# Simulation and Comparison Between Controls for Two-Wheeled Self-Balancing Robot in MATLAB/SIMULINK

**Ranish Devkota**
**Gokarna Baskota**

Tribhuvan University
Institute of Engineering
Dept. of Electronics and Computer Engineering
Purwanchal Campus (Eastern Regional Campus)

## ABSTRACT

A two-wheeled self-balancing robot is an application of the classical inverted pendulum problem of a control system. In this robot, it is obligatory to study control theory and find the best suitable controls for our system. Moreover, we have to study the behavior of current in the DC motor models to verify the Control system and end effector trade-off. In this project, we have chosen PID controllers and applied them in various ways, forming various control loops for stabilizing our system and at last studied behavior for one of the optimal controllers called LQR by simulating in MATLAB.

**Keywords:** Matrix Laboratory(MATLAB), Proportional Integral Derivative(PID), Linear Quadratic Regulator(LQR), Direct Current(DC), single input single output(SISO).

# 1. Introduction

A Two-Wheeled self-balancing robot is an unstable design where the robot has to act constantly to get its position upright. For balancing the system has to be controllable and observable. The self-balancing robot is the upgraded version of the inverted pendulum. The robot can balance its position even if an external disturbance is applied to it. This system is used in many daily applications. Here, we are going to compare the response and behavior of robots between different arrangements, number and feedback loops of PID controllers and also see current and speed characteristics of DC motors for those arrangements. At last, we will also see the implementation of one of the optimal controller LQR. And compare among all those arrangements of controllers to find the best among those for balancing our unstable robot.

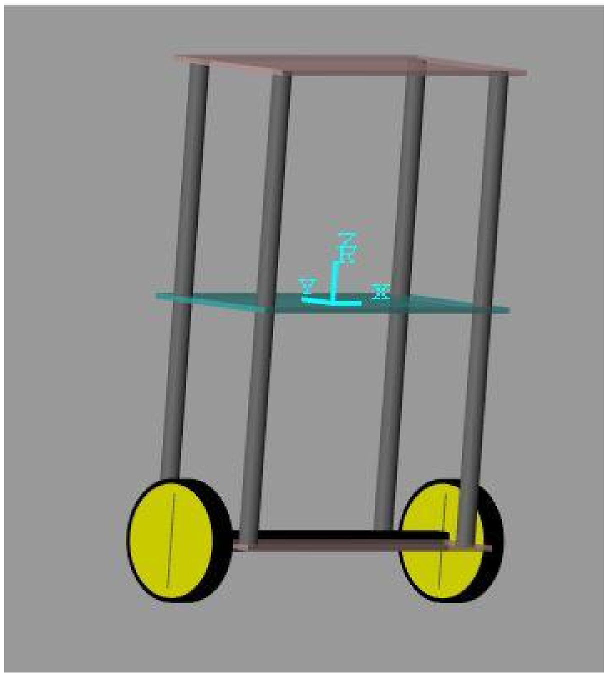


Fig: 1.1 Robot 3D model

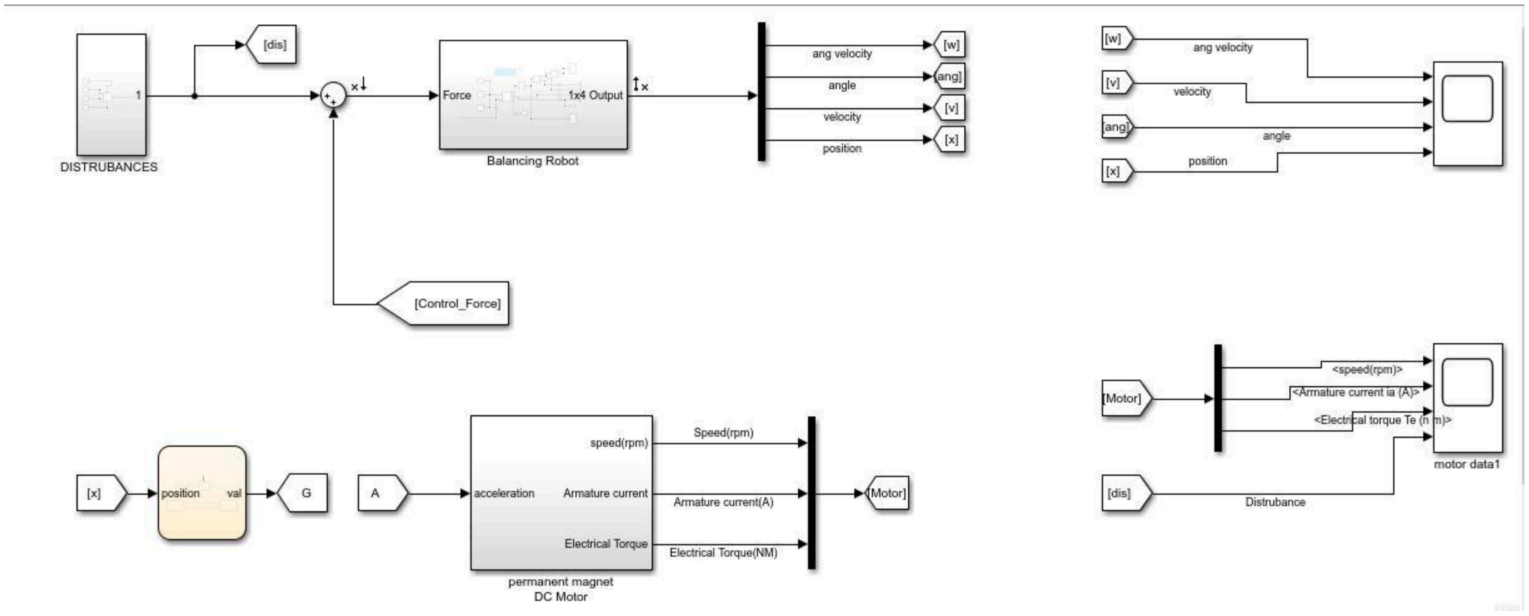


Fig: 1.2 Robot Simulink model

## 2. Specification of our Robot

| | |
|---|---|
| Mass of the chassis(M) | 1 Kg |
| Mass of the wheels and shaft(m) | 0.2 kg |
| Height of Robot(H) | 0.25 m |
| length to pendulum center of mass(l) | 0.125 m |
| Moment of inertia of the pendulum(**I**) | 0.0005 kg.m^2 |
| Estimate of viscous friction coefficient | 0.1 N.m.sec |

