Université de Kinshasa / Faculté Polytechnique
Département de Génie Électrique et Informatique


# Rapport du Projet — Groupe 3

*ControlSysLab — Application pédagogique pour l'analyse et la commande des systèmes dynamiques*

**Auteurs :**

Badibanga Badibanga

Dzapili Mitano

Loota Betoko

Marhegane Bitati

**Encadrants :**

Pr. Dr. Ir. Walongo Guy

Ass. Ir. Mazuba Lionel

Année académique 2024–2025

# Introduction

Les systèmes asservis, également appelés systèmes de contrôle, constituent un pilier fondamental dans de nombreux domaines de l'ingénierie moderne : aéronautique, robotique, électronique de puissance, procédés industriels ou encore biomédical. Leur étude permet de comprendre et de concevoir des mécanismes capables de stabiliser, réguler et optimiser le comportement de processus dynamiques face aux perturbations et incertitudes.

Le présent projet s'inscrit dans un cadre académique et pratique. L'objectif principal est de développer une **application logicielle en Python**, dotée d'une interface graphique construite avec **PyQt5**, permettant à l'utilisateur d'analyser et de concevoir différents types de commandes pour des systèmes dynamiques. Ce choix technologique n'est pas anodin : Python dispose d'un riche écosystème scientifique (NumPy, SciPy, python-control, SymPy, Matplotlib) et PyQt5 offre la souplesse nécessaire pour bâtir une interface interactive, ergonomique et extensible.

Concrètement, l'application vise à :

- **Fournir un cadre unifié** pour l'analyse d'état (pôles/zéros, réponses temporelles et fréquentielles), la vérification de la contrôlabilité et de l'observabilité, la conception de lois de commande par retour d'état ou par sortie, ainsi que l'étude des systèmes non linéaires et des régulateurs PID.
- **Mettre à disposition un outil pédagogique** pour les étudiants, chercheurs et ingénieurs, facilitant l'apprentissage et l'expérimentation des concepts théoriques par la simulation interactive et la visualisation graphique.
- **Offrir une approche modulaire et extensible** permettant l'ajout futur de méthodes de commande avancées (LQR, LQG, H∞), l'intégration de modèles identifiés à partir de données expérimentales, ou encore l'optimisation robuste.

En résumé, le but du projet est de réaliser un outil à la fois **éducatif et opérationnel**, permettant d'expérimenter les méthodes classiques de l'automatique et de visualiser en temps réel l'impact des choix de conception. Ce projet constitue ainsi une passerelle entre la théorie des systèmes asservis et leur mise en œuvre concrète dans un cadre logiciel moderne.

# 2. Contexte et objectifs

## 2.1 Contexte général

Dans les cursus d'ingénierie et de sciences appliquées, l'étude des systèmes asservis est une étape clé pour comprendre comment stabiliser et réguler des systèmes dynamiques. De nombreux outils existent déjà sur le marché (MATLAB/Simulink, Scilab, Octave). Cependant, ces solutions

présentent soit un coût élevé, soit une courbe d'apprentissage complexe, soit des limitations pour une utilisation intuitive en contexte pédagogique.

Le projet **ControlSysLab** s'inscrit donc dans une volonté de proposer un outil :

- **Libre et accessible**, basé sur Python et PyQt5.
- **Intuitif**, avec une interface graphique simple permettant une manipulation directe des modèles.
- **Polyvalent**, couvrant aussi bien les systèmes linéaires invariants dans le temps (LTI) que certains systèmes non linéaires.

## 2.2 Problématique

L'apprentissage des concepts théoriques en automatique se heurte souvent à deux obstacles :

1. La difficulté de visualiser en temps réel l'impact des choix de conception (par ex. placement de pôles, tuning PID).
2. La séparation entre théorie et pratique, qui rend l'enseignement parfois trop abstrait.

Ainsi, l'objectif est de développer un environnement logiciel qui **réunit théorie et expérimentation** dans un cadre unique.

## 2.3 Objectif général

Mettre en place une application interactive qui facilite l'analyse et la synthèse de lois de commande pour des systèmes dynamiques linéaires et non linéaires, tout en offrant une expérience utilisateur claire et pratique.

