APPLICATION OF INTELLIGENT CONTROLLER IN A BALL AND BEAM CONTROL SYSTEM

Mohd Fuaad Rahmat, Herman Wahid and Norhaliza Abdul Wahab

Control and Instrumentation Engineering Department (CIED)
Faculty of Electrical Engineering
Universiti Teknologi Malaysia
81310 Skudai, Johor Darul Takzim
Emails: fuaad@fke.utm.my, herman@fke.utm.my, aliza@fke.utm.my

ABSTRACT- Ball and beam system is one of a nonlinear and unstable control system, thus providing a challenge to the control engineers and researchers. There are a number of controllers which have been studied for years that can be used to stabilize the ball and beam system. This paper investigates the performance of few different control approaches that consist of conventional controller, modern controller and intelligent controller for a ball and beam system. It will involve the derivation of the mathematical modeling that includes the linearization of the model in order to be used with the linear controllers. The works followed with designing those controllers and simulating it in MATLAB. Each controller performance will be analyzed and compared which is based on common criteria's of the step response. An appropriate graphic user interface (GUI) has been developed to view the animation of the ball and beam system.

Index terms: ball beam, modeling, PID controller, LQR controller, neural network controller

I. INTRODUCTION

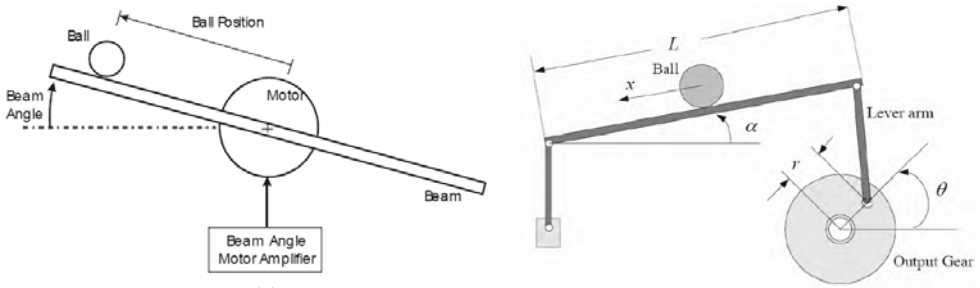
The ball and beam system is a simple mechanical system which usually difficult to control. It consists of rigid beam which is free to rotate in the vertical plane at the pivot, with a solid ball rolling along the beam. It can be categorized into two configurations. The first configuration is shown in Figure 1(a), which illustrates that the beam is supported in the middle, and rotates against its central axis. Most ball and beam systems use this type of configuration such as Hirsch (1999) [1], Rosales (2004) [2] and Lieberman (2004) [3]. This type of configuration is normally called as 'Ball and Beam Balancer'. The advantage of this form is that it is easy to build and the mathematical model is relatively simple.

The next configuration is constructed with the beam is supported on both sides by two level arms. One of the level arms acted as the pivot, and the other is coupled to motor output gear. The disadvantage is that more consideration of the mechanical parts, which meant add some difficulties in deriving a mathematical model. This type of configuration is so called 'Ball and

Beam Module'. The 'Quanser' ball and beam system uses this configuration for its commercial product as illustrated in Figure 1(b) [4]. The advantage of this system is that relatively small motor can be used due to the existing of gear box. This type of configuration will be used in this project.

The aim of ball and beam system is to position the ball at a desired point on the beam. The position of ball the ball cannot be controlled directly, but only through its acceleration. Thus, it will imply the presence of the two integrators plus the dynamical properties of the beam result in a difficult open loop unstable control problem, which is non-linear system [5]. However, the control problem can be approximate by linearised the model, hence the linear feedback control such as PID control can be applied and the stability analysis can be determined based on linear state-space model or transfer function. Besides, recent results show that the stabilization problem of the ball and beam can also be solved by nonlinear controllers, see example in [6], unfortunately this type of controller is very complex for real application. Some intelligent controllers for ball and beam can also be found, such as fuzzy control [7] and neural network control [8].

This paper will describe about designing a few types of controller for ball and beam system that consist of conventional controller, modern controller and intelligent controller. PID controller represents the conventional controller, state space controller as a modern controller, and neural network that represent the intelligence controller. An analysis of the performance will be carried out to the entire controllers, so that the best performance can be identified. Finally, a suitable general user interface (GUI) is developed to view the animation of ball and beam system.



(a) (b)

Figure 1. (a) Beam supported at the centre, (b) Beam supported at both side

II. MATHEMATICAL MODELING OF THE BALL AND BEAM SYSTEM

As illustrated in Figure 1(b), a ball is placed on a beam that free to roll along the length of the beam at horizontal plane. A lever arm is attached to the beam at one end and a servo gear at the other. The servo gear turns by an angle θ, and the lever changes the angle of the beam by α. The force that accelerates the ball as it rolls on the beam come from the component of gravity that acts parallel to the beam. The ball actually accelerates along the beam by rolling, but we can simplify the derivation by assuming that the ball is sliding without friction along the beam. The mathematical modeling of ball and beam system consists of DC servomotor dynamic, alpha-theta relation, and ball on the beam dynamic.