Table: 1 Robot Parameters

### 2.1 Mathematical Model

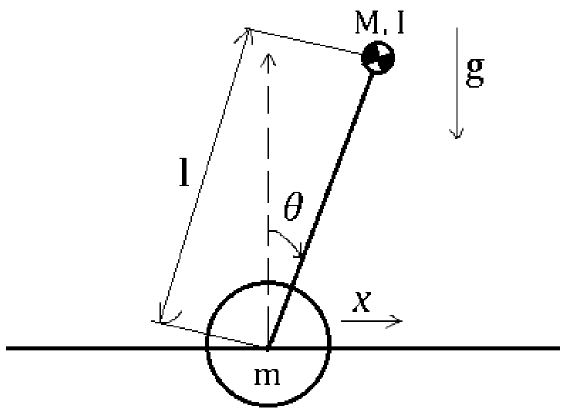


Fig:-2.1 Free body diagram for inverted pendulum on a cart

The system consists of two-part upper chases as a pendulum and lower wheeled part as a Cart of the inverted pendulum. The lower body has two-wheel and one shaft which is connected to the chases by a prismatic joint. This is only mechanical modeling of the self-balancing robots, there are no electric motors to run wheels.

We are just modeling a robot and using the different controllers (PID and LQR) and studying the robot's performance.

Consider the robot as a two-part pendulum and the cart (wheels), which reduce the degree of freedom. Both cart and pendulum have one degree of freedom (position x, theta $\theta$).
Now model Newton's equation for these two degrees of freedom.

$$\frac{d^2x}{dt^2} = \frac{1}{m}\sum_{cart} F_x = \frac{1}{m}\left(F - N - b\frac{dx}{dt}\right)$$

$$\frac{d^2\theta}{dt^2} = \frac{1}{I}\sum_{cart} \tau = \frac{1}{I}\left( Nl\cos(\theta) - Pl\sin(\theta)\right)$$

However, it is necessary to include the interaction forces N and P between the cart and the pendulum to study using the model of the dynamics. The inclusion of these forces requires modeling the x and y dynamics of the pendulum in addition to its theta dynamics. Generally, we would like to exploit the modeling power of Simulink and let the simulation take care of the algebra. Therefore, we will model the additional x and y equations for the pendulum.

$$M\frac{d^2x_p}{dt^2} = \sum_{pend} F_x = N$$

$$\therefore N = M\frac{d^2x_p}{dt^2}$$

$$M\frac{d^2y_p}{dt^2} = \sum_{pend} F_y = P - Mg$$

$$\therefore P = M\left(\frac{d^2y_p}{dt^2} + g\right)$$

Where $X_p$ and $Y_p$ are functions of theta.

So we can write values as,

$$x_p = x - l\sin(\theta)$$
$$\frac{dx_p}{dt} = \frac{dx}{dt} - l\cos(\theta)\,\frac{d\theta}{dt}$$
$$\frac{d^2x_p}{dt^2} = \frac{d^2x}{dt^2} + l\sin(\theta)\left(\frac{d\theta}{dt}\right)^2 - l\cos(\theta)\,\frac{d^2\theta}{dt^2}$$

$$y_p = l\cos(\theta)$$
$$\frac{dy_p}{dt} = -l\sin(\theta)\,\frac{d\theta}{dt}$$
$$\frac{d^2y_p}{dt^2} = -l\,cos(\theta)\left(\frac{d\theta}{dt}\right)^2 - l\,sin(\theta)\,\frac{d^2\theta}{dt^2}$$

Substitute these values to get,

$$N = m(\ddot{x} - l\dot{\theta}^2\sin\theta + l\ddot{\theta}\cos\theta)$$
$$P = m(l\dot{\theta}^2\cos\theta + l\ddot{\theta}\sin\theta + g)$$

We don’t need to solve all the values here, Simulink takes care of this algebra.

## 2.2 Simulink Model

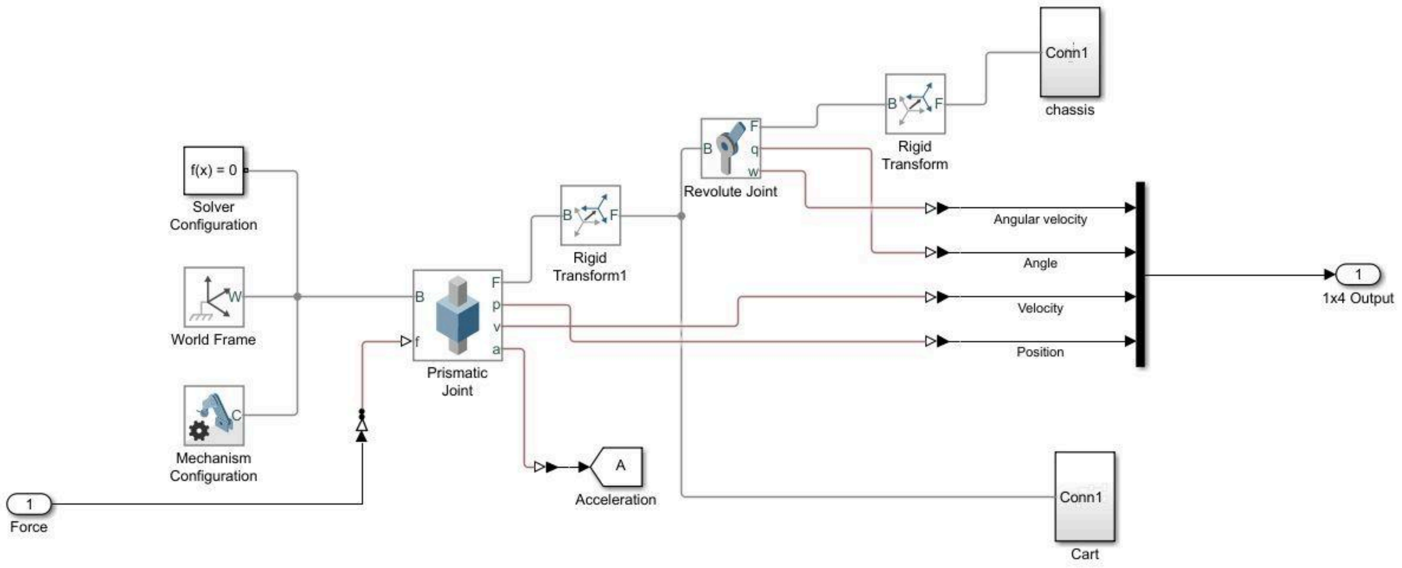


Fig: 2.2 Robot Mechanical model with Simscape multibody

Here, Apart from all transformation and Solids to make body or chassis, two key things are necessary to be explained.