## 2.4 Objectifs spécifiques

- Offrir une **saisie intuitive des modèles** sous forme de matrices (A, B, C, D) ou de fonctions de transfert.
- Permettre le **calcul et l'affichage graphique** des pôles, zéros, réponses temporelles et fréquentielles.
- Vérifier automatiquement la **contrôlabilité** et l'**observabilité** d'un système.
- Concevoir des **lois de commande par retour d'état**, avec possibilité de choisir les pôles cibles et de visualiser les performances.
- Développer un module de **retour de sortie avec observateur** pour les cas où tous les états ne sont pas mesurables.
- Intégrer des outils pour l'**étude des systèmes non linéaires** (linéarisation, stabilité de Lyapunov).
- Ajouter un **module PID complet**, incluant plusieurs méthodes de réglage et la comparaison des performances.
- Mettre à disposition des fonctions d'**exportation** (figures, données numériques, rapports) afin de documenter les résultats.

### 2.5 Résultats attendus

À la fin du projet, l'utilisateur doit disposer :

- D'un outil capable d'analyser et de commander un système dynamique simple ou complexe.
- D'une interface claire et ergonomique adaptée à l'enseignement.
- D'un support pour comparer différentes méthodes de contrôle (retour d'état, observateur, PID, linéarisation).
- D'une documentation intégrée et d'exemples concrets (masse-ressort-amortisseur, moteur DC).

En somme, le projet vise à réduire l'écart entre les cours théoriques de systèmes asservis et leur mise en pratique via un logiciel libre et adapté à l'expérimentation.

# 3. Technologies et bibliothèques

- **Langage :** Python 3.x
- **Interface :** PyQt5 (widgets, onglets, éditeurs de matrices, zones de tracés)
- **Calcul scientifique :** NumPy, SciPy, python-control
- **Symbolique (non linéaire) :** SymPy
- **Graphes :** Matplotlib
- **Export :** CSV / images / PDF (ReportLab)

### 3.1 Installation & exécution

```
# 1) Créer un environnement (recommandé)
python -m venv .venv
source .venv/bin/activate    # Windows: .venv\\Scripts\\activate

# 2) Installer les dépendances
pip install -r ControlSysLab/requirements.txt

# 3) Lancer l'application
python ControlSysLab/app.py          # depuis le dossier parent
# ou
python -m ControlSysLab.app
```