The dynamic equation of the ball on the beam has been described by Hauser [9] by using Lagrange method as given below,

$$0 = \left(\frac{J_b}{R^2} + m \right) \ddot{r} + mg \sin \alpha - mr\dot{\alpha}^2 , \qquad (1)$$

where J_b is the moment inertia of the ball, R is radius of the solid ball, $\ddot{r}$ is acceleration of the ball, m is mass of the ball, g is gravitational constant, α is beam angle and $\dot{\alpha}$ is angular velocity of the beam angle. The derivation of equation (1) is based on diagram depicted in Figure 2.

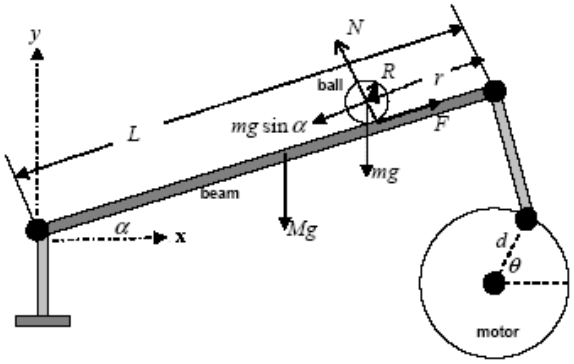


Figure 2. Force acting on the ball and beam system

Linearization of equation (1) can be estimated when the system approach the stable point. At this point $\dot{\alpha} \approx 0$, where $\dot{\alpha}$ is the angular velocity of the beam angle. Therefore, the linear approximation of the system is given by differential equation as follows,

$$\ddot{r} = \frac{-mg \sin \alpha}{\left(\frac{J_b}{R^2} + m \right)} . \qquad (2)$$

Notice that the nominal value J_b of the solid ball given by $J_b = 2mR^2/5$ kgm^2. By substituting J_b into equation (2) will give

$$\ddot{r} = -\frac{5}{7} g \sin\alpha \ . \tag{3}$$

Since α angle is small when near to stable point, thus $\sin\alpha \approx \alpha$. The transfer function for ball and beam dynamic is given as follows,

$$\frac{\Re(s)}{\alpha(s)} = -\frac{5g}{7}\frac{1}{s^2} \tag{4}$$

where $\Re$ =ball position, α =beam angle, g =gravitational acceleration.

The beam angle (α) can be related with motor gear angle (θ) by approximate linear equation $\alpha L = \theta d$, where d=lever arm offset and L=beam length. Substitute L=41.7cm and d=2.54cm will give another transfer function as follows,

$$\frac{\alpha(s)}{\theta(s)} \cong \frac{d}{L} = \frac{1}{16} \tag{5}$$

Based on voltage Kirchhoff's Law, the electrical equation of the motor is given by

$$V_{in} = IR_a + K_m\dot{\theta}_m + L_a\frac{di}{dt}, \tag{6}$$

where V_{in}=input voltage (V), I=armature current (A), R_a=armature resistance=9Ω, K_m=motor torque constant=0.0075Nm/A, $\theta_m = \omega$=angular velocity of output (rad/sec) and L_a=inductance in armature coil (mH). To simplify the motor model, the effect of the inductance L_a can be ignored because L_a contributes a very small effect in low speed application. Thus, the model will become $V_{in} = IR_a + K_m\dot{\theta}_m$. The torque produced at the motor shaft is given by

$$T_m = K_m I \ . \tag{7}$$

Assume that the load torque is the same as the produced torque, thus

$$T_l = T_m = \frac{(J_1\ddot{\theta} + B\dot{\theta})}{K_g}, \tag{8}$$

where $\dot{\theta} = \frac{\theta_m}{K_g}$, $\dot{\theta}$= angular velocity of load (rad/sec), J_1=total load inertia=7.35 x 10^{-4} Nms^2/rad, B=total load friction=1.6x10^{-3} Nms/rad and Kg=gear ratio=75. From equation (6), (7) and (8), solve for $\frac{\theta(s)}{V_{in}(s)}$, we obtain the transfer function for servomotor model as

$$\frac{\theta(s)}{V_{in}(s)} = \frac{K_m K_g / R_a J_1}{s^2 + s\left[\frac{B}{J_1} + \frac{(K_m K_g)^2}{R_a J_1}\right]} = \frac{a_m}{s^2 + b_m s}, \quad (9)$$

where a_m=85 and b_m=50. In differential equation, servomotor model can be written as

$$\ddot{\theta} = -b_m \dot{\theta} + a_m V_{in} \quad (10)$$

The linearized system equations can also be represented in state-space form. This can be done by selecting the ball's position (r) and velocity ($\dot{r}$) from equation (4) as the state variables. Besides, we select motor gear angle (θ) and motor angular velocity ($\dot{\theta}$) from equation (10) as another state variables, and the motor input voltage (V_{in}) as the input. The state-space representation is shown in equation (11), whereas equation (12) shows the output equation for this system. The mathematical model in state space form is used to design the linear quadratic regulation (LQR) controller in the next section.

$$\begin{bmatrix} \dot{r} \\ \ddot{r} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{5g}{7} * \frac{d}{L} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -b_m \end{bmatrix} \begin{bmatrix} r \\ \dot{r} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ a_m \end{bmatrix} V_{in} \quad (11)$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} r \\ \dot{r} \\ \theta \\ \dot{\theta} \end{bmatrix} \quad (12)$$