First is the prismatic joint in fig 2.2 is modeled as the robot's linear motion since both the joint and motion of the robot are linear along one axis. Second is the revolute joint that is modeled as a point about which the Robot body turns, forming an arc. Here both joints also behave as sensors along with actuators. Prismatic joints give us linear velocity, position, and acceleration while Revolute joints give us angular position and velocity. The prismatic joint always calculates force with the help of data from revolute joints and some disturbance simulated as step and pulse generator. The open-loop transfer function of our Robot without any supplementary control is found to be

```
From input "PID CONTROL SYSTEM3" to output "BALANCING_2019/From2":
                        7.412e06
  ----------------------------------------------------------
  s^4 + 2000 s^3 + 9.999e05 s^2 - 1.892e05 s - 9.459e07
```

which is very much unstable. For stabilizing the Robot, we have to use feedback loops.

## 2.3 DC Motor

Also, we have modeled the DC motor for our controller to see the motor's characteristics, features, and behavior. For this, we have a permanent magnet DC motor with the following parameters
Input Voltage(V)=6V
Armature resistance (Ra)=0.7Ω
Back emf constant (Kb)=0.08 V/rpm
Total inertia(J)=1 kg.m^2
The input Torque for DC motor is obtained by,

$$\tau = I\alpha$$

$$\alpha = \frac{dw}{dt}$$

$$w = \frac{v}{r}$$

Combining all these values, we get

$$\therefore \tau = \frac{I}{r}\frac{dv}{dt} = \frac{I}{r} * a$$

The DC motor is then connected to make H-Bridge Connection using 4 MOSFET as shown below in Fig 2.3. The gate sequence is generated such that the direction of rotation of the motor changes with the changing of the angle of tiltation(θ).

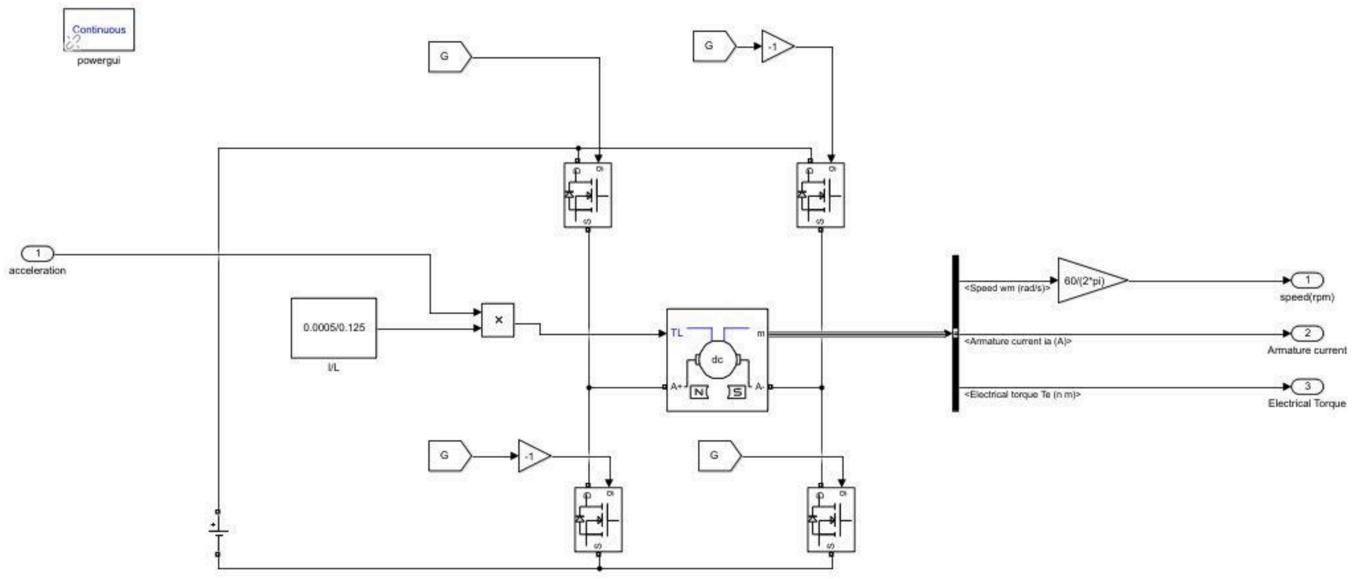


Fig: 2.3 DC motor with H-Bridge connection

# 3. Results With Different Controls

## 3.1 Single PID Controller

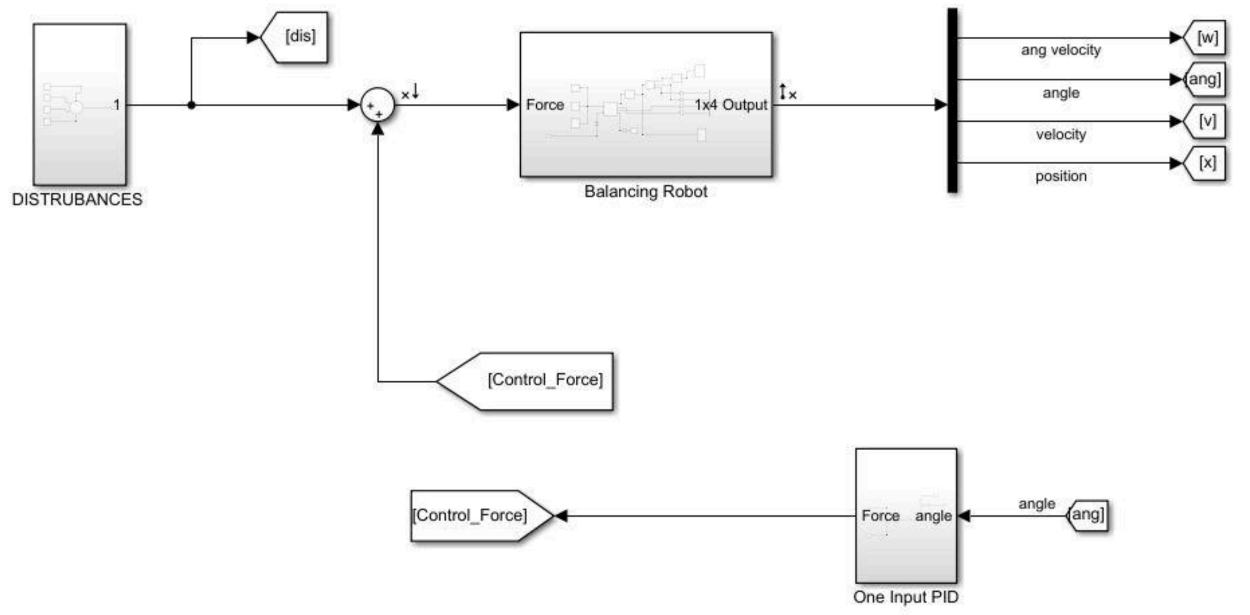


Fig: 3.1 Robot Simulink model for single input as feedback.

