# A SURVEY OF LQE AND MPC DISCRETE-CONTROL FOR ACROBOT

**Thanh-Tri-Dai Le*, Le-Thien-Bao Doan, Hoang-Anh-Thai Vo, Hoang-Loi Trinh, Tan-Dung Nguyen, Quoc-Khanh Tran, Hoai-Vong Tran, Xuan-Vinh Nguyen**

Ho Chi Minh city University of Technology and Education (HCMUTE)
Vo Van Ngan, 01, Ho Chi Minh city, Vietnam
***** Corresponding author. E-mail: 19151051@student.hcmute.edu.vn

**Abstract:** This paper surveys the Linear Quadratic Estimation (LQE) and Model Predictive Control (MPC) discrete control methods applied to the Acrobot system. Both techniques aim to achieve and maintain a balanced position for the Acrobot. Through evaluation and comparison, we highlight their strengths, limitations, and potential applications, offering insights for future robotic control research.
**Keywords:** acrobot, LQR control, MPC control, discrete control.

## 1. Introduction

The Acrobot, a benchmark model in control research [1]-[4], is intricately described by its bifurcated structure, visually represented in Fig. 1. This system encompasses two primary links: the lower link (Link1), and the elevated link (Link2), which are interconnected through an active joint, managed by a control motor. Uniquely, Link1 freely maneuvers around a passive joint at its terminal point, adding an additional layer of dynamical complexity and thus, presenting a sophisticated control challenge that has been addressed by various methodologies in the field.

Over time, a myriad of control strategies, from Reinforcement Learning (RL) to classical PID [2] controllers and Fuzzy Logic[3], have been employed to navigate the multifaceted control landscape of the Acrobot, each yielding its own set of insights and challenges. RL, while notable for its adaptive capabilities, often demands substantial computational resources and training time, whereas PID controllers and Fuzzy Logic, despite their computational efficiency and simplicity, may struggle to maintain stability due to the Acrobot's non-linearities and underactuated dynamics.

In this context, research on Linear LQE [5]- [6] and MPC [7]-[8] has emerged, both presenting formidable contenders in addressing the control challenges of the Acrobot, each offering a unique approach and distinctive advantages. This paper aims to delve comprehensively into exploring and contrasting LQE and MPC, evaluating their theoretical underpinnings, practical applications, and performance metrics in controlling the Acrobot, ultimately seeking to unearth new insights and guide future research and applications in robotic control strategies.

## 2. Mathematical Model

In Fig. 1, the x-axis of the Cartesian coordinate system is established as the baseline for zero potential energy.

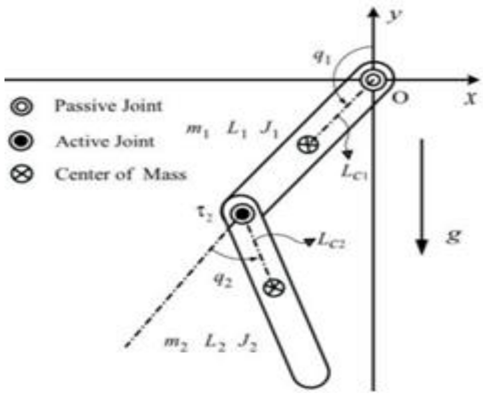


**Fig. 1.** Mathematical model of Acrobot [1]

Let $X_i = [X_i^x, X_i^y]^T \in R^2$ , represent the absolute position of the Center of Mass (COM) of the ith link, as derived in [5]:

$$X_1 = \begin{bmatrix} L_{c1}sinq_1 \\ L_{c1}cosq_1 \end{bmatrix} \quad (1)$$

$$X_2 = \begin{bmatrix} L_1 sinq_1 + L_{c2}sin(q_1+q_2) \\ L_1 cosq_1 + L_{c2}cos(q_1+q_2) \end{bmatrix} \quad (2)$$

