Modélisation et Simulation Robotique Projet de Conception, Modélisation et Commande d'un robot parallèle 3RRR

Ismail GHOMARI
Amir AHADDAD
Abithan THEVARAJAH
Sorbonne University
M1 SAR

I. INTRODUCTION

Ce projet porte sur la conception, la modélisation et la commande d'un manipulateur parallèle plan de type 3-RRR à trois degrés de liberté. Chacune de ses trois chaînes cinématiques identiques relie la base à l'effecteur à l'aide de trois liaisons rotoïdes et deux corps rigides intermédiaires. Les manipulateurs parallèles, tels que le 3-RRR, sont réputés pour leur rigidité structurelle et leur précision, mais présentent également des défis spécifiques, notamment la présence de deux types de singularités : les singularités parallèles et les singularités série.

L'objectif de ce travail est de concevoir un robot capable d'évoluer dans un espace de travail convexe, exempt de singularités, tout en maximisant la surface atteignable et en minimisant les risques de collision entre les composants mécaniques. Le projet prend aussi en compte les contraintes pratiques liées à l'impression 3D, la puissance limitée des actionneurs disponibles (servomoteurs Dynamixel AX12), ainsi que les jeux et frottements mécaniques.

Pour répondre à ces exigences, nous avons mis en œuvre une approche basée sur l'optimisation des longueurs de bras et des dimensions du triangle de l'effecteur, afin d'agrandir la zone de travail exploitable. Nous avons ensuite déterminé la plage angulaire maximale pour chaque moteur avant l'apparition de singularités, puis validé ces résultats en simulation avant de passer à la conception mécanique et à la fabrication.

Le reste de ce rapport est organisé comme suit : la Section II présente la méthode d'optimisation utilisée, la Section III détaille la simulation et l'analyse des singularités, la Section IV traite de la modélisation CAO et de l'impression 3D, et la Section V discute des résultats obtenus et des limites rencontrées.

II. OPTIMISATION DE L'ESPACE DE TRAVAIL

A. Détermination de l'espace atteignable du robot 3-RRR

Cette étape vise à identifier la zone que peut atteindre l'effecteur en tenant compte uniquement des limites articulaires des trois bras, sans tenir compte des singularités.

A.1 Structure logicielle: La démarche repose sur plusieurs modules Python :

- `ikm.py` : calcule la cinématique inverse en déterminant les angles moteurs θ_1, θ_2, θ_3 à partir de la position (x, y) de l'effecteur et son orientation θ_e. Il résout analytiquement le problème via des expressions trigonométriques issues de la géométrie du robot.
- `ikm_phi_psi.py` : calcule les angles internes ϕ_i (entre les bras) et ψ_i (entre les

bras et la plateforme), à partir des angles moteurs et de la configuration géométrique du robot. Ces angles sont essentiels pour identifier les configurations proches de singularité.

— `get_coord.py` : calcule la position de l'effecteur à partir de la chaîne vectorielle formée par les segments OA_i, A_iB_i, B_iP en tenant compte de l'orientation relative des actuateurs.

A.2 Méthodologie: Pour déterminer l'espace de travail atteignable, la méthode suivante est utilisée :

1) On calcule l'espace atteignable indépendamment pour chaque bras du robot, en fonction des limites d'angle imposées à chaque moteur (ex. $\pm 300^\circ$ pour les Dynamixel AX-12).
2) Pour chaque configuration (x, y), la fonction `ikm` vérifie si une solution réelle existe pour les trois branches.
3) Si une configuration est accessible par les trois branches, elle est conservée. Sinon, elle est rejetée.
4) On détermine l'intersection des trois espaces atteignables pour obtenir un polygone représentant l'espace **réellement exploitable** du robot.

A.3 Résultat obtenu: L'exécution du code `plot_workspace.py` permet de visualiser l'espace de travail *atteignable*, représenté ci-dessous :

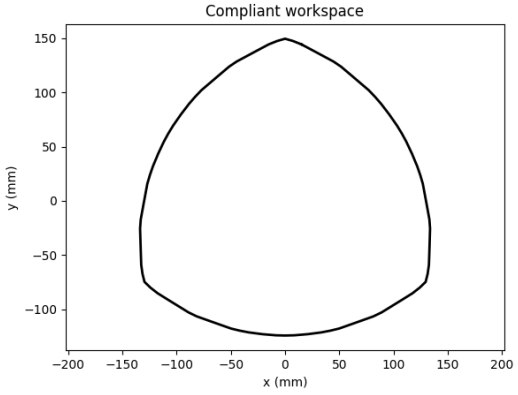

FIG. 1 – Espace de travail atteignable du robot 3-RRR

B. Optimisation des paramètres géométriques

L'objectif de cette section est de maximiser l'aire de l'espace de travail *compliant* tout en respectant plusieurs contraintes liées à la structure physique du robot.

