# Rapport du Projet de Modélisation

## Concéption et Simulation d'un Robot 3RRR

2024/2025

par

Sayf Chafik

SCIENCES
SORBONNE
UNIVERSITÉ

# Table des matières

# 1

# Objectif du projet

L'objectif principal de ce projet est de concevoir, modéliser et simuler un robot manipulateur parallèle plan de type 3RRR, capable de se déplacer avec précision dans un plan tout en contrôlant son orientation. Ce type de robot est constitué de trois bras identiques, chacun composé de deux segments reliés par des liaisons rotoïdes (R), d'où l'appellation 3RRR. L'architecture offre trois degrés de liberté : deux translations dans le plan $(x, y)$ et une rotation autour de l'axe perpendiculaire à ce plan $(\theta)$.

Ce travail s'inscrit dans une démarche complète, allant de la modélisation mécanique à la simulation interactive, avec pour ambition de proposer une solution à la fois fonctionnelle, imprimable et intelligemment optimisée.

## Objectifs détaillés

- **Conception mécanique adaptée à la fabrication** : Le robot a été modélisé sous SolidWorks en prenant soin d'optimiser sa géométrie pour l'impression 3D.
- **Intégration d'actionneurs réels** : Le robot a été conçu pour accueillir des servomoteurs Dynamixel AX-12.
- **Maximisation de l'espace de travail** : La géométrie du robot a été pensée pour obtenir une surface de travail étendue et fonctionnelle, tout en évitant les zones de singularité et en garantissant une bonne précision de positionnement.
- **Ajout d'une fonctionnalité de traçage** : L'effecteur mobile a été pensé pour accueillir un crayon, permettant au robot de
- **Simulation numérique interactive** : Une interface a été développée en Python afin de simuler le comportement du robot. Cette simulation permet :
  - la visualisation dynamique de la configuration du robot en temps réel,
  - le suivi automatique de trajectoires (cercle, carré) déclenchées par l'utilisateur,
  - le tracé de la trajectoire de l'effecteur,
  - le calcul de l'orientation optimale de l'effecteur pour éviter les singularités,
  - une détection qualitative des configurations proches de collision.

# 2

# Modélisation - Conception mécanique sous SolidWorks

Une modélisation 3D du robot a été réalisée sous **SolidWorks**, en tenant compte des contraintes suivantes :

- **Maximiser l'espace de travail** de l'effecteur pour exploiter au mieux la surface disponible.
- **Réduire les collisions** potentielles entre les bras grâce à une disposition angulaire symétrique.
- **Limiter l'encombrement** global pour permettre une fabrication en impression 3D.
- **Adapter la conception aux actionneurs Dynamixel AX12**, avec un espace dédié pour leur intégration.
- **Prévoir un support pour crayon** afin de dessiner les trajectoires.

# 3

# Simulation

La simulation a été réalisée en Python à l'aide de la bibliothèque `pygame` pour l'interface graphique. Elle s'appuie sur les fonctions MATLAB fournies (modèle géométrique inverse, trajectoire cible), réimplémentées en Python.

## 3.1 Structure générale

La simulation repose sur une boucle temps réel qui :
— Lit les commandes utilisateur (clavier).
— Calcule les angles articulaires $\alpha_i, \beta_i$ à partir de la position $(x_E, y_E, \theta_E)$.
— Vérifie la validité géométrique et cinématique.
— Détecte les singularités et collisions.
— Corrige l'orientation si nécessaire.
— Affiche graphiquement le robot et sa trajectoire.

### 3.1.1 Fonctionnalités principales

Voici les principales fonctions du code Python :
— `inverse_kinematics(xE, yE, θE)` :
Cette fonction calcule les angles articulaires $(\alpha_i, \beta_i)$ pour chaque bras à partir de la position $(x_E, y_E)$ et de l'orientation $\theta_E$ de l'effecteur. Le calcul repose sur un modèle géométrique inverse analytique. Elle construit la transformation homogène de l'effecteur puis résout chaque triangle fermé entre la base, les bras et les points d'attache à l'effecteur.
— `is_near_singularity(q)` :
Elle évalue si la configuration courante est proche d'une singularité. Deux tests sont effectués :
— $\det(A) \approx 0$ : singularité parallèle (bras alignés ou concourants).
— $\det(B) \approx 0$ : singularité série (bras déployé à sa limite).
Les matrices $A$ et $B$ sont construites à partir des directions $\gamma_i$ des bras et des leviers $d_i$ et $e_i$ selon le modèle cinématique. Si une singularité est détectée, un message d'avertissement est affiché.
— `detect_collisions(q)` :
Cette fonction vérifie la présence d'interchapters entre les segments des bras (liaisons mécaniques). Chaque bras est modélisé par deux segments consécutifs (base à coude, coude à effecteur). Pour chaque paire de bras distincts, elle teste si deux segments se croisent géométriquement à l'aide d'un test de convexité (fonction `ccw`).