In our first test, we have used PID control with feedback Angle(θ) from the revolute joint.

After tuning PID for an angle to make the robot upright using the only angle as feedback to generate force for prismatic joint, we found that angle is stable at θ = 0 approximately. The Robot goes on moving in the same direction as the disturbance applied. The graph shown in Fig 2.1 shows that after t=1 sec, the initial imbalance on tiltation of the robot settles down till t=2.5 sec where another disturbance hits the robot and from then the robot regains its stability quite fast.

But in the same fig 2.1, we see that the position changes at constant velocity.

So if we use such controls, we have to move continuously indefinitely to achieve stability in the robot.

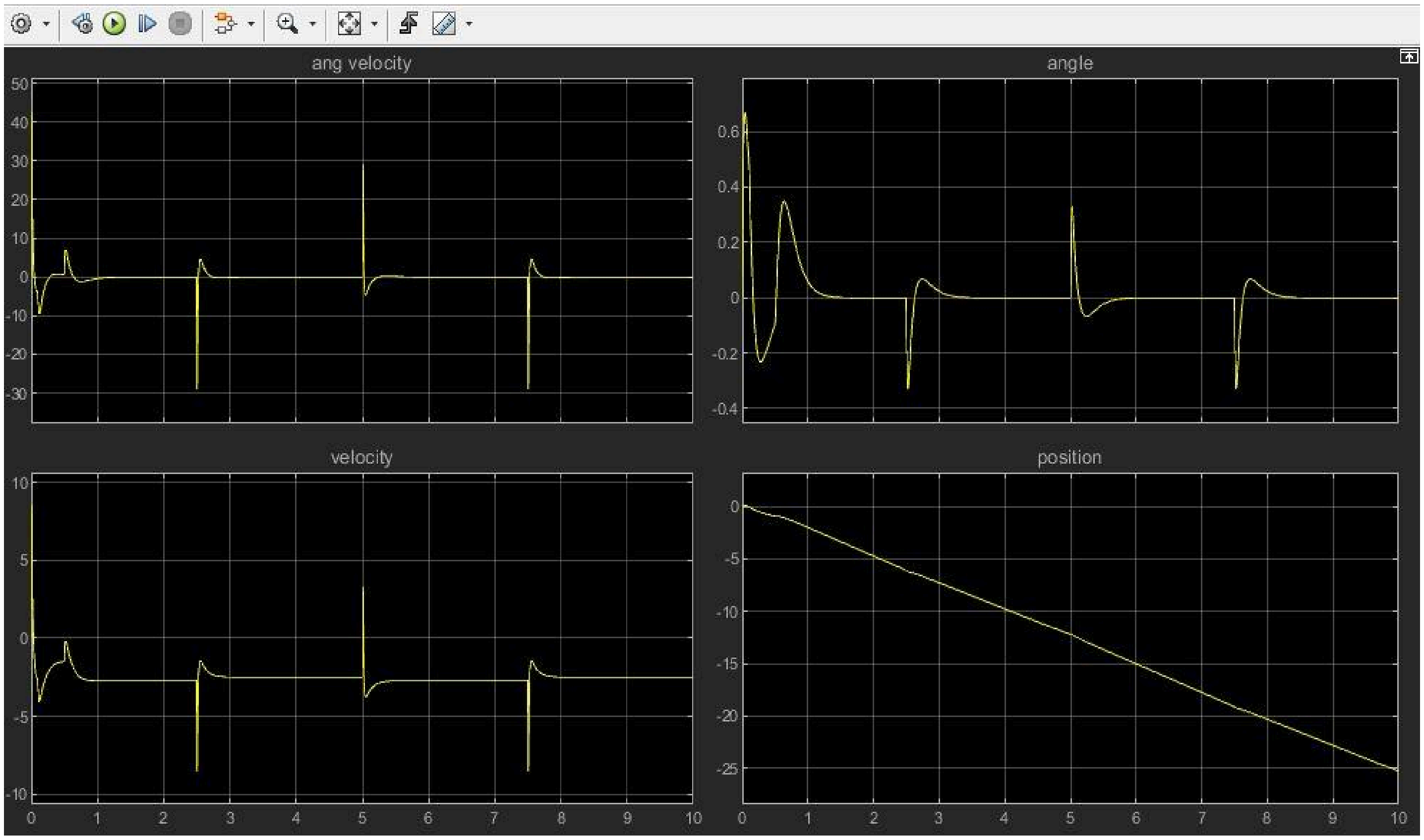


Fig: 3.1 Characteristics of Robot for Angle as feedback using PID.

Now moving towards the DC motor, we Observe that the logic on H-Bridge has worked for changing the direction of the motor as shown in speed characteristics in Fig 3.2. However, starting current is a large hitting margin that drops in an instant of time and peaks when speed changes from zero to either direction that is stimulated by the disturbances.