III. CONTROLLER DESIGN

The controller realization is implemented via three control strategies, namely PID controller, linear quadratic regulation (LQR) controller and neural network controller. The control problem is to design a controller which computes the applied voltage V_{in} for the motor to move the ball in such a way that the actual position of the ball reaches the desired position. The motor is controlled to produce the desired α (beam angle), however it should be noted that α is controlled by the angle at the output of the servomotor plant (θ angle). The simplest control strategy is the 1-DOF (Degree of Freedom) topology shown in Figure 3, where the plant is treated as the cascade connection of transfer function in equations (4), (5) and (9). Although it is possible to design the controller $C(s)$ such that the closed-loop system is stable, the existence of multiple integrator in the plant contributes - 270° phase lag to the loop gain. This provides a difficulty for obtaining a good closed-loop performance. For decision making of controller design, a few design specifications have been set. In this design, we only take two considerations to be met which are settling time less than 3 second and percentage of overshoot

is less than 3%. The three second is chosen to determine the effectiveness of the designed controllers in term of fast response. Whereas, the low overshoot is required to avoid the ball run out of the beam especially at the end points of the beam.

a. PID Controller Design

It was found that the overall system is a fourth order system. Also, it is quite tedious and difficult to design a controller to control a third order and higher order system. Therefore, to make the controller design become easier and realizable, the whole system is separated into two feedback loops as shown in Figure 4. The purpose of the inner loop is to control the motor gear angle position. Controller 1 ($C_1(s)$) should be designed so that gear angle (θ) tracks the reference signal (θ_{ref}). The outer loop uses the inner feedback loop to control the ball position. Therefore the inner loop definitely must be designed before the outer loop.

For the inner loop, PD controller is selected instead of PID because servomotor model is a second order system, thus PID controller will change the second order system to third order system which is quite hard to control whereby PD controller will preserve its second order. Thus, the equation for PD controller for inner loop is $C_1(s)=K_p+K_ds$, where K_p is proportional gain and K_d is derivative gain. By using Ziegler Nichols's method, PD parameter has been tuned to be $K_p=5$ and $K_d=0.1$.

For the outer loop, we will study the implementation of proportional (P) controller and proportional-integral-derivative (PID) controller. The variation of that controller showed in equation (13) and (14).

$$\text{P controller} \quad : C_2(s) = K_p \tag{13}$$

$$\text{PID controller} : C_2(s) = \frac{K_d s^2 + K_p s + K_i}{s} \tag{14}$$

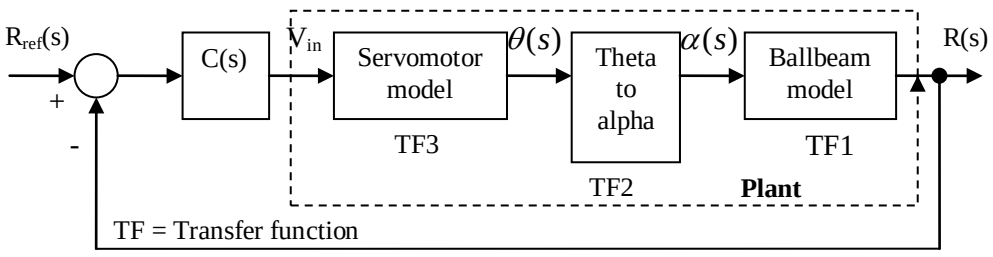


Figure 3. 1-DOF control strategy for the linearized model

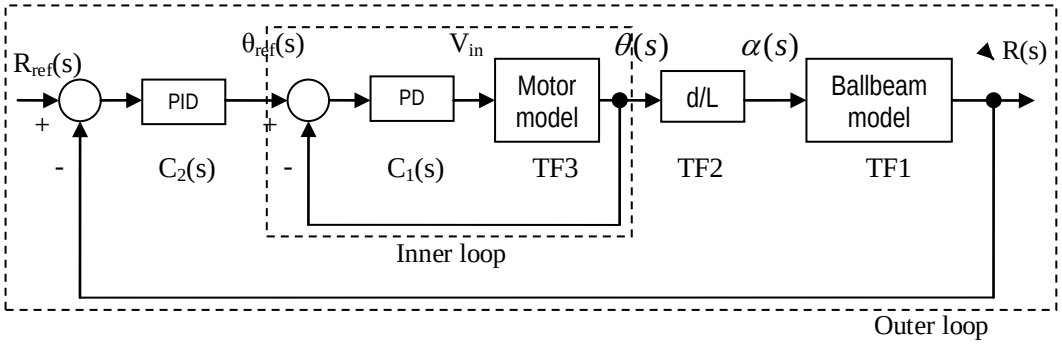


Figure 4. 2-DOF control strategy for the linearized model that utilize PID controller

b. LQR controller design

There are various types of state space controller techniques such as full state feedback control, observer control and optimal control. This paper analyzes a full state feedback controller which is used linear quadratic regulation (LQR) approach. The schematic of a full-state feedback system is shown in Figure 5(a). From this figure, $r_{ref}(t)$ is desired position, $r(t)$ is output position and K is full-state feedback gain.