En cas de collision, un message est affiché.

— `verify_geom(q, xE, yE, θE)` :
Cette fonction vérifie la validité géométrique de la configuration calculée. Elle compare la position réelle d'un point $B_i$ (calculée via les angles $\alpha_i, \beta_i$) à celle de $E_i$ (fixe sur l'effecteur). Une configuration est considérée incorrecte si la distance entre ces points dépasse un seuil tolérable. C'est un test de cohérence géométrique très utile après l'inverse.

— `optimize_orientation(prev_q, [xE, yE, θE])` :
Cette fonction est appelée lorsque la configuration souhaitée est invalide (singularité ou collision). Elle explore un ensemble de valeurs de $\theta_E$ autour de la valeur initiale, avec un pas fixe. Pour chaque orientation candidate, elle appelle l'inverse, puis teste :
  — l'absence de singularité,
  — l'absence de collision,
  — la validité géométrique,
  — et la proximité articulaire avec la configuration précédente (continuité).

  L'orientation qui minimise le "saut articulaire" tout en restant sûre est conservée.

— `trace_rob(q, θE)` :
C'est la fonction de rendu graphique. Elle trace :
  — les trois bras du robot (en deux segments chacun),
  — l'effecteur sous forme de triangle entre les trois points $P_{ij}$,
  — la trajectoire déjà parcourue,
  — et un panneau d'instructions clavier.

  L'effecteur est coloré en vert (sûr) ou jaune (danger), selon les tests de singularité/collision.

### 3.1.2 Modes de fonctionnement

Le simulateur propose :

— Un **mode manuel** (flèches directionnelles + Q/E pour l'orientation).

— Un **mode automatique** :
  — Appui sur A → trajectoire circulaire.
  — Appui sur B → trajectoire carrée.

Chaque mode est accompagné d'une visualisation temps réel de la trajectoire en bleu et de l'orientation de l'effecteur via une flèche noire.

## 3.2 Gestion des singularités et collisions

### 3.2.1 Détection des singularités

Deux types de singularités sont prises en compte :

— **Singularité parallèle** : lorsque les trois directions $A_iB_i$ deviennent parallèles ou concourantes.

— **Singularité série** : lorsqu'un bras est étendu à sa limite (détection via $\det(B) \approx 0$).

Ces configurations sont détectées dans `is_near_singularity` par les déterminants des matrices Jacobiennes $A$ et $B$.

### 3.2.2 Évitement automatique par réorientation

Lorsque la configuration actuelle est dangereuse, la fonction `optimize_orientation` explore différentes valeurs de $\theta_E$ autour de la valeur cible. Elle sélectionne celle qui minimise la discontinuité articulaire tout en évitant la singularité.

Cette stratégie fonctionne de façon fiable pour éviter les blocages et permet la poursuite du mouvement fluide.

### 3.2.3 Gestion des collisions

Une détection de collisions basique est implémentée par test d'interchapters de segments (chaque bras modélisé comme deux segments). Elle fonctionne dans la majorité des cas, mais peut échouer dans des cas limites où les segments se croisent sans partage de point. Une amélioration serait d'ajouter des zones de sécurité (cercle autour des joints).

## 3.3 Affichage graphique et interface utilisateur

L'affichage comprend :

— Les bras du robot en rouge et rose.
— L'effecteur (triangle) en vert ou jaune (quand singularité) selon la sécurité.
— La trajectoire de l'effecteur en bleu.
— Un panneau d'instructions clavier (Q/E pour rotation, A/B pour automatisme).

L'interface est simple, claire et réactive, permettant de suivre précisément les mouvements du robot.