Consequently, the expressions for kinetic energy, denoted as $K(q,\dot{q})$ and potential energy, symbolized as $V(q)$, are as follows:

$$K(q,\dot{q}) = \frac{1}{2}\sum_{i=1}^{2}[\, m_i||\dot{X}_i||^2 + J_1\dot{q}_1^2 + J_2(\,\dot{q}_1 + \dot{q}_2)^2 = \frac{1}{2}[\,\dot{q}_1\ \dot{q}_2]M(q_2)\begin{bmatrix}\dot{q}_1 \\ \dot{q}_2\end{bmatrix}] \quad (3)$$

where (4)

$$M(q_2) = \begin{bmatrix} a_1 + a_2 + 2cosa_3cosq_2 & a_2 + a_3cosq_2 \\ a_2 + a_3cosq_2 & a_2 \end{bmatrix}$$

$$\begin{cases} a_1 = m_1L_{c1}^2 + m_2L_1^2 + J_1 \\ a_2 = m_2L_{c2}^2 + J_2 \\ a_3 = m_2L_1L_{c2} \\ a_4 = (m_1L_{c1} + m_2L_1)g \\ a_5 = m_2L_{c2}g \end{cases} \quad (5)$$

$$V(q) = m_1gX_1^y + m_2gX_2^y = a_4cosq_1 + a_5cos(q_1+q_2) \quad (6)$$

When considering friction as negligible, by applying Lagrangian function to Acrobot, we obtain expression for dynamic equation of mechanical system as follows:

$$\frac{d}{dt}\left[\frac{\partial L(q,\dot{q})}{\partial \dot{q}_i}\right] - \frac{\partial L(q,\dot{q})}{\partial q_i} = \tau_i \, , i = 1,2, \quad (7)$$

where $L(q,\dot{q}) = K(q,\dot{q}) - V(q)$ and $\tau_1 = 0$ . Eq.(7) equivalent to

$$M(q_2)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \begin{bmatrix} 0 \\ \tau_2 \end{bmatrix} \quad (8)$$

From (8) , we have

$$\ddot{q} = -M^{-1}C\dot{q} - M^{-1}G + M^{-1}u \quad (9)$$

where

$$q = [q_1\, q_2]^T \quad (10)$$

$$C(q,\dot{q}) = \begin{bmatrix} -a_3\dot{q}_2sinq_2 & -a_3(\dot{q}_1+\dot{q}_2)sinq_2 \\ a_3\dot{q}_1sinq_2 & 0 \end{bmatrix} \quad (11)$$

$$G(q) = \begin{bmatrix} -a_4sinq_1 - a_5sin(q_1+q_2) \\ -a_5sin(q_1+q_2) \end{bmatrix} \quad (12)$$

$$u = \begin{bmatrix} 0 \\ \tau_2 \end{bmatrix} \quad (13)$$

From (9) , we have

$$\begin{cases} \ddot{q}_1 = f_1(q_1,q_2,\dot{q}_1,\dot{q}_2,u) \\ \ddot{q}_2 = f_1(q_1,q_2,\dot{q}_1,\dot{q}_2,u) \end{cases} \quad (14)$$

We proceed to set variables in order to reduce the order of the system:

$$\begin{cases} x_1 = q_1 \\ x_2 = \dot{q}_1 = \dot{x}_1 \\ x_3 = q_2 \\ x_4 = \dot{q}_2 = \dot{x}_3 \end{cases} \quad (15)$$

From (14) , we obtain:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = f_1(x_1,x_2,x_3,x_4,u) \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = f_2(x_1,x_2,x_3,x_4,u) \end{cases} \quad (16)$$

From equation set (16) , linearization around the operating point yield $x_0 = [\,0\;0\;0\;0]^T$. Specifically, it is : x1 = 0 ; x2 =0 ; x3 = 0 ; x4 = 0 ; u=0

$$\Rightarrow \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = A \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + B*u \quad (17)$$

⇒ The system of linear equations, when operating around the equilibrium point, is defined as:

Matrices A and B are determined as follows.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{\partial f_1}{\partial x_1}_{\tau=0;x=x_0} & \frac{\partial f_1}{\partial x_2}_{\tau=0;x=x_0} & \frac{\partial f_1}{\partial x_3}_{\tau=0;x=x_0} & \frac{\partial f_1}{\partial x_4}_{\tau=0;x=x_0} \\ 0 & 0 & 0 & 1 \\ \frac{\partial f_2}{\partial x_1}_{\tau=0;x=x_0} & \frac{\partial f_2}{\partial x_2}_{\tau=0;x=x_0} & \frac{\partial f_2}{\partial x_3}_{\tau=0;x=x_0} & \frac{\partial f_2}{\partial x_4}_{\tau=0;x=x_0} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ \frac{\partial f_1}{\partial \tau}_{\tau=0;x=x_0} \\ 0 \\ \frac{\partial f_2}{\partial \tau}_{\tau=0;x=x_0} \end{bmatrix} \quad (18)$$

(where A and B are linear matrices)

Variable names and parameters are listed in Tab. 1.

