

DOUBLE PENDULE INVERSÉ SUR UN CHARIOT

Exploration des Systèmes Non Linéaires et de la
Conception Assistée par Ordinateur (CAO)

Presented by:

- Elyes CHAMEKH
- Ayoub HEDFI
- Elyes KHECHINE

Jury

- Prof. Olfa BOUBAKER

SOMMAIRE

01

Introduction & Objectifs

02

Modélisation mathématique:

- Equations du système dynamique: Méthode de Lagrange
- Modélisation du système sur MATLAB
- Modélisation du système sur LabVIEW

03

Simulations du système:

- Simulation #1: *Commande LQR du système DIP linéarisé de Quanser avec LabVIEW*
- Simulation #2: *Swing-Up basée sur Python utilisant Gekko et Matplotlib, présentée sur un Dashboard Web interactif*
- Simulation #3: *Régulateur LQR pour un robot modélisant le système physique DIP dans ROS*
- Simulation #4 : *Apprentissage par renforcement - Recherche de stratégies de haute qualité pour le système DIP*

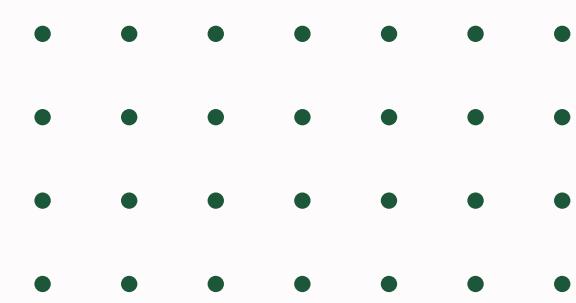
05

Conclusion Générale

06

Références

N.B: DIP = Double Inverted Pendulum





LES OBJECTIFS DU PROJET



- Mettre en évidence les applications pratiques du double pendule inversé dans la conception de systèmes de contrôle avancés et la stabilisation de plates-formes à pendule inversé
- Expliquer la méthode de Lagrange pour obtenir les équations dynamiques du système et souligner l'importance de la modélisation mathématique
- Utiliser MATLAB pour simuler et analyser le comportement du système sous différentes stratégies de contrôle.
- Appliquer les concepts théoriques à des systèmes réels en utilisant LabVIEW pour la modélisation du système.
- Simuler le système DIP linéarisé de Quanser avec LabVIEW en utilisant un régulateur LQR et analyser les performances du contrôleur
- Développer une stratégie de contrôle Swing-Up basée sur Python avec Gekko et Matplotlib, présentée sur un Dashboard Web interactif.
- Simuler le système DIP modélisé sur un chariot dans ROS et Gazebo avec un régulateur LQR pour évaluer le comportement dans un environnement virtuel réaliste.
- Mettre en œuvre un modèle d'apprentissage par renforcement pour rechercher des stratégies performantes pour le système DIP.

Introduction

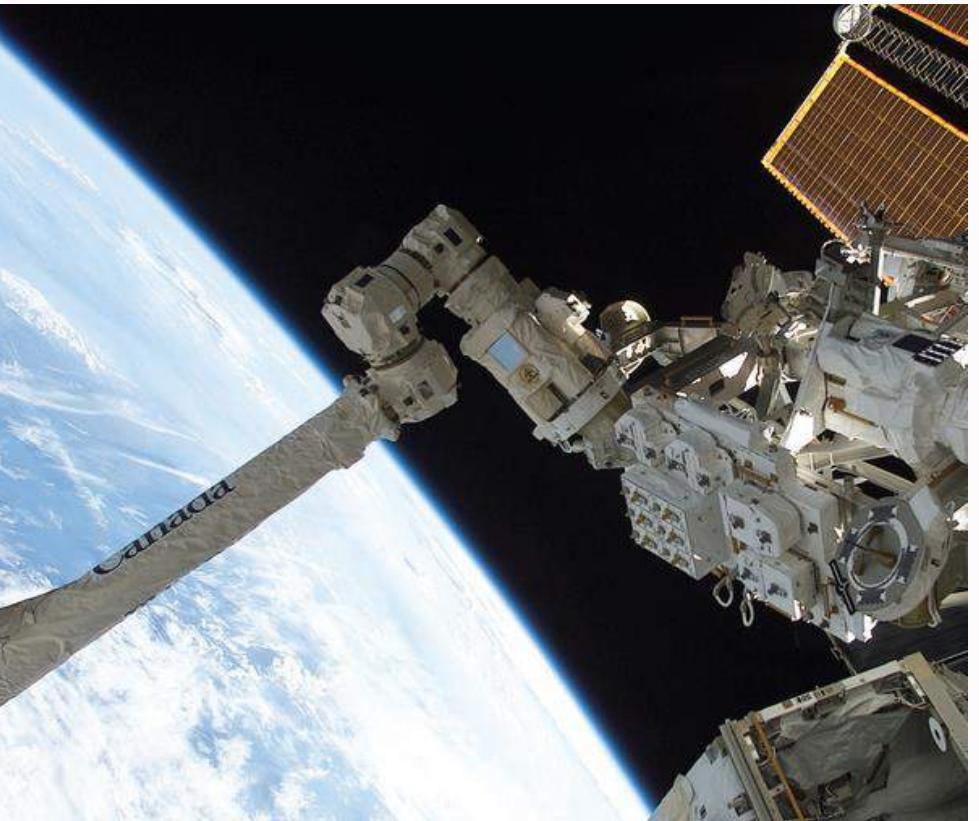
Le double pendule inversé est un système qui a suscité beaucoup d'intérêt dans le domaine scientifique et pédagogique ces dernières années.

Il est utilisé pour tester les nouvelles méthodes de commande, car il présente plusieurs caractéristiques attrayantes telles que :

- Il est un système non linéaire,
- Il est couplé,
- Il est intrinsèquement instable.

Exemples d'applications pratiques:

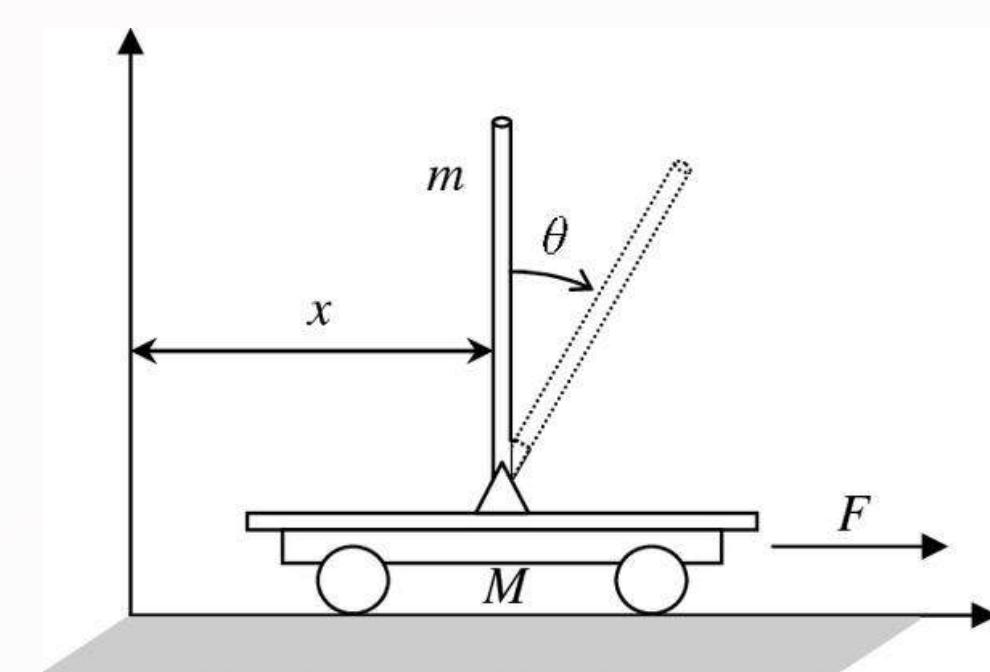
- Conception de systèmes de contrôle pour des manipulateurs robotiques avancés, permettant des mouvements précis et coordonnés.
- Développement de mécanismes de stabilisation pour les plates-formes à pendule inversé utilisées dans les transports, telles que les segways et les scooters électriques.



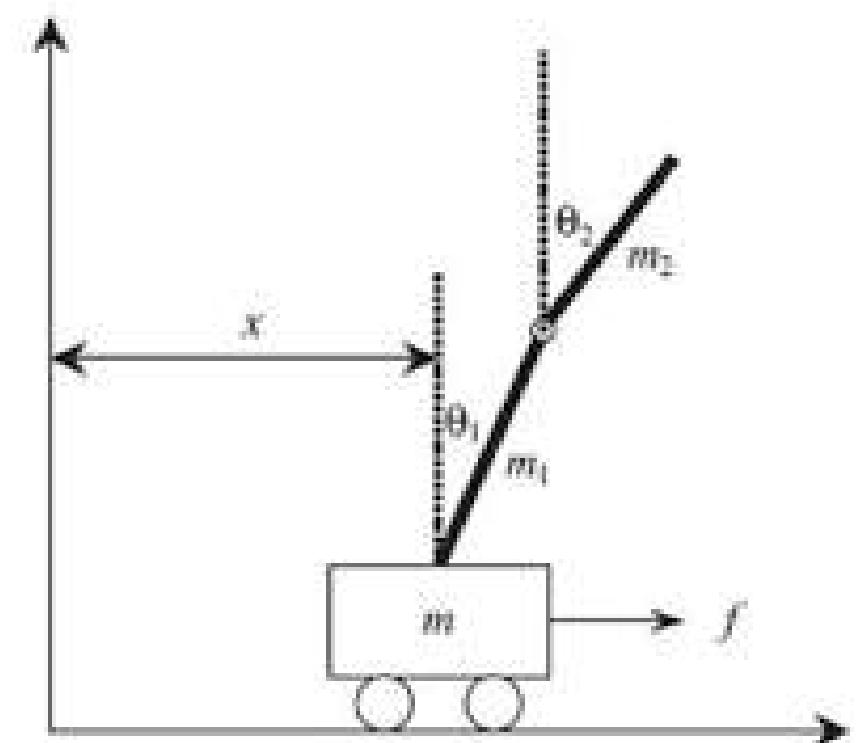
Pendule simple vs Pendule double

La principale différence entre un double pendule inversé et un pendule inversé simple est leur complexité et leur comportement.

Alors qu'un pendule inversé simple peut être stabilisé à l'aide de techniques de commande simples, la stabilisation d'un double pendule inversé nécessite des méthodes de commande plus avancées en raison de son comportement hautement non linéaire et chaotique.



Pendule simple inversée



Pendule double inversée

MODÈLISATION MATHÉMATIQUE

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\theta}} \right) - \frac{\partial L}{\partial \theta} = \mathbf{q}$$

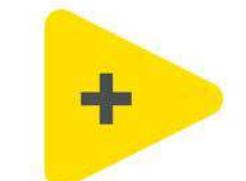
Équations du système dynamique

Nous examinerons brièvement les équations qui régissent la dynamique du système, en utilisant la méthode de Lagrange



Modélisation du système sur MATLAB

Nous avons utilisé MATLAB pour simuler le modèle mathématique et analyser le comportement du système sous différentes stratégies de contrôle.



LabVIEW™

Modélisation du système sur LabVIEW

Nous avons exploité les capacités de LabVIEW pour modéliser et simuler la dynamique du système. Cette section donne un aperçu de notre expérience en matière de modélisation du système sur LabVIEW.

EQUATIONS DU SYSTÈME DYNAMIQUE

Référence(s):

- University of New Mexico - Control Theory of The Double Pendulum Inverted on a Cart/ <https://drive.google.com/file/d/1VxT84KJXg89n7mJowL-xiMBDLQW6Sbc3/view?usp=sharing>

EQUATIONS DU SYSTÈME DYNAMIQUE: MÉTHODE DE LAGRANGE

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\theta}} \right) - \frac{\partial L}{\partial \theta} = q$$

$$\theta = \begin{pmatrix} x \\ \theta_1 \\ \theta_2 \end{pmatrix}$$

$$q = \begin{pmatrix} u(t) \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} u(t) = H u(t)$$

EQUATIONS DU SYSTÈME DYNAMIQUE: méthode de Lagrange

$$\mathbf{q} = \begin{pmatrix} u(t) \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} u(t) = \mathbf{H}u(t)$$

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{x}} \right) - \frac{\partial L}{\partial x} = u(t)$$

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\theta}_1} \right) - \frac{\partial L}{\partial \theta_1} = 0$$

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\theta}_2} \right) - \frac{\partial L}{\partial \theta_2} = 0$$

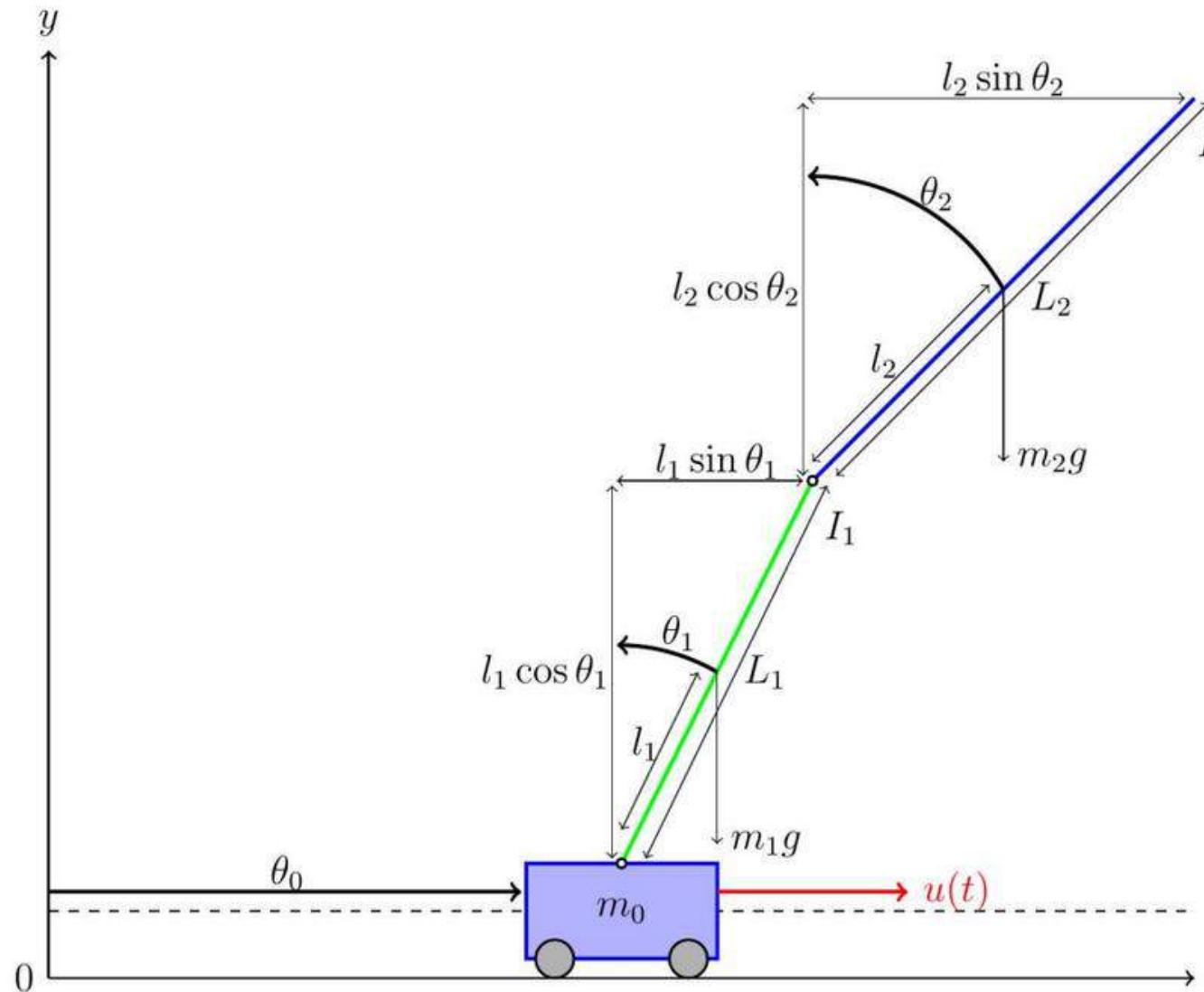
EQUATIONS DU SYSTÈME DYNAMIQUE: méthode de Lagrange

$$L = E_{kin} - E_{pot}$$

$$E_{kin} = E_{kin}^{(0)} + E_{kin}^{(1)} + E_{kin}^{(2)}$$

$$E_{pot} = E_{pot}^{(0)} + E_{pot}^{(1)} + E_{pot}^{(2)}$$

EQUATIONS DU SYSTÈME DYNAMIQUE: méthode de Lagrange



$$x_0 = x$$

$$y_0 = 0$$

The position of the midpoint of the first pendulum link

$$x_1 = x + l_1 \sin \theta_1$$

$$y_1 = l_1 \cos \theta_1$$

The position of the midpoint of the second pendulum link

$$x_2 = x + L_1 \sin \theta_1 + l_2 \sin \theta_2$$

$$y_2 = L_1 \cos \theta_1 + l_2 \cos \theta_2$$

Using these coordinates the energy components can be calculated. First, using standard results from Newtonian mechanics [11], calculate the kinetic and potential energy, $E_{kin}^{(0)}$ and $E_{pot}^{(0)}$, for the cart

$$E_{kin}^{(0)} = \frac{1}{2}m_0\dot{x}^2$$

$$E_{pot}^{(0)} = 0$$

EQUATIONS DU SYSTÈME DYNAMIQUE: méthode de Lagrange

$$E_{kin}^{(1)} = E_{kin}^{(1)}(\text{ trans }) + E_{kin}^{(1)}(\text{ rot })$$

$$E_{kin}^{(1)} = \frac{1}{2}m_1 (\dot{x}_1^2 + \dot{y}_1^2) + \frac{1}{2}I_1\dot{\theta}_1^2$$

$$E_{kin}^{(1)} = \frac{1}{2}m_1 \left\{ \left(\frac{d}{dt} [x + l_1 \sin \theta_1] \right)^2 + \left(\frac{d}{dt} [l_1 \cos \theta_1] \right)^2 \right\} + \frac{1}{2}I_1\dot{\theta}_1^2$$

$$E_{kin}^{(1)} = \frac{1}{2}m_1 \left[(\dot{x} + l_1 \dot{\theta}_1 \cos \theta_1)^2 + (-l_1 \dot{\theta}_1 \sin \theta_1)^2 \right] + \frac{1}{2}I_1\dot{\theta}_1^2$$

$$E_{kin}^{(1)} = \frac{1}{2}m_1 \left[\dot{x}^2 + 2l_1 \dot{x} \dot{\theta}_1 \cos \theta_1 + l_1^2 \dot{\theta}_1^2 \cos^2 \theta_1 + l_1^2 \dot{\theta}_1^2 \sin^2 \theta_1 \right] + \frac{1}{2}I_1\dot{\theta}_1^2$$

$$E_{kin}^{(1)} = \frac{1}{2}m_1 \left[\dot{x}^2 + 2l_1 \dot{x} \dot{\theta}_1 \cos \theta_1 + l_1^2 \dot{\theta}_1^2 \right] + \frac{1}{2}I_1\dot{\theta}_1^2$$

$$E_{kin}^{(1)} = \frac{1}{2}m_1 \dot{x}^2 + \frac{1}{2} (m_1 l_1^2 + I_1) \dot{\theta}_1^2 + m_1 l_1 \dot{x} \dot{\theta}_1 \cos \theta_1$$

$$L = E_{kin} - E_{pot}$$

$$\begin{aligned} &= \frac{1}{2} (m_0 + m_1 + m_2) \dot{x}^2 + \frac{1}{2} (m_1 l_1^2 + m_2 L_1^2 + I_1) \dot{\theta}_1^2 + \frac{1}{2} (m_2 l_2^2 + I_2) \dot{\theta}_2^2 \\ &\quad + (m_1 l_1 + m_2 L_1) \cos \theta_1 \dot{x} \dot{\theta}_1 + m_2 l_2 \cos \theta_2 \dot{x} \dot{\theta}_2 + m_2 L_1 l_2 \cos (\theta_1 - \theta_2) \dot{\theta}_1 \dot{\theta}_2 \\ &\quad - g (m_1 l_1 + m_2 L_1) \cos \theta_1 - m_2 g l_2 \cos \theta_2 \end{aligned}$$

EQUATIONS DU SYSTÈME DYNAMIQUE: méthode de Lagrange

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{x}} \right) - \frac{\partial L}{\partial x} = u(t)$$

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\theta}_1} \right) - \frac{\partial L}{\partial \theta_1} = 0$$

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\theta}_2} \right) - \frac{\partial L}{\partial \theta_2} = 0$$

$$\mathbf{D}(\boldsymbol{\theta}) = \begin{bmatrix} m_0 + m_1 + m_2 & (m_1 l_1 + m_2 L_1) \cos \theta_1 & m_2 l_2 \cos \theta_2 \\ (m_1 l_1 + m_2 L_1) \cos \theta_1 & m_1 l_1^2 + m_2 L_1^2 + I_1 & m_2 L_1 l_2 \cos (\theta_1 - \theta_2) \\ m_2 l_2 \cos \theta_2 & m_2 L_1 l_2 \cos (\theta_1 - \theta_2) & m_2 l_2^2 + I_2 \end{bmatrix}$$

$$\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = \begin{bmatrix} 0 & -(m_1 l_1 + m_2 L_1) \sin \theta_1 \dot{\theta}_1 & -m_2 l_2 \sin \theta_2 \dot{\theta}_2 \\ 0 & 0 & m_2 L_1 l_2 \sin (\theta_1 - \theta_2) \dot{\theta}_2 \\ 0 & -m_2 L_1 l_2 \sin (\theta_1 - \theta_2) \dot{\theta}_1 & \end{bmatrix}$$

$$\mathbf{G}(\boldsymbol{\theta}) = \begin{bmatrix} -(m_1 l_1 + m_2 L_1) g \sin \theta_1 \\ -m_2 g l_2 \sin \theta_2 \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

MODÉLISATION DU SYSTÈME SUR MATLAB

Référence(s):

- *Implementation of MATLAB Code for LQR/LQG for linearized and non-linearized system:* <https://github.com/nakul3112/Double-inverted-pendulum.git>

LES HYPOTHÈSES

- Il n'y a pas de frottement.
- Les tiges auxquelles les masses sont suspendues sont considérées comme sans masse et rigides.
- Le chariot se déplace linéairement le long de l'axe X uniquement.
- Le mouvement du chariot en unidimensionnel.
- Le mouvement du pendule est bidimensionnel.
- Toutes les masses ont une densité uniforme.
- Le centre de masse de tous les corps se trouve à leur centre géométrique.
- Les axes de coordonnées sont extérieurs à l'axe X à la hauteur du point de suspension, qui coïncide avec le centre de masse du chariot centre de masse du chariot.
- La rotation dans le sens des aiguilles d'une montre est direct.

