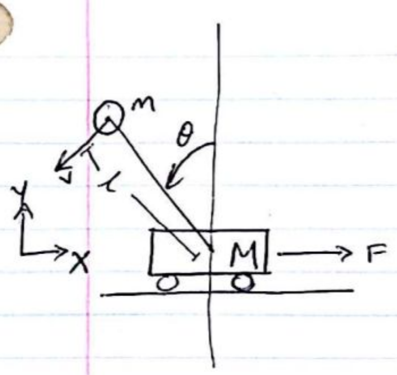
BALANCING AN INVERTED PENDULUM

Nikhil Advani

I chose to balance the inverted pendulum on a cart i.e. I used a cart-pole system. In this report, I start by showing the derivation of the dynamics, followed by the derivation and results of applying the control law using LQR, and then PID.

DYNAMICS OF THE CART-POLE SYSTEM:

The dynamic equations are derived using the Euler Lagrange equations as shown below:



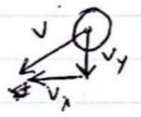
Using Euler-Lagrange to derive the dynamic equations:

$$L = KE - PE$$

$$PE = mgl\cos\theta$$

$$KE = KE_{cart} + KE_{pendulum}$$

$$= \frac{1}{2}M\dot{x}^2 + \cancel{\frac{1}{m}}\ \frac{1}{2}mv^2$$



$$= \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(v_x^2 + v_y^2\right)$$

$$= \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(\dot{x} - \dot{\theta}l\cos\theta\right)^2 + \frac{1}{2}m\left(-\dot{\theta}l\sin\theta\right)^2$$

$$= \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(\dot{x}^2 - 2\dot{x}\dot{\theta}l\cos\theta + \dot{\theta}^2 l\cos^2\theta\right) + \frac{1}{2}m\dot{\theta}^2 l^2\sin^2\theta$$

Let $\cos\theta = c\theta$

$\sin\theta = s\theta$

$$KE = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\dot{x}^2 - m\dot{x}\dot{\theta}lc\theta + \frac{1}{2}\dot{\theta}^2 mlc^2\theta + \frac{1}{2}\dot{\theta}^2 mls^2\theta$$

$$= \frac{1}{2}\dot{x}^2(M+m) - m\dot{x}\dot{\theta}lc\theta + \frac{1}{2}\dot{\theta}^2 ml\left(c^2\theta + s^2\theta\right)$$

$$KE = \frac{1}{2}\dot{x}^2(M+m) - m\dot{x}\dot{\theta}lc\theta + \frac{1}{2}\dot{\theta}^2 ml$$

$$L = KE - PE$$

$$\boxed{L = \frac{1}{2}\dot{x}^2(M+m) - m\dot{x}\dot{\theta}lc\theta + \frac{1}{2}\dot{\theta}^2 ml - mglc\theta} \quad — ①$$

<u>Euler Lagrange equations:</u>

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

$$\frac{\partial L}{\partial \dot{x}} = \dot{x}(M+m) - m\dot{\theta}lc\theta$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) = (M+m)\ddot{x} - (m\ddot{\theta}lc\theta \ast m\dot{\theta}ls\theta\dot{\theta})$$

$$= (M+m)\ddot{x} - m\ddot{\theta}lc\theta + m\dot{\theta}^2 ls\theta$$

$$\frac{\partial L}{\partial x} = 0$$

Thus

$$\boxed{(M+m)\ddot{x} - m\ddot{\theta}lc\theta + m\dot{\theta}^2 ls\theta = F} \quad — ②$$

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

$$\frac{\partial L}{\partial \dot\theta} = -m\dot{x}\,l c\theta + m\dot\theta\, l^2$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot\theta}\right) = -\left(m\ddot{x}\,l c\theta - m\dot{x}\,l s\theta\,\dot\theta\right) + m\ddot\theta\, l^2$$

$$= -m\ddot{x}\,l c\theta + m\dot{x}\dot\theta\, l s\theta + m\ddot\theta\, l^2$$

$$\frac{\partial L}{\partial \theta} = m\dot{x}\dot\theta\, l s\theta + \text{[illegible]}\ mgls\theta$$

Thus,

$$-m\ddot{x}\,l c\theta + \cancel{m\dot{x}\dot\theta\, l s\theta} + m\ddot\theta\, l^2 - \cancel{m\dot{x}\dot\theta\, l s\theta} - mgls\theta = 0$$

$$-\cancel{m}\ddot{x}\,\cancel{l}c\theta + \cancel{m}\ddot\theta\,l^{\cancel{2}} - \cancel{m}g\cancel{l}s\theta = 0$$

$$\boxed{-\ddot{x}\,c\theta + \ddot\theta\, l - g s\theta = 0} \quad - ③$$

② and ③ describe the dynamic equations of the system.