**Tab. 1.** System parameters

| Symbols | Meaning | Value |
|---|---|---|
| $q_1$ | Angle of Link1 | |
| $q_2$ | Angle of Link2 | |
| $\dot{q}_1$ | Angular velocity of Link1 | |
| $\dot{q}_2$ | Angular velocity of Link2 | |
| $m_1$ | Mass of Link1 | 0.8 Kg |
| $L_1$ | Length of Link1 | 0.18 m |
| $L_{c1}$ | Distance from Passive joint to center of mass of the Link1 | 0.11 m |
| $m_2$ | Mass of Link2 | 0.2 Kg |
| $L_2$ | Length of Link2 | 0.18 m |
| $L_{c2}$ | Distance from Active joint to center of mass of the Link2 | 0.09 m |
| $J_1$ | Moment of Inertia Link1 | 0.0022 kg.$m^2$ |
| $J_2$ | Moment of Inertia Link2 | 0.00054 kg.$m^2$ |
| $g$ | Gravitational acceleration | 9.81 m/s$^{23}$ |
| $\tau_2$ | Torque applied to Active joint | |

### 3. Design of the controller

#### 3.1. Convert from Continuous State Space Equation to Discrete

To delve deeper, refer to [5], we are starting with:

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

By applying the Laplace transform to each side:

$$sIX(s) - x(0) = AX(s) + B\frac{u}{s}$$

$$X(s) = (sI-A)^{-1}x(0) + (sI-A)^{-1}B\frac{u}{s} \quad (19)$$

By applying the inverse Laplace transform to each side, we obtain:

$$x(t) = \int_0^t e^{A\tau}d\tau Bu + e^{At}x(0) \quad (20)$$

If A is invertible, we can further simplify as :

$$x(t) = \int_0^t Ie^{A\tau}d\tau Bu + e^{At}x(0) \quad (21)$$

$$x(t) = A^{-1}(e^{At} - I)Bu + e^{At}x(0) \quad (22)$$

We assume u remains constant from 0 to t, valid only for 0 ≤t≤ Δt . Specifically, we consider:

$$x(\triangle t) = A_d x(0) + B_d u) \quad (23)$$

or

$$x[1] = A_d x[0] + B_d u)$$

We determine the state-space representation of the system for the first sampling period through $A_d$ and $B_d$. If A is non-invertible, use the Moore-Penrose inverse. System S, with defined inputs and outputs, is considered time-invariant if the corresponding conditions are satisfied. Systems described by linear differential equations with constant coefficients all possess this characteristic.

Within the span of [t,t+Δt], u is maintained at a

$$x(t + \triangle t) = A_d x(t) + B_d u) \quad (24)$$

constant value. Specifically, this informs us.

$$x((k+1) \triangle t) = A_d x(k \triangle t) + B_d u) \quad (25)$$

Or:

$$x[k+1] = A_d x[k] + B_d u[k] \quad (26)$$

### 3.2. Designing a MPC Controller

MPC is an advanced control strategy based on a system model. At each step, it predicts the system's future behavior, solves an optimization problem to determine the optimal control signal, and then applies this control action.

For the state model of the system with a single input and a single output, the representation is as follows:

$$x[k+1|k] = A_d x[k] + B_d u[k] \quad (27)$$

$$y[k|k] = Cx[k] \quad (28)$$

Now, we can determine the system's output at each stage based on the state matrix. To start, define np as the prediction period, representing how many stages ahead we anticipate the system's response. Similarly, nc is the control period, indicating how many stages we aim to adjust the control sequence. It's essential that np must be greater than or equal to nc. In this situation, they hold the same value, so n = np = nc.

Presuming we're aware of the system's starting state x[k], we set u[k] as the control input for the kth interval. Then:

$$x[k+1|k] = A_d x[k] + B_d u[k] \quad (29)$$

$$x[k+2|k] = A_d^2 x[k] + A_d B_d u_k + B_d u[k+1] \quad (30)$$

$$x[k+n|k] = A_d^n x[k] + A_d^{n-1} B_d u[k] + A_d^{n-2} B_d u[k+1] + \ldots B_d u[k+n-1] \quad (31)$$

The predicted output variables will be calculated:

$$y[k+1|k] = Cx[k+1] = CA_d x[k] + CB_d u[k] \quad (32)$$

$$y[k+2|k] = CA_d^2 x[k] + CA_d B_d u_k + CB_d u[k+1] \quad (33)$$

$$y[k+n|k] = CA_d^n x[k] + CA_d^{n-1} B_d u[k] + CA_d^{n-2} B_d u[k+1] + \ldots CB_d u[k+n-1] \quad (34)$$