LES RÉSULTATS MATHÉMATIQUES

LINEARISATION.M

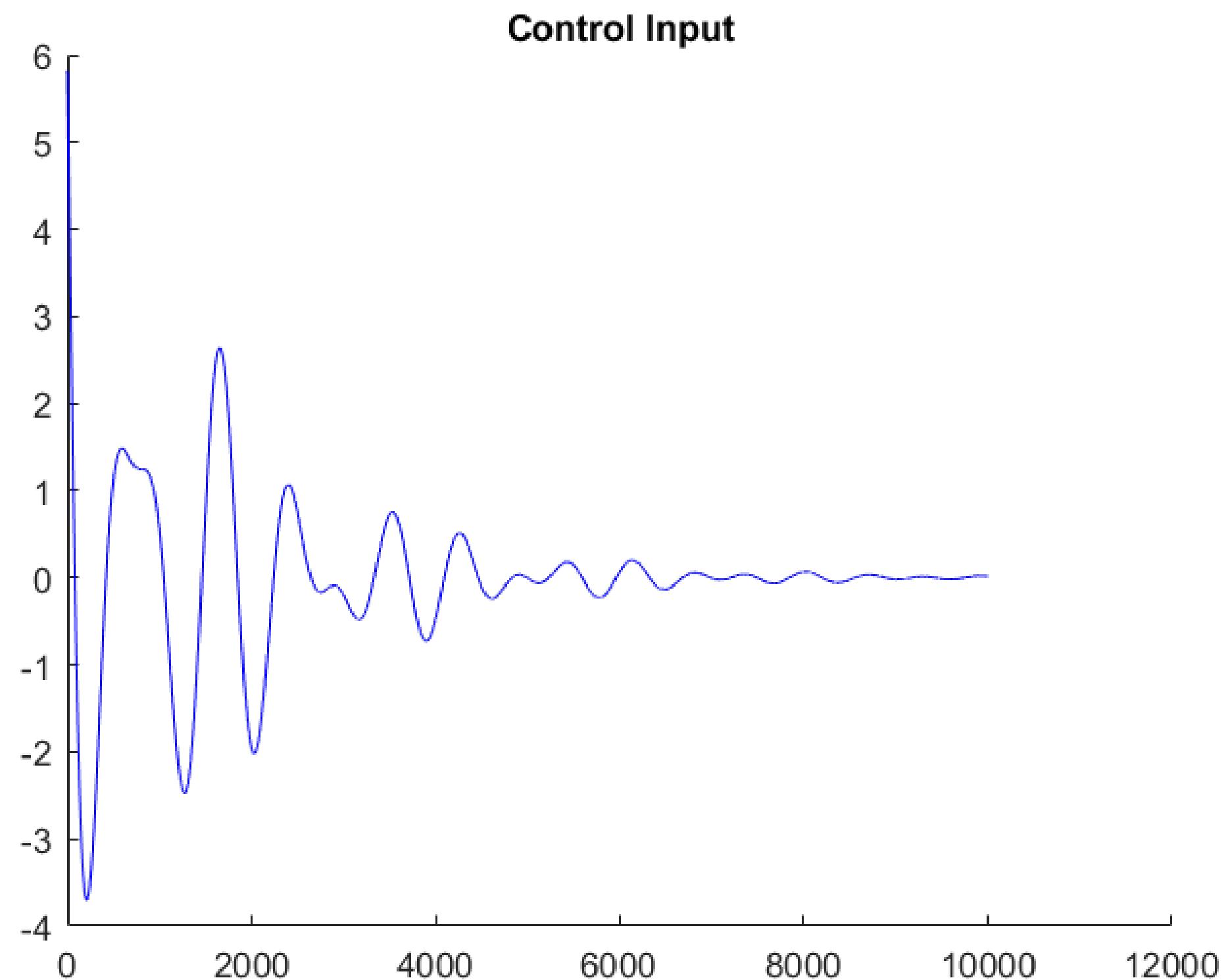
Workspace		Workspace		Workspace	
Name	Value	Name	Value	Name	Value
A_lin	6x6 double	Jacob_B	6x1 sym	~	1x1 sym
A_lin1	6x6 sym	Jacob_B_sub	6x1 sym	I1	1x1 sym
ans	[-0.7421 + 0.405...	K	[1.7291e+03,-84...	I2	1x1 sym
B_lin	[1.0000e-03;5.00...	k1	1x1 sym	M	1x1 sym
B_lin1	6x1 sym	k2	1x1 sym	m1	1x1 sym
check_rank	6	k3	1x1 sym	m2	1x1 sym
closed_loop	6x6 sym	k4	1x1 sym	Q_cost	6x6 double
controllability...	6x6 sym	k5	1x1 sym	R_cost	1.0000e-04
controllability...	6x6 double	k6	1x1 sym	s	6x6 double
D	1x1 sym	I1	1x1 sym	sim_time	100
e	[-0.7421 + 0.405...	I2	1x1 sym	t	100
F	1x1 sym	M	1x1 sym	test	6x6 double
f1	1x1 sym	m1	1x1 sym	time_step	0.0100
f2	1x1 sym	m2	1x1 sym	u	1x10001 double
f3	1x1 sym	Q_cost	6x6 double	x1	1x1 sym
f4	1x1 sym	R_cost	1.0000e-04	x2	1x1 sym
f5	1x1 sym	s	6x6 double	x3	1x1 sym
f6	1x1 sym	sim_time	100	x4	1x1 sym
g	1x1 sym	t	100	x5	1x1 sym
Jacob_A	6x6 sym	test	6x6 double	x6	1x1 sym
Jacob_A_sub	6x6 sym	time_step	0.0100	x_fin	6x10001 double
Jacob_B	6x1 sym	u	1x10001 double	x_fin_temp	[-2.2347e-06;1.6...
				x_init	[1.0000e-03;1.00...

LES RÉSULTATS MATHÉMATIQUES

NON_LINEAR_SOL.M

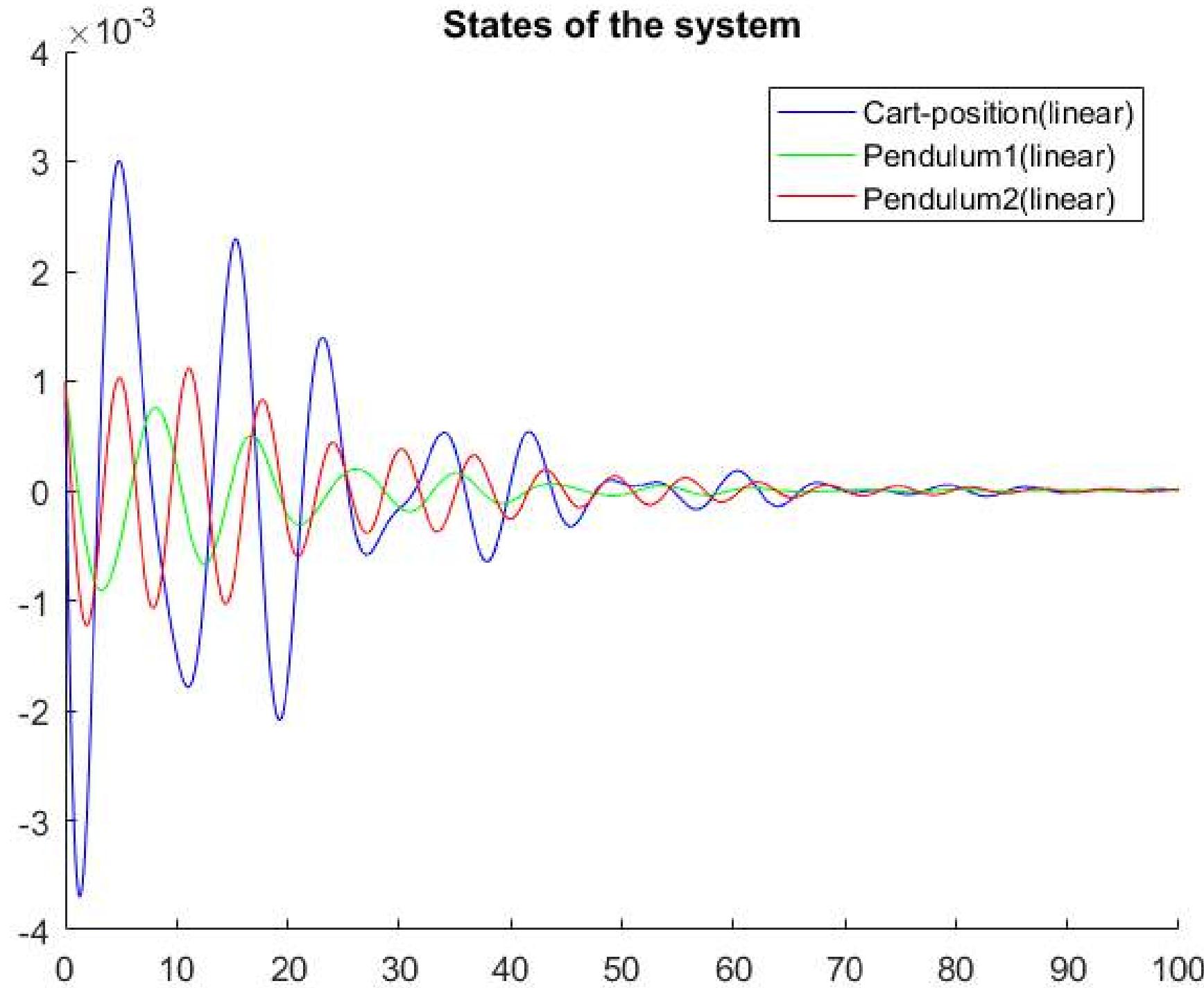
Workspace		Workspace		Workspace		Workspace	
Name	Value	Name	Value	Name	Value	Name	Value
A_lin	6x6 double	f1	1x1 sym	k3	1x1 sym	sim_time	100
A_lin1	6x6 sym	F1	6x1 sym	k4	1x1 sym	state_vec	1x1 symfun
ans	[-0.7421 + 0.405...	f1_sub	1x1 sym	k5	1x1 sym	t	353x1 double
B_lin	[1.0000e-03; 5.00...	f1_sub1_t	1x1 sym	k6	1x1 sym	test	6x6 double
B_lin1	6x1 sym	f2	1x1 sym	I1	1x1 sym	time_step	0.0100
check_rank	6	f2_sub	1x1 sym	I2	1x1 sym	u	1x10001 double
closed_loop	6x6 sym	f2_sub1_t	1x1 sym	M	1x1 sym	x1	1x1 sym
control_input	1x1 sym	f3	1x1 sym	m1	1x1 sym	x1t	1x1 symfun
controllability...	6x6 sym	f3_sub	1x1 sym	M1	6x6 sym	x2	1x1 sym
controllability...	6x6 double	f3_sub1_t	1x1 sym	m2	1x1 sym	x2t	1x1 symfun
D	1x1 sym	f4	1x1 sym	ode_fun	@(t,in2)[(in2(1,:))...	x3	1x1 sym
diff_eqn1	1x1 symfun	f5	1x1 sym	Q_cost	6x6 double	x3t	1x1 symfun
diff_eqn2	1x1 symfun	f6	1x1 sym	R_cost	1.0000e-04	x4	1x1 sym
diff_eqn3	1x1 symfun	g	1x1 sym	s	6x6 double	x4t	1x1 symfun
diff_eqn4	1x1 symfun	Jacob_A	6x6 sym	sim_time	100	x5	1x1 sym
diff_eqn5	1x1 symfun	Jacob_A_sub	6x6 sym	state_vec	1x1 symfun	x5t	1x1 symfun
diff_eqn6	1x1 symfun	Jacob_B	6x1 sym	t	353x1 double	x6	1x1 sym
e	[-0.7421 + 0.405...	Jacob_B_sub	6x1 sym	test	6x6 double	x6t	1x1 symfun
eqns	1x1 symfun	K	[1.7291e+03, -84...	time_step	0.0100	x_fin	6x10001 double
f	6x1 sym	k1	1x1 sym	u	1x10001 double	x_fin_temp	[-2.2347e-06; 1.6...
F	1x1 sym	k2	1x1 sym	x1	1x1 sym	x_init	[1.0000e-03; 1.00...
f1	1x1 sym	k3	1x1 sym	x1t	1x1 symfun	x_out	353x6 double

LES GRAPHES



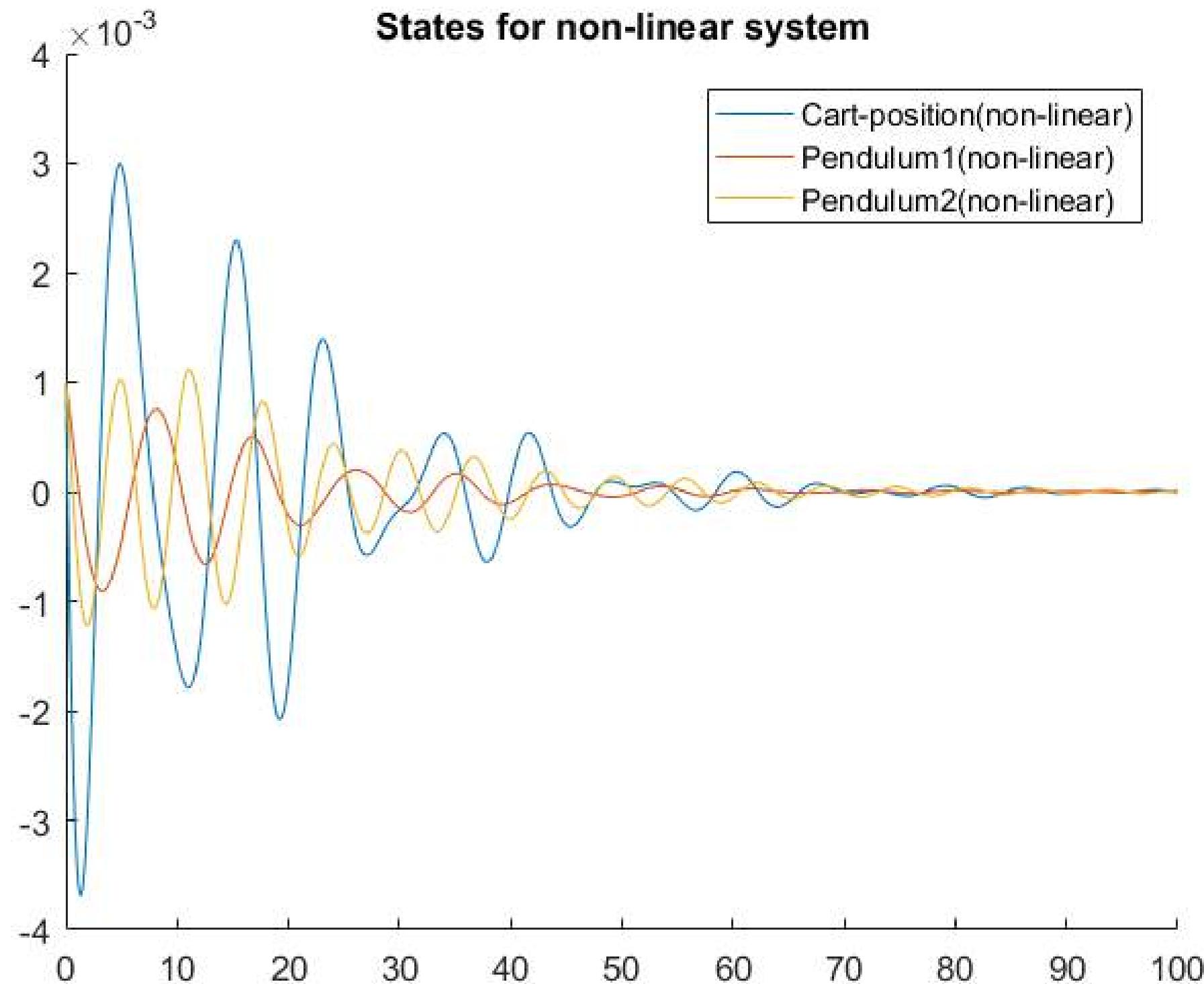
Voici l'affichage de la courbe de l'entrée de contrôle du système. On remarque qu'elle possède un comportement quasi-périodique et oscillatoire.

LES GRAPHES



Voici l'affichage de la courbe de
le statuts du système linéaire.
On remarque que les courbes des
pendules sont périodiques
oscillatoires avec une allures
sinusoïdales.

LES GRAPHES



Voici l'affichage de la courbe de
le statuts du système non-
linéaire.
On remarque que les courbes des
pendules possède les mêmes
caractéristiques que celle du
système linéaire.

MODÉLISATION DU SYSTÈME SUR LABVIEW

Référence(s):

- Quanser Linear Double Inverted Pendulum: <https://www.quanser.com/products/linear-double-inverted-pendulum/>
- Quanser - Linear Servo Control Systems Brochure: https://drive.google.com/file/d/1FyeQGBpf4AzLHVXN6yYvqwZmRYK_1f8l/view?usp=sharing

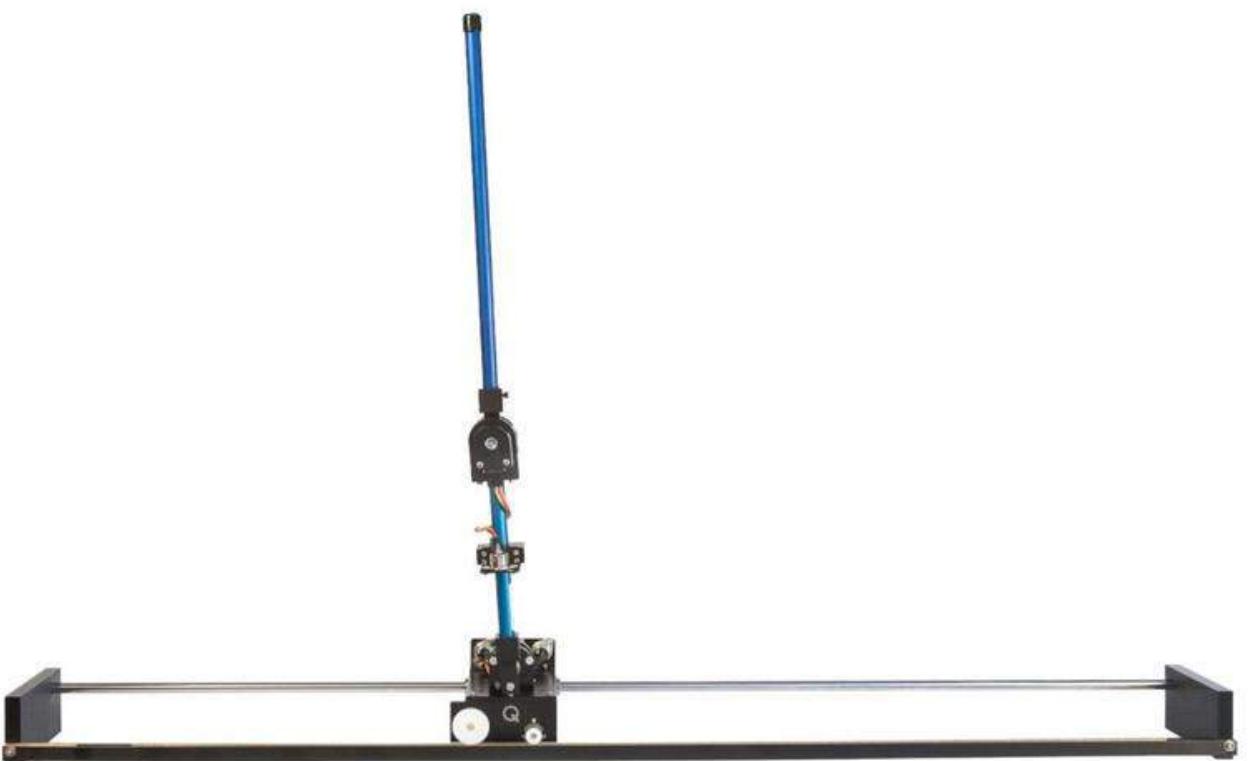
DOUBLE PENDULE LINÉAIRE INVERSÉ: POSTE DE TRAVAIL DE QUANSER

• DESCRIPTION DU SYSTÈME

- Le module du Double Pendule Inversé Linéaire se fixe à l'unité de base de servomoteur linéaire pour améliorer une expérience classique du pendule inversé.
- La conception d'un régulateur équilibrant deux liaisons ajoute un défi supplémentaire par rapport au système de pendule inversé simple. Les applications connexes de cette expérience incluent la stabilisation du décollage d'une fusée à étages multiples et la modélisation du système de posture humaine.

2. COMMENT ÇA FONCTIONNE

- Le module de Double Pendule Inversé se compose de deux tiges en aluminium bleu usinées avec précision : une de 20 cm et l'autre de 33.7 cm de longueur. Le module s'attache facilement à l'arbre du pendule avant sur le chariot de l'unité de base et peut pivoter librement à 360 degrés. L'angle du lien court est détecté à l'aide de l'encodeur de l'arbre du pendule du servomoteur linéaire, tandis que la longueur du lien moyen est mesurée à l'aide de l'encodeur situé au milieu du Double Pendule Inversé lui-même.
- En fonction de la position du chariot et des angles des pendules, le contrôle d'équilibre calcule une tension qui est appliquée au moteur du chariot. Le chariot se déplace d'avant en arrière pour équilibrer les deux pendules et maintenir une position verticale droite.



