Double Inverted Pendulum: Stable Control via Feedback Linearization with Python Simulation and Analysis

Bardia Mojra

***Abstract*—Stable control of *double inverted pendulum* (DIP) is a classic problem in control systems and is considered to be a good case for studying and control nonlinear systems. DIP is highly nonlinear and sensitive to the initial conditions which makes it particularly difficult to control. This article focuses on the derivation of *equations of motion* and *feedback linearization* method which, were used to develop a symbolic representation of the system. We show that the system characteristic matrix is positive definite and is invertible as required by *Lyapunov's direct method*. Using MATLAB Control Toolbox, we solve the continuous *Lyapunov control function* to obtain the *linear quadratic regulator* (LQR) solution which is considered to be unique, stable, and optimal. To enhance this case study, an end-to-end software suite is developed in Python to simulate, animate, and analyze the system. Moreover, a series of experiments are designed and performed to 1) study the effects of variations in model parameters (i.e. masses, pendulum lengths, and friction) on system stability, 2) and to investigate and present some evidence to confirm chaotic nature of the system. The source code developed for this project will be available at https://github.com/BardiaMojra/dip.**

I. Introduction

A. Related Work

The DIP on a cart problem has long been interesting for researchers and many stable and robust solutions have been introduced [1], [2], [3], [4], and [5]. There are nonlinear and adaptive solutions that perform better but are not the subject of this work. Among linear solutions which, require complete knowledge of the systm, LQR is proven to be an optimal solution. [2] and [5] derive the LQR solution for a known model and proves its optimality with analytic derivation. [4] compare LQR solutionwith a PD controller.

B. System Setup

Consider a cart resting motionless on a flat surface along x-axis, on top of which are stacked two inverted pendulums, figure 1. The pendulums resemble a robotic arm but rather without motors or external torque input. At the end of each pendulum is a point-mass, m_1, m_2. Each pendulum moves freely and independently and is connected by lengths l_1 and l_2 at, q_1 and q_2 to their point-masses, m_1, m_2. The mass of the cart is denoted by m_c with its position x along the x-axis. θ_1 and θ_2 denote angular deviations from upright position for q_1 and q_2, respectively.

C. State Definition

Suppose we have a continuous-time, nonlinear system presented in state space form. q is the state vector and has 3x1 dimensions. Function f is the state transition function and u is the input to the system. Function h is the state observation function.

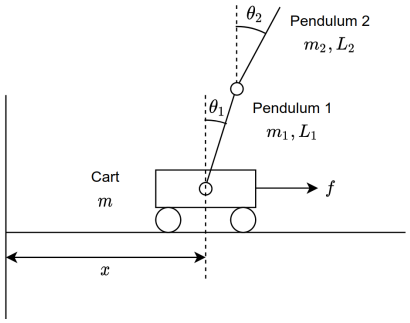


Fig. 1. Physical System

$$\dot{q} = f(q, u), \tag{1}$$

$$y = h(q). \tag{2}$$

Our goal is to stabilize the pendulums in the upright position and with the pendulum angles close to zero.

D. Noise and Friction

To keep the project simple, noise is neglected but in reality there are always multiple types of noise observed in measurement. Since we are using a simulation environment there is no noise but the system still behaves chaotically. Friction and damping terms are represented in the dynamic model with linear ansatz.

E. Global Coordinate Frame

We need to form a homogenous configuration space to present state variables x, θ_1, θ_2. We this global coordinate frame by defining the following relative transforms. We assume $q(t = 0)$ is at the origin at the start of the simulation.

$$x = \begin{bmatrix} x \\ 0 \end{bmatrix}, \quad q_1 = \begin{bmatrix} x + l_1 \sin\theta_1 \\ l_1 \cos\theta_1 \end{bmatrix}, \\ q_2 = \begin{bmatrix} x + l_1 \sin\theta_1 + l_2 \sin\theta_2 \\ l_1 \cos\theta_1 + l_2 \cos\theta_2 \end{bmatrix}, \tag{3}$$

We denote position of point-masses for m_c, m_1, and m_2 by $x, q_1, q_2 \in R^2$ on the x-y plane. We seek to define our system

in terms of these state variables. Thus, we define state variable vector $q(t)$ as

$$q = [x, q_1, q_2]^T \tag{4}$$

For simpler representation, we denote time-dependant state variable vector as $q := q(t)$.