Now, if we define:

$$Y = [\, y[k+1|k]\; y[k+2|k] \ldots y[k+n|k] \,]^T \quad (35)$$

$$U = [\, u[k]\;\; u[k+1] \ldots u[k+n-1] \,]^T \quad (36)$$

We have

$$Y = Fx[k] + GU \quad (37)$$

Here:

$$F = \begin{bmatrix} CA_d^1 \\ CA_d^2 \\ CA_d^3 \\ \vdots \\ CA_d^n \end{bmatrix} \quad (38)$$

$$G = \begin{bmatrix} CB_d & 0 & 0 & \ldots & 0 \\ CA_d^1 B_d & CB_d & 0 & \ldots & 0 \\ CA_d^2 B_d & CA_d^1 B_d & CB_d & \ldots & 0 \\ \vdots & \vdots & \vdots & \ldots & \ldots \\ CA_d^{n-1} B_d & CA_d^{n-2} B_d & CA_d^{n-3} B_d & \ldots & CB_d \end{bmatrix} \quad (39)$$

Now that we possess a clear formula for the output of the system, we can refine the control input. Assume r is the desired vector for the system's output, and let $R = [\, r\, r \ldots r \,]^T$ where there are n elements

Now, we need to define the cost function to optimize it such that it reaches its minimum value.

$$P = \text{q}||R - Y||_2^2 + \omega||U||_2^2 \quad (40)$$

$$P = (R - Y)^T Q (R - Y) + U^T W U \quad (41)$$

$$P = (R - (Fx(k) + GU))^T Q (R - (Fx(k) + GU)) + U^T W U \quad (42)$$

$$P = (R - Fx)^T Q (R - Fx) - 2U^T G^T Q (R - Fx) + U^T (G^T Q G + W) U \quad (43)$$

We apply the Newton-Raphson algorithm to find the control signal *U* such that the cost function *P* is minimized, ensuring optimal system performance.

We need to calculate:

$$\nabla P_u = \frac{\partial P}{\partial U} = -2G^T Q (R - Fx) + 2(G^T Q + W) U \quad (44)$$

$$H(P) = \nabla^2 P_u = \frac{\partial^2 P}{\partial U^2} = 2(G^T Q G + W) \quad (45)$$

Step 1 : Initialization of Control Signal:

- Start with the previously used control signal, denoted as $u_{n-1}$. If there is no previous signal u, we initialize with the value $u_0$
- Setting the initialized signal as $u_i$ with $i = 0$

Step 2 : Updating the new value using Newton - Raphson method :

$$u_{i+1} = u_i - (H(P)^{-1}) \nabla P_u \quad (46)$$

Step 3 : Check the stopping condition :

During the execution of the algorithm, the loop will terminate if any of the following conditions are met:

- Convergence: The loop will stop when the result no longer changes significantly or when the gradient (or derivative) approaches 0 : $||\nabla P_u(i)||_2^2 < \in$
- The number of iterations i does not exceed the specified number of iterations j : $i < j$

Step 4 : Store the value

The value stored is the final value of U returned after checking the stopping condition, and this value is denoted as $U_n$

The resultant U is a series of predicted signals U; we select the initial U from the column following a Receding Horizon Control strategy

**3.3. Designing a LQR Controller**

In a model with a clear mathematical equation, detailed system parameters, and a fixed operating point, LQR stands out as a preferred choice. Its prowess is showcased through its streamlined structure, efficient computation - especially with the tools provided by Matlab, and its flexibility in adjustments based on the weight matrix. This makes LQR a top pick for balancing robot control. And this approach has been employed in our study.

The cost function is selected as

$$J = \sum_{t=0}^{\infty} (x(k)^T Q x(k) + u(k)^T R u(k) \quad (47)$$

where: Q is positive define matrix(or semi positive definite); R is positive definite matrix ; matrix K is optimized from the Riccati equation in the form:

$$K = (R + B_d^T P B_d)^{-1} B_d^T P A_d \quad (48)$$

The control law u(t) is computed as

$$U = -Kx = -(R + B_d^T P B_d)^{-1} B^T P A_d x \quad (49)$$

where P is the semi-positive definite solution of the Riccati algebraic equation:

$$A_d^T P + P A_d + Q - A_d^T P B_d (R + B_d^T P B_d)^{-1} B_d^T d P A_d = 0 \quad (50)$$

where, Q matrix represents the control object, R matrix represents the control signal.