DOUBLE PENDULE LINÉAIRE INVERSÉ: POSTE DE TRAVAIL DE QUANSER

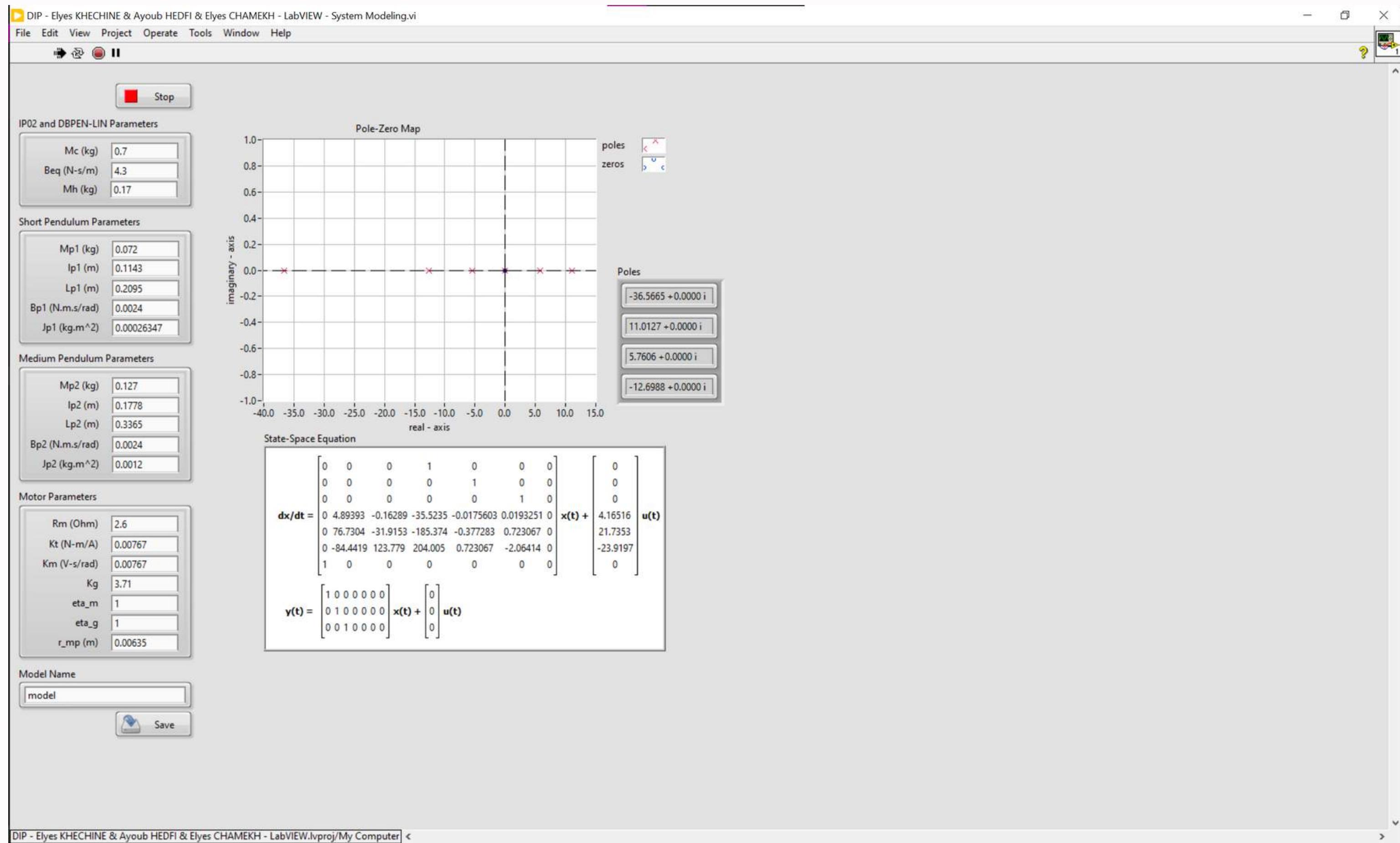
3. SPÉCIFICATIONS DE LA MAQUETTE

SPECIFICATION	VALUE	UNITS
Mass of linear double pendulum assembly	0.364	kg
Medium pendulum mass (with T-fitting)	0.127	kg
Medium pendulum length (pivot to tip)	33.7	cm
Short pendulum mass (with T-fitting)	0.097	kg
Short pendulum length (pivot to tip)	20.0	cm
Mass of encoder hinge	0.14	kg
Hinge encoder resolution (in quadrature)	4096	counts/rev

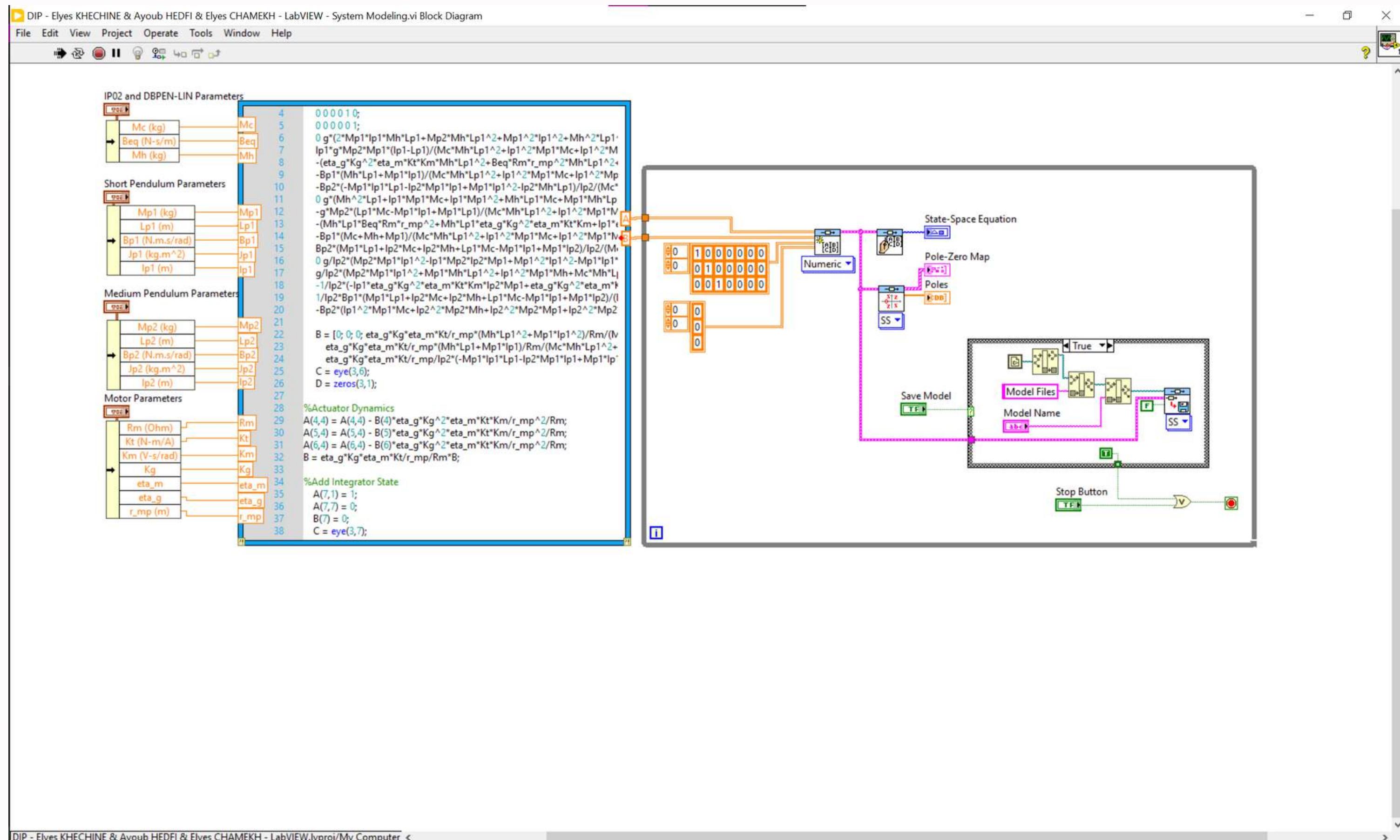
4. COMPOSANTS DU POSTE DE TRAVAIL

Component	Description
Plant	<ul style="list-style-type: none">• Linear Servo Base Unit (IP02)• Linear Double Inverted Pendulum module
Controller Design Environment ¹	<ul style="list-style-type: none">• Quanser QUARC® add-on for MATLAB®/Simulink®• Quanser Rapid Control Prototyping (RCP) Toolkit® add-on for NI LabVIEW™
Documentation ²	<ul style="list-style-type: none">• Laboratory Guide• User Manual• Quick Start Guide
Targets ¹	<ul style="list-style-type: none">• Microsoft Windows® or NI CompactRIO
Data Acquisition Board	<ul style="list-style-type: none">• Quanser Q8-USB, QPID/QPIDe, NI PCI/PCIe DAQ device or Quanser Q1-cRIO
Amplifier	<ul style="list-style-type: none">• Quanser VoltPAQ-X1
Others	<ul style="list-style-type: none">• Complete dynamic model• Simulink® pre-designed controllers• LabVIEW™ pre-designed controllers

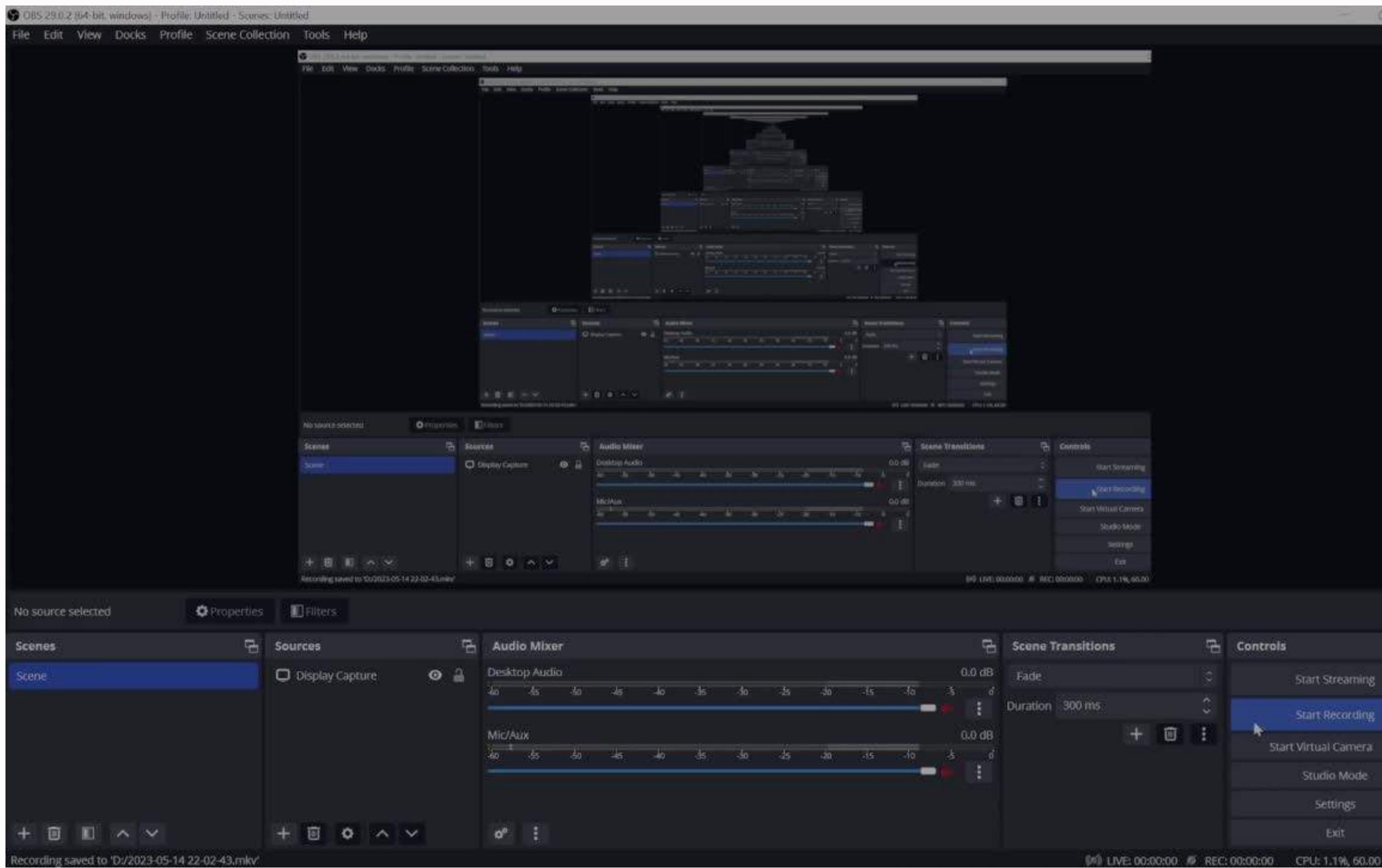
MODELISATION DU SYSTEME AVEC LABVIEW: FACE AVANT



MODELISATION DU SYSTEME AVEC LABVIEW: DIAGRAMME FONCTIONNEL

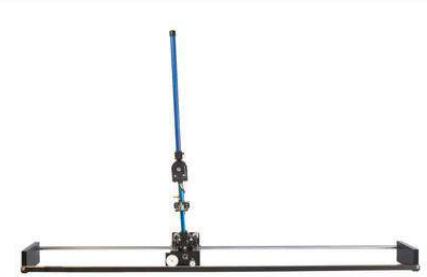


MODELISATION DU SYSTEME AVEC LABVIEW: VIDEO DÉMONSTRATIF



SIMULATIONS DU SYSTEME

Simulation #1:
Commande LQR du système DIP linéarisé de Quanser avec LabVIEW



Q QUANSER
INNOVATE • EDUCATE



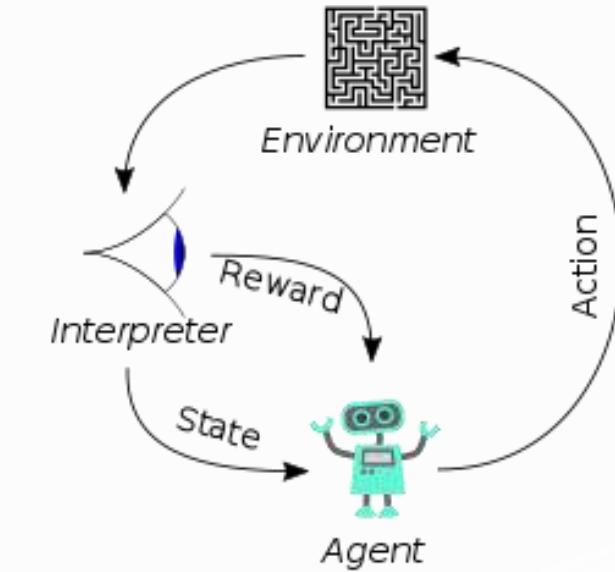
Dans cette section, nous présentons une simulation LabVIEW d'un régulateur LQR appliqué au système réel du double pendule inversé. Nous avons utilisé la maquette DIP de Quanser et le modèle décrit dans la partie modélisation. Grâce à cette simulation, nous avons pu étudier et analyser les performances du régulateur LQR dans la régulation du système. Cette approche expérimentale démontre notre capacité à appliquer les concepts théoriques à des systèmes réels.

Dans cette simulation, nous avons développé une stratégie de contrôle Swing-Up en utilisant Python avec les bibliothèques Gekko et Matplotlib. Nous avons créé un tableau de bord Web interactif permettant de visualiser et d'analyser les résultats en temps réel. Cette approche met en évidence notre capacité à concevoir une solution efficace et à fournir des informations visuelles claires sur le comportement du système.

Simulation #3:
Régulateur LQR pour un robot modélisant le système physique DIP dans ROS



Simulation #4 :
Apprentissage par renforcement - Recherche de stratégies de haute qualité pour le système DIP



Dans cette section, nous avons mis en œuvre un modèle d'apprentissage par renforcement (RL) afin de rechercher des stratégies de haute qualité pour le système à double pendule inversé. Notre objectif était d'exploiter les capacités de l'apprentissage par renforcement pour découvrir des stratégies efficaces et performantes pour la régulation du système DIP. Cette simulation démontre notre volonté d'explorer des approches innovantes et d'optimiser les performances du système.

SIMULATION #1:

COMMANDÉ LQR DU SYSTÈME DIP

LINÉARISÉ DE QUANSER AVEC LABVIEW

Référence(s):

- Quanser - Linear Double Inverted Pendulum - Laboratory Guide: <https://drive.google.com/file/d/1QVa5jelx7D5Yzznd775VXKbliXpe-aos/view?usp=sharing>

SYSTÈME DIP DE QUANSER: LINÉARISATION DU SYSTÈME

Les équations linéaires de l'espace d'état sont les suivantes:

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

où x est l'état, u est l'entrée de commande, A, B, C et D sont des matrices de l'espace d'état.

Pour le système linéaire à double pendule l'état et la sortie sont définis comme suit $x^T = [x_c \ \alpha \ \gamma \ \dot{x}_c \ \dot{\alpha} \ \dot{\gamma}]$

où $x_{c,d}$ est la position souhaitée du chariot, et $y^T = [x_1 \ x_2 \ x_3]$

Après avoir linéarisé les équations non linéaires du mouvement autour de la position d'angle zéro (équilibrée), et substitué les états données par la modélisation LabVIEW dans l'équation, nous obtenons les matrices espace-état présentées ici:

$$\begin{aligned} \frac{dx}{dt} &= \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 4.89393 & -0.16289 & -35.5235 & -0.0175603 & 0.0193251 & 0 \\ 0 & 76.7304 & -31.9153 & -185.374 & -0.377283 & 0.723067 & 0 \\ 0 & -84.4419 & 123.779 & 204.005 & 0.723067 & -2.06414 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 4.16516 \\ 21.7353 \\ -23.9197 \\ 0 \end{bmatrix} u(t) \\ y(t) &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} u(t) \end{aligned}$$

Remarque: Les vitesses des angles d'asservissement et de pendule peuvent être calculées dans le régulateur numérique, par exemple en prenant la dérivée et en filtrant le résultat par l'intermédiaire d'un filtre passe-haut.

SYSTÈME DIP LINÉARISÉ DE QUANSER: RÉGULATEUR LINÉAIRE-QUADRATIQUE (LQR)

Si (A, B) sont contrôlables, la méthode d'optimisation linéaire quadratique régulière peut être utilisée pour trouver un gain de contrôle de rétroaction. Étant donné le modèle d'état, il faut trouver une entrée de commande u qui minimise la fonction de coût:

$$J = \int_0^{\infty} x(t)'Qx(t) + u(t)'Ru(t) dt,$$

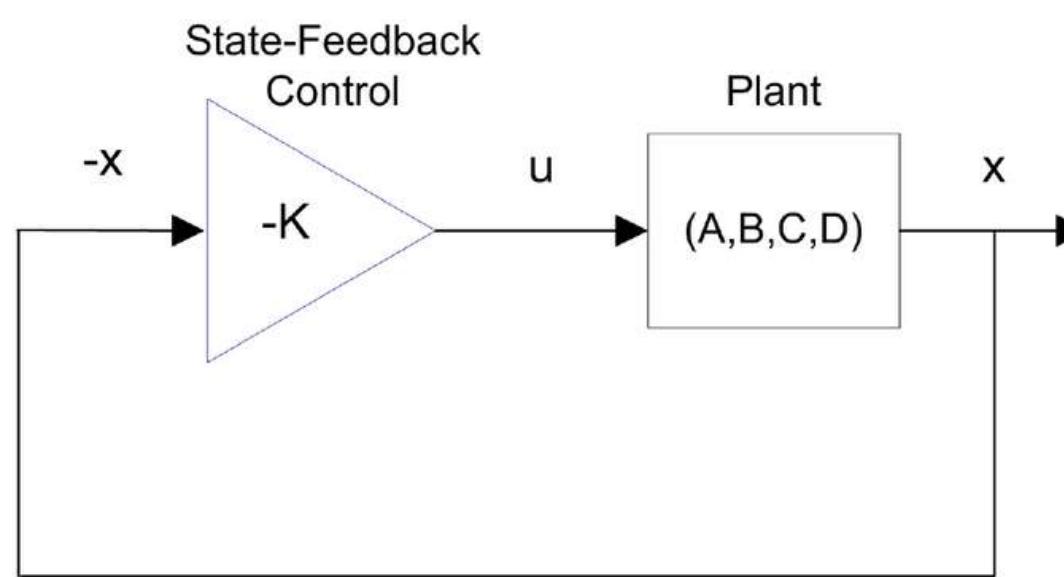
où Q et R sont les matrices de pondération. Les matrices de pondération affectent la façon dont LQR minimise la fonction et sont, essentiellement, des variables de réglage,

Compte tenu de la loi de commande $u = -Kx$, l'espace d'état devient:

$$\begin{aligned}\dot{x} &= Ax + B(-Kx) \\ &= (A - BK)x\end{aligned}$$

SYSTÈME DIP LINÉARISÉ DE QUANSER: CONTRÔLE PAR RÉTROACTION

La boucle de rétroaction de la figure suivante est conçue pour contrôler la position du chariot linéaire IP02 tout en équilibrant les pendules rigides et flexibles. L'état de référence est défini comme suit: $x_d = [x_{c,d} \ 0 \ 0 \ 0 \ 0 \ 0]$. Le régulateur est donc:



Lorsque $x_d = 0$, alors $u = -Kx$, qui est le régulateur utilisé dans l'algorithme LQR. Pour éliminer l'erreur de régulation du chariot linéaire, nous pouvons augmenter le système pour inclure un intégrateur tel que $\dot{\eta} = \begin{bmatrix} A & 0 \\ 1 & 0 \end{bmatrix} \eta + \begin{bmatrix} B \\ 0 \end{bmatrix} u$ où A et B sont les matrices de l'espace-état définies précédemment et où les états sont:

$$\eta^T = [x_c - x_{c,d} \ \alpha \ \theta \ \dot{x}_c \ \dot{\alpha} \ \dot{\theta} \ \int(x_c - x_{c,d}) dt]$$

Ceci introduit les termes d'intégration $\eta_7(t) = \int(x_c - x_{c,d}) dt$ dans le régulateur de rétroaction $u = -K(\eta)$ pour aider à compenser les dynamiques non modélisées dans le système réel qui font dériver le chariot de sa position souhaitée.

