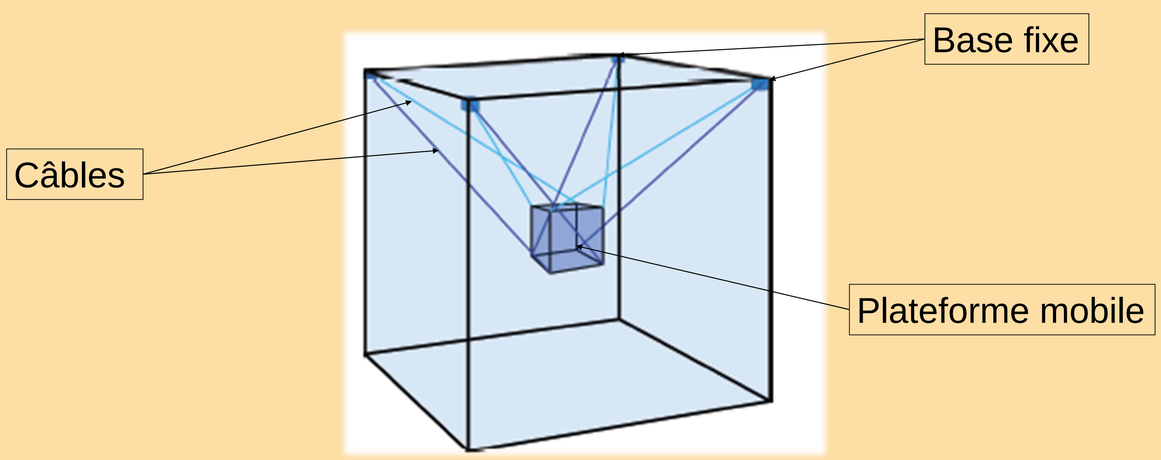
Robot manipulateur à câbles

Projet universitaire

Master 1 Automatique, Robotique Parcours Systèmes Avancés et Robotiques

Introduction

Les chaines cinématique très particulières des robots parallèles en font une classe à part de la robotique parallèle traditionnelle, en effet, les robot manipulateurs parallèle présentés dans cette fiche ont les éléments mécanique qui reliant la plateforme mobile à la base fixe ne sont pas rigides, mais composés de cables flexibles enroulés pour transmettre un movement et une force.

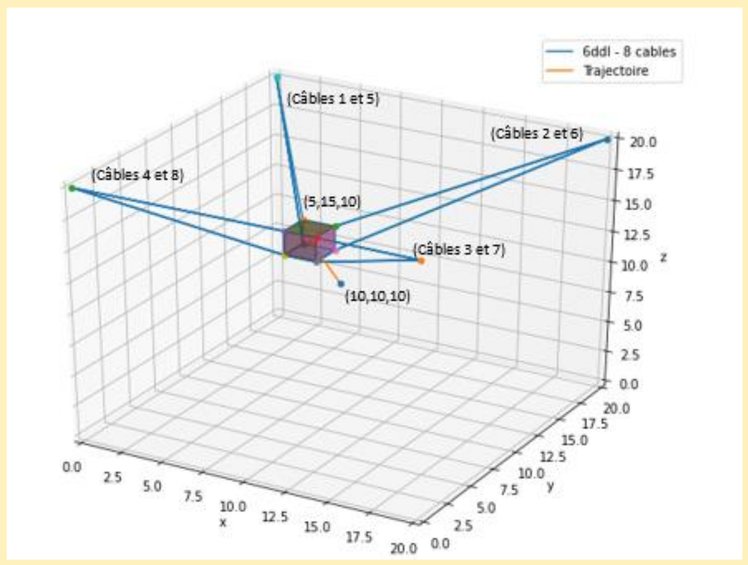


Objectifs

- Proposition d'une modélisation géométrique, cinématique, statique et dynamique.
- Proposition d'une simulation d'un robot manipulateur à cables (8 cables, 6 DDL).
- Visualisation de l'évolution de movement de ces robots en fonction de ses paramètres articulaires et opérationnels.
- Exposer l'évolution de ses variables sous forme des courbes en fonction du temps.
- Controle de la vitesse de la plateforme mobile.
- Déduire la distribution des tensions dans les cables et proposition d'un algorithme d'optimisation.

Démarches

1) - Simulation sous Python pour visualiser un robot maniplulateur suspendu à 8 câbles.



2) - Définition du modèle géométrique direct (MGD)**:** $\mathbf{x = f^{-1}(}\boldsymbol{l}\mathbf{)}$

- Trouver la position et l'orientation de la plateforme mobile connaissant la longueur des câbles.