II. Euler-Lagrangian Model

A. Equations of Motion

Per *Lagrangian mechanics*; first we derive the *equations of motion* by modeling of the physical system and find the *Lagrangian*, $\mathcal{L} = K - P$. K and P represent *kinetic* and *potential* energies of the system. K is the total kinetic energy from all three masses. Here, we define K as

$$K = \frac{1}{2}mv^2 = \frac{1}{2}m_c\|\dot{x}\|^2 + m_1\|\dot{q_1}\|^2 + m_2\|\dot{q_2}\|^2. \tag{5}$$

P denotes the overall potential energy of the system; but since the cart rolls flat along the x-axis, it is inherently incapable of storing energy so we discard it. P is obtained by

$$P = g\left[m_1 l_1 \cos\theta_1 + m_2(l_1\cos\theta_1 + l_2 cos\theta_2)\right] \tag{6}$$

Where $g \in R$ denotes gravitational acceleration. Next, we use the *Euler-Lagrange* equation to write the following ODE system.

The Lagrange Equations of Motions are defined as

$$Q = \frac{\partial \mathcal{L}}{\partial f_i} - \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{f}_i}\right). \tag{7}$$

Where Q represents the vector of sum of external forces acting on the plan. We solve for an *linearized* solution by setting Q to zero. This is also called the *zero dynamics* of the system.

$$\mathcal{L} = \mathcal{L}(t, q, \dot{q}) \tag{8}$$

$$\begin{cases} u - d_1\dot{q} &= \frac{d}{dt}\{\frac{\partial \mathcal{L}}{\partial \dot{q}}\} - \{\frac{\partial \mathcal{L}}{\partial q}\} \\ -d_2\dot{\theta_1} &= \frac{d}{dt}\{\frac{\partial \mathcal{L}}{\partial \dot{\theta_1}}\} - \{\frac{\partial \mathcal{L}}{\partial \theta_1}\} \\ -d_3\dot{\theta_2} &= \frac{d}{dt}\{\frac{\partial \mathcal{L}}{\partial \dot{\theta_2}}\} - \{\frac{\partial \mathcal{L}}{\partial \theta_2}\} \end{cases} \tag{9}$$

So far our derivation has made our expressions more complicated but these steps are necessary for capturing the full dynamic characteristics of the system. In the following section, we linearize the continuous-time nonlinear model about the upright position.

B. Continues-Time, Nonlinear Model

We used MATLAB Symbolic Math Toolbox to derive the resultant ODE. Obtaining symbolic or implicit expressions of our ODE will allow us to model a general DIP system.

$$\begin{aligned} u - d_1\dot{q} &= (m_c + m_1 + m_2)\ddot{q} + l_1(m_1 + m_2)\ddot{\theta_1}\cos\theta_1 \\ &\quad + m_2 l_2\ddot{\theta_2}\cos\theta_2 - l_1(m_1 + m_2)\dot{\theta_1}^2\sin\theta_1 \\ &\quad - m_2 l_2\dot{\theta_2}^2\sin\theta_2 \\ -d_2\dot{\theta_1} &= l_1(m_1 + m_2)\ddot{(q)}\cos\theta_1 + l_1^2(m_1 + m_2)\ddot{\theta_1} \\ &\quad + l_1 l_2 m_2\ddot{\theta_2}\cos(\theta_1 - \theta_2) + l_1 l_2 m_2\dot{\theta_2}^2\sin(\theta_1 - \theta_2) \\ &\quad - g(m_1 + m_2)l_1\sin\theta_1 \\ -d_3\dot{\theta_2} &= l_2 m_2\ddot{q}\cos\theta_2 + l_1 l_2 m_2\ddot{\theta_1}\cos(\theta_1 - \theta_2) + l_2^2 m_2\ddot{\theta_2} \\ &\quad - l_1 l_2 m_2\dot{\theta_1}^2\sin(\theta_1 - \theta_2) - l_2 m_2 g\sin\theta_2 \end{aligned} \tag{10}$$

C. Linearization

First, we rewrite the equations of motion into matrix form to batch acceleration, velocity, and position terms into separate matrices. A second order representation of the physical system is sufficient to provide an accurate enough estimate, per *Taylor Series Linearization* method. This is an important step as identifying system dynamics lies at the heart of *Optimal Control*.

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = Hu \tag{11}$$

where matrix functions $D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q)$ represent systems dynamics with respect to time, current state and previous state. Time variable t is omitted to simplify notation. The system energy state (left hand side of above EOM) is presented in relation to external forces Hu acting on the system. Vector $u \in R^1$ represents external input forces vector and matrix H represents its relationship with state variable functions. Matrix functions $D(q),\ C(q, \dot{q}),\ G(q)$ and vector H are define as presented in [2].

