Fun Control Experiments with Matlab and a Joystick

Marc Bodson*
Electrical & Computer Engineering, University of Utah
50 S Central Campus Dr Rm 3280, Salt Lake City, UT 84112, U.S.A.

Abstract

The paper describes real-time simulation programs that were developed for an undergraduate control systems course. The implementation of three classical experiments (the ball and beam, the inverted pendulum, and the flexible beam) as Matlab m-files is described. The visualization and animation capabilities of Matlab provide a realistic perception of the behavior of the testbeds, without actual hardware being needed. A joystick interface enables users to control the systems manually, providing a fun and educational experience. Automatic controllers can be designed to gain insight into a variety of concepts, including stabilization of unstable systems, root-locus properties, frequency-domain analysis and design, robustness, and discrete-time implementation of continuous-time systems.
The paper is submitted to the conference as a *poster/interactive* paper. Attendees will view the experiments on a laptop, interact with them through a joystick, and ask any questions that they may have. The software is available from the web at: *www.ece.utah.edu/~bodson/fun*. The joystick interface is a dynamic link library file requiring a Windows operating system.
Keywords: control education, real-time visualization, Matlab, ball and beam, inverted pendulum, flexible beam.

1. Introduction

Undergraduate control laboratories typically rely on a number of experiments to illustrate the principles of feedback design. These experiments are costly, whether one decides to develop the systems from scratch or to acquire them from a vendor. The testbeds also require constant care to stay in satisfactory operating condition within a teaching lab environment. Real-time experimentation is, however, greatly beneficial in terms of providing an understanding of control concepts, and giving the motivation to study the abstract material of control theory.

In this paper, we consider three standard experiments found in teaching labs: the ball and beam system (sometimes called beam and ball), the inverted pendulum, and a flexible beam. Instead of physical implementations, we discuss real-time simulations with visualization that avoid the costs and risks associated with actual hardware (see Fig. 1). Because the code is based on Matlab, students can easily and rapidly modify the code and experiment with a variety of controllers. They can also pursue these efforts at home. In contrast to the typical commercial systems, the simulated systems can be controlled manually through a joystick interface. Manual control is a great opportunity to have fun and to get better insight into the challenges of control. The simulation programs can also easily be modified to incorporate more complex models, or more elaborate systems.

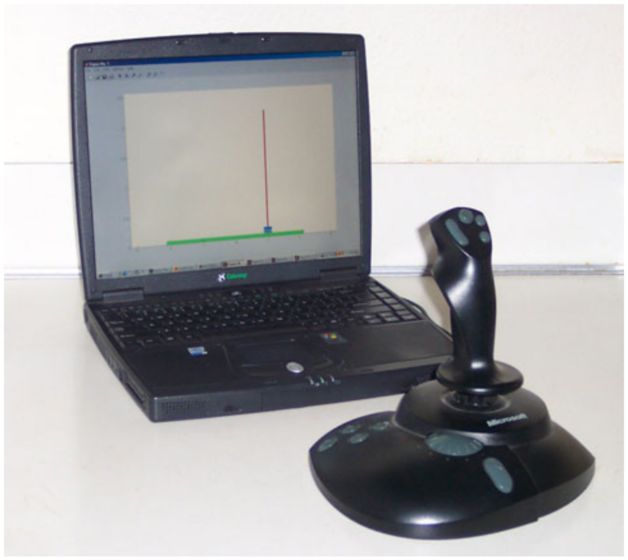
Figure 1: Controlling a simulated inverted pendulum through a joystick

2. Animation, Visualization, and Real-Time Control in Matlab

The simulations are coded as m-files in Matlab, taking advantage of simple, yet powerful Matlab functions for drawing and animation. The following macro illustrates the basic functions used in the simulations:

```
figure(1);clf;axis([-0.48 0.48 -0.16 0.16]);
set(1,'pos',[20 350 940 300]);hold on;
xbeam=[0.4;0.4;-0.4;-0.4];
```

*The author acknowledges the support of the National Science Foundation under Grant No. ECS0115070.