Solving equations ② and ③:

$$\boxed{\ddot{x} = \frac{1}{(M+ms^2\theta)}(mgs\theta c\theta - m\dot{\theta}^2 ls\theta + F)} \quad — ④$$

$$\boxed{\ddot{\theta} = \frac{1}{l(M+ms^2\theta)}(M+m)gs\theta - m\dot{\theta}^2 ls\theta c\theta + Fc\theta} \quad —⑤$$

These 2nd order differential equations could be solved using ode45 in MATLAB, but since I am using Python, I decided to use:

$$\ddot{x}_t = \frac{\dot{x}_t - \dot{x}_{t-1}}{t-(t-1)}$$

$$\ddot{x}_t = \frac{\dfrac{(x_t - x_{t-1})}{(t-(t-1))} - \dfrac{(x_{t-1} - x_{t-2})}{((t-1)-(t-2))}}{t-(t-1)}$$

$$x_t = x_{t-1} + \left[a(t-(t-1))^2 + \left(\frac{x_{t-1} - x_{t-2}}{(t-1)-(t-2)}\right)(t-(t-1))\right] \quad ⑥$$

I had started by assuming that (t-(t-1)) = ((t-2)-(t-1)). Experimentally, I found out that this assumption does not hold.

I used equation 6, with the results from equations 4 and 5, to find the values of x(t) and θ(t) respectively.

As a sanity check, to ensure that the dynamics of the system are correct, I simulated the system without any force input to the cart. A video of this can be seen in ‘no_force_input.mp4’. The simulator environment has been developed using the OpenCV library in Python. The other libraries used have been described in the README file.

CONTROL LAW USING LINEAR QUADRATIC REGULATORS (LQR):

This is an optimal control technique where the pole placement is optimized (given some penalty for the state not being at the desired state and cost of applying a control input).

The system must be represented as $\dot{x}_s = Ax_s + Bu$, since A and B are required to calculate the optimal pole placement. This controller can only be used on systems that have linear dynamics, and thus the dynamic equations need to be linearized. Since the pendulum needs to be balanced vertically I have used Jacobian linearization at θ = 0.

$$\frac{d}{dt} x_s = \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \boxed{eq^n 4} \\ \dot{\theta} \\ \boxed{eq^n 5} \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}$$

Now that we have f_1, f_2, f_3 and f_4, we can calculate the Jacobian as..

$$A = \left. \frac{\partial f}{\partial x_s} \right|_{\theta=0}$$

$$= \left. \begin{bmatrix} \partial f_1/\partial x & \partial f_1/\partial \dot{x} & \partial f/\partial \theta & \partial f/\partial \dot{\theta} \\ \partial f_2/\partial x & \ddots & & \\ \partial f_3/\partial x & & \ddots & \\ \partial f_4/\partial x & & \cdots & \partial f_4/\partial \dot{\theta} \end{bmatrix} \right|_{\theta=0}$$

After taking the partial derivative and substituting the value of θ = 0, we get the following:

$$\text{Thus, } A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{mg}{M} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{(M+m)g}{ML} & 0 \end{bmatrix}$$

And since the input is the Force applied to the cart. Thus, u = F. Thus, we get B..

$$B = \begin{bmatrix} 0 \\ 1/M \\ 0 \\ 1/LM \end{bmatrix}$$

(This are the same values of A and B that we would have got if we would have just assumed θ is almost 0).

The cost function to be minimized is given by:

$$J = \int_0^\infty \left(x^T Q x + u^T R u \right) dt$$

$$\text{where} \quad u(t) = -\left(K x_s(t) - x_d \right)$$

Where x_s is the current state and x_d is the desired state. Q is a positive semi-definite matrix that represents the penalty if the state variables are not at the desired value. In this example, since we have 4 state variables, Q is a 4x4 matrix. The R matrix represents the cost of applying different control inputs. In this example, since we have only 1 control input, this is a 1x1 matrix.

