Design and control of a two-wheel self-balancing robot

Alex Mathew, R Ananthu, P Binsy, Abdul Vahid, Ciby Thomas and S Sidharthan

Department of Mechanical Engineering, Rajiv Gandhi Institute of Technology, Kottayam, Kerala, India

Email: abdulvahid.k849@gmail.com

Abstract. Designing the self-balancing two wheeled mobile robots and stabilizing it has gained a lot of momentum in research areas. Instead of complex control system that are mostly scene in these research papers this paper introduces a new design of two wheeled robot structure which fulfills the required level of stabilization without any complex control system. To achieve this, the structure is designed with a vertical straight line motion. After deriving the mathematical model, the model is simulated in Matlab Simulink. The simulation is carried for step, ramp and uneven terrain input conditions.

1. Introduction
The transformation of human based missions to automatic process has surfaced the study of robotics. Now we can see so many robots, anything that is capable of doing a particular task. It is a very powerful field that influenced the world a lot. Among the Robot category, the least popularized model is the Two Wheeled Robot. The main issue with this is the structural balance and complexity in control. But, if we look into research field, the Two Wheeled Robot has great importance because it is a typical device of testing various control theories and control methods. Even though on Product basis a Two Wheeled Robot doesn’t have much importance, it has good significance in Research and Development. The advantage of simple structure, flexibility, power consumption, no turn radius etc. makes it suitable for Military and Exploring Missions. JOE, nBot and Eyebot are the 3 product that are available as commercial product. JOE use a linear state space controller utilising sensory data from a gyroscope and motor encoders for stabilization of the system. The nBot is controlled by a simple idea. It drives the wheels in the direction that the upper body of the robot is falling or leaning. If the wheels can be driven in such a way to stay under the robot’s Centre of Gravity, the robot remains balanced. The robot contains two feedback sensors which are a tilt or angle sensor and wheel encoder. A tilt or angle sensor is to measure the tilt of the robot with respect to gravity and wheel encoders to measure the position of the base of the robot.

This paper concerns the design of Two Wheel Robot as the research object, which uses the Newtonian mechanics equation method and the linear method near the balance point to establish the linearized mathematical model of the system. The design of the robot is based on “Segway, nBot, and JOE”. In view of the mathematical model of the system, LQR controller is designed to stabilise the system. Then PID is used to make the system controllable. This method has better control effect by simulation tests on different inputs. In this paper, structure of the robot is presented in Section 2, followed by the
mathematical model and the derivation of the transfer function of the system in Section 3. Simulation results for different inputs are highlighted in next section and finally Conclusions in Section 6.

2. Structure
The two-wheeled and self-balancing robot structure is mainly composed of the body and the two wheels, connected by 8 links, four on each wheel, and the wheels are coaxial, driven by the independent motor. The links provides a vertical straight line motion, so that the Centre of Gravity of the robot doesn’t lean to forward or back while moving in a terrain. The main purpose of this design is to have stability in all directions. The robot has two wheels and it has been driven by two stepper motors. The control of robot for back and forth motion is by these motors. These also help in making a turn. The gyro sensor on the robot body will sense the angular tilt and this input is fed into the controller, which determines the calculated movements and make itself stable.

3. Mathematical model

| Symbol | Parameter |
|--------|-----------|
| $X_m$  | Displacement in X direction |
| $Y_m$  | displacement in Y direction |
| $X$    | systems current position |
| $l$    | height of the system |
| $\theta$ | angular displacement with gravity |
| $M$    | mass of the top body |
| $M$    | mass of wheel |
| $f(x)$ | Force on wheel |
| $T$    | Torque |
| $R$    | Radii of wheel |

In order to simulate the two-wheel robot in Matlab, the mathematical model of the system is derived using Lagrange’s equation. Lagrange’s equation is:

$$\frac{\partial}{\partial t}\left(\frac{d}{dq}\right) - \left(\frac{d}{dq}\right) = Q, \quad (I = K.E - P.E)$$  (1)

where, $K.E = \frac{1}{2}(M + m)\dot{X}^2 + \frac{1}{2}m(l^2\dot{\theta}^2 - Xl\dot{\theta}^2\cos\theta)$ and $P.E = mgY_m = mgl\cos\theta$
The system of equations that defines the system are

\[
\frac{1}{2}(M + m) + \frac{1}{2}ml\theta \cos \theta - \frac{1}{2}ml\theta^2 \sin^2 \theta = f(x) \quad (2)
\]

\[
mgl\theta - m\ddot{x}\cos \theta - mgsin\theta = 0 \quad (3)
\]

Taking Laplace transformation, the transfer function is obtained as shown in equation (4). The transfer function is then converted into State space format for simulating the system in Matlab Simulink.

\[
\text{Transfer function} = \frac{398s}{s^3 + 0.2s^2 - 214.85s - 39} \quad (4)
\]

4. Design of control system

Modules available in MatLab were used for simulation and further interpretation which enabled to make the system controllable.

The Eigen values are very much significant in predicting the controllability of the system. MatLab is very much helpful on prediction of the Eigen values of the transfer function by using the code. The control theory says that the Eigen values should be negative so that the system will be controllable. ‘eig (i)’ is the MatLab command used, where i the transfer function;

\[
\text{Eigen values} = \begin{bmatrix} -11.6112 & 11.5931 & -0.0018 \end{bmatrix}
\]

Here the second Eigen value is a positive and thus the system is uncontrollable. The step input plot is given in Figure 2.