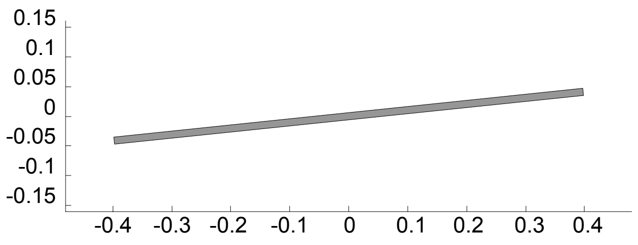

Figure 2: Snapshot of Matlab figure

```
ybeam=[-0.006;0.006;0.006;-0.006];
beam=fill(xbeam,ybeam,'green','EraseMode',...
   'background');
dt=0.05;t=0;tm=0;nt=0;done=0;tic;
while (done==0);
   while tm<nt;tm=toc;end;t=nt;nt=t+dt;
   theta=0.2*sin(t);
   cth=cos(theta);sth=sin(theta);
   rbeam=[xbeam ybeam]*[cth sth;-sth cth];
   xrbeam=rbeam(:,1);yrbeam=rbeam(:,2);
   set(beam,'Xdata',xrbeam,'Ydata',yrbeam);
   drawnow;
end;hold off;
```

Running the macro in Matlab will create a figure, reproduced on Fig. 2. The rectangle on the figure will oscillate around its center by ±11 degrees, with a period of about 6 seconds. The rectangle represents the beam in the ball and beam experiment to be discussed later. The code uses Matlab's ability to draw polygons (function *fill*), to rapidly change the coordinates of their vertices (function *set*) and to update their representation in a figure for animation (function *drawnow*). The argument 'pos' in the function *set* defines the size of the figure on the screen, with the argument in brackets being: [position from left, position from bottom, width, height]. The values are appropriate for a monitor with 1024x768 pixels, but may be changed if needed.

The program will run continuously until stopped by typing "Ctrl-C.". It can easily be modified to run for a fixed period of time, if desired. The *tic* command in Matlab initializes the timer and the *toc* command returns the new value of time. In an iteration of the "while" loop, t is the current value of time. dt is the sampling period. The program waits the new value of time specified by nt before going through a new iteration. One difficulty with Windows machines is that the precision of the timer is low. On the Windows 98 machines that we tested, the *toc* function returned several values equal to 0, then 0.05, 0.11, 0.16, and so on. Therefore, sampling rates above 20Hz could not be implemented. On a Windows XP machine purchased recently, the minimum sampling time was somewhat lower (0.016s). Fortunately, a sampling frequency of 20Hz was found adequate for visualization of the experiments described in this paper.

3. Ball and Beam

3.1 Simulation Model

The ball and beam (or beam and ball) system is an educational experiment that is fun to watch and play with. Several research papers have used it as an example, including [5] and [11]. A diagram is shown in Fig. 3. A beam is attached to a motor so that its angle θ with respect to the horizontal can be controlled at will. A ball is placed on the beam and is free to roll under the action of gravity (a small channel in the beam may keep the ball from rolling sideways). The distance of the ball from the center of the beam is denoted x. The ball can be placed at any location on the beam, and will stay there if its velocity v is zero and the beam angle is zero.

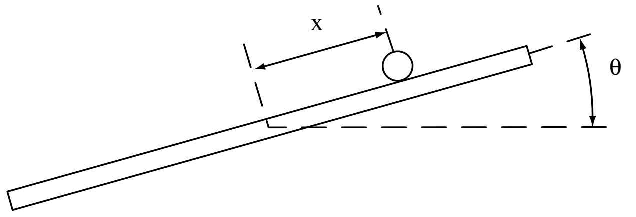

Figure 3: Ball and beam system

Assuming that the only force acting on the ball is gravity and assuming that the ball slides on the beam, the movement of the ball is determined by Newton's law

$$m\frac{d^2x}{dt^2} = -mg\sin(\theta) \qquad (1)$$

where g is the acceleration of gravity and m is the mass of the ball. If the ball rolls on the beam, instead of sliding, the rotational inertia of the ball adds itself to the translational inertia, resulting in a factor of $5/7$ in the right-hand side of the equation of motion. The state-space model describing the movement of the ball is then

$$\frac{dx}{dt} = v, \quad \frac{dv}{dt} = -\frac{5}{7}g\sin(\theta). \qquad (2)$$

where x is the position of the ball and v is its velocity.

More detailed models have been used in the literature (see, *e.g.,* [12]), accounting for the beam dynamics and for nonlinear effects such as the centrigugal force acting on the ball. We have found that nonlinear effects were small in a typical laboratory experiment, and that controlling the system manually with motor torque as an input was very difficult, if not impossible.