B.1 Problème d'optimisation posé: Soit R_b le rayon de la base (zone sur laquelle sont fixés les moteurs), l_1 et l_2 les longueurs des bras, et R_e le rayon de la plateforme mobile.

Nous posons le problème suivant :

$$\begin{aligned} \underset{l_1, R_e}{\text{maximiser}} \quad & \mathcal{A}(l_1, R_e) \\ \text{s.c.} \quad & l_1 = l_2 \\ & l_1 \leq R_b \\ & R_e \leq 0.6 R_b \\ & R_e \leq 0.8 l_1 \\ & \mathcal{W}(l_1, R_e) \subseteq \mathcal{D}(R_b) \end{aligned} \tag{1}$$

où :

— $\mathcal{A}(l_1, R_e)$ est l'aire de l'espace *compliant* obtenu avec les paramètres donnés,

— $\mathcal{W}(l_1, R_e)$ est le polygone représentant l'espace atteignable,

— $\mathcal{D}(R_b)$ est le disque de rayon R_b centré à l'origine,

— les contraintes assurent que le workspace reste contenu dans la base et que la géométrie reste réalisable mécaniquement.

B.2 Résolution par évolution différentielle: Le problème ci-dessus est résolu numériquement à l'aide de l'algorithme d'optimisation *Differential Evolution*, disponible dans la bibliothèque `scipy.optimize`. L'espace de recherche est défini par les bornes suivantes :

— $l_1 \in [50, 180]$ mm

— $R_e \in [10, 100]$ mm

La fonction objectif retourne l'opposé de l'aire de l'espace *compliant* si toutes les contraintes sont respectées, et une valeur pénalisante très élevée (10^6) sinon.

L'exécution du programme principal retourne la configuration géométrique optimale ainsi que l'aire maximale atteinte.

B.3 Résultat de l'optimisation: Après 20 itérations de l'algorithme, le résultat obtenu est :

— $l_1 = l_2 = 98.19$ mm
— $R_e = 77.89$ mm
— Aire maximale : $55\,444$ **mm²**

Ce résultat respecte les contraintes et permet d'obtenir un espace de travail large, convexe et utilisable mécaniquement.

C. Détermination de l'angle moteur maximal avant singularité

C.1 Principe de détection des singularités: Pour garantir la stabilité et la contrôlabilité du robot, il est crucial d'éviter les configurations singulières. On distingue :

— les singularités de type I (parallèles), où $\det(J_\theta) = 0$ — perte de contrôlabilité interne.
— les singularités de type II (sérielles), où $\det(J_x) = 0$ — blocage de mouvement de l'effecteur.

L'espace de travail est discrétisé en une grille (ex. 100×100), et les deux déterminants sont calculés en chaque point (x, y) pour les huit modes de fonctionnement. Après normalisation, on obtient deux cartes thermiques représentant les zones potentiellement dangereuses.

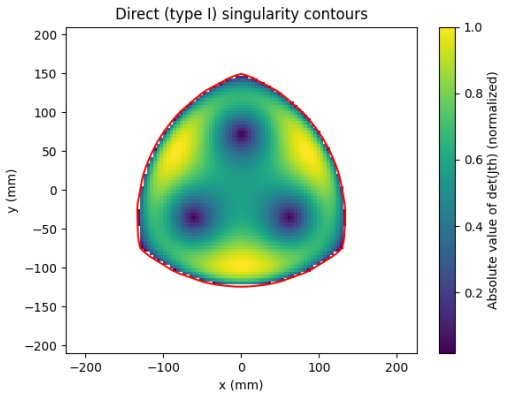

FIG. 2 – Contours des singularités de type I (parallèles) sur l'espace *compliant*

