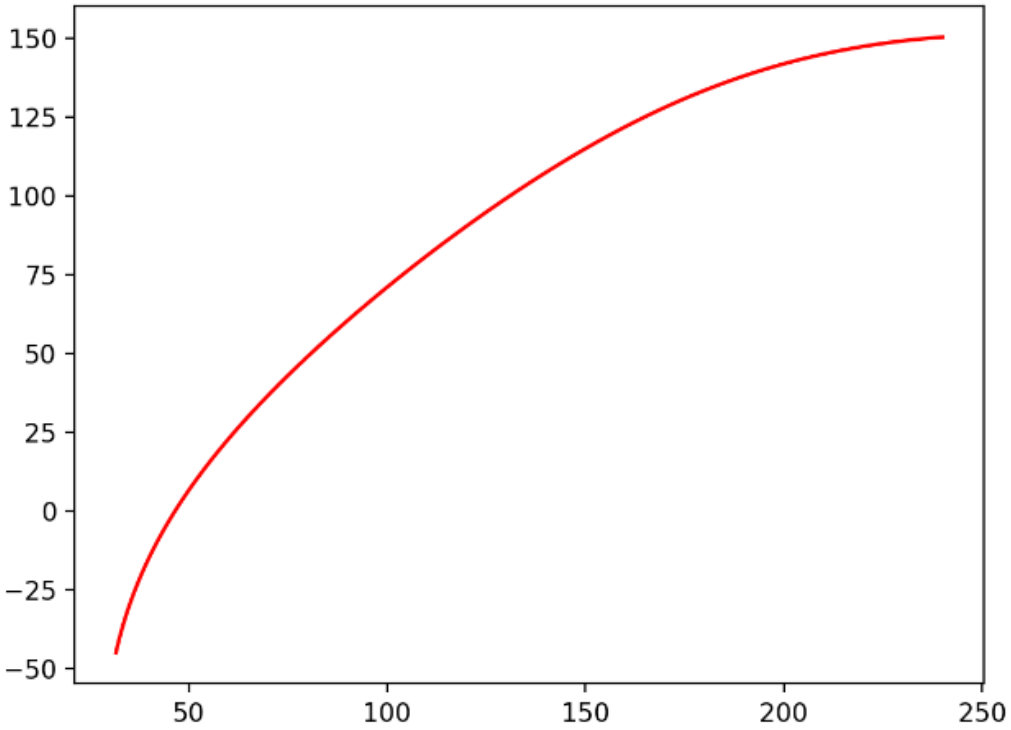
$$f(x,y) = -a\cos x + b\cos y - c\cos(y-x) + d = 0$$

$$\begin{cases} a = 2L_0L_1 \\ b = 2L_0L_3 \\ c = 2L_1L_3 \\ d = {L_0}^2 + {L_1}^2 - {L_2}^2 + {L_3}^2 \end{cases}$$

Résolution

Il est possible de résoudre cette équation par le calcul, vous pouvez jeter un œil à la démonstration jointe à ce sujet, c'est TRES LOURD ! Nous réalisons donc ici une résolution approchée par dichotomie (sujet de résolution dédié en ligne sur Eduscol). Pour toute valeur de x dans un intervalle choisi, on obtient y par résolution dichotomique de l'équation $f(x,y) = 0$. Il est alors possible d'obtenir deux listes, Lx des valeurs de x et Ly des valeurs de y associées.

Un code élève vous est fourni, il réalise cette résolution et trace la fonction $g: x \to g(x) = y$, avec des angles en degrés sur 100 points.



Code élèves

Vous avez à disposition un code élèves créant les listes Lx et Ly et affichant la courbe.
Exécutez le code fourni et visualiser la courbe obtenue.

Mise en place du réseau de neurones

Voyons comment utiliser le module scikit-learn pour construire un réseau de neurones permettant d'obtenir la fonction :

$$g: x \to g(x) = y$$

Si besoin, pour installer le module, tenter dans la console « `pip install scikit-learn` » ou « `python -m pip install scikit-learn` ».
Nous allons utiliser un réseau de neurones de type « Regressor » permettant de renvoyer une valeur continue.
Quelques rappels :

Import module et réseau de neurones multicouches	`from sklearn.neural_network import MLPRegressor`
Création du modèle	`MLPRegressor(max_iter=20000)`
Entrainement du modèle	`rn.fit(Lxt,Ly)`
Prédiction `x` : Flottant	`rn.predict([[x]])[0]`
Sauvegarde de modèle	`from joblib import dump` `dump(rn,'rn.joblib')`
Chargement de modèle	`from joblib import load` `rn = load('rn.joblib')`

Avec Lxt une liste de listes contenant chaque valeur de x
Vous allez :

- Créer le réseau de neurones
- L'entraîner sur les données
- Créer la fonction g
- Tracer la fonction g et la comparer à la résolution dichotomique
- Trouver le nombre de points nécessaire pour une bonne approximation

Question 1: Créer le code attendu

Vous savez maintenant interpoler n'importe quelle fonction représentée par des listes de ses données.