We need to rewrite the EOM in the following matrix form to obtain the final ODE. Per *Lyapunov Control Function theory* (LCF) for autonomous dynamical systems and *LaSalle's*, we can assume there exists a *Optimal Control Trajectory* or *Hamiltonian Flow* for a valid initial condition if the LCF is a *positive definite* and *positive semi-definite* energy function. Moreover, the energy function describing the system must be *symmetric* and *positive definite* in order to form *stable* control trajectory.

$$M(q)\dot{q} = A(q)q + Bu \tag{12}$$

Where *positive definite* or *positive semi-definite* constraints are evaluated by computing the determinant of M.

$$det(M) > 0;\ \forall q \tag{13}$$

Such that, the final ODE can be written as

$$\dot{q} = M(q)^{-1}(A(q)q + Bu) \tag{14}$$

where A and B are defined as

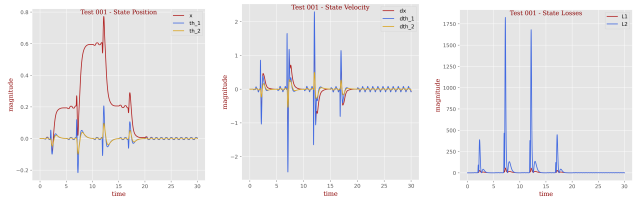
$$A = \begin{bmatrix} 0 & I \\ D(^{-1}\frac{\partial G}{\partial q} & 0 \end{bmatrix} \tag{15}$$

$$B = \begin{bmatrix} 0 \\ D^{-1}H \end{bmatrix} \tag{16}$$

D. Linear Quadratic Regulator Solution

So far, we have modeled, linearized, and discretized our system using state space general form. In the process, we systematically reduced the solution space complexity by replacing time t and velocity dimension as linear products of current and previous states. This allows us to analytically derive a general form optimal solution. But before we reach that step, we need to define a general *Cost Function* J that only depends on absolute minimal parameters, which we will later optimize to arrive to a *minimal surface*. Quadratic performance index J is defined as

$$J(x, u) = \frac{1}{2}x^T Q x + \frac{1}{2}u^T R u \tag{17}$$



(a) $q(t)$ (b) $\dot{q}(t)$ (c) $J(t)$

Fig. 2. Test 001

whose solution is the minimal cost or minimum-energy input required for the system to reach local minima, for any given initial state. Next, we minimize J which will yield the state feedback weight vector K and is obtained by recursively solving the dynamic *Riccati* equation. It is important to note that calculation of Quadratic losses are carried out by performing dot product of state vector with its transpose and energy magnitude.

III. Experiments

A series of 18 tests are performed to investigate nonlinearities and stability of the mentioned DIP system with a linear closed-loop feedback controller. The tests are divided to two experiments; *experiment I* investigates system dynamics and LQR stability, given tuned and untuned control vector (eigenvalues), *experiment II* investigates the chaotic nature of the system. Moreover, a brief description of the simulation and animation implementation is provided.

Each test was simulated and animated in real-time to ensure both validity and diversity of tests for both experiments. Moreover, for each test the simulation and controller data, L1 and L2 losses, and all relevant test configurations are systematically captured and stored under `out` and `config` directories on the provided repository. The data for each test is analyzed and presented by 3 plots, *state position*, *state velocity*, and *state losses*. Test 001, figure 2, is the control for all tests performed in both experiments. Control vector K used in all experiments are tuned using `lyap()` function from MATLAB and parameters from Test 001 configuration were used.

It is important to note that, in order to construct a more appropriate analytic tools, we decided to convert radians to degrees and omit cart position values when computing post-precessing loss functions. This is not to be confused with the LQR solution which uses quadratic errors (similar to L2). The reason for changing from radians to degrees is that, for small changes (less than 1) a quadratic loss function quadratically decreases the magnitude instead of increasing it.

A. Simulation and Animation

For this project, both MATLAB and Python were used at different stages of the development. Besides symbolic math derivation, MATLAB Control Toolbox was used to check Python ODE simulation output and to obtain the LQR solution. MATLAB is the gold-standard of scientific numerical analysis and Python is versaltile and rich with high level feature.