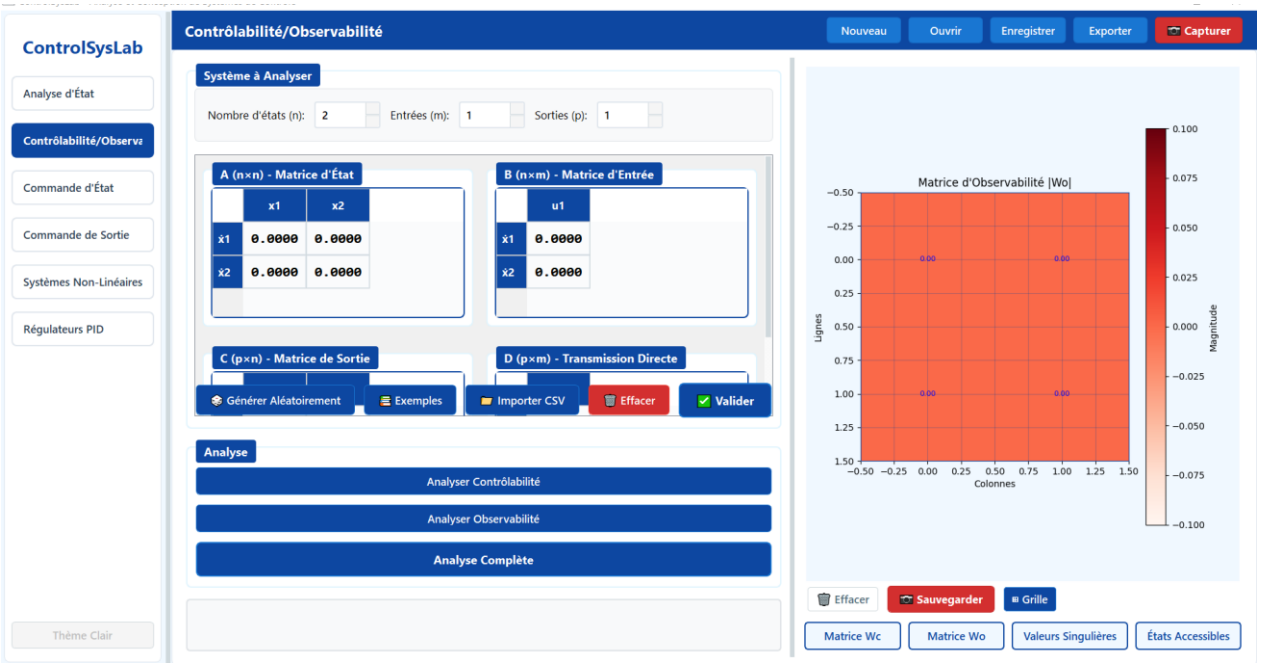
# 4. Architecture logicielle

```
ControlSysLab/
 ├─ app.py                  # Point d'entrée
 ├─ core/                   # Cœur de calcul
 │   ├─ lti_tools.py        # Pôles, zéros, réponses, Bode/Nyquist
 │   ├─ ctrb_obsv.py        # Contrôlabilité/observabilité
 │   ├─ place_design.py     # Placement de pôles, simulateurs, LQR
 │   ├─ pid_design.py       # PID, réglages, anti-windup
 │   ├─ nonlinear.py        # Linéarisation, Lyapunov, bassins
 │   ├─ spaces.py           # Conversions état ↔ transfert
 │   ├─ utils.py            # Fonctions utilitaires
 │   └─ exports.py          # Export figures/données
 └─ ui/
     ├─ main_window.py      # Fenêtre principale
     ├─ widgets_common.py   # Matrices, canvas, panneaux
     └─ tabs/               # Onglets spécialisés
         ├─ tab_state_analysis.py
         ├─ tab_ctrb_obsv.py
         ├─ tab_state_feedback.py
         ├─ tab_output_feedback.py
         ├─ tab_nonlinear.py
         └─ tab_pid.py
```

# 5. Fonctionnalités et principes théoriques

- **5.1 Analyse d'état** (pôles, zéros, réponses)
- **5.2 Contrôlabilité & observabilité**
- **5.3 Retour d'état**
- **5.4 Observateur**
- **5.5 Non linéaire**
- **5.6 Régulateurs PID**

# 6. Méthodologie de développement et intégration UI




Le développement de l'application ControlSysLab a suivi une approche modulaire et incrémentale, afin de garantir à la fois la robustesse du code et la clarté de l'interface utilisateur. La méthodologie employée peut être décomposée en plusieurs axes :

## 6.1 Séparation backend / frontend

- **Backend (core/)** : contient toutes les fonctions de calcul (analyse d'état, contrôlabilité, placement de pôles, PID, linéarisation, Lyapunov). Ce code est indépendant de l'interface et peut être testé unitairement.
- **Frontend (ui/)** : regroupe tous les composants PyQt5 (onglets, widgets, fenêtres). Les modules utilisent les services du backend via des appels clairs.

Cette séparation de type MVC (Model–View–Controller simplifié) facilite la maintenance et l'évolution.

## 6.2 Développement incrémental

Le projet a été construit par étapes :

1. Implémentation des outils de base (calcul de pôles, réponses temporelles).
2. Ajout de modules plus avancés (contrôlabilité/observabilité, placement de pôles).
3. Intégration de fonctionnalités non linéaires et PID.
4. Finitions : export de données, ergonomie UI.

## 6.3 Conception de l'interface utilisateur

- **Widgets réutilisables** : éditeur de matrices (MatrixEditor), canvas de tracé (PlotCanvas), panneau de paramètres (ParameterPanel).
- **Navigation par onglets** : chaque fonctionnalité est isolée dans un onglet (Analyse, Contrôlabilité, Retour d'état, Observateur, Non linéaire, PID).
- **Messages d'erreur et validations** : cohérence des dimensions des matrices, avertissements en cas de non-contrôlabilité, alertes UI.