Assuming that the angle θ is small, and replacing $\sin(\theta)$ by θ, the transfer function of the linearized system becomes the so-called *double integrator*, which is often encountered in control applications. Newton's law, $F = m.a$, generally yields this transfer function if force is the control variable and position is the output variable. Moving a spacecraft with thrusters is a practical example of such a system. Therefore, the control problem for the double integrator is very common, and representative of many applications.

The model of the ball and beam system is implemented in the Matlab simulation *bbeam.m*, available from the web. The length of the beam matches a ball and beam system available at the University of Utah and developed as part of an earlier effort [9]. An Euler approximation of (2) is implemented. The sampling period is set to 0.05s, or a frequency of 20Hz. The control signal is the angle of the beam and is limited in the code to ±5 degrees. The position of the ball and the velocity of the ball are available for feedback, although a practical design requires reconstruction of the velocity. In the simulation program, the position of the ball is limited to the length of the beam ($\pm 0.4m$) by setting the velocity to zero when the end of the beam is reached. Friction of the ball rolling on the beam is simulated by setting the velocity to zero if the ball velocity and the beam angle are small enough.

3.2 Manual & Automatic Control

The simulation program gives the option of manual control or automatic control. Manual control is immediately available through the joystick interface, while automatic control requires that the user provide the appropriate m-files. Manual control is a good opportunity for students (as well as the instructor!) to have fun and to get a sense for the challenges of controlling the system. Two lines are drawn on the beam and an objective is to move the ball from one line to the other as fast as possible. A difficulty is the tendency to overshoot the target position, due to the velocity of the ball. After a few trials, one learns to incorporate a perception of the ball velocity in the control strategy.

For automatic control, two m-files must be written: a file called *bbeamc.m* containing the control algorithm, and an initialization file called *bbeamcinit.m.* The initialization file is called once before the simulation starts, while the control algorithm is called at the same rate as the ball and beam simulation. The program does not store the time histories of the signals, but it is a simple exercise to add the appropriate instructions to the code. As a controller, a proportional-derivative (PD) controller can be chosen, so that

$$\theta = k_p e + k_v \frac{de}{dt}, \qquad \text{with } e = x_{REF} - x \tag{3}$$

Assuming that both the ball position and the ball velocity can be measured, and neglecting dx_{REF}/dt, the control law may be implemented simply with

$$\theta = k_p e - k_v v \tag{4}$$

Choosing the parameters k_p and k_v so that the two closed-loop poles are real and equal, with an associated time constant of 0.3 seconds, yields good results. The ability of a simple control law to rapidly move the ball from one side of the beam to the other is surprising, and particularly impressive after having struggled with the challenges of manual control.

4. Inverted Pendulum

4.1 Simulation Model

The inverted pendulum system is a favorite experiment in control system labs. It is used as an example in many textbooks, including [4] and [8]. The highly unstable nature of the plant enables an impressive demonstration of the capabilities of feedback systems. The inverted pendulum is also considered a simplified representation of rockets flying into space.

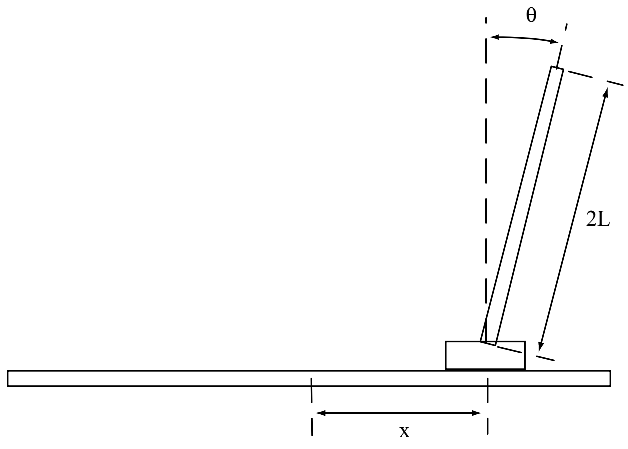

Figure 4: Inverted pendulum system

Fig. 4 shows a diagram of the experiment. A cart rolls along a track, with its position x being controlled by a motor. A beam is attached to the cart so that it rotates freely at the point of contact with the cart. The angle of the beam with the vertical is denoted θ, and an objective is to keep the angle close to zero. Since the pendulum may be stabilized at any position on the track, a second objective is to specify the track position. However, recovery from non-zero beam angles may require significant movements of the cart along the track, and stabilization may be impossible if an insufficient range of motion remains.