COMMANDÉ LQR DU SYSTÈME DIP LINÉARISÉ DE QUANSER AVEC LABVIEW: INTRODUCTION ET FONCTIONNALITÉS

- La simulation a été réalisée à l'aide d'un VI qui simule le contrôle en boucle fermée du système du double pendule linéaire. Le modèle linéaire du système, présenté dans précédemment, a été utilisé: gain de rétroaction K est déterminé à l'aide de la commande LQR du Control Design and Simulation Toolkit. L'objectif est de s'assurer que le gain utilisé permet de stabiliser avec succès le système (c'est-à-dire de maintenir les pendules en équilibre) sans saturer le moteur à courant continu.

Les étapes suivantes ont été suivies pour simuler le système :

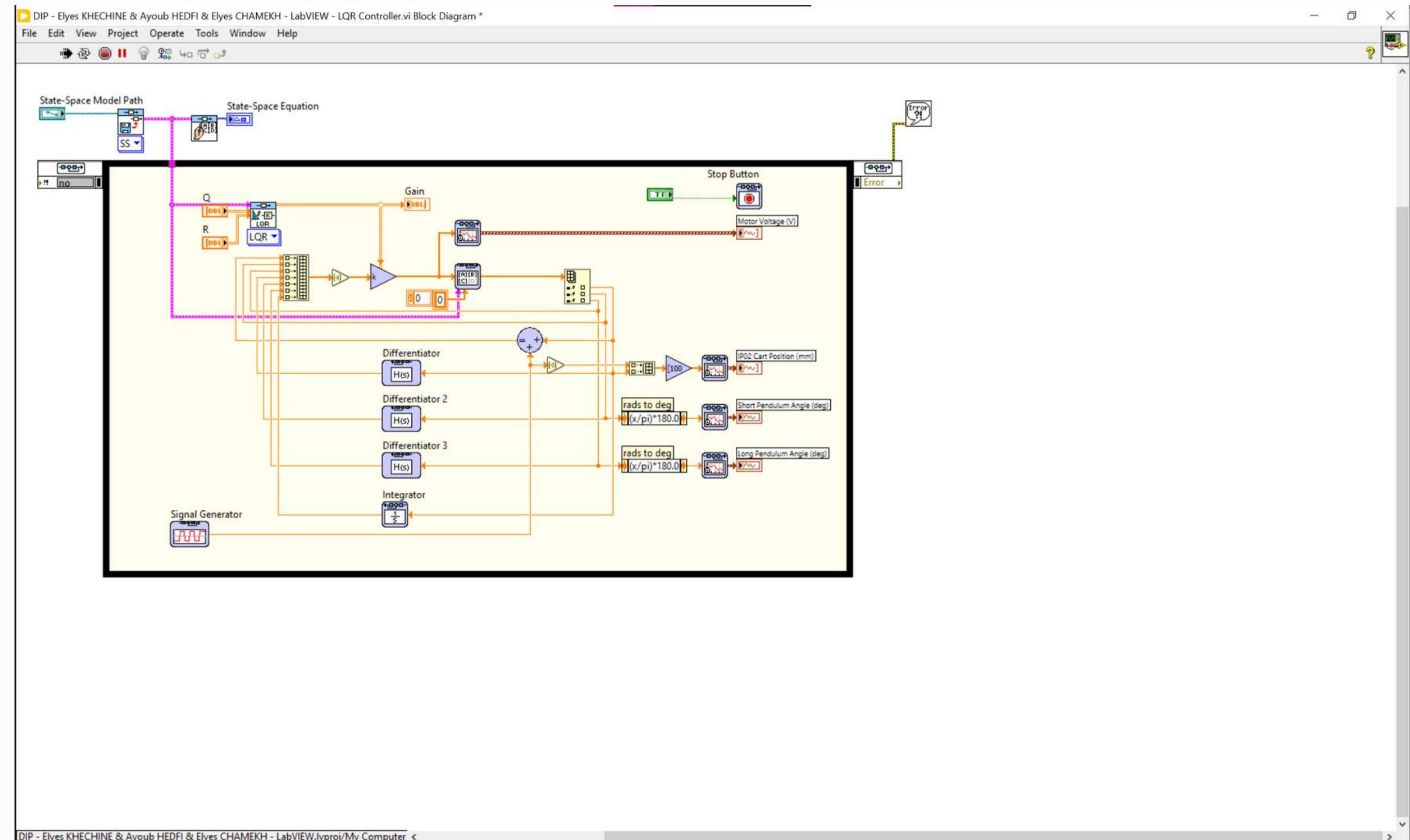
- Ouvrez et exécutez le VI "DIP - Elyes KHECHINE & Ayoub HEDFI & Elyes CHAMEKH - LabVIEW - LQR Controller.vi". S'assurez de choisir le fichier modèle en utilisant le contrôle "Model Path". Le fichier modèle est généré à l'aide du VI "DIP - Elyes KHECHINE & Ayoub HEDFI & Elyes CHAMEKH - LabVIEW - System Modeling.vi" en entrant le modèle d'espace d'état et en exportant le fichier résultant.
- Par défaut, la matrice Q est initialisée comme une matrice identité. On va définir les matrices de pondération LQR comme suit :

$$Q = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 50 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

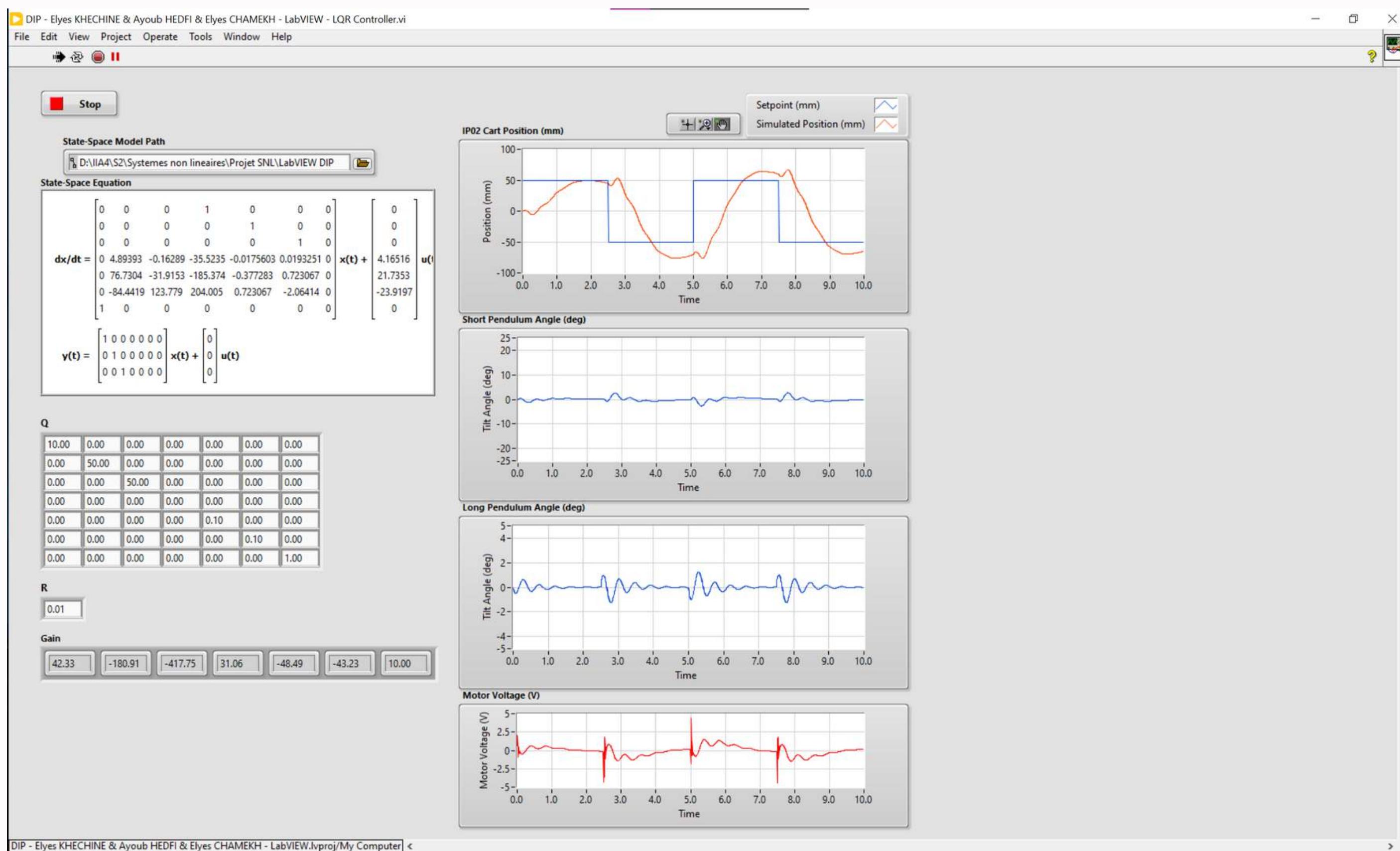
$$R = 0.01.$$

- Cela génère automatiquement le gain K.
- On exécute le VI. Les oscilloscopes affichent les réponses du système.
- On clique sur le bouton "STOP" pour arrêter l'exécution du VI.

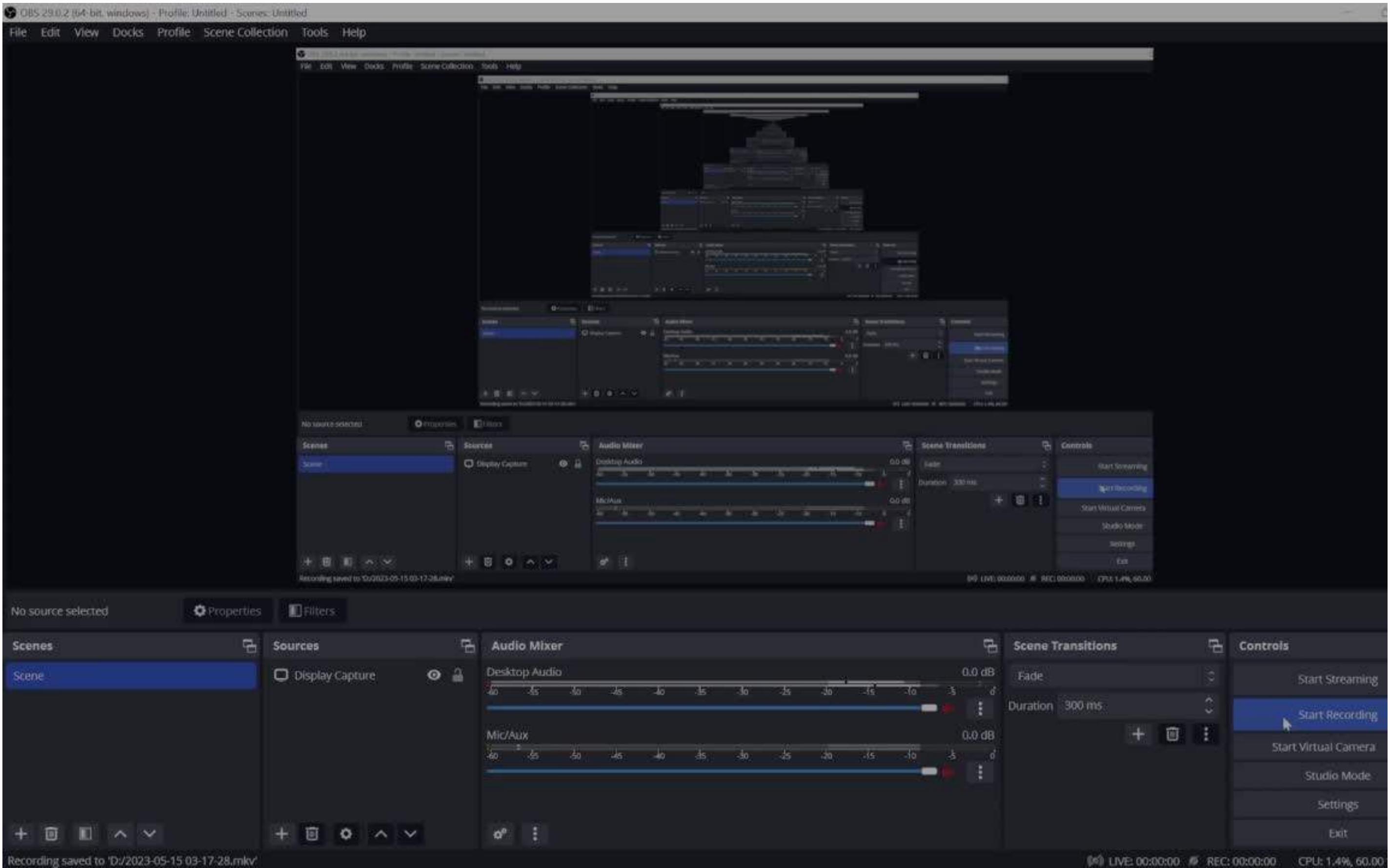
COMMANDÉ LQR DU SYSTÈME DIP LINÉARISÉ DE QUANSER AVEC LABVIEW: DIAGRAMME FONCTIONNEL



COMMANDÉ LQR DU SYSTÈME DIP LINÉARISÉ DE QUANSER AVEC LABVIEW: FACE AVANT (APRÈS SIMULATION)



COMMANDÉ LQR DU SYSTÈME DIP LINÉARISÉ DE QUANSER AVEC LABVIEW: VIDÉO DÉMONSTRATIF



ANALYSE DES RÉSULTATS, AMÉLIORATIONS POSSIBLES ET CONCLUSION

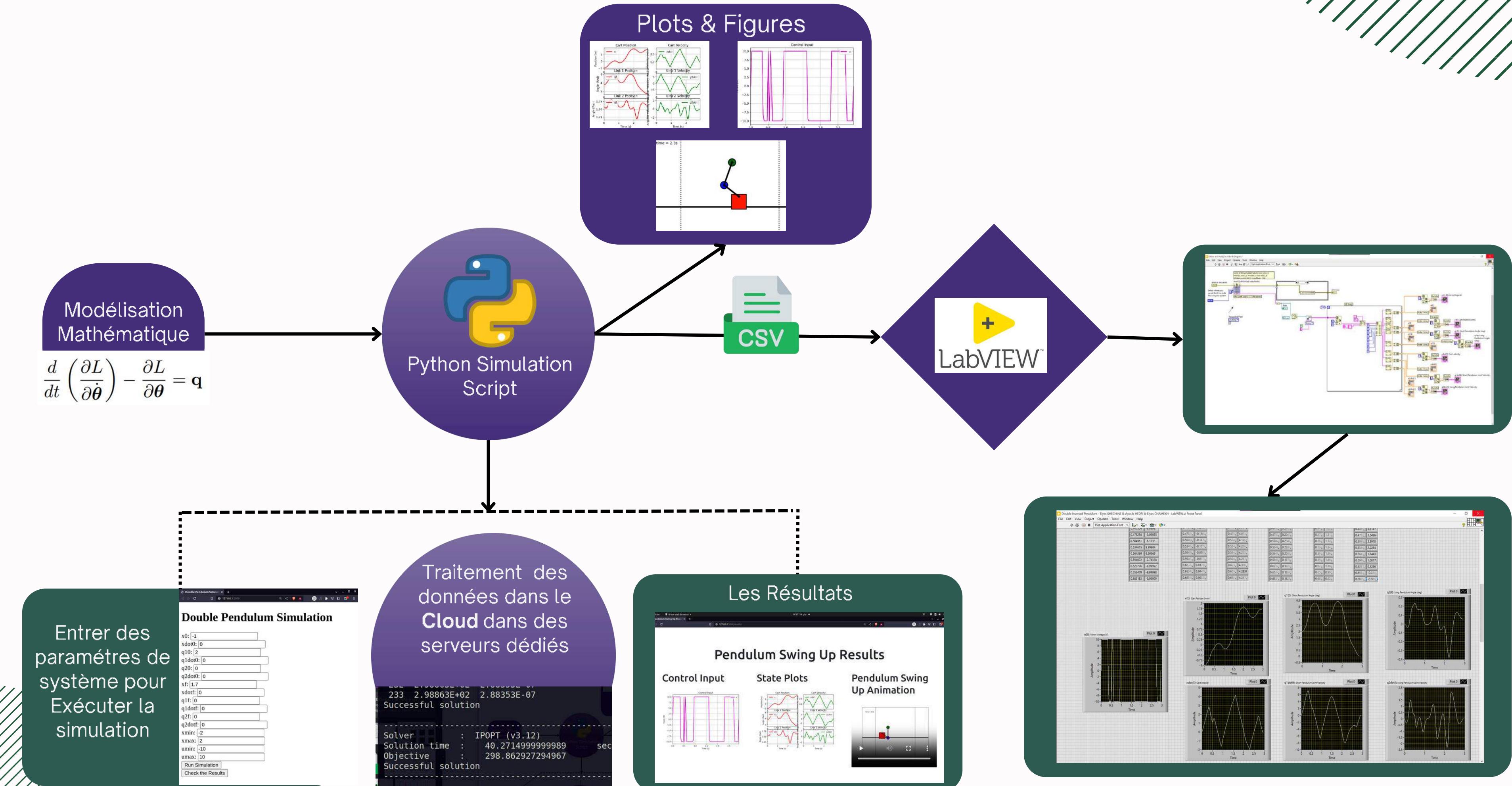
- Comme le montre la réponse de la simulation présentée dans les figures, les pendules maintiennent leurs positions d'équilibre malgré une perturbation de position du chariot de 50 mm. Cette simulation démontre l'efficacité du régulateur LQR dans la régulation du système du double pendule inversé. Les pendules parviennent à maintenir leurs positions d'équilibre malgré les perturbations, ce qui confirme l'efficacité de la commande en boucle fermée pour assurer la stabilité du système. De plus, la réponse du système est rapide, ce qui indique une bonne performance du régulateur LQR.
- Des améliorations possibles peuvent être envisagées pour cette simulation. Tout d'abord, il serait intéressant d'étudier l'impact de différentes matrices de pondération dans la commande LQR afin d'optimiser les performances du système. Cela pourrait inclure des expérimentations avec des valeurs différentes dans la matrice Q pour ajuster les gains de pondération en fonction des objectifs spécifiques du système.
- En outre, il serait utile d'ajouter des fonctionnalités supplémentaires à la simulation, telles que la possibilité de visualiser et d'analyser les performances à l'aide de différentes métriques telles que le temps de montée, le temps de stabilisation, etc. Cela permettrait une évaluation plus détaillée du régulateur et faciliterait l'optimisation des paramètres.
- En conclusion, la simulation présentée démontre l'efficacité du régulateur LQR dans la régulation du système du double pendule inversé. Les pendules parviennent à maintenir leurs positions d'équilibre malgré les perturbations, ce qui atteste de la capacité du régulateur à assurer la stabilité du système. Des améliorations futures peuvent être apportées pour optimiser les performances du régulateur et faciliter l'analyse détaillée du système. Cette simulation constitue une base solide pour l'étude et le développement de stratégies de contrôle plus avancées pour des systèmes similaires.

SIMULATION #2:
**SWING-UP BASÉE SUR PYTHON UTILISANT GEKKO
ET MATPLOTLIB, PRÉSENTÉE SUR UN DASHBOARD
WEB INTERACTIF**

SWING-UP BASÉE SUR PYTHON: INTRODUCTION ET FONCTIONNALITÉS

- Dans cette simulation, nous avons développé une stratégie de contrôle Swing-Up pour un double pendule inversé en utilisant Python avec les bibliothèques Gekko et Matplotlib. La simulation permet de trouver une solution efficace pour équilibrer le double pendule en partant d'une position initiale donnée et en atteignant une position finale souhaitée.
- Une caractéristique intéressante de la simulation est la possibilité pour l'utilisateur de choisir d'exécuter la simulation sur des serveurs cloud ou sur l'ordinateur local. L'option "remote" dans la fonction Gekko du script Python permet de sélectionner cette option. L'exécution sur les serveurs cloud peut être avantageuse en termes de rapidité d'exécution, ce qui peut être particulièrement utile pour les ordinateurs lents. Cependant, cela peut dépendre de la qualité de la connexion Internet de l'utilisateur.
- L'utilisateur a également la possibilité de définir les paramètres initiaux du système à partir du tableau de bord Web. Les paramètres tels que la position et la vitesse du chariot, la position et la vitesse des deux maillons, ainsi que les limites des conditions initiales et finales peuvent être ajustés en fonction des besoins de l'utilisateur.
- Une fois que la simulation est lancée avec les paramètres choisis, les calculs sont effectués par le script Python pour trouver une solution réussie. Lorsque l'utilisateur clique sur le bouton "Vérifier les résultats" dans le tableau de bord, il est dirigé vers une page affichant plusieurs graphiques générés par Matplotlib. Ces graphiques comprennent l'entrée de contrôle, la position du chariot, sa vitesse, la position des maillons 1 et 2, leur vitesse, ainsi qu'une vidéo illustrant la simulation de l'expérience de swing-up du système à double pendule inversé.