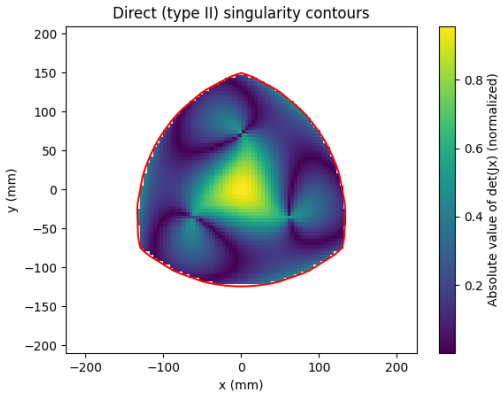

FIG. 3 – Contours des singularités de type II (sérielles) sur l'espace *compliant*

On observe que les singularités de type II limitent sévèrement l'espace atteignable du robot, tandis que les singularités de type I apparaissent plutôt en périphérie, dans des zones déjà interdites par les contraintes précédentes.

C.2 Détermination numérique de l'angle admissible: L'idée est de fixer un seuil ε (par exemple 0.5) en dessous duquel $\det(J_x)$ est considéré comme indiquant une singularité sérieuse.

La méthode consiste à :

1) Identifier tous les points de la carte $\det(J_x)$ où la valeur est inférieure à ε (zones bleues foncées).
2) Tester, pour chaque angle moteur $\theta_{lim} \in [1°, 180°]$, si une des singularités tombe dans l'espace *compliant* correspondant.
3) Conserver le plus grand θ_{lim} pour lequel aucune singularité n'est contenue dans l'espace atteint.

En pratique, le code utilise une fonction `isinterior()` qui teste chaque point singulier par rapport au polygone de workspace généré pour un angle donné.

C.3 Résultat obtenu: En fixant $\varepsilon = 0.5$, l'algorithme retourne un angle maximal admissible de :

$$\boxed{\theta_{max} = 33°}$$

Cela garantit que l'espace de travail reste convexe, sans singularités internes de type II, donc mécaniquement sûr.

D. Resultat obtenu sous le simulateur Pygame

Pour valider expérimentalement les longueurs optimisées des bras ainsi que la limitation angulaire déterminée dans la section C, nous avons utilisé notre simulateur Pygame développé en Python.

La figure ci-dessous montre la configuration du robot dans le simulateur avec les paramètres géométriques optimaux ($l_1 = l_2 = 98.19\,\text{mm}$, $R_e = 77.89\,\text{mm}$) et une limitation angulaire de $\theta_{max} = 33°$.

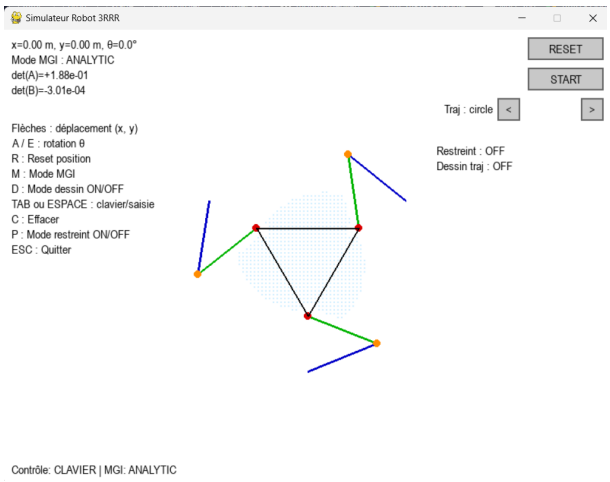

FIG. 4 – Résultat simulé avec les longueurs optimales et limitation angulaire dans Pygame

Cette configuration permet de garantir un **espace de travail convexe**, tel que démontré dans les cartes de singularités précédentes. Par ailleurs, la limitation angulaire appliquée aux moteurs contribue à **réduire fortement les risques de collision** entre les bras du robot.

La vérification de l'absence de collision a été réalisée sous **SolidWorks**, à travers une étude de mouvement complète avec les longueurs retenues. Les résultats confirment que les bras ne se croisent pas dans la zone de fonctionnement définie par l'angle limite.