The LQR finds the value of K. This is done by solving the Riccati equation for P.

$$A^T P + PA + Q - P R^{-1} B^T P = 0$$

$$\text{and } K \text{ is} \qquad K = R^{-1} B^T P.$$

Empirically, the value of Q is found out to be:

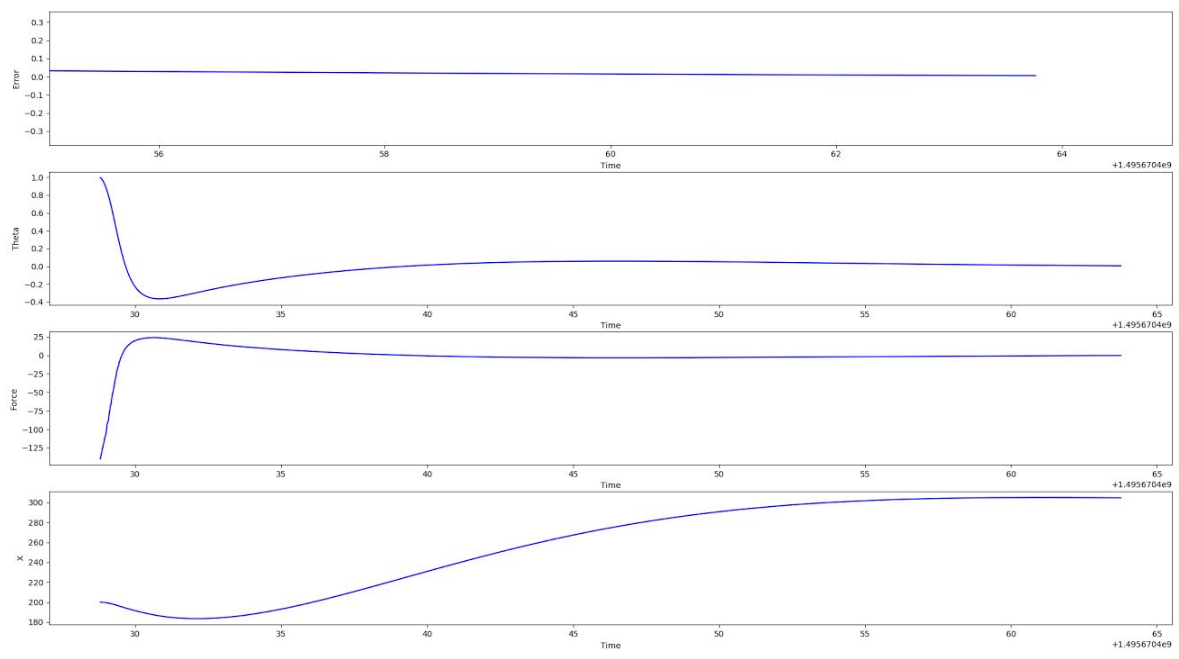
Q = [[10,0,0,0]
[0,1,0,0]
[0,0,10000,0]
[0,0,0,100]]

This reflects the fact that the highest priority is placed on θ reaching its desired value. Since a desired state needs to be specified, the desired value for x was chosen to be 300. Being able to specify desired values for numerous state variables is a big advantage of modern control techniques.

Results:

Since the Jacobian linearization has been done about θ = 0, the controller is designed to work only if θ = 0 (i.e. -1 radian < θ < 1 radian). For the results presented below, the pendulum starts at 1 radian. The graphs presented are:

- error vs time,
- theta value (of the pendulum) vs time,
- force input vs time,
- X position of the cart vs time.