A model of the inverted pendulum may be derived using standard techniques. A careful derivation [6] gives the equation

$$(I + mL^2)\frac{d^2\theta}{dt^2} + mL\cos(\theta)\frac{d^2x}{dt^2} = mgL\sin(\theta) \tag{5}$$

where m is the mass of the beam, $2L$ is the length of the beam, $I = mL^2/3$ is the moment of inertia of the beam around its center of gravity, and g is the acceleration of gravity. For small angles θ, it follows that

$$\frac{4}{3}L\frac{d^2\theta}{dt^2} - g\theta = -\frac{d^2x}{dt^2} \tag{6}$$

and, the transfer function of the system is

$$\frac{\Theta(s)}{X(s)} = \frac{-s^2}{(4/3)Ls^2 - g} \tag{7}$$

The system has poles at $s = \pm\sqrt{3g/4L}$. The positive root is unstable, and instability worsens when the beam is short. Another tricky problem is that the system has two zeros at $s = 0$. This is due to the fact that an *acceleration* of the cart is required to impact the beam angle. On the other hand, this property makes stabilization possible for arbitrary cart positions.

More complex models assume that the control variable is the force applied to the cart, rather than its position. This assumption is more realistic, but makes manual control very difficult. Instead, the simulation program *invpend.m* assumes that some type of inner control loop provides tracking of position commands for the cart. The delay in the motion of the cart is represented by the response of a first-order system

$$X(s) = \frac{f}{s+f}\, X_{com}(s) \tag{8}$$

where x_{com} is the commanded cart position, x is the cart position, and $f > 0$. The overall transfer function of the system is then

$$\frac{\Theta(s)}{X_{com}(s)} = \frac{bs^2}{(s+a)(s-a)(s+f)} \tag{9}$$

where $a = \sqrt{3g/4L}$, $b = -3f/4L$.

The program assumes that $L = 2$ meters, and that x is limited to $\pm L$. The beam angle is prevented from exceeding $\pm 30°$. The value of f is 5, so that the cart responds relatively fast to commands. Even so, the system is difficult to control manually. The program has a "*cheat*" variable that allows one to make the problem easier. The variable reduces the magnitude of the gravity constant, and the choice *cheat* $= 6$ is preset in the program (it is as if one controlled the pendulum on the moon!). The system is simulated in discrete-time by assuming a sampling rate of 20Hz and using a zero-order-hold equivalent system.

4.2 Manual and Automatic Control

It takes some time to learn how to keep the beam balanced. Because of the instability, the system requires constant attention. As for the ball and beam system, one finds that perception of the angular velocity of the beam is critical to the success of the control strategy. Once one feels comfortable with balancing the beam, one may try to reduce the "*cheat*" variable, or to move the cart from one line drawn on the track to the other.

For automatic control, one should first design a stabilizing controller. The controller transfer function must be implemented in a file *invpendc.m*, with its initialization in a file *invpendcinit.m.* Given (9), an interesting exercise is to show that the system can only be stabilized by an *unstable* controller. Among possible choices, the second-order controller

$$C(s) = \frac{X_{com}(s)}{-\Theta(s)} = k\frac{(s+a)(s+f)}{(s-a)(s+c)} \tag{10}$$

is a simple one. After cancellations, the closed-loop system only has three poles. It is possible to choose the controller parameters k and c so that the three closed-loop poles are placed at $s = -a/2$. The problem is a good opportunity to apply root-locus methods, the Nyquist criterion, gain and phase margin concepts, and discrete-time implementation of continuous-time systems.

As a second step, one may design a controller such that an arbitrary set-point of the cart position can be imposed. The previous controller may be replaced by

$$X_{com}(s) = C_f(s)X_{ref}(s) - C(s)\Theta(s) \tag{11}$$

This is an unusual configuration for a feedback system, because the reference command is applied to the plant input rather than the controller input. Without $C_f(s)$, the response exhibits overshoot, due to a low-frequency zero of the transfer function from $X_{ref}(s)$ to $X(s)$. $C_f(s)$ can be chosen as a first-order filter with unity DC gain and a pole placed to cancel the low-frequency zero.

5. Flexible Beam

5.1 Simulation Model

Systems with lightly-damped complex poles (resonances) are encountered in many applications. An example is a large robotic arm in space, whose transversal dimensions are made small to reduce weight. The arm will bend and oscillate if moved rapidly. In a computer disk drive, a read/write head is attached to the end of a small, rigid structure that is rotated rapidly to access various tracks. When the head is positioned within fractions of microns, even such a rigid structure behaves like a flexible structure.