The control law is computed using the function in Matlab as follows:

$$K = dlqr(A_d, B_d, Q, R) \quad (51)$$

**4. Result and Simulation**

In this simulation section, we will select initial value for the process as follows :

$x_0 =$ [ 0.02 0 0.001 0.01] ;

To simulate fairness, we set the parameters of the LQR and MPC controllers to be the same; the difference here is that the MPC uses prediction steps equal to np.

The control parameter values selected for simulation are:

- t=0.01 (conversion time from continuous domain to discrete domain)
- $Q = \begin{bmatrix} 1000 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
- R = 100
- n_p = 200 ( Prediction Horizon )

Thence, simulation results are shown in Fig. 2 to Fig. 5.

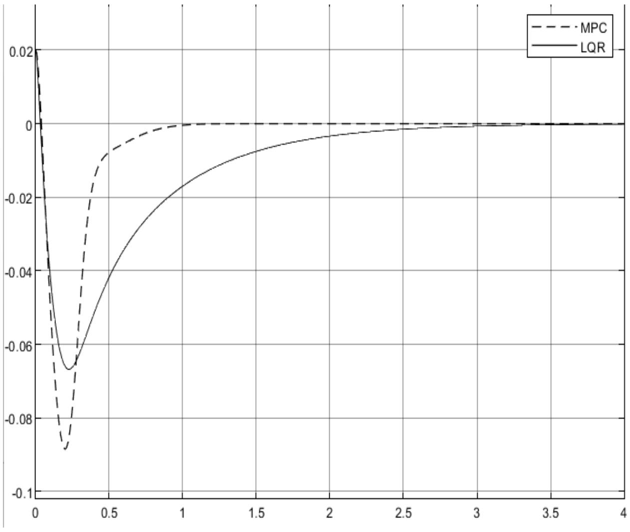


**Fig. 2.** Angle of Link1 (rad)

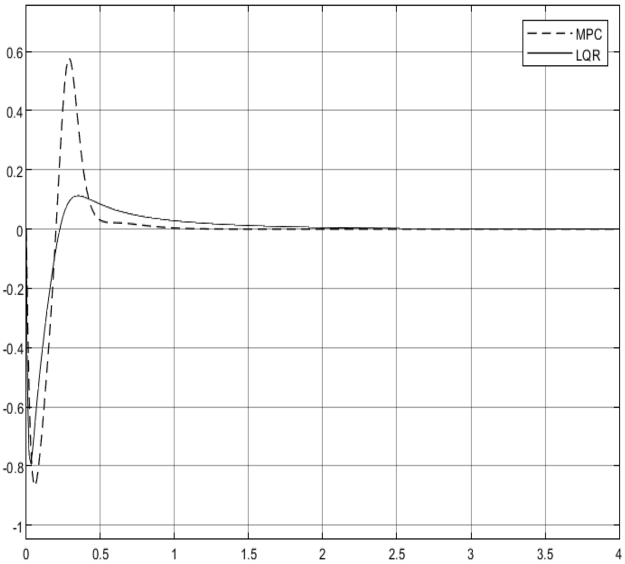


**Fig. 3.** Angular velocity of Link1 (rad/s)

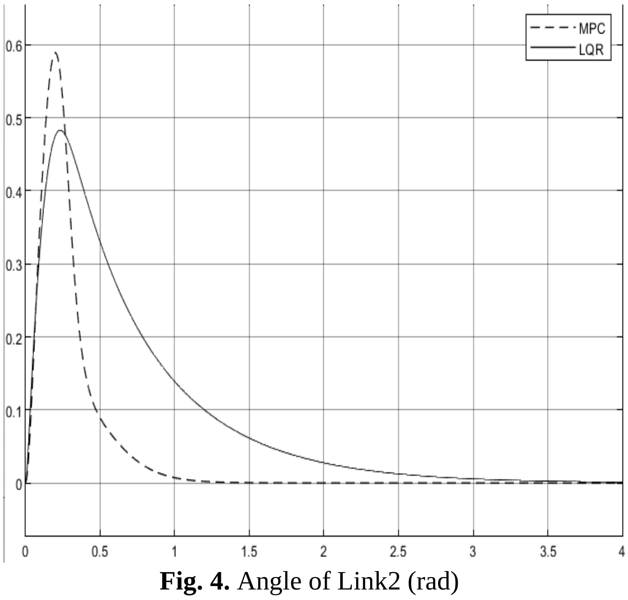


**Fig. 4.** Angle of Link2 (rad)

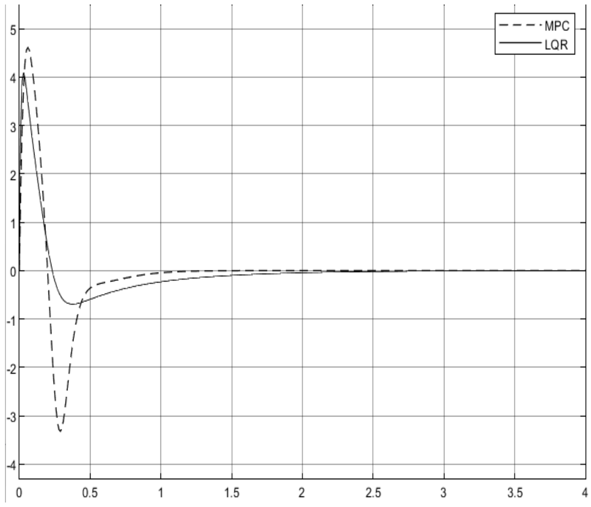


**Fig. 5** Angular velocity of Link2 (rad)

Observation:

- ✓ In Fig. 2, angle of link 1 under MPC controller has a settling time of 1s, which is faster compared to LQR at 3s. However, maximum oscillation amplitude of MPC is 0.09, which is larger than LQR at 0.065.
- ✓ In Fig. 3, velocity of link 1 quickly approaches 0 under MPC controller in 1s, faster than under LQR controller at 1.5s.
- ✓ In Fig. 4, angle of Link 2 under MPC controller has a response time of 1.1s, faster than under LQR controller at 3.5s, and peak oscillation amplitude of MPC is 0.59, greater than LQR at 0.48.
- ✓ In Fig. 5, velocity of Link 2 under MPC controller rapidly decreases to 0 after 1s, quicker than under LQR controller at 2s, and maximum oscillation amplitude of MPC's velocity is 4.5, which is greater than under LQR at 4.