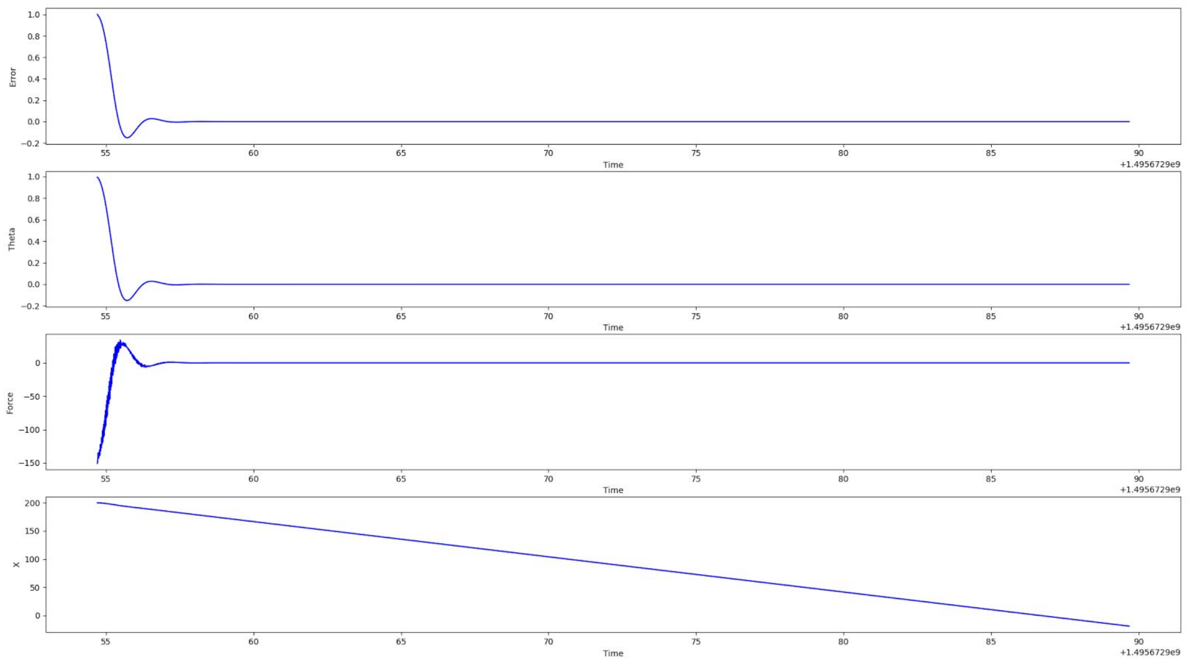
As seen in the results (x vs time), when the simulation starts, the cart moves away from its desired position, since a greater importance is given to getting the pendulum to its desired position. So the controller does what is needed to get the pendulum angle to 0, and puts much lesser importance on getting the cart to its desired position. Once the pendulum is in its desired position, the carts moves slowly (to make sure that the pendulum maintains its angle) to its desired position.

CONTROL LAW USING PROPORTIONAL-INTEGRAL-DERIVATIVE (PID) CONTROLLER:

After the dynamics of the system were derived, and once the error was calculated (while accounting for the ‘wrap around’), I empirically tuned the gains for the proportional, integral, and derivative terms. Since PID is a classical control method, we can use it to control only 1 state variable (In this case, we are using it to control theta, and we have no control over the position of the cart). It can also be used with only 1 input (which is good in this case since we have only 1 input).

For the results presented below, the pendulum starts at 1 radian. The graphs presented are:
- error vs time,
- theta value (of the pendulum) vs time,
- force input vs time,
- X position of the cart vs time.

From the results it is clear that the error in pendulum angle quickly converges to 0. It does oscillate a tiny about and this can be eliminated by further tuning it. It is interesting to note that x continues to change slightly even after the error of the pendulum angle is 0. And this cannot be explicitly controlled. It can be eliminated by further tuning the gain values.



CONCLUSION:

- I used Euler-Lagrange to derive the dynamic equations of a cart-pole system. I encoded these dynamics in a simulator developed in Python using the OpenCV libraries.
- I implemented 2 controls laws to balance the pendulum i.e. LQR and PID. For our application both would do, but LQR has advantages like being able to find the optimal input to take numerous state variables to their desired position. PID is limited by just being able to take 1 state variable to its desired position.

OTHER COOL THINGS THAT COULD HAVE BEEN TRIED:

- Use the K matrix from LQR to find the gains for the PID.
- Use Zeigler-Nichols or Twiddle to auto-tune the PID gains.
- Use Feedback Linearization so that the LQR could be used for any initial θ.
- If the initial position is closed to pi, use a energy based controller to swing the pendulum upwards and then the LQR can kick in to balance it at 0 degrees. (http://www.quanser.com/Content/CoursewareNavigators/qubeservo_matlab/Courseware/Ancillary%20Material/Profiles/Astrom/Swing-up%20Pendulum%20Energy%20Control.pdf)
- Add a constraint on the movement of the cart along x.
- Using Reinforcement Learning to balance the inverted pendulum.