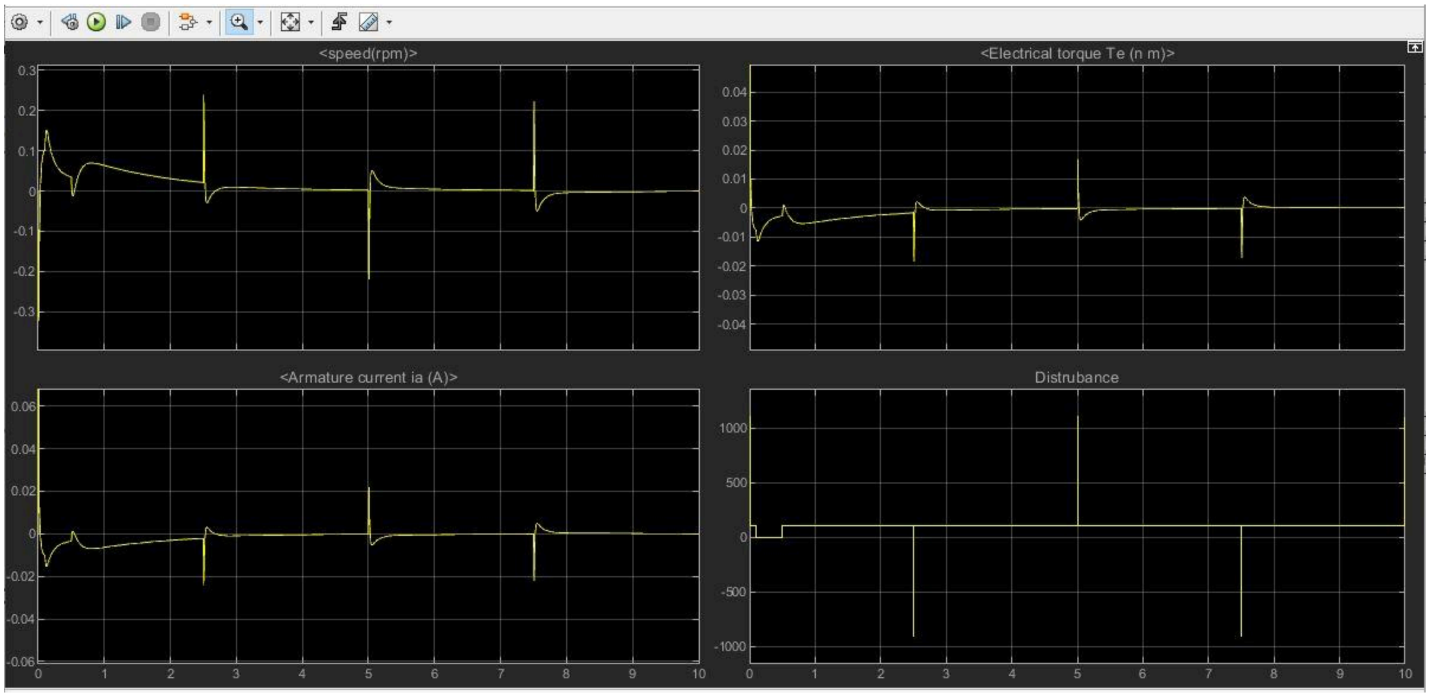


Fig:-3.2 Motor characteristics for single PID system with Angle as feedback.

## 3.2 Four PID Controllers

In our second test, we have different arrangements. Now we have chosen to feedback 4 signals (linear position and linear velocity) from prismatic joint and (Angular position/tiltation and rate of change of tiltation) from Revolute Joint to feedback to the prismatic joint so that optimum force is generated to move and keep the system stable despite periodic disturbances in the system.
As PID controllers are SISO systems, It requires 4 PID controllers to feedback 4 signals.

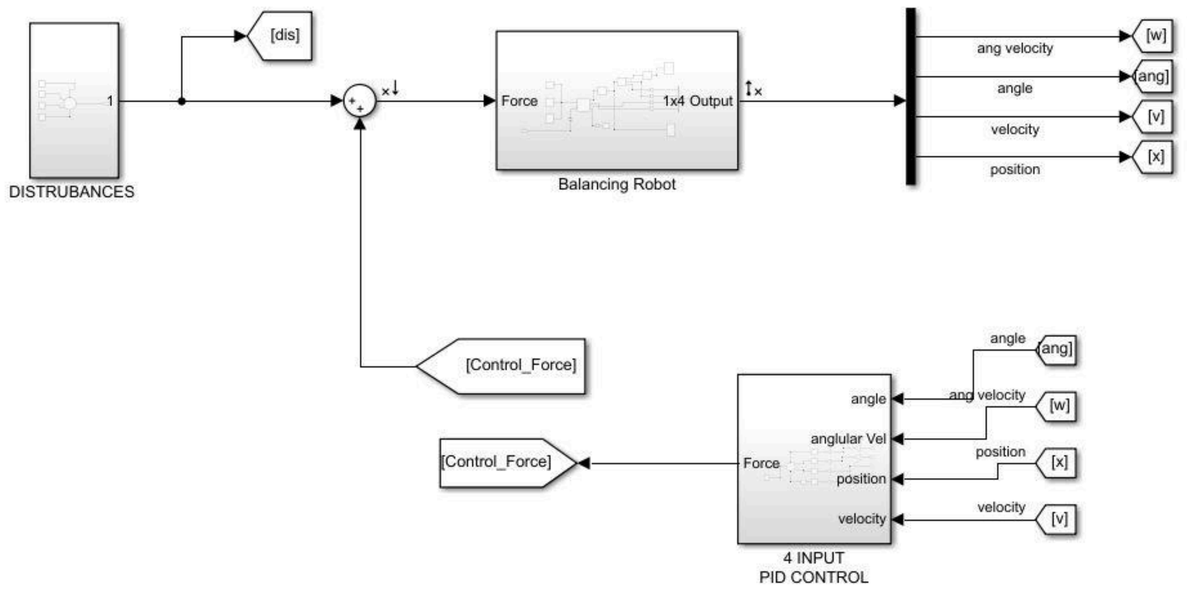


Fig:-4.1 PID controllers with 4 input as feedback

In this test, we observed that the angle is more stable with fast settling time, and initial oscillation has been reduced. Since in this system, the angular velocity has been taken into account, the rate at which tiltation changes is also reduced, and the system becomes more stable.

However drastic change could be observed in control of the position of the robot as the position is more stable with subtle motion in response to disturbance and the robot doesn't move indefinitely along the same direction of force as we have faced in the above test which is verified on the graph in Fig 4.2.