## 5. Conclusion

In the context of Acrobot, both MPC and LQR offer vital control solutions for positioning and maintaining the robot's balanced position. However, MPC exhibits notable advantages, including its ability to converge rapidly, especially in situations requiring precise and meticulous control. The uniqueness of MPC, with its mechanism of predicting and optimizing the system's behavior over time, not only aids it in achieving and sustaining a balanced state but also ensures stability and high performance during extended operation periods. Notably, its flexibility and adaptability to various models and systems make MPC an attractive choice in developing control strategies for robotic systems, like Acrobot, which demand high precision and reliability.

**Acknowledgement**

We want to give thanks to PhD. Van-Dong-Hai Nguyen (Faculty of Electrical and Electronics Engineering - HCMUTE) due to his supports in giving ideas to this contribution.

## 6. References

[1] Tran H. C. et al: "Genetic algorithm implementation for optimizing linear quadratic regulator to control acrobot robotic system", Robotica & Management, Vol 23(1), p32-35, 2018.

[2] Nguyen C.D.: "Neural Networks Application for the Data of PID Controller for Acrobot", The Scientific World Journal, Vol 2022, pp. 3-6, 2022.

[3] Smith M.H. et al.: "Dynamic fuzzy control and system stability for the Acrobot", IEEE International Conference on Fuzzy Systems Proceedings. IEEE World Congress on computational intelligence, 1998.

[4] Lai X.-Z. et al: "Control of acrobot based on lyapunov function", Journal of Central South University of Technology, vol 11, pp. 210-215, 2004.

[5] Purnawan H.: "Design of Linear Quadratic Regulator (LQR) Control System for Flight Stability of LSU-05", Journal of Physics: Conference Series, 2017.

[6] Setiawan N., Pratama G.N.P.: "Application of LQR Full-State Feedback Controller for Rotational Inverted Pendulum" Journal of Physics: Conference Series, 2111(1), 012006, pp. 3-7, 2021.

[7] Askari M. et al: "Model predictive control of an inverted pendulum", IEEE, 2010.

[8] Joshi H., Paulose N.: "Discrete time model predictive control approach for inverted pendulum system with input constraints", International Journal of Electronics and Electrical Engineering, 3(3), Article 8, pp. 186-190, 2015.