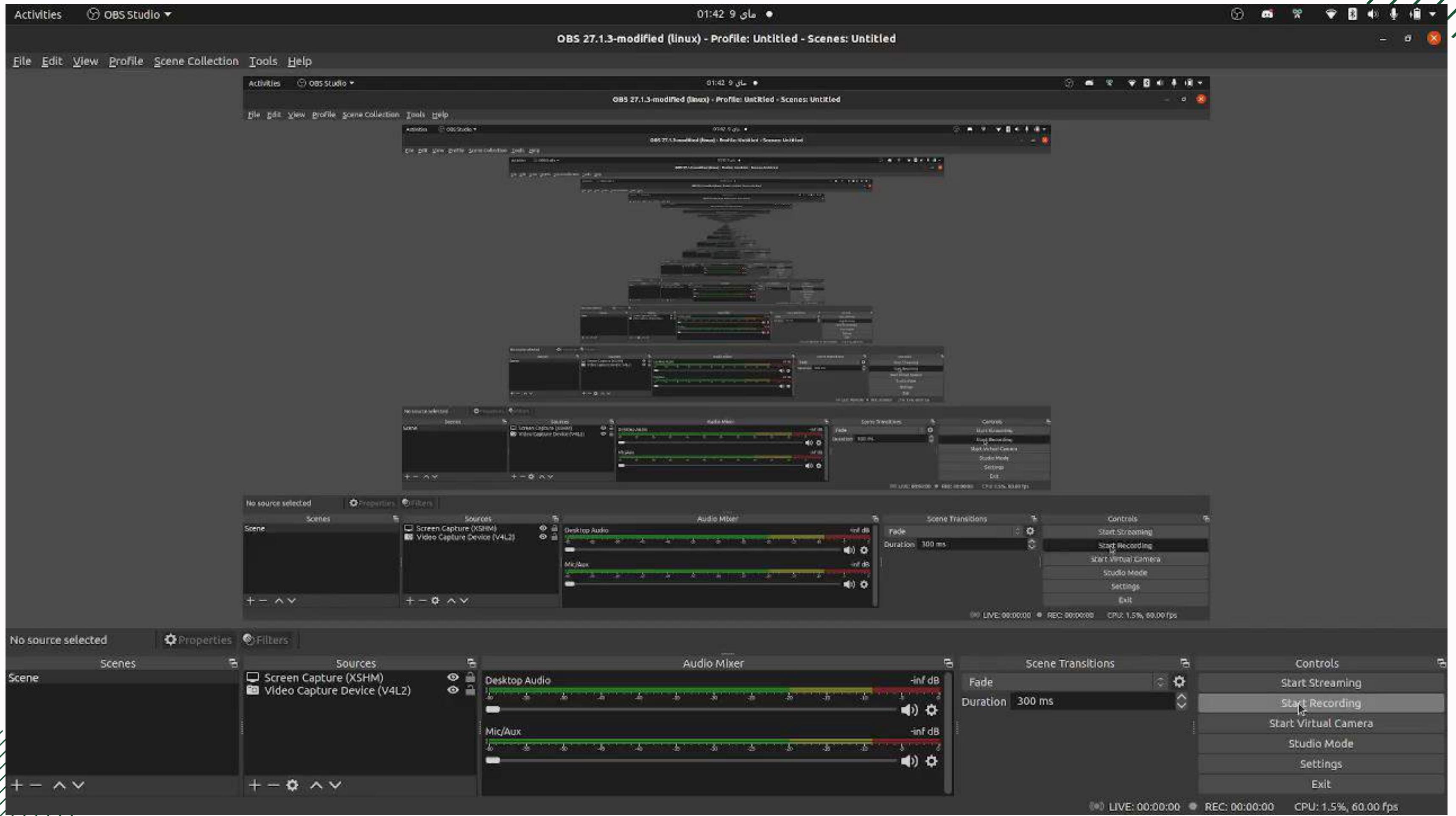
DIAGRAMME & PROCESS DE LA SIMULATION



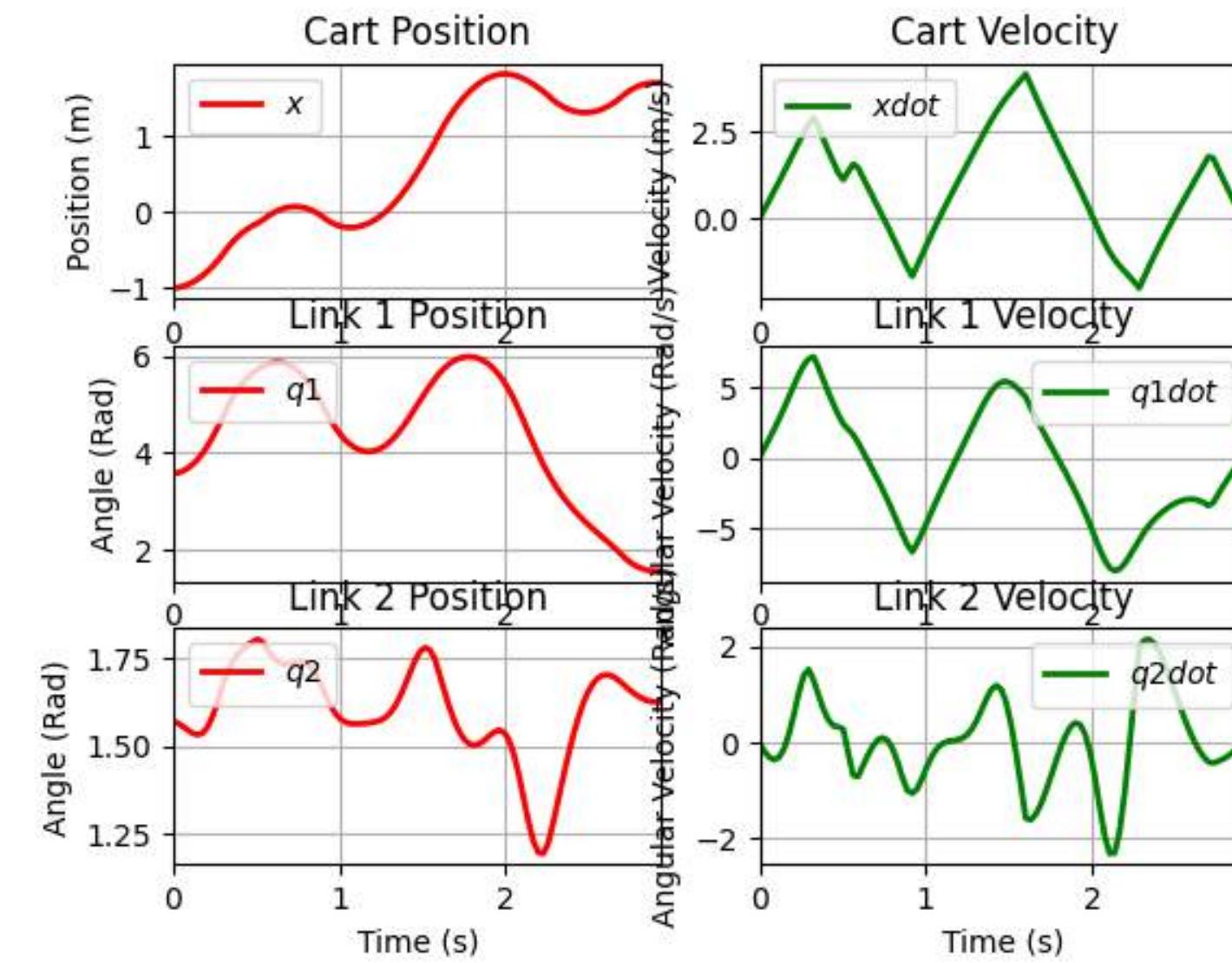
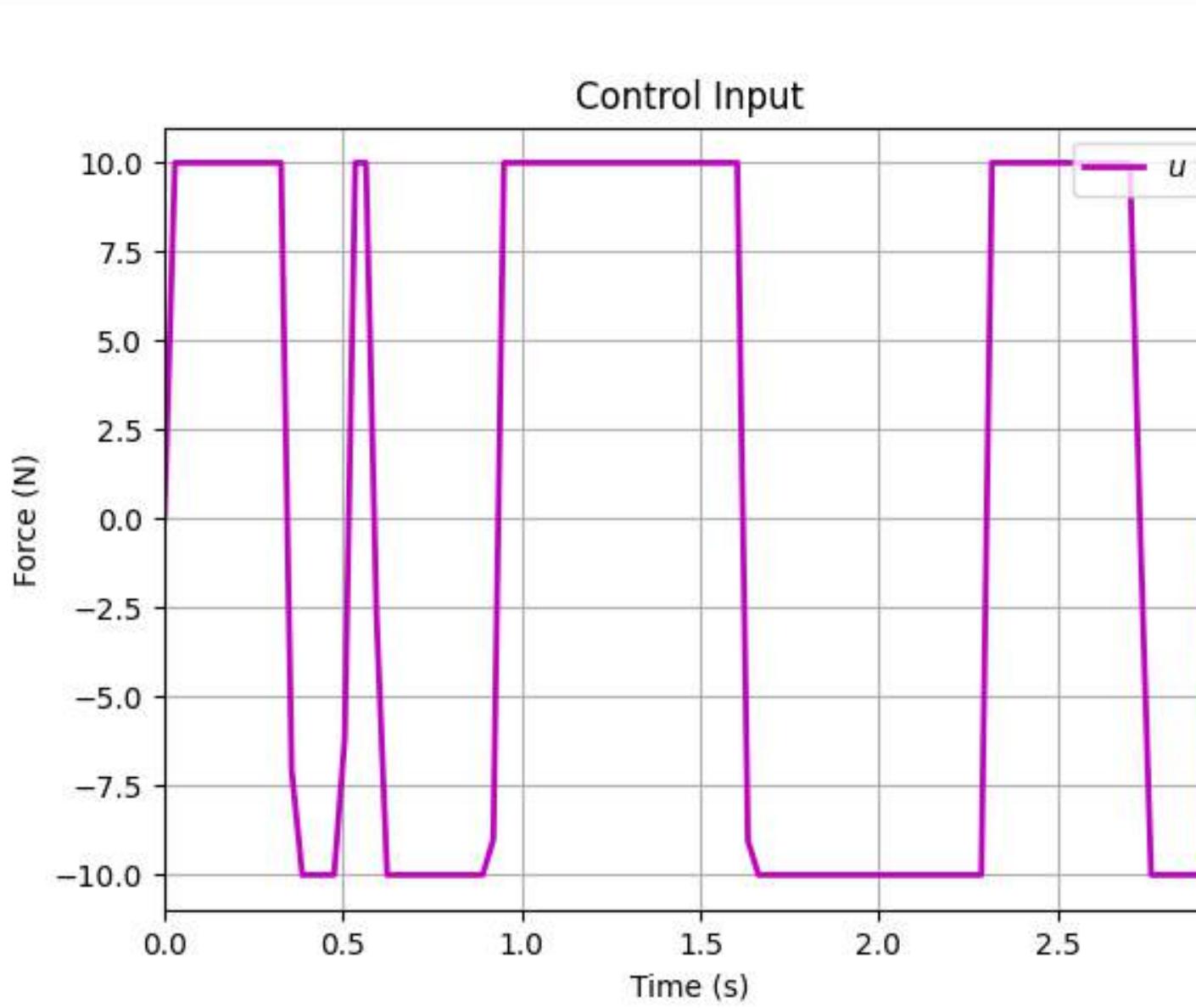
**VIDÉO DÉMONSTRATIF DE
LA SIMULATION**



PYTHON SIMULATION



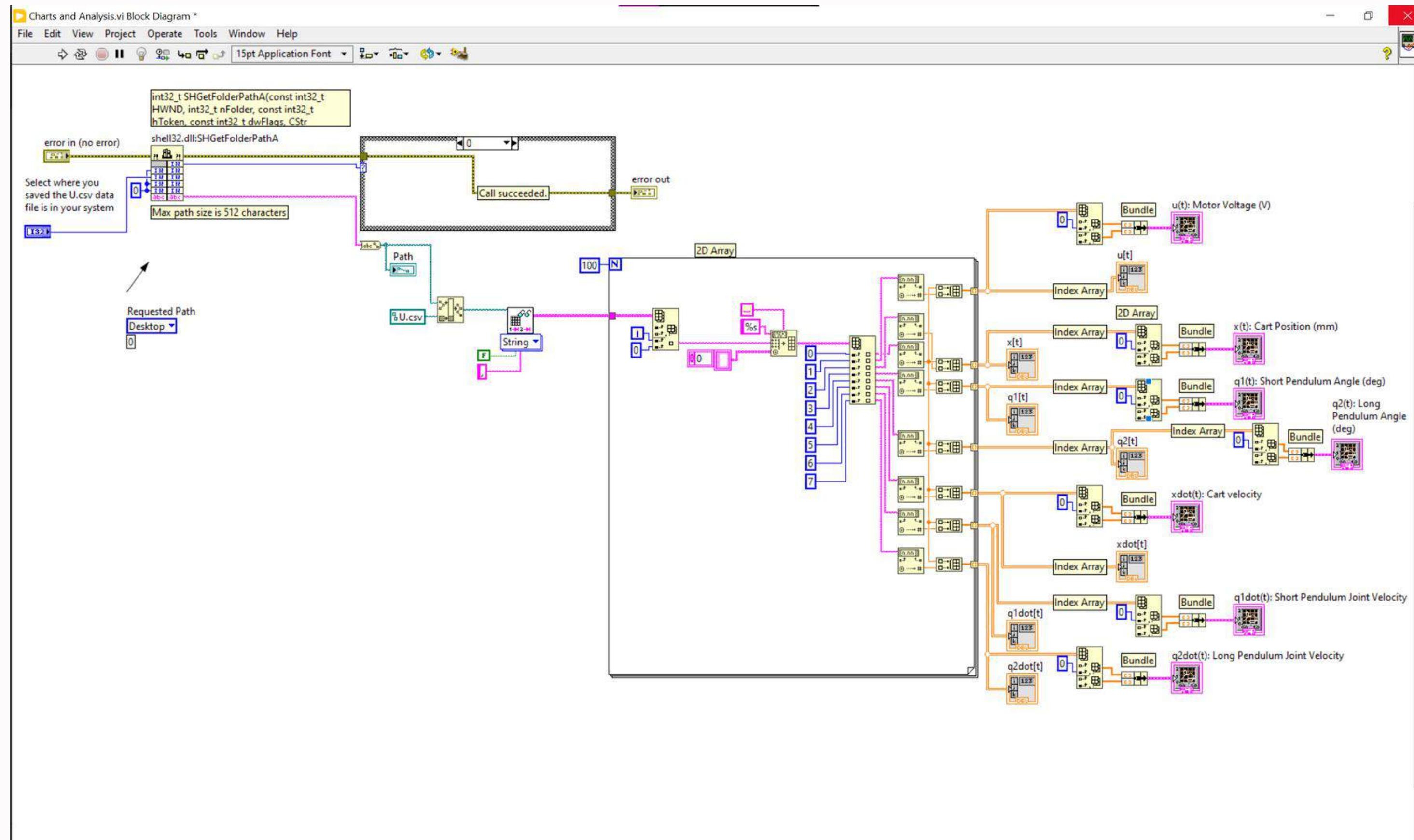
SWING-UP BASÉE SUR PYTHON: FIGURES GÉNÉRÉES PAR LA SIMULATION



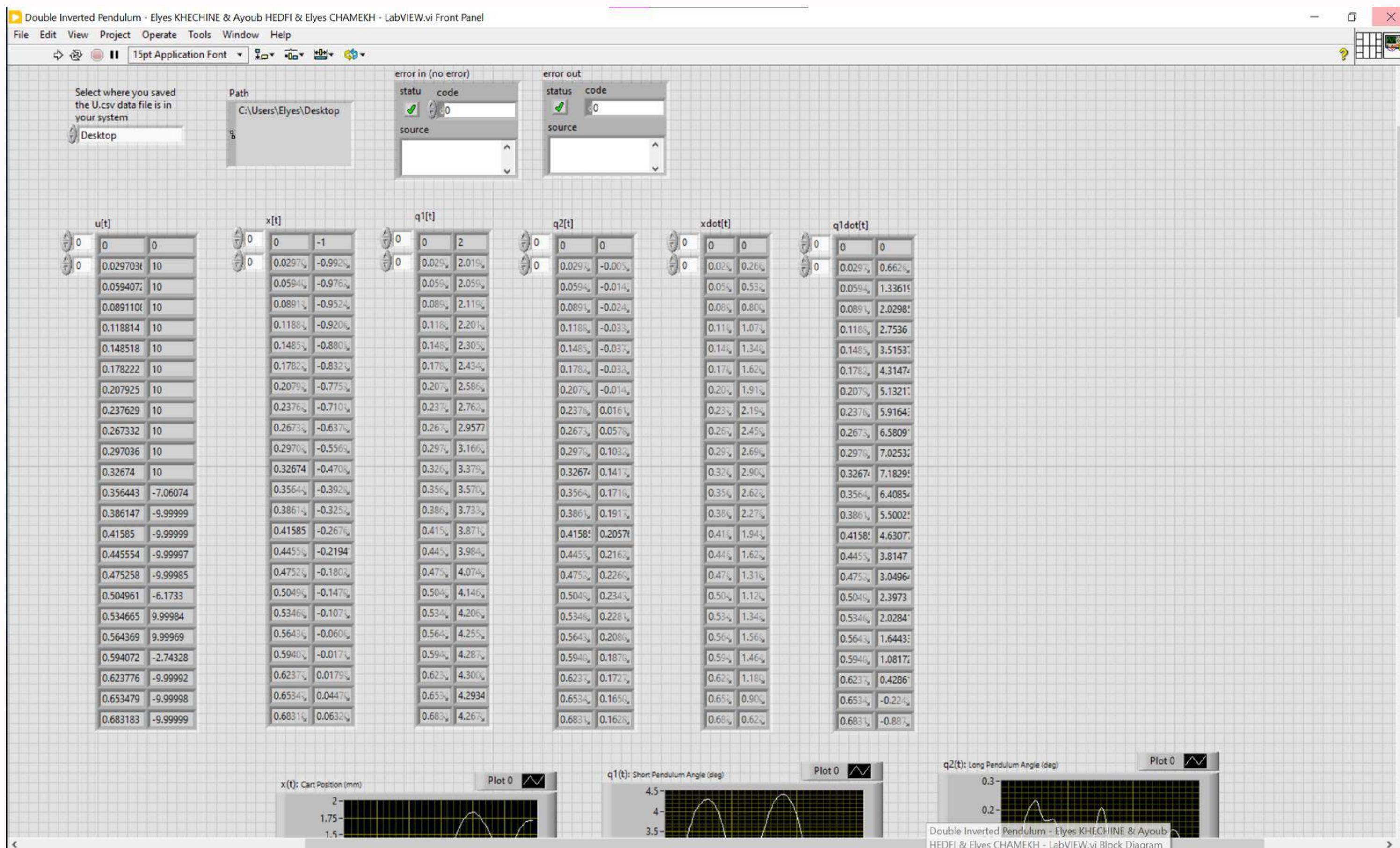
ANALYSE DES RÉSULTATS, AMELIORATIONS POSSIBLES ET CONCLUSION

- En analysant les résultats de la simulation, nous observons que le chariot du double pendule inversé part de la position -1 et se déplace vers la droite jusqu'à atteindre la position +1. Pendant ce temps, le premier maillon descend de la position basse ($+\pi$ rad) à la position 0, tandis que le deuxième maillon se redresse de la position 0 à la position verticale.
- Les graphiques générés montrent que le contrôle de l'entrée varie au fil du temps pour équilibrer le système. La position et la vitesse du chariot, ainsi que la position et la vitesse des maillons 1 et 2, évoluent également pour atteindre une configuration d'équilibre stable. Ces résultats mettent en évidence l'efficacité de la stratégie de contrôle Swing-Up mise en œuvre dans la simulation.
- Pour des améliorations futures, il serait intéressant d'ajouter des fonctionnalités supplémentaires au tableau de bord Web, telles que la possibilité de modifier les paramètres en cours d'exécution et d'explorer différentes configurations initiales et finales.
- En conclusion, la simulation développée avec Python, Gekko et Matplotlib offre une approche pratique et flexible pour résoudre le problème de l'équilibrage du double pendule inversé. Elle met en évidence l'intégration des bibliothèques Python Gekko et Matplotlib, qui fournissent des fonctionnalités avancées pour la modélisation et la visualisation des systèmes dynamiques. Cette simulation peut servir d'outil d'apprentissage, de recherche et de développement de nouvelles stratégies de contrôle pour des systèmes similaires.