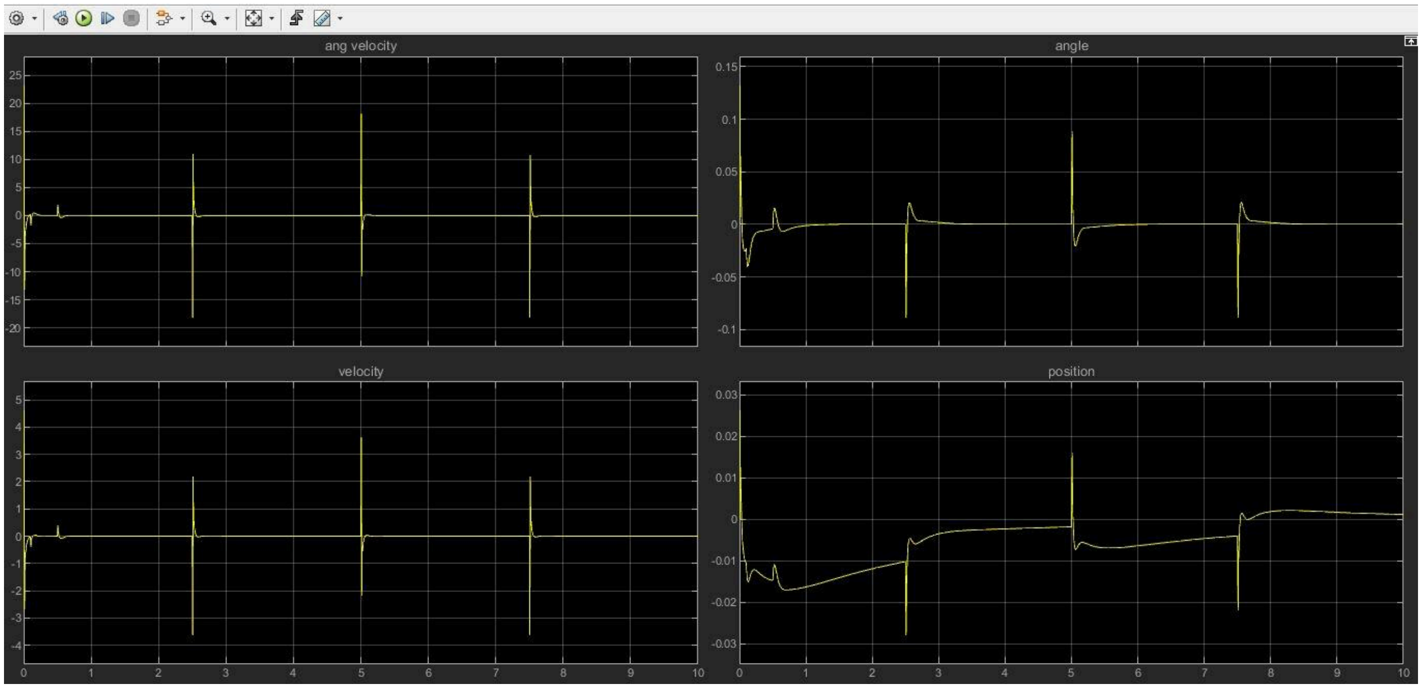


Fig 4.2:- Characteristics of Robot with 4 PID Controllers

While in the motor we observe that the initial response in current stabilizes a bit with low current consumption as the motor is more controlled and the effect of disturbance is quite low during the transient state.

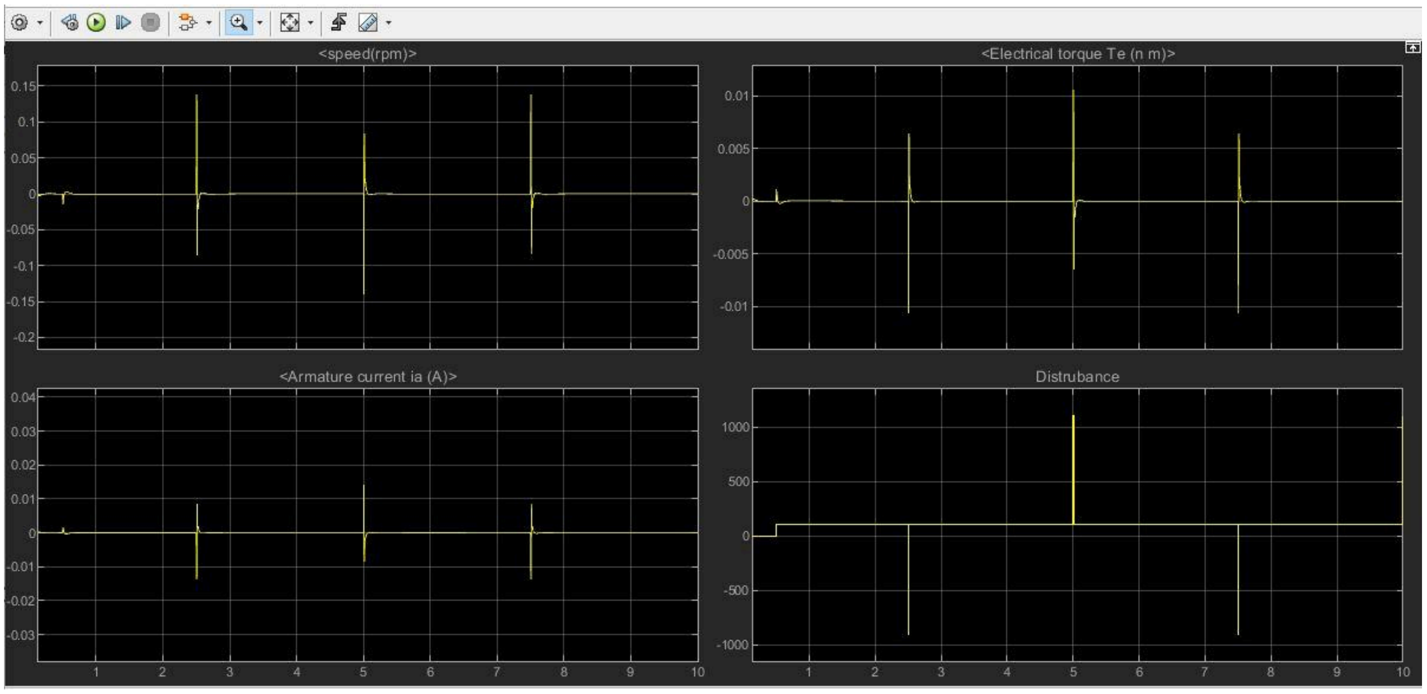


Fig:-4.3 Motor characteristics for 4-PID Controllers.

## 3.3 Optimal Control using LQR

The approach for optimal Control is a bit different. In this approach, the main motive is to reduce the cost function of the system given by

$$J = \int_0^\infty \left(x^T Q x + u^T R u + 2x^T N u\right) dt$$

where R is a symmetric positive definite or positive semi-definite matrix, R is a symmetric positive definite matrix and is unconstrained. The final control law can be derived as:

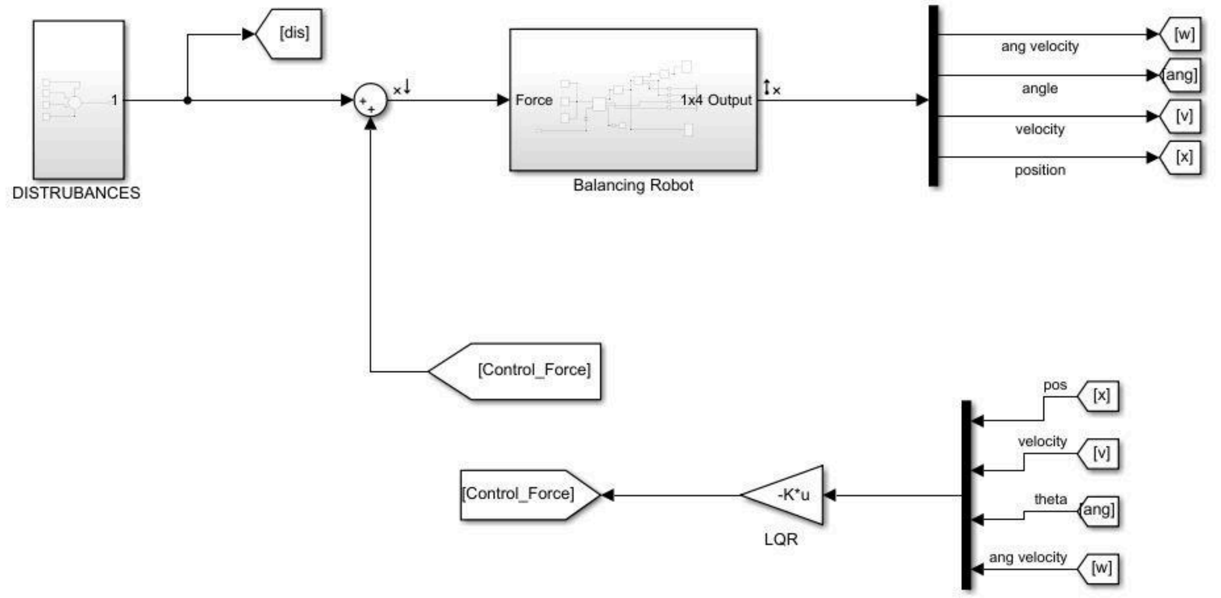
$$u = R^{-1} B^T P\, x(t) = -K\, x(t)$$

Therefore, we have verified that our system is controllable and thus we should be able to design a controller that achieves the given requirements. Specifically, we will use the linear quadratic regulation method for determining our state-feedback control gain matrix (K). The MATLAB function lqr() allows you to choose two parameters, R and Q, which will balance the relative importance of the control effort (u) and error (deviation from 0), respectively, in the cost function that you are trying to optimize. The simplest case is to assume R=1, and Q = C'C. The cost function corresponding to this R and Q places equal importance on the control and the state variables which are outputs (the pendulum's angle and the cart's position). Essentially, the lqr() method allows for the control of both outputs. In this case, it is pretty easy to do. The controller can be tuned by changing the nonzero elements in the Q matrix to achieve the desired response.

For this method, the state vectors for self-balancing Robot is calculated by

$$p = I.(M+m) + M.m.l^2$$

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -(I + m.l^2).\frac{b}{p} & \frac{m^2.g.l^2}{p} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\left(\frac{m.l.b}{p}\right) & \frac{m.g.l.(M+m)}{p} & 0 \end{bmatrix} . \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{I + m.l^2}{p} \\ 0 \\ \frac{m.l}{p} \end{bmatrix} . [u]$$ .......(1)



5.1:-LQR controlled system

In this test, the 4 sensor data are fed to the controller. The gain (K) is obtained such that it reduces the cost function. Here in MATLAB, we can obtain K using the function

K=lqr(A,B,Q,R); ...............(2)

Where A and B are the Robot characteristics that are obtained as a 4*4 matrix and 4*1 matrix

In equation 1 if we generalize it as

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

$$Q = \begin{bmatrix} 1000 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 1000 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix}$$

$$R = 0.25$$

In this test, we observed that the angle is stable with fast settling time, and oscillation has been reduced. In this system, penalizing the angular velocity less than angle, there is some subtle oscillation which is bearable in our Robot. However, we could see the position has been drastically improved so that overall, the system seems to be more stable than the 4 PID system which is also supported by graphs in Fig 5.2.

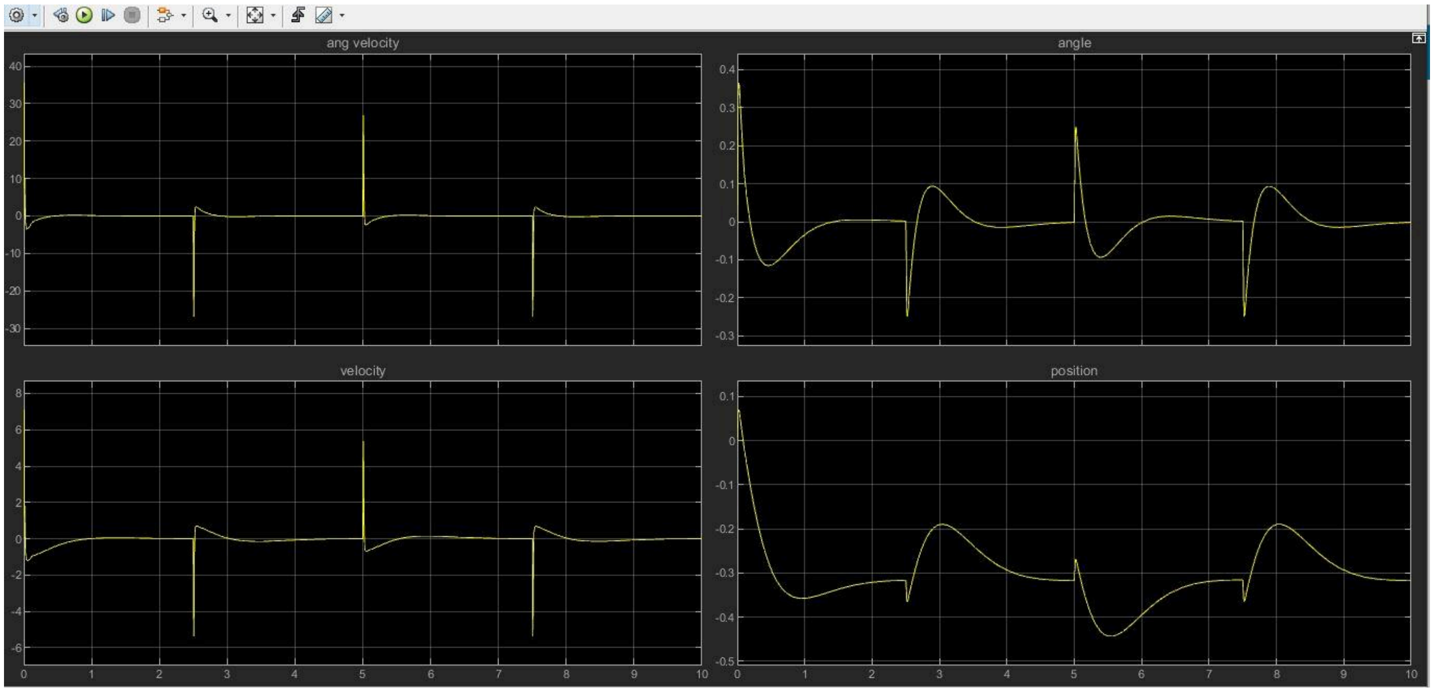


Fig:5.2:-Characteristics of Robot with LQR controller

Now, when we are looking at the motor behavior,  the transient current is a bit more which quickly drops in microseconds and stabilizes. The speed and current characteristics are moreover similar and H-bridge works fine for changing the direction of the motor when disturbance hits the system. During the change in direction, the current is moreover within its limit suggesting that the motor would work fine under such circumstances as the current characteristics shown in Fig 5.3