For the LQR controller design, we will utilize state feedback equation and output equation as in (11). The characteristic equation for the closed-loop system is given by the determinant of $[sI-(A-BK)]$, where I is identity matrix, while A and B are the system matrix and input matrix respectively from state space equation in (11). For this system the A and $B*K$ are both 4×4 matrices. Hence, there should be four poles for this system.

LQR will give the optimal controller under certain assumptions. The *'lqr'* function (in Matlab) allows us to choose two parameters, regulator (R) and quadratic (Q), which will balance the relative importance of the input and state in the cost function that we are trying to optimize. The simplest case is to assume R=1, and $Q=C^T*C$, where C is output matrix from equation (12) and C^T is matrix transpose of C. The controller can be tuned by changing the nonzero elements in the Q matrix to get a desirable response. Thus, that element will be used to weight the output response. The strategy is described in Matlab command as shown below:

```
R = 1;
Q =[x 0 0 0; 0 0 0 0; 0 0 0 0; 0 0 0 0];
K = lqr (A, B, Q, R)                                    (15)
```

From above command, by increasing x, the settling time (T_s) and rise time (T_r) can be decreased. For this design, the value of x is set to 40000 and let R remain one. The following value of K was found:

```
K = [-200.0000    -79.3501    110.1876    1.9928]
```

Basically by using State Feedback method only, we can found a very large steady state error. Therefore, in order to eliminate the steady-state error we have to enhance the design by adding a constant gain $\bar{N}$ after the reference input as depicted in Figure 5(b). The value $\bar{N}$ can be determined by using the user defined function '*rscale*'.

$$\bar{N} = rscale\ (A, B, C, D, K) \qquad (16)$$

In this case matrix D is equal to zero. Finally we will get the new matrix equation for *A*, *B*, *C* and *D* through this controller which are $A_c=A–B*K$, $B_c=B*\bar{N}$, $C_c=C$ and $D_c=D$.

c. Neural network Controller Design

A neural controller can be created for the case where the mathematical model is not available. In this study, we will use 'model reference control' strategy that utilize Levenberg-Marquardt backpropagation for the training process [10]. For model reference control we want to control a system so that its output follows the output of a reference model. In this study, we will train a neural network controller which will drive the ball and beam system to follow a linear reference model. Figure 6 shows the entire neural network controller structure with the ball and beam system.

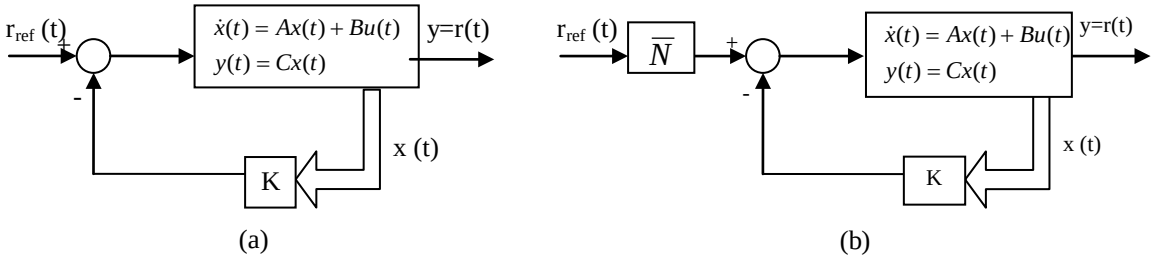


Figure 5. (a) State feedback control configuration, (b) State feedback control configuration with the input gain, $\bar{N}$

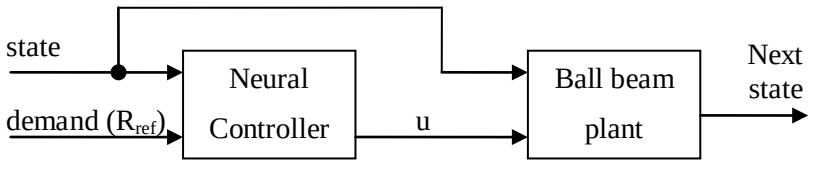


Figure 6: Neural network controller configuration for the ball and beam system

From Figure 6, we would like the ball and beam system to respond with target states, T_c from the initial state, B_c. The problem is that the error between the actual *ball beam* behavior and the desired linear behavior occurs on the outputs of the ball and beam system [11]. Hence,

these errors will be used to adjust the controller. It can be done by replacing the *ball beam* with the '*model network*', for training the controller. Figure 7 shows the diagram of the neural network controller, with the inclusion of 'model network'.

From Figure 7, the derivatives of the error can be back-propagated through the model network to the control network. The derivatives are then back-propagated through the controller and used to adjust its weights and biases (in this case, the *model network's* weights and biases are not changed). Thus the control network must learn how to control the ball and beam system (that represented temporarily by the *model network*) so that it behaves like the linear reference model [11].

Before the controller network can be trained, the model network and the linear reference model need to be defined. Base on equation (3), the ball and beam system can be represented as below state equation:

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{5}{7}gu \end{bmatrix}, \tag{17}$$

where $\dot{x}_1 = r = position$, $\dot{x}_2 = dr/dt = velocity$, $u = beam\_angle$ and g=gravitational constant.