From Figure 2, it is clear that if a unit input is given to the system it is highly unstable and it is attaining a value of $10^3$ with in 10s. It is because of the positive Eigen value. The following gives the root locus of our current. According to the control theory our root locus graph should have all the poles on their negative x axis. However, we have a positive pole on it (Figure 3). So our system can’t be controlled so far.
Pole placement is a widely used method on such cases. Here if there is an undesired pole that pole can be displaced into the desired position. If the poles are correctly placed, then our system will be controllable. The pole placement is done by multiplying a gain value $K$ in the feedback system and giving a compensating matrix with input.

**Figure 2.** Step response before LQR

**Figure 3.** Root Locus before LQR

**Figure 4.** LQR Model
But the problem with the pole placement is that it is not very efficient. Thus the LQR method is used here. The LQR stands for Linear Quadratic Regulation. It’s a similar approach like pole placement and the difference is that here there is a gain matrix instead of the gain value. Therefore, it is more controllable. The LQR gain matrix consist of two matrices R and Q where R is effort (How much effort our system need) and Q is performance (with R effort what should be its performance). These two terms are contradicting and it is the designer’s choice to determine a suitable value for these matrices.

4.1 Applying the LQR to our design
An optimum value for R and Q are selected for applying LQR to the system. After applying the LQR, Eigen values $\begin{bmatrix} -101.3236 & -1.5228 & -0.6481 \end{bmatrix}$. Here, all the values are negative. The root locus of the system is given in Figure 5.

![Figure 5. Root Locus after LQR](image1)

![Figure 6. Step Response after LQR](image2)

All the poles are at the negative axis so the system is controllable. The step response of our system is shown in Figure 6. Observing the system, it could be noticed that that the system is having a response of $10^{-4}$ with 1s it’s not yet being controlled. But the system is degrading and it’s a sign that it is controllable.

5. Simulation results
To make the new system controllable, the PID controller algorithms are being used. For this special tuning purpose, we are using the MatLab built-in application Simulink. Here the system is graphically modelled and analyzed.
Figure 7. Schematic Before PID Tuning

The Figure 7 shows the general layout of the entire system. The plant indicates the transfer function (state space). A PID controller is added to the system and the result is shown in the Figure 8.

Figure 8. Step Response Before PID Tuning

The value of the output is very negligible compared to the input step signal. The given PID values are: P = 1, I = 1, D = 0. So the automatic tuning property of MatLab is used.

Figure 9 shows the response of the system after PID tuning. The resulting PID values are as follows P = 3548.90313422479, I = 24875.2773279152, D = 58.9074245437089. The resulting response of the system is shown in Figure 11. It has a steady state error of .01. To compensate it reduce the gain value to 0.98. The resulting response is shown in Figure 12.

Now the system is having zero error. It’s various properties are; setting time is 1s and overshoot is 15%. Now, the system is tested for Ramp and Sine Input. For both the inputs, the system behaves in line with the input with negligible error as shown in Figure 14 and Figure 15.
Figure 9. PID Tuning

Figure 10. PID Tuned Values

Figure 11. Step Response after PID Tuning-1

Figure 12. Step Response after PID tuning -2
Figure 13. Tuned PID Schematic

Figure 14. Ramp Response after Tuning

Figure 15. Sine wave response after tuning
6. Conclusions
The robot has better structural balance while in motion due to the vertical straight line mechanism. The wheel travels in a straight line while moving over an obstacle thus avoiding unbalance due to moment on the wheel. On the control part, with the help of Matlab Simulink an optimal PID controller was designed. It can be found from the simulation result that the design response speed of the controller is faster with less overshoot amount. Also, it can keep the steady-state error zero, thereby enabling a better control.

References
[1] Boubaker, Olfa (The inverted pendulum: A fundamental benchmark in control theory and robotics) International conference on education and e-learning innovations. IEEE 2012.
[2] Wang, Jia-Jun (Simulation studies of inverted pendulum based on PID controllers.) Simulation Modelling Practice and Theory. 19.1 440-449.
[3] Kurdekar, Vijayanand, and Samarth Borkar (Inverted pendulum control: A brief overview.) International Journal of Modern Engineering Research. 3.5 2924-2927.
[4] Velásquez-Lobo, M. F., Ramirez-Cortés, J. M., de Jesus Rangel-Magdaleno, J., & Vázquez-González, J. L (Modeling a biped robot on MatLab/simmechanics.) International Conference on Electronics, Communications and Computing 203-206.
[5] Ooi, Rich Chi (Balancing a two-wheeled autonomous robot.) University of Western Australia 3.
[6] Grassner, Felix, et al. (JOE: a mobile, inverted pendulum.) IEEE Transactions on industrial electronics 49.1 107-114.
[7] J. Wu (Design of fuzzy logic controller for two-wheeled self-balancing robot)Proc. 2016th Int. Forum Strateg. Technol. 1266–1270.
[8] Unluturk, Ali, Omer Aydogdu, and Ufuk Guner (Design and PID control of two wheeled autonomous balance robot.) International Conference on Electronics, Computer and Computation 2013.
[9] A. N. K. Nasir, M. Z. M. Tumari, and M. R. Ghazali. (Performance comparison between Sliding Mode Controller SMC and Proportional-Integral-Derivative PID controller for a highly nonlinear two-wheeled balancing robot)6th Int. Conf. Soft Comput. Intell. Syst. 13th Int. Symp. Adv. Intell. Syst., pp. 1403–1408.
[10] An, Wei, and Yangmin Li. (Simulation and control of a two-wheeled self-balancing robot.) IEEE International Conference on Robotics and Biomimetics (ROBIO). 2013.