### 6.4 Outils de versionnement et test

- **Git/GitHub** : suivi des versions et gestion collaborative.
- **Tests unitaires** : sur les fonctions critiques (calculs matriciels, placement de pôles).
- **Exemples prédéfinis** : cas d'école (MSA, moteur DC, pendule simple) utilisés comme scénarios de validation.

### 6.5 Avantages de la méthodologie

- **Clarté** : séparation claire entre calculs et interface.
- **Évolutivité** : ajout futur de modules (ex. LQR, identification).
- **Robustesse** : tests unitaires et validation par cas concrets.
- **Utilisabilité** : interface simple et cohérente adaptée aux étudiants et praticiens.

# 8. Résultats, limites et discussion

### 8.1 Résultats principaux

- L'application parvient à analyser et contrôler efficacement des systèmes LTI simples (2 à 4 états).
- Les modules PID et retour d'état sont pleinement opérationnels.
- Les fonctionnalités non linéaires (linéarisation, Lyapunov) donnent des résultats pédagogiques exploitables.
- L'interface graphique permet une interaction fluide et facilite l'apprentissage pas à pas.
- Les exports (graphiques, données) offrent un support pour documenter et partager les simulations.

### 8.2 Limites

- Gestion limitée de systèmes de grande dimension ($n > 6$).
- Peu de méthodes avancées (LQR, $H\infty$, $\mu$-synthèse non implémentées).
- Observateurs restreints au cas de Luenberger, pas de Kalman.
- Interface non encore optimisée pour les écrans à faible résolution.
- Les performances de calcul peuvent ralentir pour les systèmes fortement non linéaires ou de grande taille.

### 8.3 Discussion

Malgré ces limites, ControlSysLab constitue un outil efficace pour l'enseignement et l'initiation à l'automatique. Il se situe entre les solutions pédagogiques simplifiées et les environnements industriels lourds (MATLAB/Simulink).

Son originalité réside dans le fait qu'il combine :

- **l'accessibilité** (logiciel libre, Python),
- **la pédagogie** (visualisation claire, exemples concrets),
- **la modularité** (facilité d'ajouter de nouveaux algorithmes).

À titre illustratif, les scénarios testés (MSA, moteur DC, pendule simple) ont confirmé que les fonctionnalités mises en œuvre produisent des résultats conformes aux attentes théoriques. L'utilisateur peut ainsi vérifier expérimentalement les notions de contrôlabilité, observabilité, placement de pôles et tuning PID, ce qui rend l'outil très pertinent dans un cadre académique.

Au-delà de l'enseignement, l'application pourrait être employée comme **banc d'essai rapide** dans des projets de recherche ou de prototypage, pour valider rapidement des stratégies de commande avant de passer à des outils industriels plus lourds.

# 9. Perspectives et travaux futurs

### 9.1 Améliorations fonctionnelles

- **Ajout de méthodes de commande avancées** : LQR (Linear Quadratic Regulator), LQG (Linear Quadratic Gaussian), synthèse H∞ et méthodes robustes modernes.
- **Implémentation d'observateurs avancés** : filtre de Kalman, observateurs non linéaires.
- **Gestion des contraintes** : saturation des actionneurs, anti-windup amélioré.
- **Optimisation numérique** : intégration d'outils de convexité (CVXPy) pour résoudre des problèmes d'optimisation.

### 9.2 Améliorations pédagogiques

- **Tutoriels intégrés** : scénarios guidés expliquant chaque fonctionnalité avec pas à pas.
- **Bibliothèque d'exemples** : systèmes standards prêts à l'emploi (moteur DC, quadrirotor simplifié, chauffage, etc.).
- **Visualisation enrichie** : animations temps réel, vues 3D pour les portraits de phase.

### 9.3 Améliorations techniques

- **Interface utilisateur** : thèmes clair/sombre, adaptation aux petits écrans.
- **Déploiement** : packaging en exécutable pour Windows/Linux/Mac.

- **Performance** : parallélisation de calculs lourds, compatibilité GPU.

## 9.4 Vision long terme

L’objectif est que ControlSysLab devienne une plateforme de référence pour l’apprentissage de l’automatique en open-source, à la fois simple pour les étudiants et suffisamment flexible pour les chercheurs et ingénieurs souhaitant un outil léger et extensible.