The ballbeam model is summarized with the function '*bmodel*' in MALAB file, which takes the current time (*t*), ball position (*position*), ball velocity (*vel*), and the beam angle (*angle*), and returns the derivatives of position, velocity, and force.

x = [position; vel; angle]
dx = bmodel(t,x)

Then, we will simulate the ball beam *model network* from 0 to 0.05 seconds using *ode23* by using this command; *[time, X] = ode23 ('bmodel',[0 0.05],x).* This function returns a row vector of times, and the matrix *X* of state vectors associated with those times.

Next, the examples of ballbeam behavior must be created so that the network can be trained. The MATLAB code is comprised in *bnb1a.m*, which defines several different ballbeam position, ballbeam velocities, and beam angle. By taking all possible combinations of these values, we get a matrix B_m. The model network has two states (position and velocity) and one input (angle), thus the network requires three inputs and one output. The function '*newff*' is used to create a two-layer tansig/ purelin network with three inputs and one output and eight hidden neurons, where '*mnet*' is the ballbeam *model network*. These commands are comprised in MATLAB function, *bnb1c.m.* The *mnet* has to be trained in order to get the optimum combination of the weights and biases. Levenberg-Marquardt training function '*trainlm*' is used to obtain a solution

in the minimum amount of time. The network is trained for up to 500 epochs, displaying progress every epoch, and with a typical error of 0.0037.

Next, we will set the closed loop system to respond with the dynamics given by the Linear Reference Model (LRM) as follows,

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -9x_1 + 7x_2 + 9u \end{bmatrix}, \qquad (18)$$

where x_1 is the desired output position. The desired linear reference model, described mathematically above, is comprised in MATLAB file, *blinear.m*.

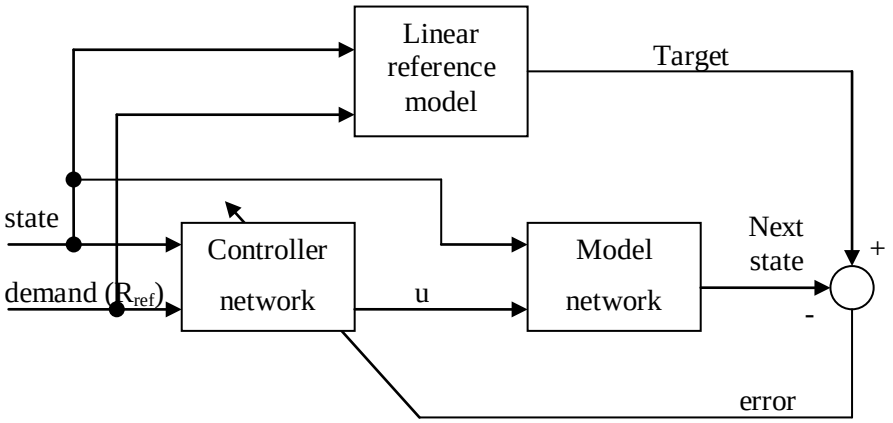


Figure 7: Configuration for training the neural network controller

Same as *model network* design strategy, the *controller network* has a tansig/ purelin network, which will have eight hidden neurons, and one output, where '*cnet*' is the neural controller network. This function is comprised in MATLAB file *bnb2c.m.*

Next, we will set up a combination network that includes both the *model network* and the *controller network* that produces *tnet* as the total network. We will use the quasi-Newton backpropagation training function, '*trainbfg*' to train the network to minimize the mean square error. During the training process, the network contains the weights and biases of both the *controller network* and *model network*, but the *model network* weights do not change during this process, whereby only the controller weights are updated. Finally, we will place back the optimum controller weights and biases into the controller network in Figure 6.

IV. GUI FOR BALL AND BEAM SYSTEM

The aim of constructing the Graphical User Interface (GUI) is to allow the user to view an animation of the ball and beam system according to desired set-point. This allows the user to see the correlation between the plot and the systems physical response. Once the satisfactory compensators were obtained, it wills then being interfaced with the graphical user interface

(GUI). The GUI has been designed by m-files coding in the MATLAB features. The designed GUI was integrated with the 3 types of controller which are PID controller, LQR controller and neural network controller. Figure 8 shows the developed GUI for a ball and beam system. Basically, it consists of three main panels which are controller setup, plotting response for ball and beam, and ball and beam animation. The full source code for the GUI is available in MATLAB file, *bnb.m*.

V. RESULT AND ANALYSIS

a. Results for Proportional (P) and Proportional-Integral-Derivative (PID) Controller

Proportional controller is the most basic strategy for the feedback control law. The controller output is made proportional to the error and the proportionality constant is called the proportional gain (K_p). Unfortunately, this controller is not capable of maintaining the output steady state value at the desired value as shown in Figure 9(a). From this figure, K_p is set to be 0.1 and setpoint at 0.2 meter. While increased K_p gain, the output will response faster because it decreases the rise time (T_r). However, K_p gain is limited by the dynamics of the system, where for ball and beam system the response is limited by the length of the beam (0.4 meter). Therefore, when the response oscillates beyond this value, the system will stop as the ball reaches the maximum distance.