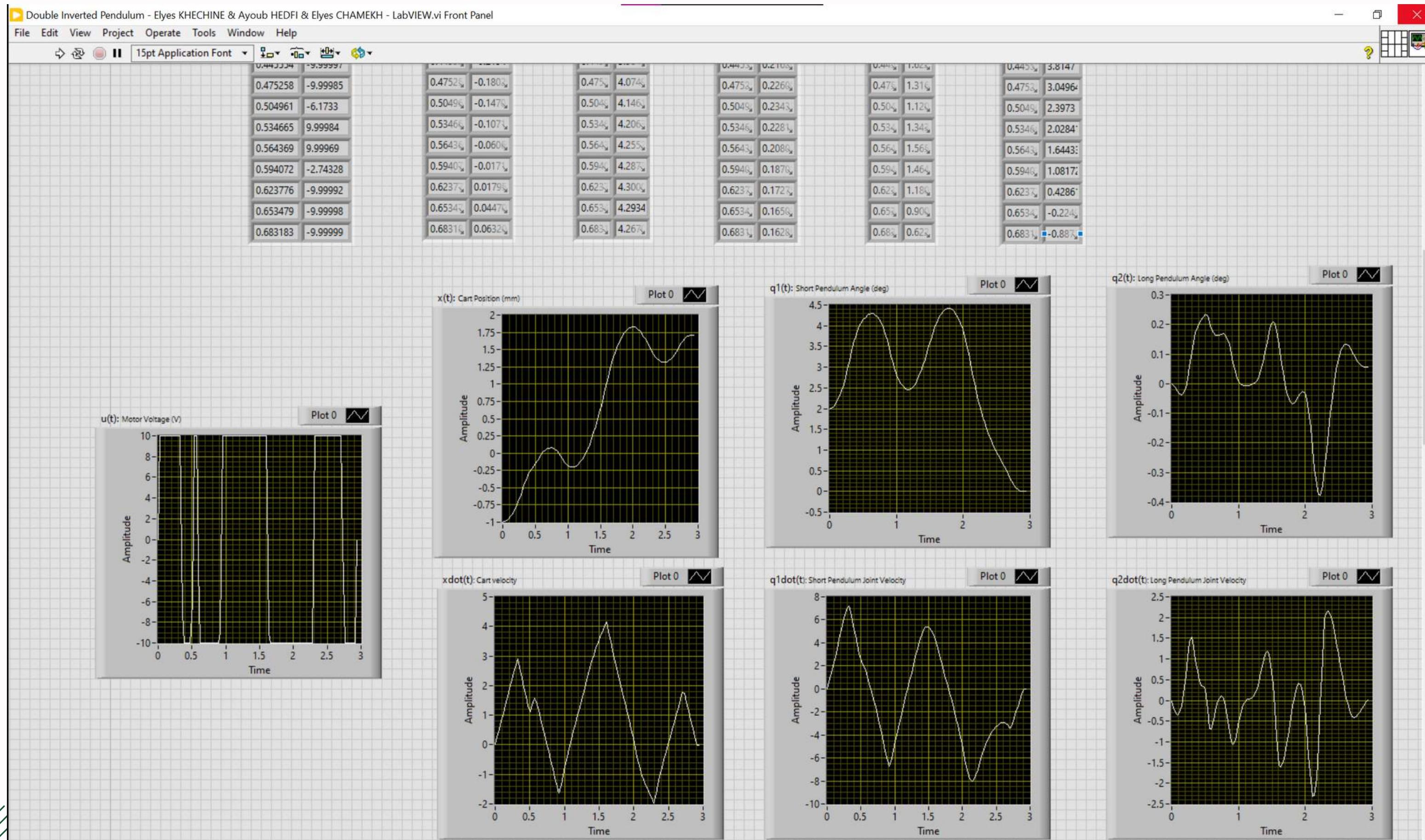
LIEN ENTRE LA SIMULATION PYTHON & LABVIEW: DIAGRAMME FONCTIONNEL



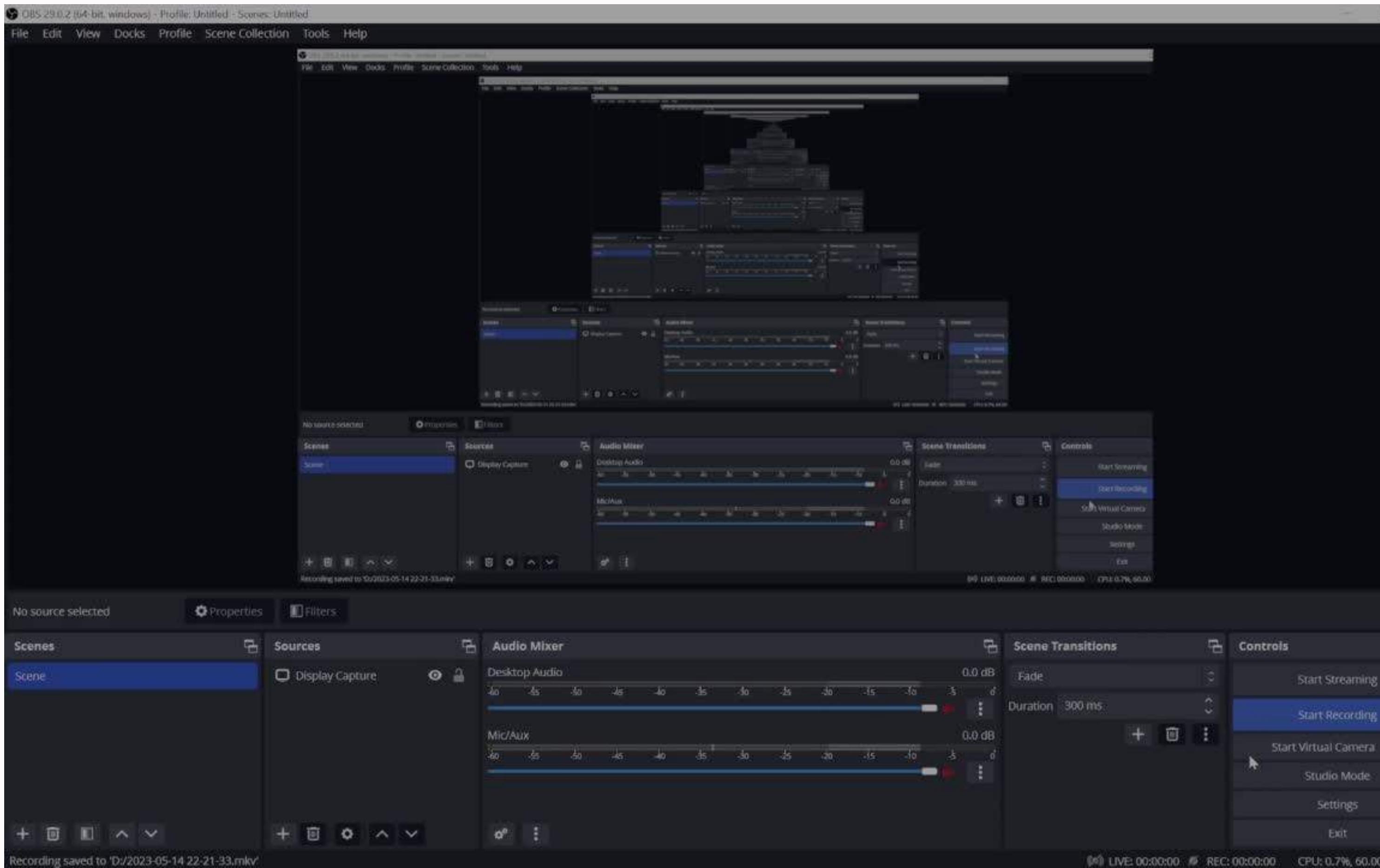
LIEN ENTRE LA SIMULATION PYTHON & LABVIEW: FACE AVANT (1)



LIEN ENTRE LA SIMULATION PYTHON & LABVIEW: FACE AVANT (2)



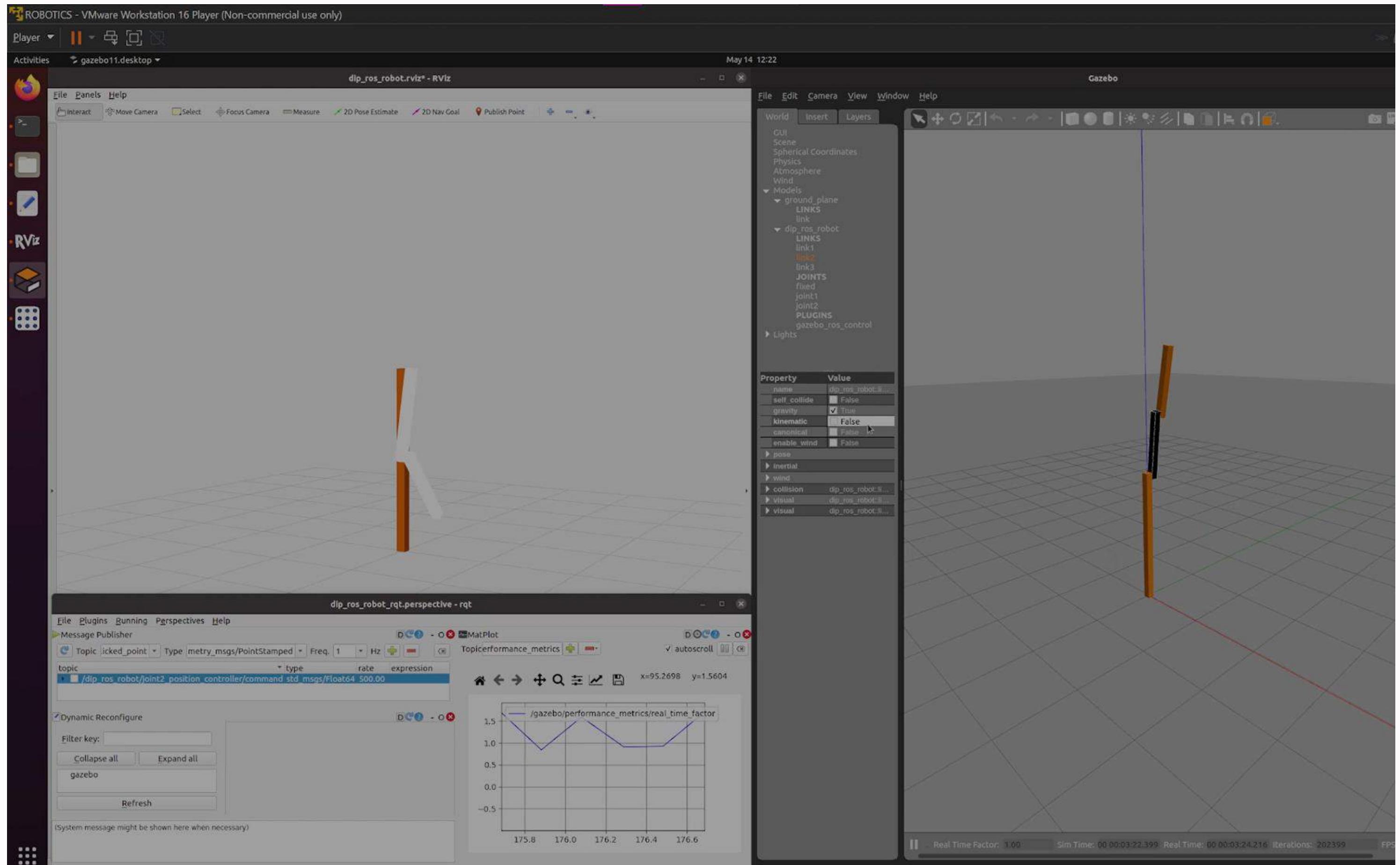
LIEN ENTRE LA SIMULATION PYTHON & LABVIEW: VIDÉO DÉMONSTRATIF



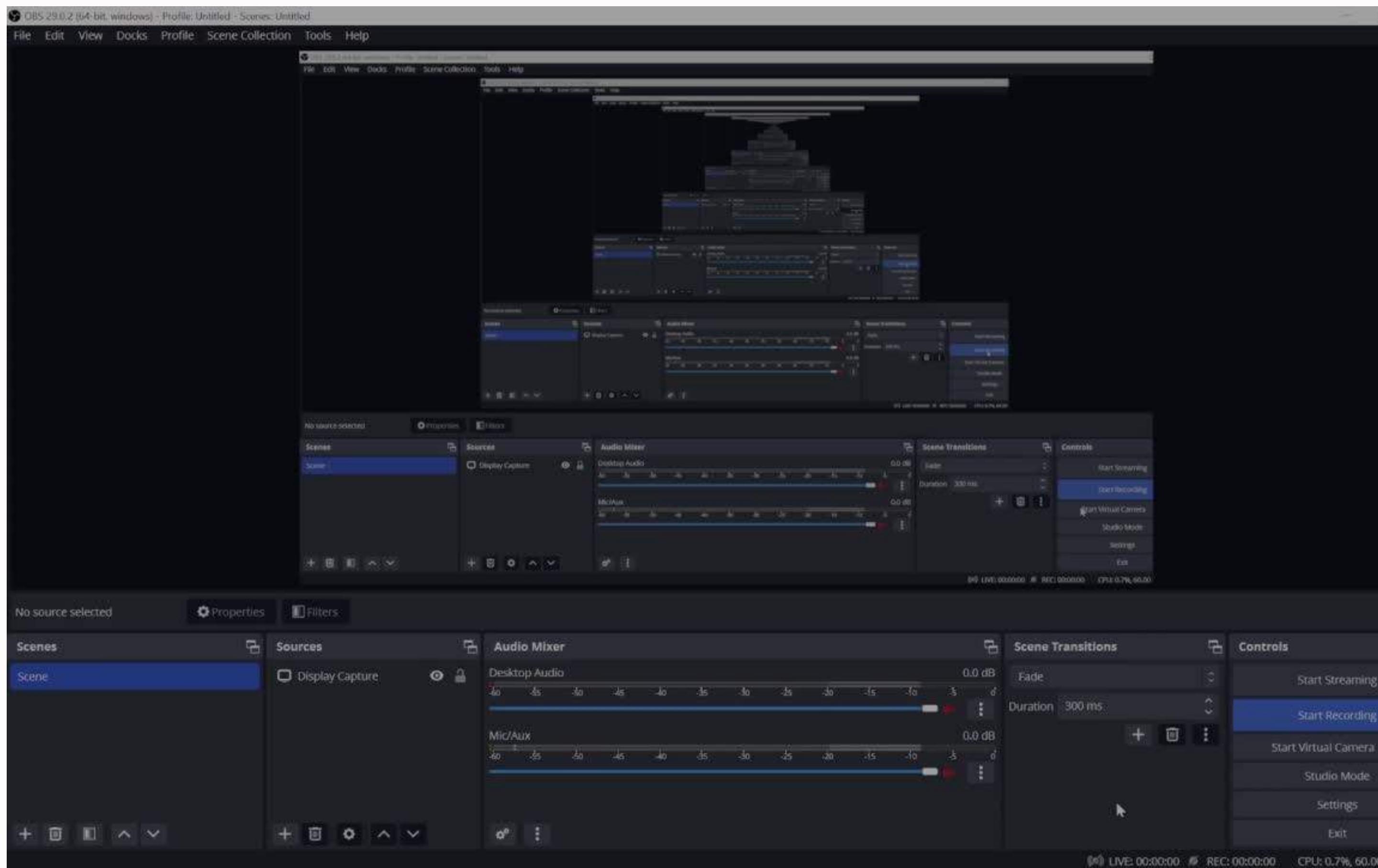
SIMULATION #3:

RÉGULATEUR LQR POUR UN ROBOT MODÉLISANT LE SYSTÈME PHYSIQUE DIP DANS ROS

RÉGULATEUR LQR POUR UN ROBOT MODÉLISANT LE SYSTÈME PHYSIQUE DIP DANS ROS: SIMULATION ROS, RVIZ & GAZEBO



RÉGULATEUR LQR POUR UN ROBOT MODÉLISANT LE SYSTÈME PHYSIQUE DIP DANS ROS: VIDÉO DÉMONSTRATIF



ANALYSE DES RÉSULTATS, AMÉLIORATIONS POSSIBLES ET CONCLUSION

- Lors de la simulation dans Gazebo, les maillons supérieurs du robot oscillent et ne parviennent pas à se stabiliser en position de balancement malgré l'utilisation du régulateur LQR. La cause de cette difficulté peut être liée à des paramètres mal réglés, des erreurs de modélisation ou une précision insuffisante du régulateur. Néanmoins, la plateforme de simulation offre une flexibilité considérable pour choisir l'environnement et les paramètres physiques des articulations, et la métrique de performance gazebo real_time_factor permet d'évaluer la stabilité du système en montrant ses oscillations.
- Malgré les difficultés rencontrées, cette simulation a été précieuse pour évaluer le comportement du système dans un environnement virtuel réaliste et mettre en évidence les défis associés à la régulation d'un système instable. L'incapacité à stabiliser les maillons supérieurs en position de balancement souligne la nécessité d'améliorer le régulateur LQR et d'optimiser les réglages du modèle pour obtenir de meilleures performances.
- Plusieurs axes d'amélioration peuvent être envisagés pour améliorer les résultats de la simulation. Il serait pertinent d'analyser en détail les paramètres du régulateur LQR, d'ajuster les gains, les limites et les poids des différentes composantes du système. Une étude approfondie de la modélisation physique et de l'interaction des articulations permettrait d'obtenir une représentation plus précise du système réel. L'utilisation de techniques de contrôle avancées telles que le modèle prédictif, l'apprentissage par renforcement ou l'optimisation numérique pourrait également ouvrir de nouvelles perspectives pour la stabilisation du double pendule inversé sur un chariot.
- En conclusion, cette simulation physique d'un régulateur LQR pour le système du double pendule inversé sur un chariot, réalisée dans le cadre de ROS et Gazebo, met en évidence les défis auxquels on peut être confronté lors de la régulation d'un système instable. Bien que la stabilisation en position de balancement n'ait pas été atteinte, cette expérience permet d'évaluer le comportement du système dans un environnement virtuel réaliste et identifier les axes d'amélioration potentiels. Il est essentiel de poursuivre les recherches et de développer des approches de contrôle plus sophistiquées pour surmonter les difficultés posées par les systèmes instables.

SIMULATION #4:

APPRENTISSAGE PAR RENFORCEMENT - RECHERCHE DE STRATÉGIES DE HAUTE QUALITÉ POUR LE SYSTÈME DIP

Référence(s):

- Inverted Double pendulum: <https://github.com/Julien-Gustin/Inverted-Double-pendulum>
- Université de Liège - Julien Gustin, Joachim Houyon - Searching High-Qaulity Policies to Control an Unstable Physical System: <https://drive.google.com/file/d/1ZZP9JLJm16gNSK-q3P0HJkgEw7KP5H6O/view?usp=sharing>
- Google Deepmind. "Continuous control with deep reinforcement learning".: <https://arxiv.org/pdf/1509.02971.pdf> (pages 2, 4, 8).

APPRENTISSAGE PAR RENFORCEMENT: RECHERCHE DE STRATÉGIES DE HAUTE QUALITÉ POUR LE SYSTÈME DIP

INTRODUCTION ET OBJECTIFS

- Dans cette section, nous avons utilisé un environnement de simulation de notre système en utilisant les bibliothèques OpenAI Gym et PyBulletGym. L'objectif principal de ce programme est de permettre l'entraînement d'un modèle d'apprentissage par renforcement pour trouver la commande optimale stabilisant le système DIP.
- Le programme offre à l'utilisateur la possibilité de choisir parmi différents algorithmes, tels que le Deep Deterministic Policy Gradient (DDPG), le Deep Q-learning (DQL) et la Fitted Q-iteration (FQI). Ces méthodes d'apprentissage par renforcement utilisent des réseaux neuronaux et des fonctions de valeur pour apprendre de l'environnement et optimiser la politique de contrôle.
- De plus, l'utilisateur peut ajuster certains paramètres tels que le gamma, les échantillons et les actions. Ces paramètres influencent le taux d'apprentissage, le nombre de points de données et le nombre d'actions discrètes utilisés par les algorithmes. Le programme offre également la possibilité de charger un modèle pré-entraîné du double pendule et de visualiser sa performance.
- Grâce à cette approche, nous sommes en mesure d'explorer et d'expérimenter différentes stratégies de contrôle pour le double pendule inversé. Cela démontre notre capacité à appliquer les concepts théoriques à des systèmes réels et à repousser les limites de notre projet.

TECHNIQUES DE RECHERCHE DE POLITIQUES : PRÉSENTATION DE L'ALGORITHME DDPG

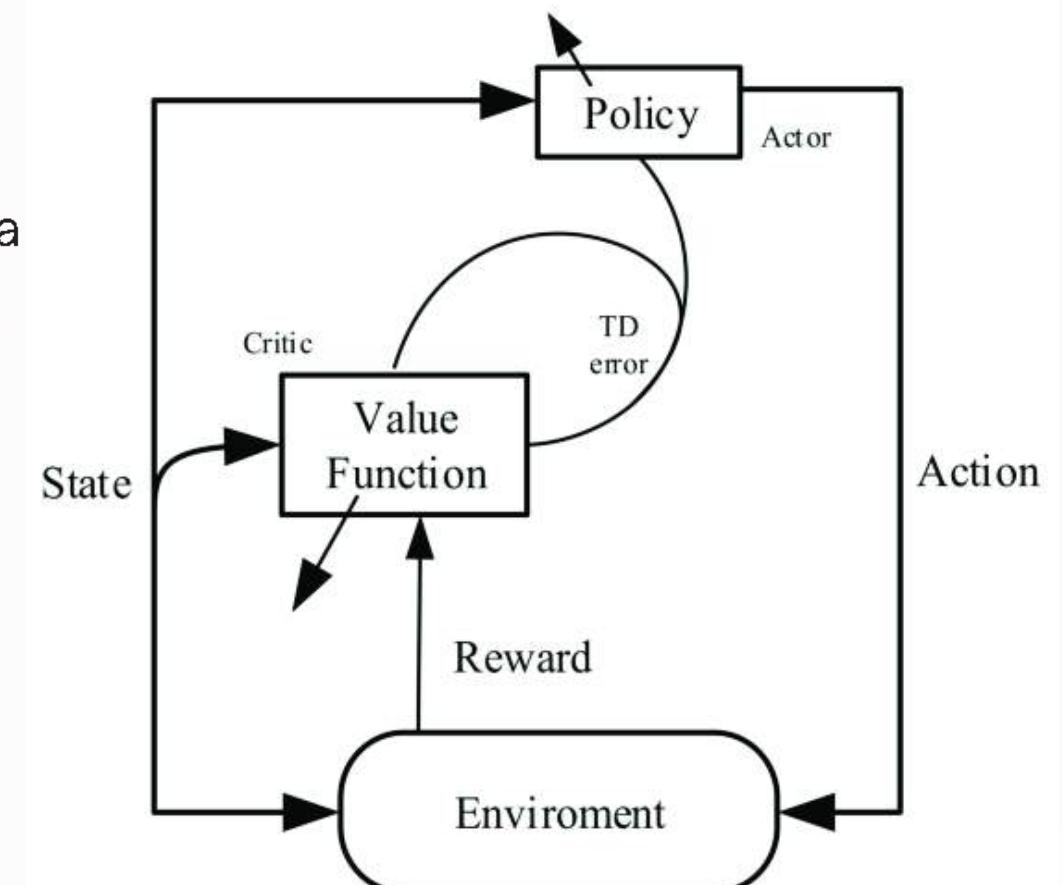
DEEP DETERMINISTIC POLICY GRADIENT (DDPG)

- Le Deep Deterministic Policy Gradient (DDPG) est un algorithme d'apprentissage qui utilise deux fonctions paramétrées : une fonction $Q(s, a|\theta^Q)$ et une politique $\mu(s, \theta^\mu)$, où θ^Q et θ^μ sont les paramètres respectifs de la fonction Q et de la politique. La fonction Q est également appelée le "critique", tandis que la politique est appelée l'"acteur".
- DDPG est un algorithme d'apprentissage hors politique, ce qui signifie que l'amélioration de la politique apprise dépend d'une autre politique pour la sélection des actions. De plus, DDPG est spécifiquement conçu pour les espaces d'action continus. Cependant, étant donné que DDPG utilise des approximations de fonctions non linéaires, il n'y a aucune garantie de convergence.
- Étant donné un ensemble de transitions D , l'apprentissage de la fonction Q se fait en minimisant la fonction d'erreur moyenne quadratique de Bellman (MBSE) :

