BALL AND BEAM MODELING USING MATLAB/SIMULINK

Gîlcă Gheorghe, Lecturer PhD, “Constantin Brâncuși” University from Tîrgu-Jiu, ROMANIA

ABSTRACT: This paper presents ball and beam system, this is laboratory equipment with high nonlinearity in its dynamics. The main ideas of the paper are to model the ball and beam system considering nonlinear factors and coupling effect and to design controller to control the ball position. MATLAB/SIMULINK software program has been used to plot instant system response and to determine the system characteristics with different values of controller parameters in order to choose parameters values which obtained best performance for the system.

KEY WORDS: nonlinearity, ball position, lagrangian equation, lever arm

1. INTRODUCTION

The ball and beam system has been studied and used in the tests of new methods by many researchers. In particular, I. Petkoviç, M. Brezak, and R. Cupec have conducted artificial vision based tests of the system [1]. P. Dadios and R. Baylon did a similar study on fuzzy logic [2]. J. Whelan and J. V. Ringwood, on the other hand, used the same system for position tracking through a camera instead of sensors, using PID controllers [3]. Sıtkı KOCAOĞLU, Hilmi KUŞCU they Design and Control Ball and Beam System in which they used PID controllers [4].

2. PROBLEM SETUP

A ball is placed on a beam, see figure below, where it is allowed to roll with 1 degree of freedom along the length of the beam. A lever arm is attached to the beam at one end and a servo gear at the other. As the servo gear turns by an angle theta, the lever changes the angle of the beam by alpha. When the angle is changed from the horizontal position, gravity causes the ball to roll along the beam. A controller will be designed for this system so that the ball’s position can be manipulated.

Figure 1. The Ball and Beam mechanism

For this problem, we will assume that the ball rolls without slipping and friction between the beam and ball is negligible. The constants and variables for this example are defined as follows:

- \( m \) mass of the ball: 0.11 kg
- \( R \) radius of the ball: 0.015 m
- \( d \) lever arm offset: 0.03 m
- \( g \) gravitational acceleration: 9.8 m/s^2
- \( L \) length of the beam: 1.0 m
(J)  ball's moment of inertia  9.99e-6 kg.m^2  
(r)  ball position coordinate  
(alpha)  beam angle coordinate  
(theta)  servo gear angle  

The design criteria for this problem are:  
- Settling time less than 3 seconds  
- Overshoot less than 5%  

The second derivative of the input angle alpha actually affects the second derivative of r. However, we will ignore this contribution. The Lagrangian equation of motion for the ball is then given by the following:  

\[ 0 = \left( \frac{1}{J} + m \right) \ddot{r} + mg \sin(\alpha) - mr \dot{\alpha}^2 \]  

(1)  

The beam angle (\( \alpha \)) can be expressed in terms of the angle of the gear (\( \theta \)).  

\[ \alpha = \frac{d}{L} \theta \]  

(2)  

3. BUILDING THE MODEL IN SIMULINK  

In this example, rather than expressing all the forces and geometric constraints (which is difficult to model in Simulink for dynamic systems with constraints) we will model the nonlinear Lagrangian equation of motion directly. This equation gives \( \dot{r} \) as a function of the state and input variables, \( r \), \( \dot{r} \), \( \alpha \), and \( \dot{\alpha} \). We will make use of the Fcn block to express this function. First, we must express the derivatives of the output, \( r \).  

- Open a new model window in Simulink.  
- Insert an Integrator block from the Continuous library.  
- Insert a second Integrator to the right of the first, and connect the two with a line.  
- Label the line connecting the two "d/dt(r)". To label a line, double-click near the line where you want the label (in this case, just below the line)  
- Draw a line from the second Integrator and label it "r".  
- Insert an Out1 block from the Sinks library and connect it to the "r" signal line. This will form the output of the system.  
- Change the label of the Out1 block to "r" by single-clicking on the existing "Out1" label. Now, we will insert the function which takes the vector \([r \ \dot{r} \ \alpha \ \dot{\alpha}]\) and returns \( \dot{r} \). Insert a Fcn block from the User-Defined Functions library and connect its output to the input of the first Integrator. Edit the Fcn block by double clicking it, and change it's function to the following:  

\[ \dot{r} = \left(-\frac{1}{J} + m\right)(mg \sin(\alpha) - mr \dot{\alpha}^2) \]  

(3)  

This function block takes an input vector, \( u \), where each component is referred to as \( u[1] \), \( u[2] \), etc. In our case, \( u[1] = r \), \( u[2] = \dot{r} \), \( u[3] = \alpha \), and \( u[4] = \dot{\alpha} \).  