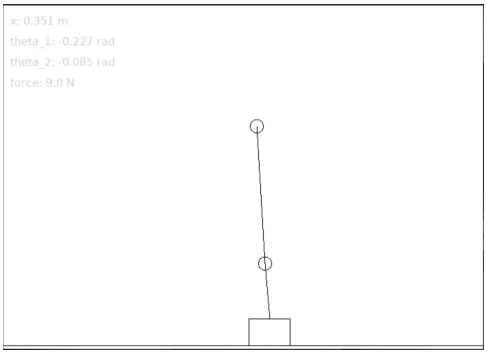


Fig. 3. Animation of Double Inverted Pendulum

Various Python-based control system libraries were explored and considered but were abandoned after observing variations in the *control vector* parameters. The LQR solution is expected to be optimal for *Lyapunov energy systems* and variations in the solution raises doubts about numerical integrity of the computation process. For that reason, derivations and control solutions used in this work are consistent with MATLAB.
Various Python libraries are used to simulate, animate and analyze model dynamics. Pymunk and Pygame libraries are used create a 2D physics-based simulation environment. Pyglet library (based on OpenGL) is used to create the animation, figure 3. Source code for the animation and graphics are base on examples and documentation provided by the authors [6] [7].

B. Experiment I

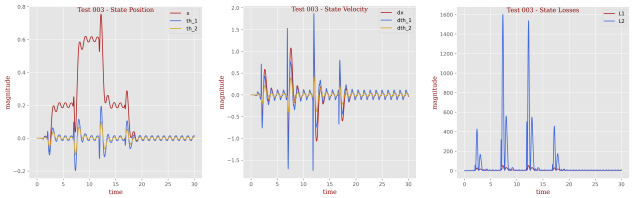
In the first experiment, we perform a set of 14 tests with the goal of studying system stability with respect to changes in model parameters. The control vector, K, obtained from MATLAB is tuned for model parameters used in Test 001 (figure 2). This control vector is used without retuning for all 14 tests in this experiment. Moreover, the duration of the test, friction and initial momentums are 30s, 0.2, and $0.001N/m$, respectively. These values remained the same for all 14 tests. We changed model parameters whose dynamics play a dominant role in the overall system dynamics. Thus, model parameters l_1, m_1, l_2, and m_2 were changed one at the time to see how each affects system dynamics and controller stability. Table 1 I presents model parameters used for each test and changes from each test to the following is highlighted in **bold**.

Test 001 proves *practical stability* of LQR solution obtained from MATLAB by bounding θ_1 and θ_2 to small positive magnitude angle close to zero degrees. There is no noise added to the simulation but quantization error itself acts as process noise. This noise is significant enough to make the simulated system behave chaotically similar to a physical system. We can only achieve *Uniformly Ultimately Bounded* (UUB) with a linear controller, figure 2.

Tests 002-006 and 012 resulted in UUB stability and were bounded close to zero with various bounds. Test 007 and 0013 were interesting due to their unique results. Test 007 has heavier top and starts with a tighter bound and lower frequency but

TABLE I
EXPERIMENT I

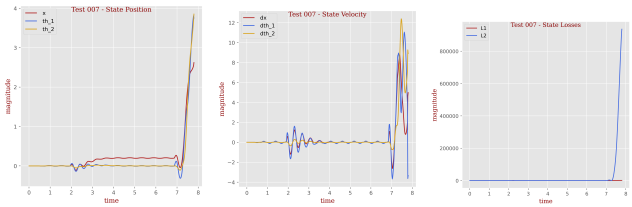
Test ID	l_1	m_1	l_2	m_2	stability
001	**0.6**	**0.2**	**0.6**	**0.2**	UUB
002	0.6	**0.3**	0.6	0.2	UUB
003	0.6	**0.4**	0.6	0.2	UUB
004	0.6	0.4	0.6	**0.3**	UUB
005	0.6	0.4	0.6	**0.4**	UUB
006	0.6	**0.3**	0.6	0.4	UUB
007	0.6	**0.2**	0.6	0.4	marginal-UUB
008	0.6	0.2	**0.5**	0.2	unstable
009	0.6	0.2	**0.7**	0.2	unstable
010	0.6	0.2	**0.8**	0.2	unstable
011	**0.5**	0.2	**0.6**	0.2	UUB
012	**0.4**	0.2	0.6	0.2	UUB
013	**0.3**	0.2	0.6	0.2	marginal-UUB
014	**0.7**	0.2	0.6	0.2	unstable



(a) $q(t)$ (b) $\dot{q}(t)$ (c) $J(t)$

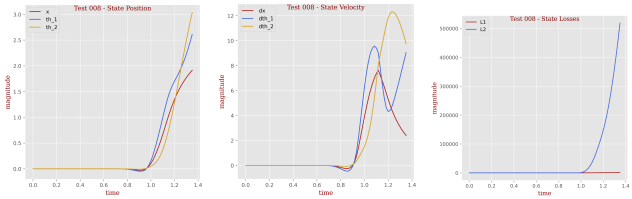
Fig. 4. Test 003

after a few seconds it suddenly becomes unstable. Test 0013 has shorter lower pendulum and moved violently from left to right to reach stability and remained bounded throughout the duration of the test. Both cases are at the margin of UUB stability; thus, we named it *marginal-UUB*. Tests 008-010 and 014 were unstable.