E. Résumé visuel de la démarche

Voici une figure pour résumer notre démarche d'optimisation de l'espace de travail du robot 3-RRR :

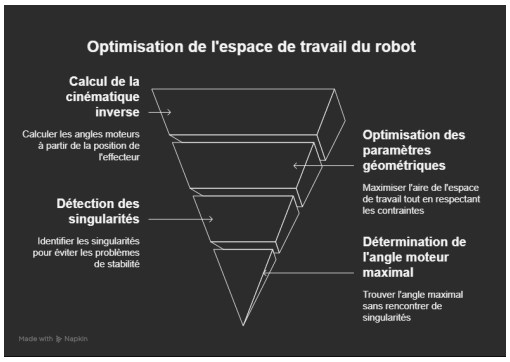

FIG. 5 – Résumé graphique de la démarche adoptée

III. SIMULATION D'UN ROBOT 3-RRR AVEC INTERFACE INTERACTIVE EN PYTHON

A. Introduction

Ce rapport présente le développement d'un simulateur interactif pour un robot parallèle 3-RRR. Le but est de visualiser la configuration du robot, simuler ses mouvements, détecter les singularités et suivre des trajectoires dans un environnement virtuel.

Le simulateur est développé en Python avec Pygame. Il permet une interaction directe par clavier ou saisie manuelle et offre un affichage graphique en temps réel.

B. Structure Générale du Simulateur

Le simulateur repose sur une boucle principale Pygame, qui exécute :

- la lecture des entrées clavier ou souris,
- la mise à jour de la pose de l'effecteur,
- les calculs de cinématique inverse,
- le rendu graphique du robot.

L'architecture logicielle est modulaire :

- `main.py` : boucle principale,
- `robot.py` : cinématique et dessin du robot,
- `graphics.py` : affichage des éléments graphiques,
- `controls.py` : gestion des événements clavier,
- `trajectoires.py` : trajectoires prédéfinies.

C. Affichage Graphique

Le simulateur affiche :

— les bras articulés du robot,
— le triangle formé par les points de l'effecteur,
— les points d'articulation B_i et E_i,
— les déterminants $\det(A)$ et $\det(B)$,
— les messages d'erreur (en rouge en cas de blocage).

D. Modes de Contrôle

Deux modes sont disponibles :

— **Mode clavier** : permet le déplacement avec les touches fléchées et la rotation avec A/E.
— **Mode manuel** : permet la saisie numérique de (x, y, θ) via une zone de texte.

Un indicateur visuel permet de savoir quel mode est actif.

E. Détection des Singularités

Le simulateur identifie automatiquement les singularités :

— Sing. parallèle si $|\det(A)| < 10^{-4}$,
— Sing. série si $|\det(B)| < 10^{-4}$.

Dans ces cas, le robot ne se déplace pas, s'affiche en rouge, et un message d'erreur est affiché.

F. Suivi de Trajectoire et Mode Dessin

Le robot peut :

— suivre automatiquement des trajectoires comme `cercle`, `ligne`, `cœur`, `8`,
— tracer sa trajectoire avec le mode dessin activé.

Les points suivis sont connectés visuellement pour afficher le chemin parcouru.

G. Mode Restreint et Espace Atteignable

Le mode restreint interdit toute configuration où un point B_i entre dans le triangle $E_1E_2E_3$, simulant ainsi les collisions internes.

L'espace atteignable est précalculé à l'initialisation et affiché comme un nuage de points bleus clairs en fond d'écran.