Now, we will begin to construct the function input vector \( u \) by feeding back the state signals from the integrators and forming a vector from them with a Mux block.  

- Insert a Mux block from the Signal Routing library and connect its output to the input of the Ball-Beam block.  
- Edit the Mux block (by double-clicking on it) and change its number of inputs to 4. The Mux block should now have four inputs.  
- Tap a line off the d/dt(r) signal (hold Ctrl while drawing) and connect it to the second input of the Mux block.  
- Tap a line of the r signal and connect it to the first input of the Mux block.
Now we will construct the signals $\alpha$ and $\dot{\alpha}$ from the input $\theta$.

- Insert an In block on the left side of your model window. Change its label to "theta".
- Insert a Gain block and connect it to the theta block. Change its gain value (double-click on it) to "$d/L$".
- Connect the output of the gain block to the third input of the Mux block. Label this line "alpha".
- Insert a Derivative block from the Continuous library and place it underneath the alpha signal line.
- Tap a line off the output of the Gain block and connect it to the input of the Derivative block.
- Connect the output of the Derivative block to the fourth input off the Mux block.
To generate the open-loop response, it is helpful to first contain this model in a subsystem block.

- Create a new model window (select New from the File menu in Simulink or hit Ctrl-N).
- Insert a Subsystem block from the Ports & Subsystems library.
- Open the Subsystem block by double clicking on it. You will see a new model window labeled "Subsystem".
- Open your previous model window named ball.slx. Select all of the model components by selecting Select All from the Edit menu (or hit Ctrl-A).
- Copy the model into the paste buffer by selecting Copy from the Edit menu (or hit Ctrl-C).
- Paste the model into the Subsystem window by selecting Paste from the Edit menu (or hit Ctrl-V) in the Subsystem window.
- Close the Subsystem window. You will see the Subsystem block in the untitled window with one input terminal labeled theta and one output terminal labeled r.
- Resize the Subsystem block to make the labels visible by selecting it and dragging one of the corners.
- Label the Subsystem block "Ball and Beam Model".
- Insert a Step block (from the Sources library) and connect it to the input of the Ball and Beam Model.
- Edit the Step block (by double clicking on it to bring up the dialog box) and change the Step Time value to 0. Close the Step block dialog box.
- Insert a Scope block (from the Sinks library) and connect it to the output of the Ball and Beam Model.

Figure 3. The ball and beam model in simulink
4. SIMULATION RESULTS

Before obtaining a step response, we must set the physical parameters. Enter the following commands at the MATLAB prompt.

\[ m = 0.111; \]
\[ R = 0.015; \]
\[ g = -9.8; \]
\[ L = 1.0; \]
\[ d = 0.03; \]
\[ J = 9.99e-6; \]

We are now ready to run the simulation. Start the simulation by selecting **Run** from the **Simulation** menu (or hit **Ctrl-T**). When the simulation is finished, open the Scope by double clicking on it. You will see the following response.

Figure 4. Adding the step function and oscilloscope to the model

Figure 5. The open-loop response of ball and beam system
From this plot it is clear that the system is unstable in open-loop causing the ball to roll right off the end of the beam. Therefore, some method of controlling the ball’s position in this system is required.

5. CONCLUSIONS

A mathematical model of the ball and beam system was developed using physical laws. A simplified mathematical model was derived through system parameters. The controller parameters values were obtained by using manual tuning method from practical model so as to perform best system response. The accuracy of the system is tested by adjusting the position of the ball at three different points and it found that the accuracy doesn’t affected by changing the set point.

REFERENCES