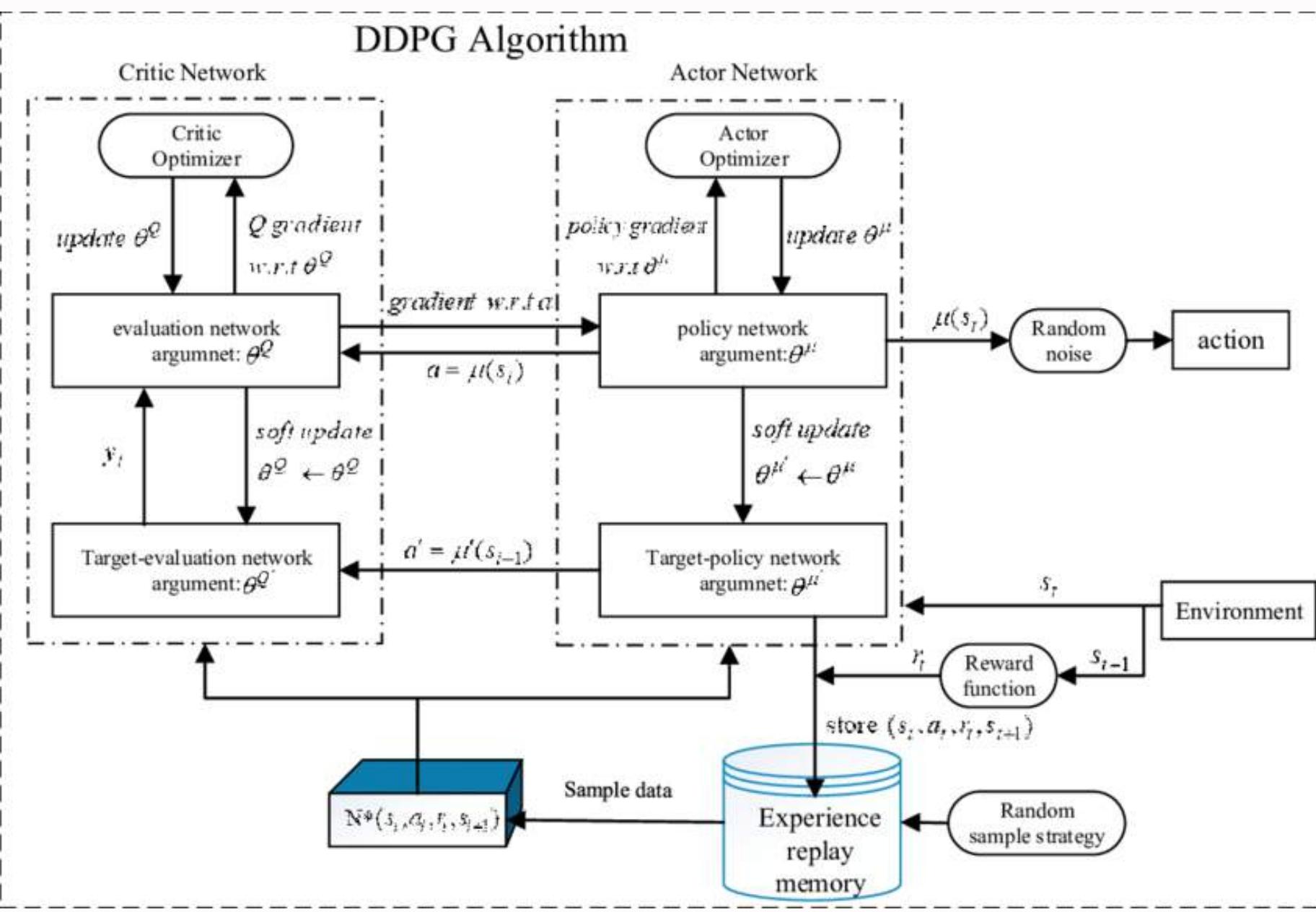
$$L(\theta^Q, \mathcal{D}) = \mathbb{E}_{(s,a,r,s',d) \sim \mathcal{D}} [(Q(s, a|\theta^Q) - (r + \lambda(1-d)(Q(s', \mu(s'|\theta^\mu)|\theta^Q))))^2]$$

où $Q(s', \mu(s'|\theta^\mu)|\theta^Q) \approx \max_{a'} Q(s', a'|\theta^Q)$, et d indique si nous sommes dans un état terminal ou non. état terminal. Dans ce cadre, on considère un espace d'action continu. Par conséquent, l'application de l'apprentissage Q dans ce cadre est irréalisable car l'apprentissage Q implique le calcul de l'action a qui maximise la valeur Q actuelle pour un état s à l'instant t . Au lieu de cela, l'acteur est formé de manière à échantillonner les actions qui maximisent $Q(s, \mu(s|\theta^\mu)|\theta^Q)$. Par conséquent, nous pouvons effectuer une ascension de gradient telle que

$$\theta^\mu = \operatorname{argmax}_\theta \mathbb{E}_{s \sim \mathcal{D}} [Q(s, \mu(s|\theta)|\theta^Q)]$$



TECHNIQUES DE RECHERCHE DE POLITIQUES : PRÉSENTATION DE L'ALGORITHME DDPG



Algorithm 1 DDPG algorithm

```

Randomly initialize critic network  $Q(s, a | \theta^Q)$  and actor  $\mu(s | \theta^{\mu})$  with weights  $\theta^Q$  and  $\theta^{\mu}$ .
Initialize target network  $Q'$  and  $\mu'$  with weights  $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^{\mu}$ 
Initialize replay buffer  $R$ 
for episode = 1, M do
    Initialize a random process  $\mathcal{N}$  for action exploration
    Receive initial observation state  $s_1$ 
    for t = 1, T do
        Select action  $a_t = \mu(s_t | \theta^{\mu}) + \mathcal{N}_t$  according to the current policy and exploration noise
        Execute action  $a_t$  and observe reward  $r_t$  and observe new state  $s_{t+1}$ 
        Store transition  $(s_t, a_t, r_t, s_{t+1})$  in  $R$ 
        Sample a random minibatch of  $N$  transitions  $(s_i, a_i, r_i, s_{i+1})$  from  $R$ 
        Set  $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1} | \theta^{\mu'}) | \theta^{Q'})$ 
        Update critic by minimizing the loss:  $L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i | \theta^Q))^2$ 
        Update the actor policy using the sampled policy gradient:
            
$$\nabla_{\theta^{\mu}} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a | \theta^Q) |_{s=s_i, a=\mu(s_i)} \nabla_{\theta^{\mu}} \mu(s | \theta^{\mu}) |_{s_i}$$

    end for
    Update the target networks:
        
$$\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}$$

        
$$\theta^{\mu'} \leftarrow \tau \theta^{\mu} + (1 - \tau) \theta^{\mu'}$$

end for

```

APPRENTISSAGE PAR RENFORCEMENT : SIMULATION AVEC L'ALGORITHME DDPG

ARCHITECTURES DE RÉSEAUX NEURONNAUX

- Le réseau critique Q est un MLP qui comporte deux couches cachées de 400 et 300 neurones respectivement. L'action n'est incluse que dans la première couche cachée. L'optimiseur Adam a été utilisé avec un taux d'apprentissage de 10^{-3} . Les poids des dernières couches sont initialisés à partir d'une distribution uniforme $[3 \times -10^{-4}, 3 \times 10^4]$.
- Le réseau d'acteurs μ est un MLP qui possède 2 couches cachées de 400 et 300 neurones respectivement. La dernière couche du réseau est une couche **tanh** afin que l'action soit bornée dans $[-1, 1]$. L'optimiseur Adam a été utilisé avec un taux d'apprentissage de 10^{-2} . Les poids sont initialisés à partir d'une distribution uniforme $[3 \times -10^{-3}, 3 \times 10^3]$.
- Pour les deux réseaux, toutes les couches cachées utilisent ReLU comme fonction d'activation. Les réseaux cibles sont mis à jour avec $\tau = 0,001$. L'architecture de ces réseaux est inspirée de la référence suivante :

Google Deepmind. "Continuous control with deep reinforcement learning". URL : <https://arxiv.org/pdf/1509.02971.pdf> (pages 2, 4, 8).

APPRENTISSAGE PAR RENFORCEMENT : SIMULATION AVEC L'ALGORITHME DDPG

EVALUATION DES PERFORMANCES

Les performances de DDPG ont été évaluées sur 500 épisodes, chaque épisode représentant 1000 pas dans l'environnement. Le tampon de relecture a une capacité de 106 échantillons et des lots de 64 échantillons y sont prélevés.

L'algorithme semble converger après ~ 200 épisodes avec un rendement attendu de ~ 930 . Lorsque l'on simule la politique du double pendule, elle n'échoue presque jamais et semble avoir une politique relativement rationnelle. En effet, il continue d'osciller en allant à gauche et à droite pour maintenir le double pendule stable. Une vidéo est disponible à l'adresse [gif/optimal_policy.gif](#) dans le dossier du projet.

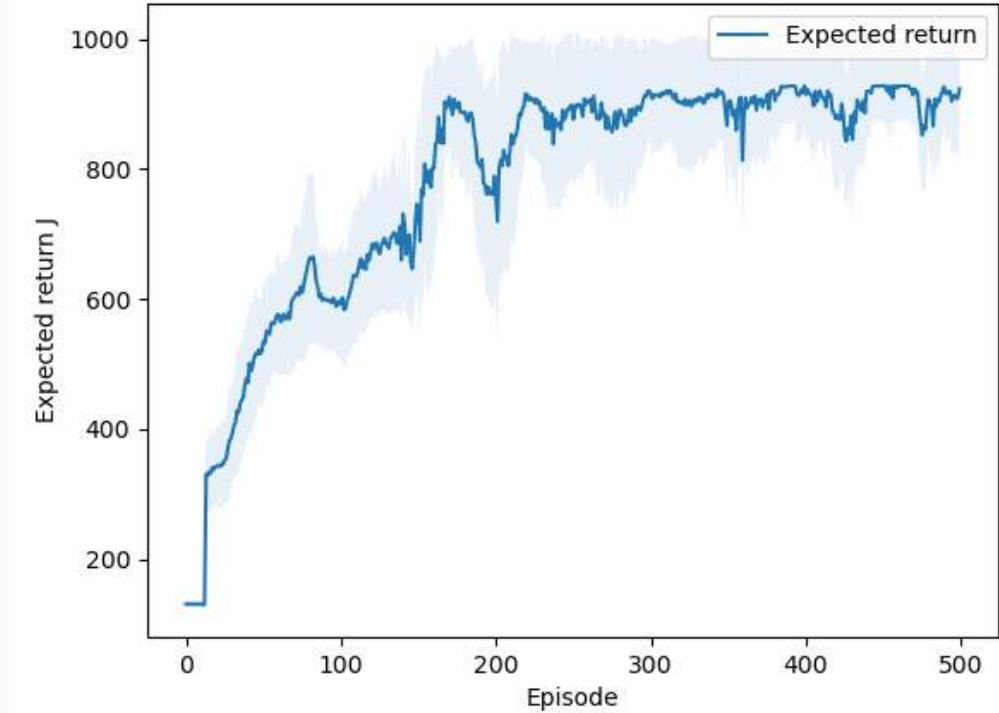


Figure: Entrainement DDPG attendue en utilisant $\gamma = 0,99$

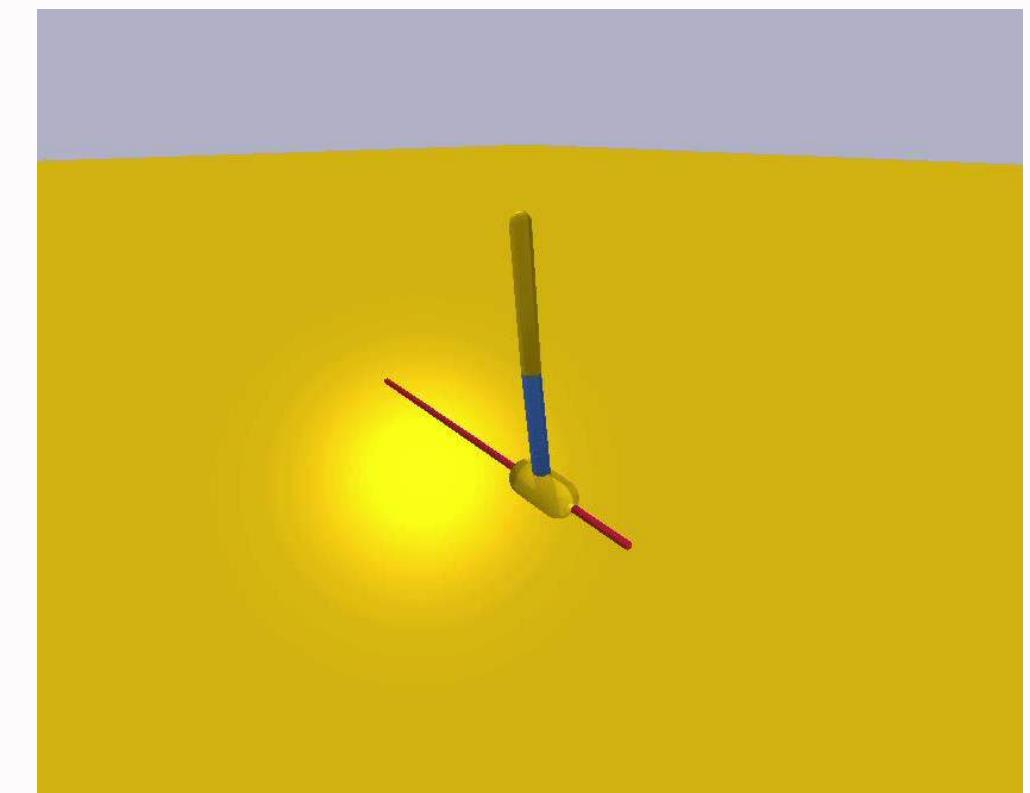
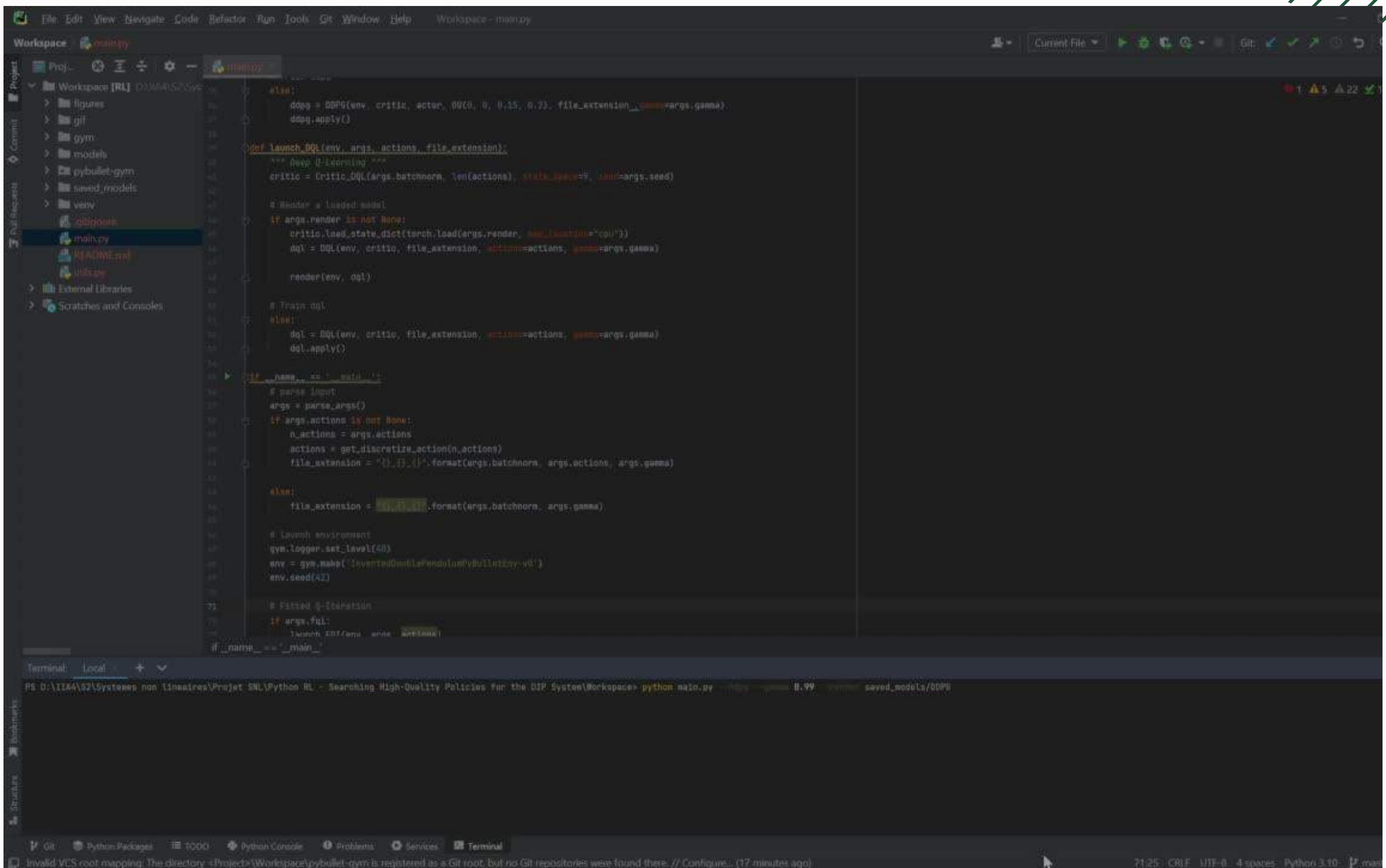


Figure: Animation du double pendule à l'aide d'une politique entraînée par DDPG

APPRENTISSAGE PAR RENFORCEMENT : EXPERIENCES AVEC L'ALGORITHME DDPG VIDÉO DÉMONSTRATIF

Commande utilisé:
python main.py --ddpg --gamma 0.99 --render saved_models/DDPG
charge un modèle préenregistré de l'algorithme DDPG



The screenshot shows the PyCharm IDE interface with the following details:

- Project:** Workspace [RL] contains subfolders like figures, git, gym, models, pybullet-gym, saved_models, venv, and ddpgcode.
- Code Editor:** The main.py file is open, showing Python code for a DDPG agent. It includes imports for numpy, torch, gym, and DDPG components. The code handles launching the environment, training, and rendering.
- Terminal:** At the bottom, a terminal window shows the command being run: `PS D:\XIAO\52\Systemes non linéaires\Projet SNI\Python RL - Searching High-Quality Policies for the DDPG System> python main.py --ddpg --gamma 0.99 --render saved_models/DDPG`.

TECHNIQUES DE RECHERCHE DE POLITIQUES : PRÉSENTATION DE L'ALGORITHME DQN

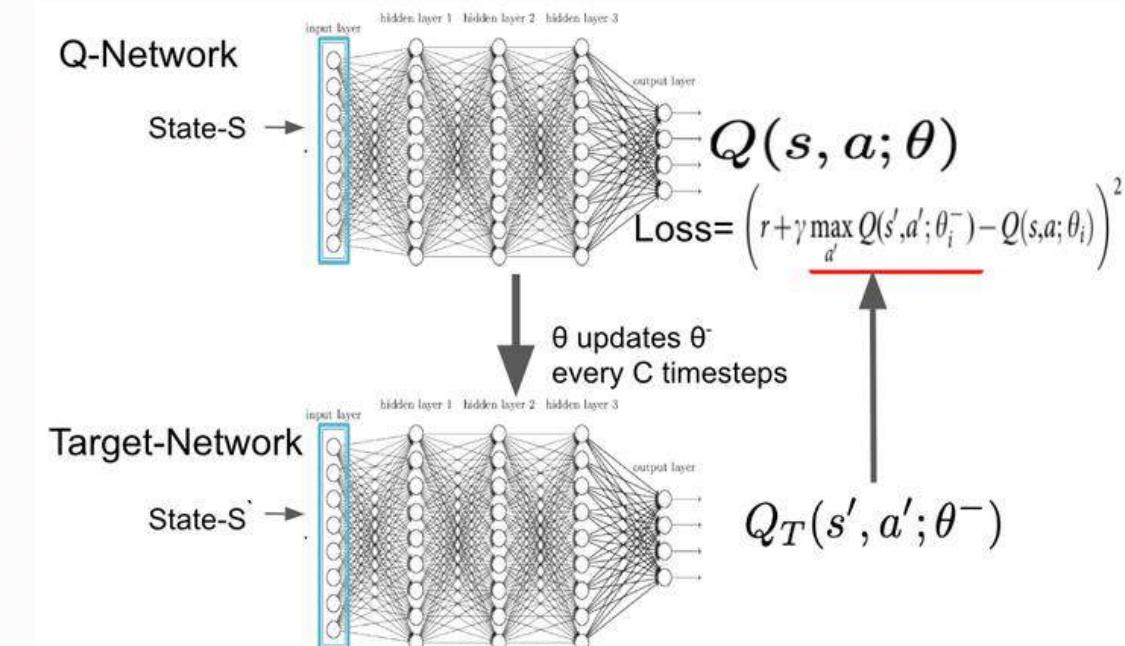
DEEP Q-NETWORK (DQN)

- Le Deep Q-Network (DQN) est une adaptation de DDPG pour les espaces d'action discrets. Contrairement à DDPG, DQN n'apprend pas de politique, mais se concentre uniquement sur l'apprentissage de la fonction Q. Cela signifie que DQN apprend une fonction paramétrée $Q(s, a|\theta)$, où θ représente les paramètres de la fonction. DQN utilise également un réseau cible $Q'(s, a|\theta')$ qui est mis à jour de la même manière que décrit précédemment, ainsi qu'un tampon de relecture pour stocker les expériences passées. Il convient de noter que la politique d'exploration utilisée dans DQN diffère de celle utilisée dans DDPG, car elle est adaptée aux espaces d'action discrets.