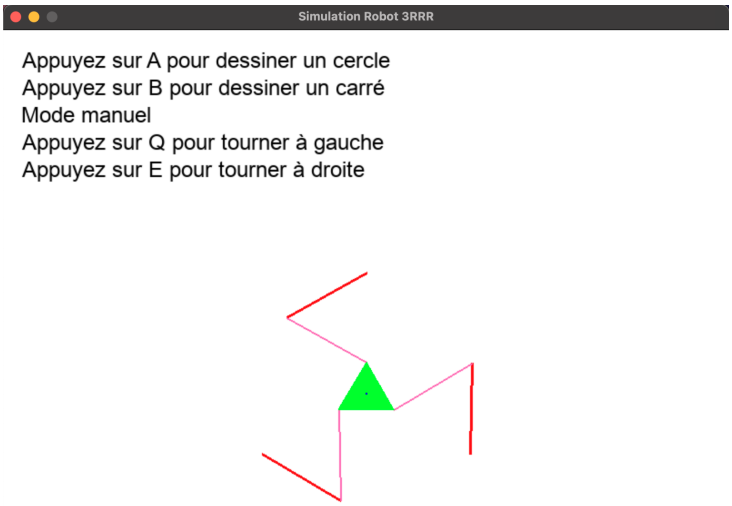


FIGURE 3.1 – Interface graphique en mode manuel. Les instructions clavier sont affichées à l'écran pour guider l'utilisateur.

## 3.4 Résultats et validation

Les tests menés sur les deux trajectoires (cercle et carré) montrent que :

— Les mouvements sont fluides et cohérents avec la cinématique du robot.
— Les singularités sont systématiquement évitées.
— Les collisions sont détectées dans la majorité des cas, mais nécessitent encore du perfectionnement.
— L'orientation de l'effecteur est bien ajustée automatiquement.

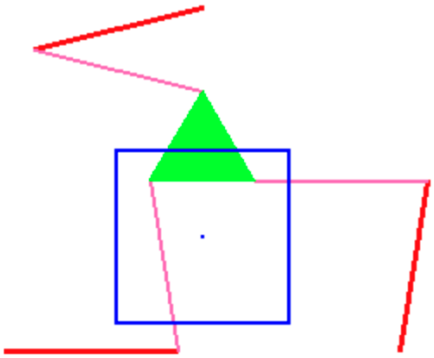

FIGURE 3.2 – Trajectoire carrée suivie par l'effecteur — la configuration est stable, l'effecteur suit bien les sommets.

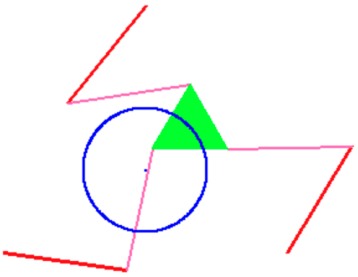

FIGURE 3.3 – Trajectoire circulaire réalisée — l'orientation de l'effecteur est automatiquement adaptée pour éviter les singularités.

.

## 3.5 Limites et perspectives

### 3.5.1 Limites actuelles

— La gestion des collisions reste partielle.
— L'espace de travail réel n'est pas affiché.
— Le repère local de l'effecteur pourrait être enrichi (axes $x/y$ en plus de la flèche).

### 3.5.2 Perspectives

— Ajout d'une visualisation des zones de singularité.
— Export de la trajectoire sous forme CSV.
— Interface graphique interactive plus avancée (Tkinter, sliders, etc.).
— Mode de replay ou enregistrement de la session.

# Conclusion

Ce projet a permis de modéliser et simuler un robot parallèle 3RRR de manière complète, en prenant en compte les aspects mécaniques, géométriques et algorithmiques.

La simulation s'est révélée robuste et intuitive à utiliser, notamment grâce aux différentes fonctions implémentées pour le calcul inverse, la détection des configurations dangereuses, et l'ajustement automatique de l'orientation de l'effecteur. Le système est réactif, fluide, et permet d'explorer les comportements caractéristiques d'un robot parallèle dans un cadre pédagogique.

Cependant, plusieurs axes d'amélioration peuvent être envisagés pour enrichir le projet :

- **Améliorer la détection et la gestion des collisions** : Bien que des tests géométriques de base soient en place, certaines collisions complexes peuvent encore échapper au système. L'intégration de zones de sécurité (par exemple, des cercles autour des articulations ou volumes d'exclusion) permettrait de renforcer cette partie.
- **Afficher dynamiquement l'espace de travail** : Une visualisation en temps réel des zones atteignables par l'effecteur serait précieuse, notamment pour anticiper les limites de mouvement et optimiser l'orientation.
- **Enrichir l'interface graphique** : Un affichage plus ergonomique (curseurs de réglage, boutons de commande, affichage des angles articulaires) pourrait rendre le simulateur plus accessible.
- **Coupler la simulation à la partie réelle** : Une perspective à moyen terme serait de connecter le simulateur à une version physique du robot via communication série, pour passer du virtuel au réel.

En somme, ce projet constitue une base solide pour l'étude des robots parallèles, alliant modélisation , contraintes et implémentation algorithmique efficace. Il m'a permis de développer une compréhension fine de la cinématique des robots parallèles, ainsi qu'une expérience concrète de leur simulation et visualisation. Le travail accompli ouvre des perspectives d'évolution intéressantes.