3) - Définition du modèle géométrique indirect (MGI) : $\boldsymbol{l}$ **= f(x)**

- $l_i = \|\overrightarrow{A_iB_i}\| = \|(\mathrm{p} + \mathrm{Qb}_i - \mathrm{a}_i)\|,\ \mathrm{i} \in [1, 2, \ldots, \mathrm{m}]$

4) - Contrôle du vitesse de la plateforme mobile (MCI) :

- Paramétrage : x(t), y(t), z(t)
- Dérivation :
- $\dot{\boldsymbol{l}}_i = \mathbf{J}^{-1}.\ \dot{\boldsymbol{x}}$

```
#création de vecteur paramètres opérationnelles:
x_dot = np.array([[x_point,y_point,z_point,phi_point,théta_point,psi_point]])
#trouver le vecteur des paramètre articulaire à partir du produit entre vecteur
#paramètre opérationnels et la matrice Jacobienne inverse
l_dot = np.dot(J-1,x_dot.T)
```

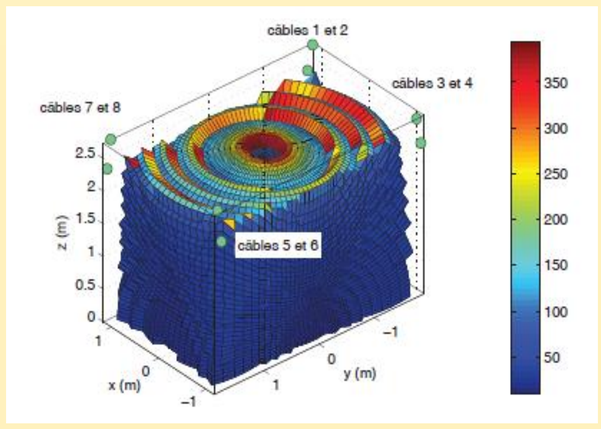
5) - Algorithme d'optimisation et minimisation de vecteur de tensions.

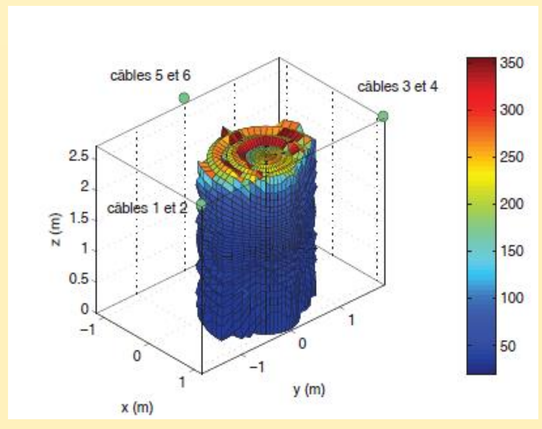
- Solution optimale : (Scipy.optimize)

```
     x: array([62.82896269,  5.86512617,  0.3203536 ,  5.86512617,  6.05985933,
       18.76105781, 24.30756467, 18.76105781])
   fun: 85.43366780969596
   jac: array([0.90803719, 0.03020573, 0.04009151, 0.03020573, 0.01656628,
       0.10322475, 0.38787842, 0.10322475])
message: 'Optimization terminated successfully.'
  nfev: 228
   nit: 15
  njev: 15
status: 0
success: True
```

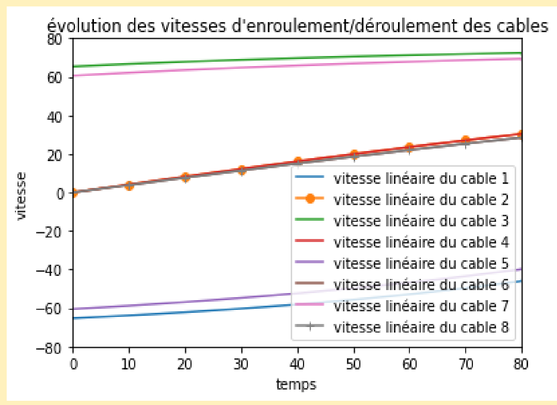
Résultats

- Espace de travail (robot à 6 cables) **<** Espace de travail (robot à 8 cables) (Singularités !!)

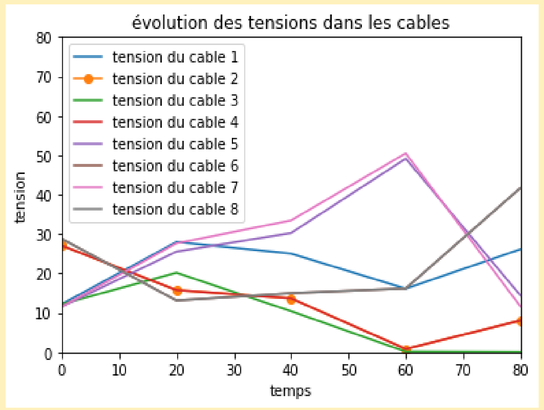




- Evolution des vitesses articulaires (enroulement / déroulement des câbles)



- Distribution des tensions dans les câbles selon des contraintes.



Conclusion

- Plus grand nombre de câbles :
  - augmenter le volume de l'espace de travail.
  - mieux répartir les tensions dans les câbles.
- Réunion des câbles aux points d'attache permet d'augmenter l'espace de travail et éviter qu'ils se croisent.
- Des problèmes mathématique se cachaient derrière cette simplicité.
- N'existe pas encore une fonction explicite pour la détermination des tensions optimales dans les câbles.