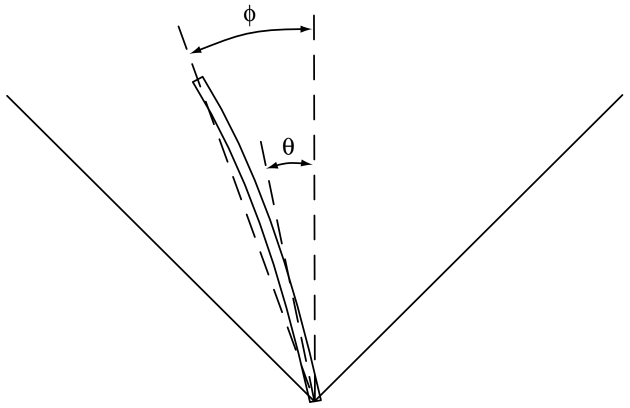

Figure 5: Flexible beam

The diagram of the flexible beam is shown on Fig. 5. Research papers on this subject include [2], [3], [7], [10], and [13]. The angle of the beam at the shaft is denoted θ, while ϕ is the angle at the tip. Experimental data was collected on a flexible beam of length $0.4m$ available at the University of Utah and used in previous experiments [1]. Fig. 6 shows the frequency response that was measured from the motor current to the angular acceleration of the shaft, while Fig. 7 shows the response measured from the motor current to the angular acceleration of the tip. The acceleration at the shaft was obtained by measuring the position with an encoder (and multiplying the frequency response by $-\omega^2$ to obtain the acceleration), while the acceleration at the tip was obtained with an accelerometer. The plots show the experimental data (solid lines), as well as the approximate fits obtained with fourth-order models (dashed lines).

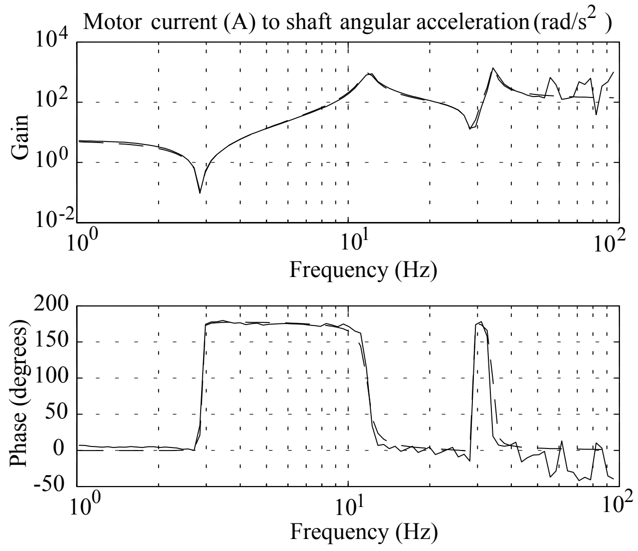

Figure 6: Frequency response of the flexible beam from motor current to shaft angular acceleration

The approximate models shown on the plots are given by

$$\begin{aligned} \frac{s^2\Theta(s)}{I(s)} &= \frac{k_\theta(s-z_1)(s-z_1^*)(s-z_2)(s-z_2^*)}{(s-p_1)(s-p_1^*)(s-p_2)(s-p_2^*)} \\ \frac{s^2\Phi(s)}{I(s)} &= \frac{k_\phi(s-z_3)(s-z_4)(s-z_5)(s-z_6)}{(s-p_1)(s-p_1^*)(s-p_2)(s-p_2^*)} \end{aligned} \quad (12)$$

where $p_1 = -3+74j$, $p_2 = -3+215j$, $z_1 = -0.07+18j$, $z_2 = -0.07+180j$, $z_3 = 100$, $z_4 = -120$, $z_5 = 200$, and $z_6 = -300$ (note that the poles are very lightly damped). The input variable is the current in the motor, measured in A, and the angles θ and ϕ are measured in radians. The constants k_θ and k_ϕ are such that the DC gains of the transfer functions are equal, with

$$k_p = \left(\frac{s^2\Theta(s)}{I(s)}\right)_{s=0} = \left(\frac{s^2\Phi(s)}{I(s)}\right)_{s=0} = 5.5 \qquad (13)$$

The equality for θ and ϕ follows from the fact that there is no bending of the beam near zero frequency. For low frequencies, the transfer functions are approximately given by

$$\frac{\Theta(s)}{I(s)} \simeq \frac{\Phi(s)}{I(s)} \simeq \frac{k_p}{s^2} \qquad (14)$$

This approximation of the system is the double integrator encountered with the ball and beam system. The feedback design is more difficult than for the ball and beam, however, because of additional poles close to the $j\omega$-axis, and because of zeros close to the $j\omega$-axis and in the right half-plane.