H. Illustrations

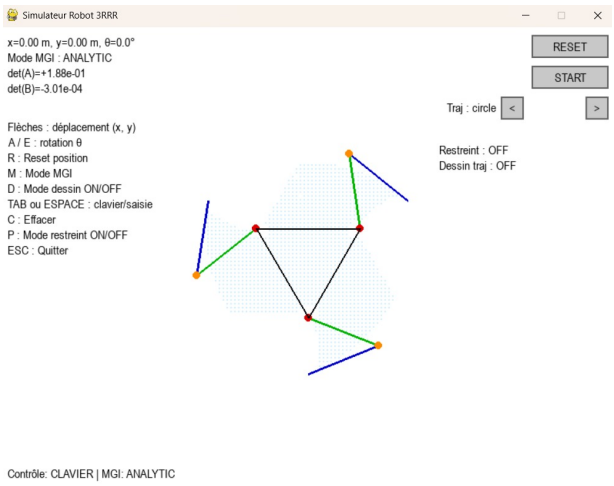

FIG. 6 – Interface du simulateur - Configuration initiale

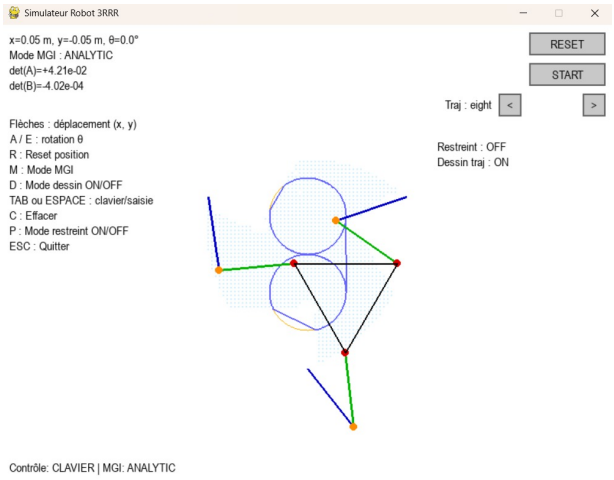

FIG. 7 – Exécution d'une trajectoire prédéfinie

FIG. 8 – Erreur affichée en cas de singularité

FIG. 9 – Saisie manuelle des coordonnées

I. Conclusion

Ce simulateur permet de tester visuellement et dynamiquement la cinématique d'un robot 3-RRR. Il offre une interface complète pour tester les mouvements, éviter les singularités, tracer des trajectoires et évaluer l'espace atteignable. Ce projet constitue une base solide pour des extensions futures telles que la commande en boucle fermée ou la connexion à un bras réel.

IV. MODÉLISATION CAO ET IMPRESSION 3D

Pour garantir la faisabilité rapide et modulaire du robot, nous avons conçu une architecture mécanique fondée sur la **reproductibilité** et la **modularité**. L'idée principale est de faciliter les ajustements géométriques (comme le rayon de la base ou la longueur des bras) sans devoir réimprimer entièrement le robot à chaque itération.

A. Modularité géométrique

Nous avons imaginé une structure parallèle 3-RRR dans laquelle le **rayon de base** (positionnement des moteurs) peut être modifié simplement en changeant la position de quelques vis et en perçant de nouveaux trous dans les plaques découpées au laser. Cela permet d'adapter rapidement le robot à différentes configurations sans réimpression longue ni coûteuse.

En cas d'erreur ou de modification nécessaire (par exemple un mauvais choix de longueur de bras), la nouvelle version peut être imprimée et montée en moins d'une heure.

B. Supports moteurs et assemblage

Pour fixer les servomoteurs Dynamixel AX-12, nous avons modélisé des supports spécifiques en 3D. Ceux-ci sont fixés sur la base du robot via des vis standards. Les deux bras identiques sont ensuite attachés d'un côté au moteur (via le support) et de l'autre au triangle mobile (la plateforme de l'effecteur).

Le triangle de l'effecteur ainsi que les bras ont été conçus pour permettre un montage simple, symétrique et rigide. Toutes les pièces ont été pensées pour être découpées au laser ou imprimées en PLA.

C. Dimensions retenues

Les longueurs géométriques utilisées dans la conception finale du robot sont directement issues des résultats de la Section **B** (optimisation). Nous avons donc utilisé :

— $l_1 = l_2 = 98.19\,\mathrm{mm}$
— $R_e = 77.89\,\mathrm{mm}$
— $R_b = 150\,\mathrm{mm}$ (fixé comme contrainte)

Ces dimensions assurent à la fois une bonne surface exploitable, l'absence de singularité critique, et une réalisation pratique dans le temps imparti.

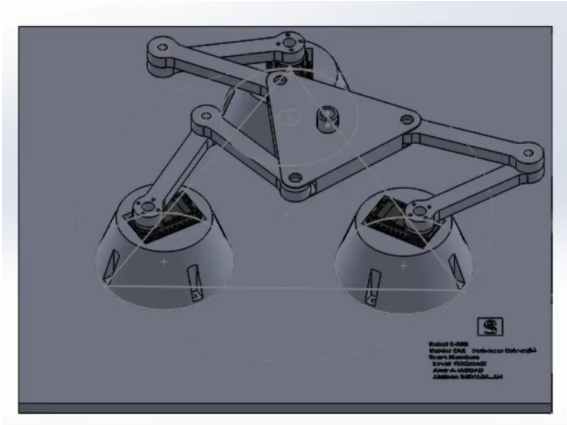

FIG. 10 – Assemblage CAO final du robot 3-RRR. Les trois servomoteurs Dynamixel sont fixés sur des supports imprimés, permettant le montage symétrique des bras et de la plateforme triangulaire.

