Zhichen Lu:21117174
Jiawei Luo:21109757
Amina Ferrad:21101473

Présentation du code:

Nous choisissons la méthode de classe pour compléter le contenu du code de l'ensemble du projet. Parce que nous pensons qu'un dessin doit être complété pour chaque méthode , s'il est réalisé séparément , cela augmentera la quantité de code, et il y aura des problèmes lorsqu'il devra être modifié de manière uniforme.

De plus, nous avons essayé d'écrire chaque méthode dans une classe, mais après avoir teriné, nous avons constaté que pour chaque méthode, nous devions appeler la classe à laquelle elle appartient, et finalement nous avons décidé de mettre toutes les méthodes dans une seule classe. De cette façon, un appel peut être effectué directement, ce qui sera plus simple et plus pratique pour les opérations externes.

Vient ensuite une introduction au contenu du code :

1. Méthode Générale de plot.

- __init__()
  a. Fournit des angles de joint initiaux :self.angles_ini = [0, 0] ;
  b. Chaque longueur de bras pour ce 2ddl robot :self.bras_longueur = [0.5, 0.5] ;
  c. Une matrice est créée qui itère sur les angles d'articulation :self.angles = [0, 0] ;
  d. Une matrice est créée qui contient les coordonnées des points communs : self.joint1 = self.joint2 = [0, 0].

- direct()
  a. L'angle de joint 0 : q0 = self.angle[0] ;
  b. L'angle de joint 1 : q1 = self.angle[1] ;
  c. La longueur de bras 0 : L0 = self.bras_longueur[0] ;
  d. La longueur de bras 1 : L1 = self.bras_longueur[1] ;
  e. Les coordonnées des point communs :
  self.joint1 = [L0*cos(q0), L0*sin(q0)] ;
  self.joint2 = [L0*cos(q0) + L1*cos(q0+q1), L0*sin(q0) + L1*sin(q0+q1)].

- plot()
  a. Plot une figure pour chaque instant

- solutionexist()
  a. La fonction de J : def residu(X, goal) ;
  b. Juger s'il exsite une solution : jugez de la valeur de l'équation J , car la signification géométrique de l'équation J est la distance entre le point cible et le point que le mouvement réel atteint finalement.

2. 4 méthodes.

a. Première méthode: <<root>>

- m_root(goal)
    a. Echantillon N points pour tracer la trajectoire ;
    b. Calculs des angles pour chaque point sur la trajectoire ;
    c. Créer une animation pour la trajectoire.
- methode1_root(goal)
    a. Utilisation de <<root>>.

b. Deuxième méthode: <<minimize>>

- m_minimize(goal, X0)
    b. Echantillon N points pour tracer la trajectoire ;
    c. Calculs des angles pour chaque point sur la trajectoire ;
    d. Créer une animation pour la trajectoire.
- methode2_minimize(goal, X0)
    a. Utilisation de <<minimize>> par rapport à X0.

c. Troisième méthode: Gradient à pas fixe

- m_gradient(goal, X0, nmax, alpha)
    a. Echantillon N points pour tracer la trajectoire ;
    b. Calculs des angles pour chaque point sur la trajectoire ;
    c. Créer une animation pour la trajectoire ;
    d. Tracer les figures des isovaleurs.
- methode3_gradient(goal, X0, nmax, alpha)
    a. Utilisation de la méthode de gradient à pas fixe.

d. Quatrième méthode: Newton

- m_newton(goal, X0)
    a. Echantillon N points pour tracer la trajectoire ;
    b. Calculs des angles pour chaque point sur la trajectoire ;
    c. Créer une animation pour la trajectoire ;
    d. Tracer les figures des isovaleurs.
- methode4_newton(goal, X0)
    a. Utilisation de la méthode de newton.

3. Execution de code

a. Terminez l'importation du package Python;
b. Courir la classe ProjetOpt ;
c. Donner une exemple : result = ProjetOpt() ;
d. Fonctionner la méthode de classe solutionexist ;
e. Pour chaque méthode, modifiez les paramètres d'entrée requis en dehors de la classe. Si vous souhaitez modifier les point cibles, veuillez ajouter ces points à la matrice de points cibles déjà définie au lieu de les créer en externe.