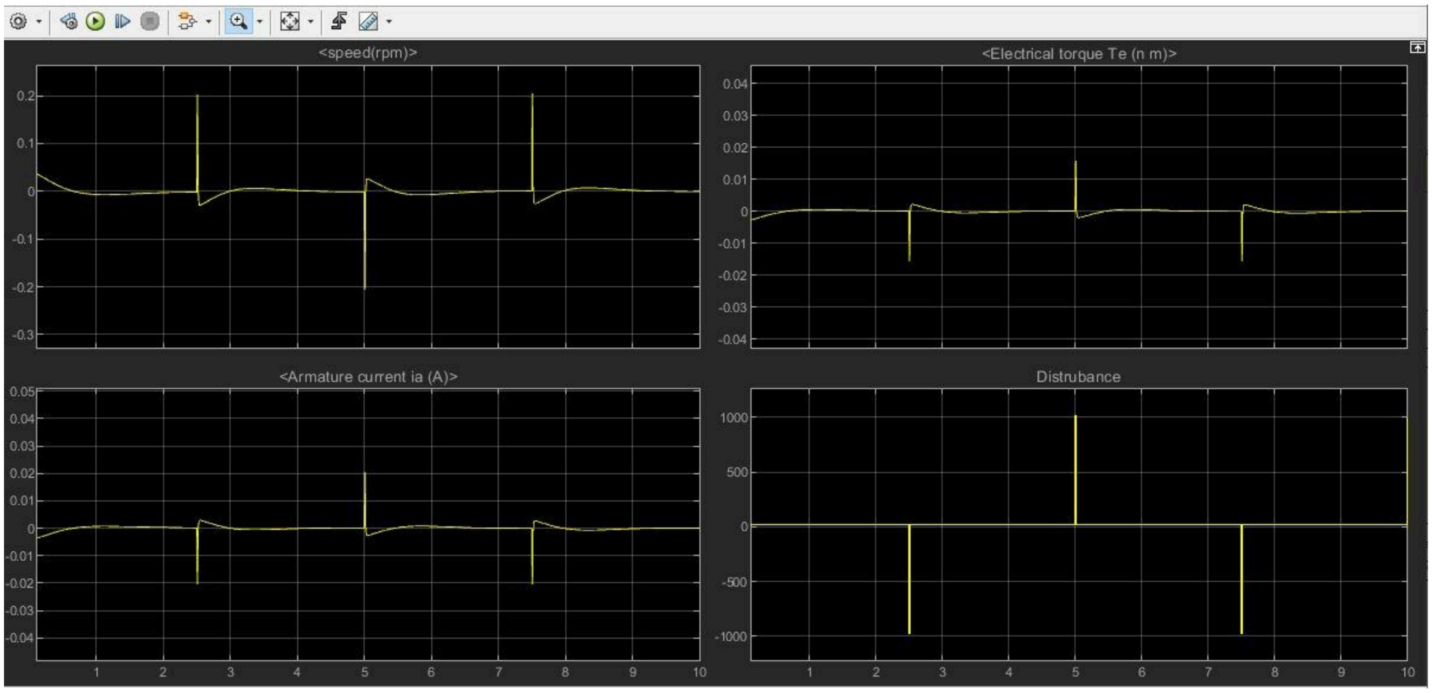


Fig 5.3:- Motor characteristics for LQR Controller

## 4. Conclusions

After testing our system in simulation with different arrangements, numbers, and types of controller we have found that the performance of a single PID system with only tiltation angles is the worst.

However, performance drastically increases when we increase the PID in the system and Feedback loops. The 4 PID system is even more tempting than LQR as since for tuning LQR we mostly perform hit and trial while we’ve used the auto-tuner app in Matlab for tuning PID for which the parameter may be sometimes impractical. However, we could safely say that LQR is more optimal as the hardware implementation of a 4 PID system is more costly and performs nearly similar to LQR for balancing. But as far as we consider the linear motion, amount of energy required, and motor performances for performing those tasks, we found the LQR controller is the best among all.

## References

[1] A.N.K. Nasir, M.A. Ahmad, and M.F. Rahmat , "Performance Comparison between LQR and PID controllers for an Inverted Pendulum System"

[2] Int. J. of Applied Mechanics and Engineering, 2017, vol.22, No.3, pp.739-747

[3] Jian Fang, "The LQR Controller Design of Two-Wheeled Self-Balancing Robot Based on the Particle Swarm Optimization Algorithm", *Mathematical Problems in Engineering*, vol. 2014, Article ID 729095, 6 pages, 2014.https://doi.org/10.1155/2014/729095

[4] mouad boumediene (2021)." two wheeled self-balancing robot" https://www.mathworks.com/matlabcentral/fileexchange/88768-two-wheeled-self-balancing-robot, MATLAB Central File Exchange. Retrieved July 20, 2021.

[5] 'Modeling of Two-wheeled Self-balancing Robot driven by DC gearmotors'
Int. J. of Applied Mechanics and Engineering, 2017, vol.22, No.3, pp.739-747 DOI: 10.1515/ijame-2017-0046

[6] 'Modeling, Simulation, and Optimal Control for Two-Wheeled Self-Balancing Robot '
Modestus Oliver Asali, Ferry Hadary, Bomo Wibowo Sanjaya

[7] 'Two wheeled Self-balancing robot'
https://www.mathworks.com/matlabcentral/fileexchange/88768-two-wheeled-self-balancing-robot

[8] 'Inverted Pendulum: State-Space Methods for Controller Design'
https://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=ControlStateSpace

[9] 'Modeling and Control of a Two Wheeled AutoBalancing Robot: a didactic platform for control engineering education'
Fabián R. Jiménez L., MSc. , Ilber A. Ruge R., MSc., and Andrés F. Jiménez L., PhD