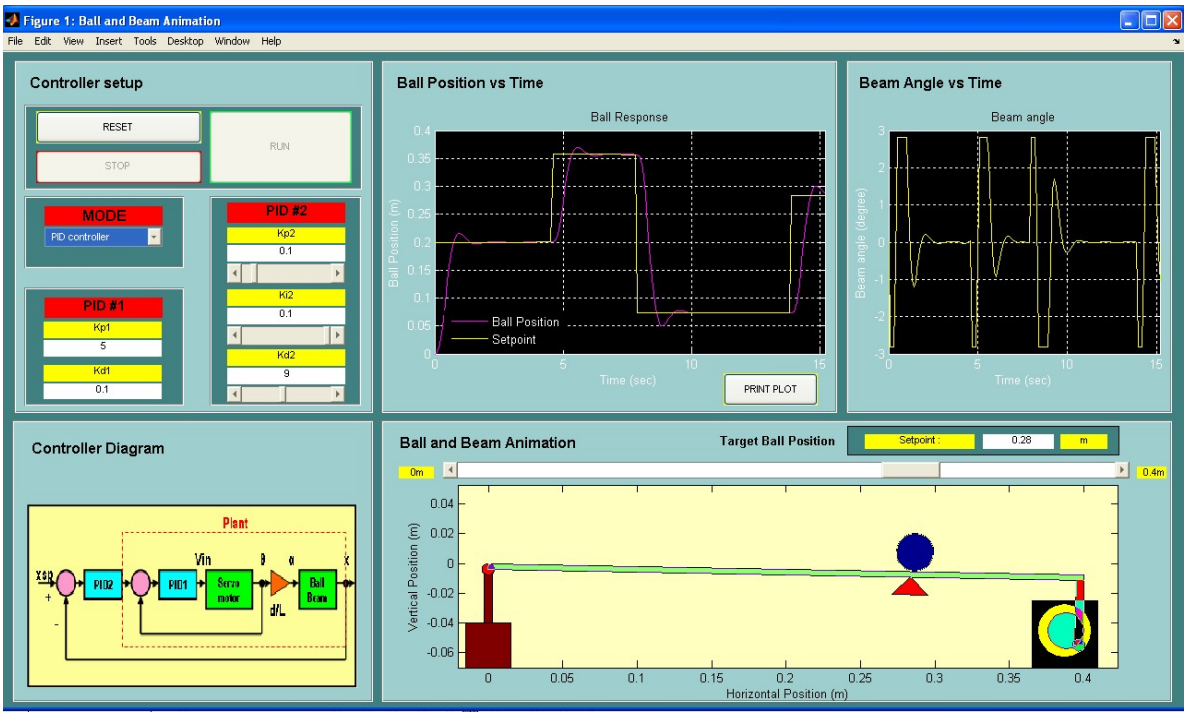


Figure 8. Graphical user interface layout in MATLAB

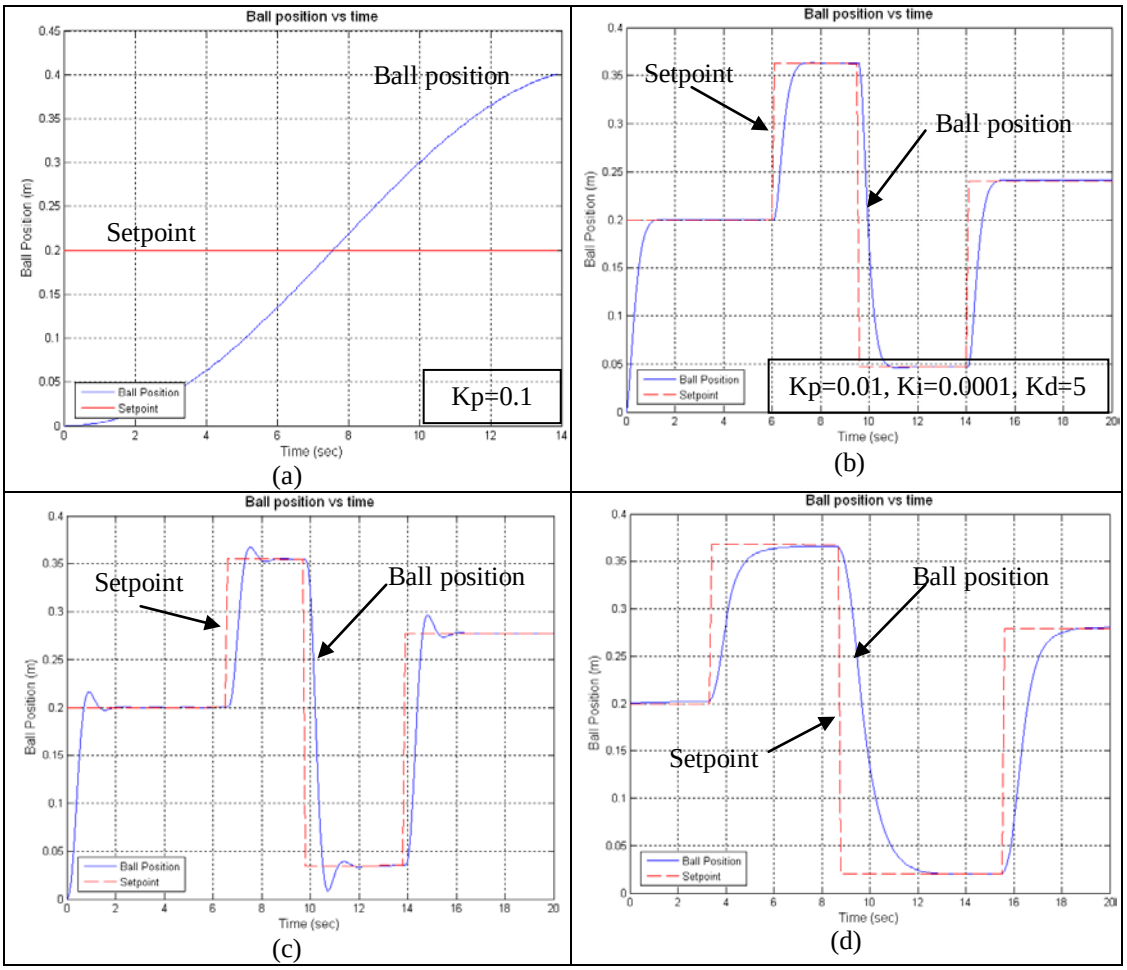


Figure 9. (a) Result for P controller, (b) Result for PID controller, (c) Result for LQR controller, (d) Result for neural network controller

The effect of K_p, K_i and K_d terms in PID controller tend to make the closed loop system become more stable. K_p, K_i and K_d are dependent of each other, so changing one of these variables can change the effect of the other two. A proportional controller (K_p) will have the effect of reducing the rise time and will reduce but never eliminate, the steady-state error. An integral control (K_i) will have the effect of eliminating the steady-state error, but it may make the transient response worse. A derivative control (K_d) will have the effect of increasing the stability of the system, reducing the overshoot, and improving the transient response. The result for PID controller is shown in Figure 9(b). The PID controller gives a good steady state response, whereby the steady state error is only 0.065 with slightly small overshoot of 0.2% from its final value.

b. Results for State Feedback Controller: LQR controller

Figure 9(c) shows the simulation result for a ball and beam system that utilizing the LQR controller. The dashed line is referring to the set point (reference) whereas the solid line is

referring to the ball position on the beam. The graph shows that the output tracks the changes of the reference input. The controller gives a very fast response with the settling time of 1.3 second and rising time that less than 0.5 second. In addition, the LQR tend to produce small steady state error of 0.0014, however it produces a high overshoot of about 8.2%. Thus, the controller is not satisfied the design requirements, even it produces a very fast response.

The LQR controller gives the best response by the optimization process. We can only tunes the response time (rise time T_r, peak time T_p, and settling time T_s) by changing the coefficient value in the matrix Q. It also can be done by tuning the value of R and the best combination value of R and matrix Q will give a satisfactory response. In this design, we fix the value of R to one in order to simplify the design process. By increasing the value of x in matrix Q, we should able to get a better settling time and rising time. Unfortunately, it will increase the percentage of overshoot in the output response. In our case, if we decrease the x value, the overshoot specification can be met, however it will take a longer response time.

c. Results for Neural Network Controller