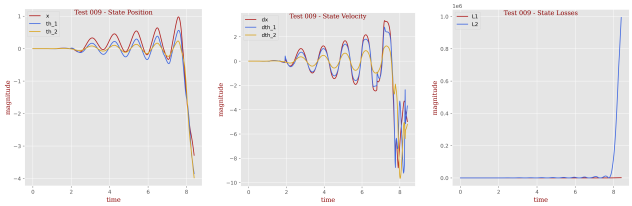
(a) $q(t)$ (b) $\dot{q}(t)$ (c) $J(t)$

Fig. 5. Test 007



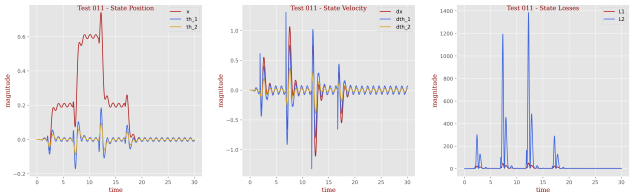
(a) $q(t)$ (b) $\dot{q}(t)$ (c) $J(t)$

Fig. 6. Test 008



(a) $q(t)$ (b) $\dot{q}(t)$ (c) $J(t)$

Fig. 7. Test 009



(a) $q(t)$ (b) $\dot{q}(t)$ (c) $J(t)$

Fig. 8. Test 011

TABLE II
EXPERIMENT II

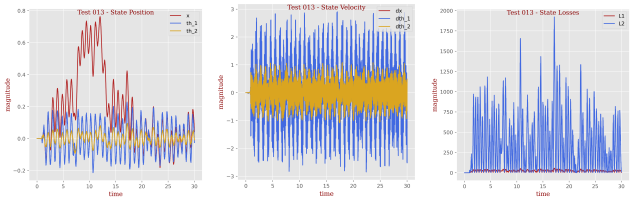
Test ID	duration	sum(L1)	sum(L2)
Test 101	20.0	6596.9514	71827.5496
Test 102	20.0	6596.9514	71827.5496
Test 103	**100.0**	13095.306	77780.4293
Test 104	100.0	**13096.591**	**77830.3157**

C. *Experiment II*

In the second experiment, we perform a set of 4 tests with the goal of discoving some evidence of *chaotic nature* of the system. Although all multi-body physical systems are chaotic in nature, they vary in degree and it often depends on system nonlinearities. We wanted to see how fast repeated experiments start to contradict previous tests with the same configurations and initial conclusions.

IV. CONCLUSION

In this work, we studied the dynamics of a double inverted pendulum on cart system and examined its nonlinearities with an optimal LQR controller. We designed this linear controller with complete knowledge of the system and were only able to achieve UUB stability. This is indicative of how nonlinear the



(a) $q(t)$ (b) $\dot{q}(t)$ (c) $J(t)$

Fig. 9. Test 013

DIP system is. Moreover, the high degree system nonlinearity manifests itself in form of *chaos* and makes the system unrepeatable only after 100 seconds.

ACKNOWLEDGMENT

We want to thank Dr. Frank Lewis for his great lectures, and notes, books, and materials he made available us (students) throughout the semester.

REFERENCES

[1] C. A. Ibanez, O. G. Frias, and M. S. Castanon, "Lyapunov-based controller for the inverted pendulum cart system," *Nonlinear Dynamics*, vol. 40, no. 4, pp. 367–374, 2005.

[2] A. Bogdanov, "Optimal control of a double inverted pendulum on a cart," *Oregon Health and Science University, Tech. Rep. CSE-04-006, OGI School of Science and Engineering, Beaverton, OR*, 2004.

[3] T. DABRETAU and A. DAREINI, "Control of double inverted pendulum first approach," 2015.

[4] I. J. Crowe-Wright, "Control theory: The double pendulum inverted on a cart," 2018.

[5] I. M. Mehedi, U. Ansari, and U. M. AL-Saggaf, "Three degrees of freedom rotary double inverted pendulum stabilization by using robust generalized dynamic inversion control: Design and experiments," *Journal of Vibration and Control*, vol. 26, no. 23-24, pp. 2174–2184, 2020.

[6] V. Blomqvist, "Pymunk: A easy-to-use pythonic rigid body 2d physics library (version 6.2.1)," 2007.

[7] P. Shinners, "Pygame." http://pygame.org/, 2011.