- Dans un espace d'action discret, le calcul d'une action a qui maximise la valeur Q actuelle pour un état s à un moment t est réalisable. Par conséquent, étant donné un ensemble de transitions D , l'apprentissage de la fonction Q se fait en minimisant la fonction MBSE :

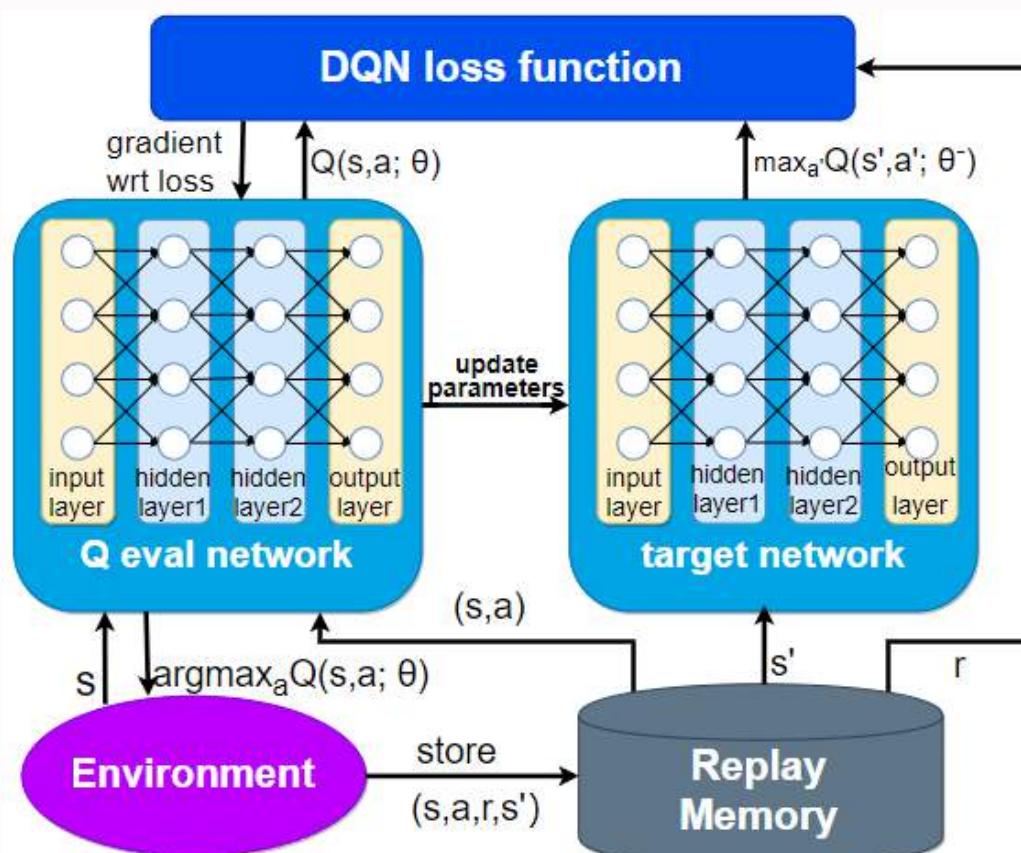
$$L(\theta, \mathcal{D}) = \mathbb{E}_{(s, a, r, s', d) \sim \mathcal{D}} [(Q(s, a|\theta) - (r + \lambda(1-d)\max_{a'} Q'(s', a'|\theta')))^2]$$

- La politique d'exploration est construite de telle sorte qu'au début, elle doit explorer beaucoup, et diminuer le taux d'exploration au fur et à mesure que le temps passe. Ce comportement peut être obtenu à l'aide d'un algorithme d'avidité epsilon où ϵ recuit linéairement à partir de $\epsilon_0 = 1$, diminue le taux d'exploration jusqu'à 0,1 de sorte que $\epsilon_t \leq \epsilon_{t-1}$ et $\epsilon_t \geq 0,1 \forall t \in [1, T]$ pour les 50 premiers pas du nombre total de pas utilisés pour l'apprentissage. Ensuite, ϵ reste à 0,1.



TECHNIQUES DE RECHERCHE DE POLITIQUES : PRÉSENTATION DE L'ALGORITHME DQN

DEEP Q-NETWORK (DQN)



Algorithm 1 Deep Q-learning with Experience Replay

```
Initialize replay memory  $\mathcal{D}$  to capacity  $N$ 
Initialize action-value function  $Q$  with random weights
for episode = 1,  $M$  do
    Initialise sequence  $s_1 = \{x_1\}$  and preprocessed sequenced  $\phi_1 = \phi(s_1)$ 
    for  $t = 1, T$  do
        With probability  $\epsilon$  select a random action  $a_t$ 
        otherwise select  $a_t = \max_a Q^*(\phi(s_t), a; \theta)$ 
        Execute action  $a_t$  in emulator and observe reward  $r_t$  and image  $x_{t+1}$ 
        Set  $s_{t+1} = s_t, a_t, x_{t+1}$  and preprocess  $\phi_{t+1} = \phi(s_{t+1})$ 
        Store transition  $(\phi_t, a_t, r_t, \phi_{t+1})$  in  $\mathcal{D}$ 
        Sample random minibatch of transitions  $(\phi_j, a_j, r_j, \phi_{j+1})$  from  $\mathcal{D}$ 
        Set  $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$ 
        Perform a gradient descent step on  $(y_j - Q(\phi_j, a_j; \theta))^2$  according to equation 3
    end for
end for
```

TECHNIQUES DE RECHERCHE DE POLITIQUES : SIMULATION AVEC L'ALGORITHME DQN

ARCHITECTURES DE RÉSEAUX NEURONAUX

Le réseau critique a la même architecture, le même optimiseur et la même initialisation que le critique de l'algorithme DDPG, sauf qu'il ne prend pas d'action en entrée. Au lieu de cela, la couche de sortie du critique contient un nombre de neurones égal à l'espace d'action :

Le i ème neurone de la couche de sortie représente l'estimation de la sortie attendue de la i ème action lorsqu'elle se trouve dans un état s .

APPRENTISSAGE PAR RENFORCEMENT : SIMULATION AVEC L'ALGORITHME DQN

EVALUATION DES PERFORMANCES

- L'évaluation de DQN a été effectuée dans les mêmes conditions que l'évaluation de DDPG. En ce qui concerne la discréétisation de l'espace d'action, elle a été évaluée sur différentes tailles en utilisant des nombres uniformément espacés sur l'intervalle [-1,1].
- D'après la figure suivante, on peut constater que l'utilisation de DQN avec une discréétisation diverse nécessite près de 200 épisodes de plus que DDPG pour converger, de plus, DQN atteint un plateau à la valeur d'environ 810, soit environ 10 % de moins que DDPG. Ce résultat peut s'expliquer par le fait que la discréétisation de l'espace d'action ne donne pas suffisamment de degrés de liberté à l'agent.
- DQN semble également être moins stable que DDPG en raison des pics observés. Curieusement, l'augmentation du nombre d'actions ne change pas beaucoup la récompense cumulative attendue de 3 à 13 actions, et il est intéressant de regarder une simulation de ces différents modèles car selon l'espace d'action, le modèle adapte ses stratégies, par exemple avec 11 actions, le réseau a appris une stratégie consistant à rester du côté gauche de la corde et à appliquer de petites forces à gauche et à droite pour maintenir le double pendule stable. Cela ressemble à de la triche, mais cela fonctionne ! En revanche, lorsque très peu d'actions sont disponibles, sa stratégie consiste à faire de grandes oscillations gauche-droite.

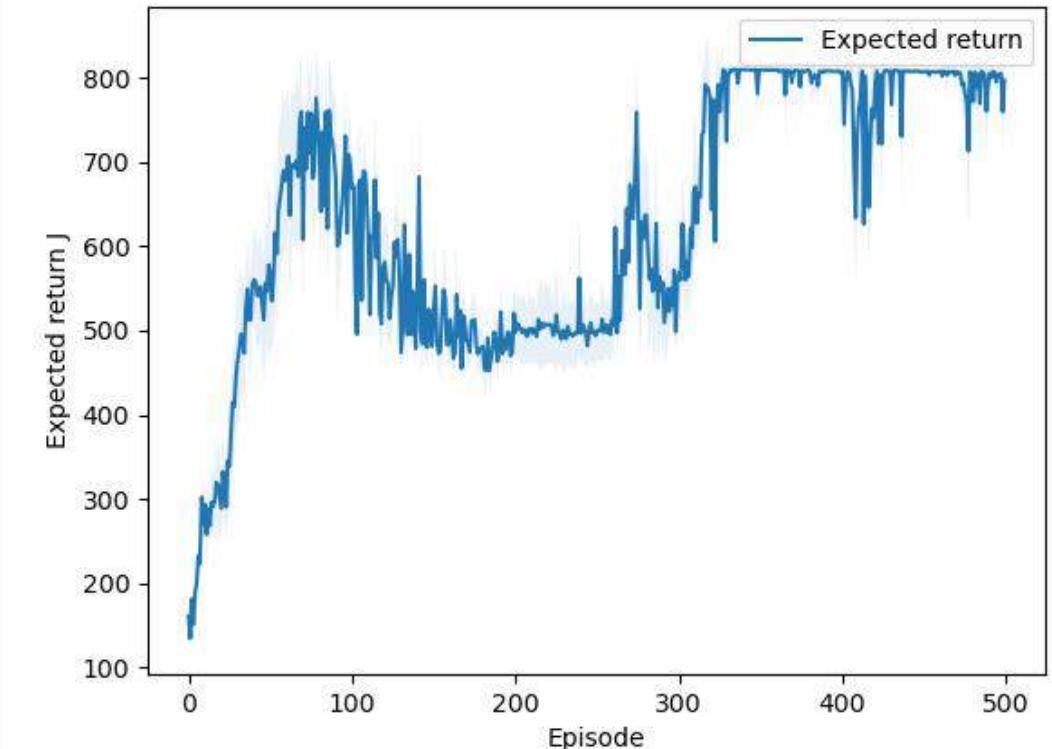


Figure: Algorithme DQN sur un espace d'action diversifié avec $\gamma = 0,99$, et 11 actions discrètes

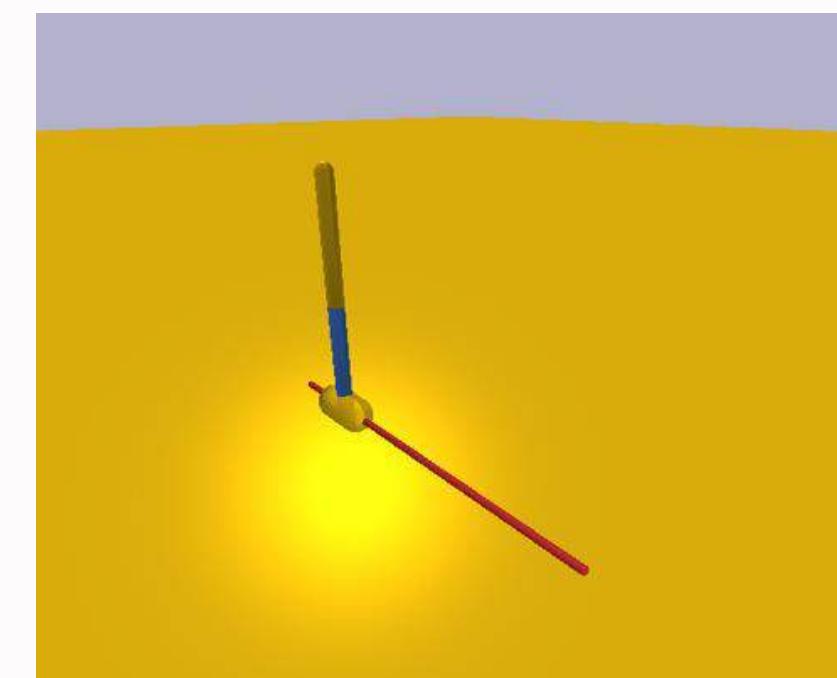
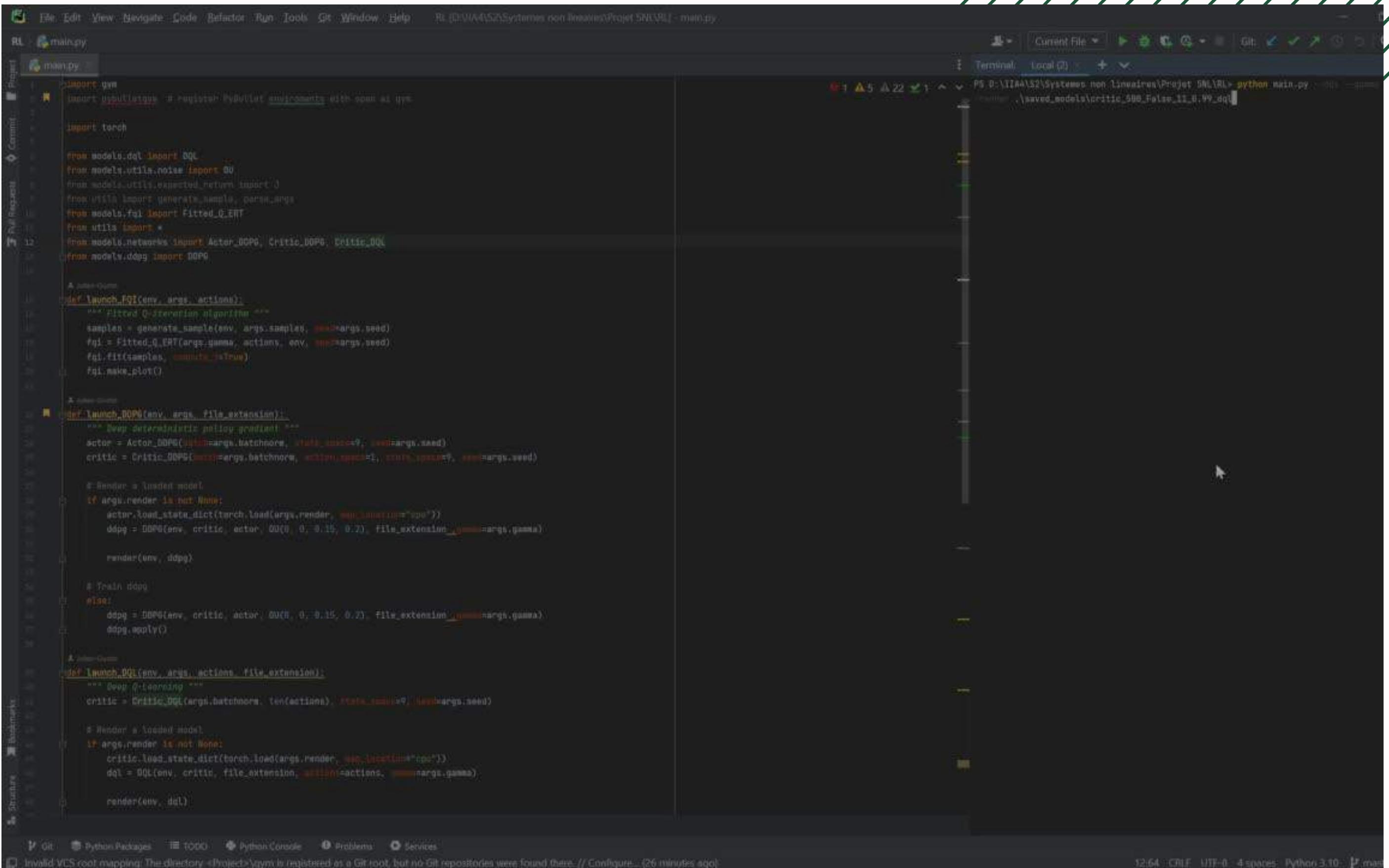


Figure: Stratégie de DQN avec 11 actions gauche-droite.

APPRENTISSAGE PAR RENFORCEMENT : EXPERIENCES AVEC L'ALGORITHME DQN VIDÉO DÉMONSTRATIF

Commande utilisé:
python main.py --dqn --gamma 0.99 --render saved_models/critic_500_False_11_0.99_dql
charge un modèle préenregistré de l'algorithme DQN avec 11 actions



The screenshot shows a code editor with a dark theme. The main window displays a Python script named 'main.py'. The code implements various reinforcement learning algorithms, including Fitted-Q-Iteration (FQI), Deep Deterministic Policy Gradient (DDPG), and Q-Learning (DQL). It imports modules from 'gym', 'torch', and 'models'. The code includes functions for launching each algorithm with specific arguments and rendering environments. A terminal window to the right shows the command being run: 'python main.py --dqn --gamma 0.99 --render saved_models/critic_500_False_11_0.99_dql'. The terminal output indicates the script is running.

```
File Edit View Navigate Code Refactor Run Tools Git Window Help R: D:\VIA\LS2\Systemes non linéaires\Projet SNCVIE\main.py
RL main.py
main.py
import gym
# Import utilitaires à registered pour utiliser environments avec gym as gym
import torch
from models.dql import DQL
from models.utils.noise import OU
from models.utils.expected_return import E
from utils import generate_sample, parse_args
from models.fqi import Fitted_Q_EHT
from utils import *
from models.networks import Actor_DDPG, Critic_DDPG, Critic_DQL
from models.ddpg import DDPG

# Launch FQI(env, args, actions):
#   * Fitted Q-Iteration algorithm
#   samples = generate_sample(env, args.sample, env, args.seed)
#   fqi = Fitted_Q_EHT(args.gamma, actions, env, env, args.seed)
#   fqi.fit(samples, env, args)
#   fqi.make_plot()

# Launch DDPG(env, args, file_extension):
#   * Deep deterministic policy gradient
#   actor = Actor_DDPG(actions=actions, batchnorm=batchnorm, env=env, args=args)
#   critic = Critic_DDPG(actions=actions, batchnorm=batchnorm, env=env, args=args)
#
#   # Render a loaded model
#   if args.render is not None:
#       actor.load_state_dict(torch.load(args.render, map_location='cpu'))
#       ddpg = DDPG(env, critic, actor, OU(0, -0.15, 0.2), file_extension, args, gamma)
#
#       render(env, ddpg)
#
#   Train ddpg
#   else:
#       ddpg = DDPG(env, critic, actor, OU(0, -0.15, 0.2), file_extension, args, gamma)
#       ddpg.train()
#
#   # Render a loaded model
#   if args.render is not None:
#       critic.load_state_dict(torch.load(args.render, map_location='cpu'))
#       dql = DQL(env, critic, file_extension, actions=actions, env=env, gamma)
#
#       render(env, dql)

# Launch DQL(env, args, actions, file_extension):
#   * Deep Q-learning
#   critic = Critic_DQL(batchnorm, env, actions, env, args, gamma)

# Render a loaded model
if args.render is not None:
    critic.load_state_dict(torch.load(args.render, map_location='cpu'))
    dql = DQL(env, critic, file_extension, actions=actions, env=env, gamma)

    render(env, dql)
```

ANALYSE DES RÉSULTATS, AMÉLIORATIONS POSSIBLES ET CONCLUSION

- Nous avons mis en œuvre un modèle d'apprentissage par renforcement pour découvrir des stratégies efficaces pour la régulation du système à double pendule inversé, démontrant notre volonté d'explorer des approches innovantes et d'optimiser les performances du système.
- Plusieurs techniques de recherche de politique ont été utilisées pour contrôler le système instable, aboutissant à des politiques décentes, bien que non optimales.
- En comparant DDPG et DQN, DDPG a trouvé une politique avec une récompense cumulative attendue plus élevée. DDPG est meilleur pour trouver de bonnes politiques, bien que l'ajustement des hyperparamètres soit plus fastidieux en raison de l'utilisation d'un réseau supplémentaire. DQN pourrait offrir un apprentissage plus stable.
- Pour améliorer les politiques, des techniques telles que la relecture d'expériences priorisées et l'utilisation de politiques basées sur des images pourraient être explorées avec DDPG ou DQN en utilisant des réseaux neuronaux convolutifs.
- En conclusion, malgré certaines limites, l'algorithme DDPG a démontré sa capacité à équilibrer les bras du double pendule verticalement sur une longue période. Pour améliorer les performances du système, des stratégies d'exploration différentes, l'ajustement des hyperparamètres et l'utilisation de données plus diverses et robustes sont recommandés.

CONCLUSION GÉNÉRALE

En résumé, notre projet sur le double pendule inversé sur un chariot présente des simulations approfondies dans divers environnements, mettant en évidence des approches de contrôle innovantes. Nous sommes convaincus que ces simulations impressionneront par la qualité du travail accompli, les résultats obtenus et l'effort considérable déployé.

Les implications de ce projet vont au-delà de la réussite technique, ouvrant de nouvelles perspectives pour la régulation de systèmes instables dans divers domaines d'application. Ces résultats prometteurs renforcent aussi notre conviction en l'importance de l'apprentissage par renforcement et des simulations software pour résoudre des problèmes réels complexes.

RÉFÉRENCES

Modélisation mathématique:

- University of New Mexico - Control Theory of The Double Pendulum Inverted on a Cart/
<https://drive.google.com/file/d/1VxT84KJXg89n7mJowL-xiMBDLQW6Sbc3/view?usp=sharing>

Modélisation du système sur MATLAB:

- Implementation of MATLAB Code for LQR/LQG for linearized and non-linearized system:
<https://github.com/nakul3112/Double-inverted-pendulum.git>

Modélisation et simulation LabVIEW (Simulation #1):

- Quanser Linear Double Inverted Pendulum: <https://www.quanser.com/products/linear-double-inverted-pendulum/>
- Quanser - Linear Servo Control Systems Brochure:
https://drive.google.com/file/d/1FyeQGBpf4AzLHVXN6yYvqwZmRYK_1f8l/view?usp=sharing
- Quanser - Linear Double Inverted Pendulum - Laboratory Guide:
<https://drive.google.com/file/d/1QVa5jelx7D5Yzznd775VXKbliXpe-aos/view?usp=sharing>

Apprentissage par renforcement (Simulation #4):

- Inverted Double pendulum: <https://github.com/Julien-Gustin/Inverted-Double-pendulum>
- Université de Liège - Julien Gustin, Joachim Houyon - Searching High-Qaulity Policies to Control an Unstable Physical System: <https://drive.google.com/file/d/1ZZP9JLJm16gNSK-q3P0HJkgEw7KP5H6O/view?usp=sharing>
- Google Deepmind. "Continuous control with deep reinforcement learning". : <https://arxiv.org/pdf/1509.02971.pdf> (pages 2, 4, 8).



**MERCI POUR
VOTRE
ATTENTION**