The simulation result for neural network controller is shown in Figure 9(d). It shows that the output can track with the changes of the set-point. With the intelligent controller, it gives a promising results that nearly same to the conventional controller. A small steady state error of 0.004% is generated without the existing of the overshoot. However, the response time is a little bit slower than the other controllers due to the time consume on the learning and training process. This controller gives the settling time of 2.4 second and rising time of 1.9 second at the reference input of 0.1 meter. In further, if the set-point is increased to a maximum limit (0.4 meter), it takes a longer response time with the settling time is equal to 3.1 second. Hence, we can summarize that neural network controller is able to control the ball and beam system, but the response time is a little bit out from the design specification.

d. Overall comparison of the controller performance

The graph in Figure 10 shows the set-point and the output response for comparison of the entire controllers. It is quite surprising that the designed PID controller has an overall better performance than P, LQR and neural network controller, though it seems that the PID gives fastest response time with the reasonable percentage of overshoot and steady state error. The comparison of the response's characteristics is depicted in Table 2.

As we can observed, the P does not able to control the system, hence given a remaining unstable system. The other compensation schemes produce very smooth results, but with noticeably different shapes to the response curves. In term of the settling time, the entire controller satisfies the design tolerance. The best settling time (T_s) is given by PID controller and the worst by neural network controllers. Even though, LQR shows a very good rising time (T_r) which is less than 0.5 second, it tends to generate inadequate transient response and also steady state response. LQR controller produces 7% of overshoot and 1.03% of steady state error which is the worst among the controllers. The steady state error given by PID method is better than other controllers.

Besides, neural network is another interesting and feasible method for control system design and can be another alternative for the conventional control and the modern control. In this limited study, it produced surprisingly promising results, which gives almost zero overshoot and very less steady state error. Even it takes longer response time as compared to others, but it is still within the specified design tolerance.