D. Assemblage des pièces et problèmes rencontrés

L'ensemble des pièces que nous avions modélisées a été imprimé avec succès, marquant une étape importante du projet. L'assemblage nous a ensuite permis de confronter notre conception virtuelle à la réalité, révélant plusieurs opportunités d'amélioration, notamment en matière de dimensionnement.

Les bras se sont avérés plus longs que prévu par rapport à la taille de la base du support de traçage, rendant l'ajustement délicat. Pour un fonctionnement optimal, une base d'environ 45 × 45 cm aurait été plus appropriée. De plus, les câbles des servomoteurs étant un peu courts, nous avons d'abord envisagé de les rallonger, avant de réaliser que le principal défi venait des proportions générales de la structure.

Nous avons également travaillé sur la fluidité des mouvements des bras, en tentant de réduire au maximum les frottements. L'effecteur, bien que conforme à nos attentes en termes de forme, s'est révélé un peu trop grand et lourd pour les servomoteurs utilisés. Cela a mis en évidence l'importance de prendre en compte les contraintes mécaniques réelles dès la phase de conception.

D'autres aspects techniques ont enrichi notre compréhension : certains servomoteurs présentaient des zones mortes inattendues, et l'un d'eux semblait légèrement moins performant, possiblement en raison de l'usure. Ces situations nous ont permis d'apprendre à diagnostiquer des écarts de performance et à ajuster notre système en conséquence.

Face aux contraintes d'espace et de performance, nous avons su faire preuve de réactivité et d'adaptabilité. Nous avons modifié la disposition des moteurs en les positionnant sur une plaque de 38,6 × 38,6 cm, formant un triangle équilatéral : deux dans des coins opposés et le troisième centré sur le côté opposé.

La configuration a été réalisée à l'aide du logiciel Dynamixel Wizard. Malgré des protections activées par surcharge (overload) après quelques instants de fonctionnement, nous avons pu prendre le contrôle des moteurs et observer leur comportement. Cela nous a permis de comprendre finement les limites à respecter pour garantir un fonctionnement stable.

Ces défis techniques ont enrichi notre démarche, en nous poussant à analyser, ajuster et améliorer constamment notre robot. Plusieurs idées d'optimisation ont émergé de cette phase, que nous détaillons dans la conclusion.

V) CONCLUSION

Ce projet a représenté une expérience d'apprentissage exceptionnelle, tant sur le plan technique que méthodologique. Il nous a permis de mobiliser nos compétences en mécanique, électronique et informatique, dans le cadre d'un projet concret, stimulant et complet.

Nous avons appris à gérer les contraintes de fabrication, les limites des équipements disponibles au Fablab, ainsi que les capacités réelles des servomoteurs Dynamixel. Ce projet nous a aussi plongés dans une démarche de résolution de problèmes, où chaque difficulté rencontrée a été une occasion de progresser.

Par manque de temps, nous n'avons pas pu pousser la réalisation aussi loin que nous l'aurions souhaité. Avec un peu plus de semaines de travail, nous aurions pu finaliser un système

plus optimisé et proche d'un fonctionnement parfait.

Pistes d'amélioration identifiées :

- Réduire les dimensions et la masse de l'effecteur pour éviter la surcharge des servomoteurs.
- Concevoir des bras plus courts et plus rigides pour améliorer la stabilité.
- Optimiser l'agencement des moteurs dès la phase de conception.
- Alléger les pièces imprimées sans compromettre leur solidité.
- Anticiper les zones mortes des servomoteurs dans la modélisation.
- Améliorer la gestion logicielle des erreurs (overload) pour éviter les redémarrages fréquents.
- Privilégier une configuration plus compacte pour limiter les efforts mécaniques.

Malgré tout, nous avons su nous adapter, collaborer efficacement, et tirer le maximum des ressources à notre disposition. Ce projet nous a permis de développer une compréhension concrète de la robotique et de renforcer notre capacité à gérer un projet technique de bout en bout.

Code source et vidéo disponible sur GitHub :
`https://github.com/damiski200240/projet_modelisation-.git`