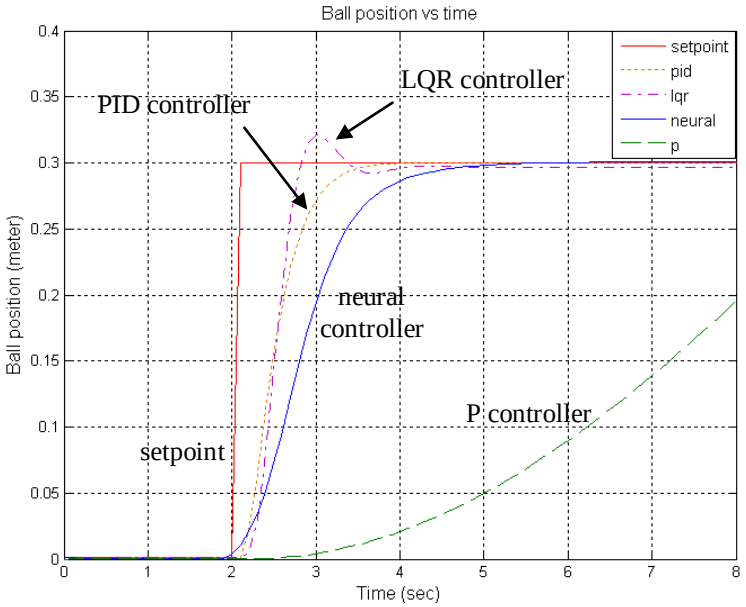


Figure 10. Output response for the entire controllers

Table 2: Comparison of the output response's characteristics

Controller	Overshoot (%)	Peak time, T_p (s)	Rise time, T_r (s)	Settling time, T_s (s)	Steady state error, e_{ss} (%)	System status
P	-	-	-	-	-	Unstable
PID	0.1	2.5	0.8	1.5	0.097	Stable
LQR	7.0	1.0	0.4	1.9	1.030	Stable
Neural Network	0.2	3.1	1.3	2.5	0.240	Stable

VI. CONCLUSIONS

The mathematical model for a ball and beam system has been derived successfully. The plant is consists of three main components which are servomotor model, angle conversion gain, and ball on the beam dynamic equation. Both servomotor and ball beam dynamic has the second order transfer function. Besides, the state space equation was derived in order to design the state space controller. Several controllers of conventional, modern and intelligent scheme have been successfully designed to control the ball and beam system. It is quite tedious to design the fourth order system, thus for conventional method, two controllers have been implemented to control those second order components. The modern controller implements the full state feedback control that utilizes of LQR method, whereas neural network was utilized for the intelligent control strategy. Furthermore, an interesting ball and beam GUI has been successfully designed by using MATLAB program. The analysis results had shown that PID controller shows better performance among the others, however, the surprising result is may be due to the implementation of cascade approach in the PID controller. If the simplest control strategy (1-DOF) is selected, it is possible that the PID will yield the worst performance, or may not be able to stabilize the system. With the basic configuration, seem like the intelligent controllers not giving a good transient response, but still can be an alternative to replace the conventional and modern controller. The results for intelligent controllers are possible to be improved further by using advance configuration and better tuning method. Notice that the implementation of LQR controller and neural network controller are both at basic configurations, and not at the optimal parameters.

REFERENCES

[1] Robert Hirsch, "Mechatronic Instructional Systems Ball on Beam System", *Shandor Motion Systems*, 1999.

[2] Evencio A. Rosales, Bennett T. Ito, Katie A. Lilienkamp, Kent H. Lundberg, "An Open-Ended Ball-Balancing Laboratory Project for Undergraduates", *Proceeding of the 2004 American Control Conference Boston. Massachusetts*, July 2004.

[3] Jeff Lieberman, "A Robotic Ball Balancing Beam", *MIT Media Lab Publications*, February 2004.

[4] Wen Yu, Floriberto Ortiz, "Stability analysis of PD regulation for ball and beam system", *Proceedings of the 2005 IEEE Conference on Control Applications Toronto, Canada*, pp. 28-31, August 2005.

[5] Peter E. Wellstead, "Introduction to Physical System Modelling", *Academic Press Limited*, pp. 221-227, 2000.

[6] R. M. Hirschorn, "Incremental Sliding Mode Control of the Ball and Beam System", *Automatic Control IEEE Transaction Volume 47*, October 2002.

[7] H. K. Lam, F. H. F. Leung, P. K. S. Tam, "Design of a Fuzzy Controller for Stabilizing a Ball-and-Beam System", *IEEE Proceeding in Control*, 1999.

[8] Yuhong Jiang C. Mc Corkell and R. B. Zmood, "Application of Neural Networks for Real Time Control of a Ball-Beam System", *Proceedings of. IEEE International Conference on* Neural Networks, *1995.*

[9] J. Hauser, S. Sastry and P. Kokotovic, "Nonlinear control via approximate input-output linearization: ball and beam example", *IEEE Transaction on Automatic Control*, Vol.37, 1992.

[10] M. T. Hagan, M. B. Menhaj, "Training Feedforward Networks with the Marquardt Algorithm", *IEEE Transactions on Neural Networks*, vol. 5, no. 6, 1994

[11] Howard Demuth, Mark Beale, "Neural Network Toolbox for use with Matlab Version 3.0", *The MathWork Inc*., 1998.