The continuous-time model was discretized assuming a sampling period of 200Hz. Since the program runs at a rate of approximately 20Hz, the visualization slows the dynamics by a factor of 10. This result is helpful, because the dynamics of the actual system are too fast to be controlled manually. For visualization, the deflection of the beam was assumed to be quadratic.

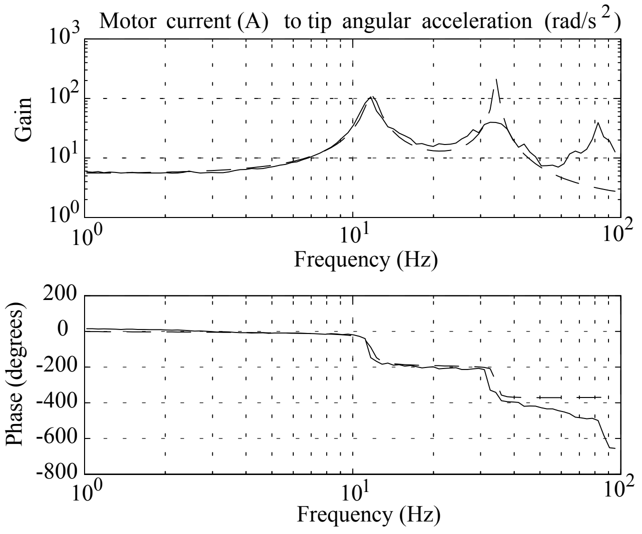

Figure 7: Frequency response of the flexible beam from motor current to tip angular acceleration

5.2 Manual and Automatic Control

Similarly to the ball and beam and inverted pendulum simulations, two lines are drawn on the screen to provide an objective of moving the beam from one line to the other. Two difficulties are encountered: the $1/s^2$ behavior and the flexibility of the beam. Because of the double integrator, the user will tend to stop the beam too late and to apply large commands, exciting vibrations. Resonances can be avoided by moving the beam slowly, but performance is unimpressive. It is enlightening (and fun) to excite the resonances by applying commands in the same frequency range as the flexible modes. Once this mechanism is understood, one may return to the task of rapidly moving the beam from side to side without exciting such resonances.

For automatic control, files *flexc.m* and *flexcinit.m* must be written. The flexible beam is a good opportunity to explore design in the frequency-domain. A *lead controller* may be designed with

$$C(s) = \frac{I(s)}{\Phi_{ref}(s) - \Phi(s)} = k_c \frac{(s+b)}{(s+a)} \qquad (15)$$

where the tip angle ϕ is the output to be regulated and

ϕ_{ref} is the reference value. Keeping only the first resonant mode in (12), one may compute analytically the values of the controller parameters k_c, a, and b such that the phase margin is 60°, the gain margin is 10, and the phase crossover frequency matches the first resonant frequency ω_n (approximately). A discrete-time equivalent of the control system must be obtained, for implementation. However, discretization should be based on a 200Hz sampling frequency (visualization will show the system at a rate slowed down by a factor of 10).

The lead controller design produces a relatively slow response and a large overshoot due to the low frequency zero at $s = -b$. The results can be improved by cascading the lead controller with a notch filter and by prefiltering the reference input, so that

$$I(s) = C(s)C_n(s)\left(C_f(s)\Phi_{ref}(s) - \Phi(s)\right) \qquad (16)$$

$C(s)$ is the lead controller (15) and

$$C_f(s) = \frac{1.2b}{s+1.2b}, \quad C_n(s) = \frac{s^2+\omega_n^2}{s^2+2\omega_n s+\omega_n^2} \qquad (17)$$

where b is the zero of $C(s)$, and ω_n is the natural frequency of the first resonant mode. The prefilter eliminates the overshoot, and the notch filter allows one to increase the crossover frequency and the speed of response. With these modifications, the response of the system is much improved and, as with previous designs, impressive compared to what can be achieved manually.

6. Conclusions

Three Matlab simulation programs were described that emulate the experience gained with classical testbeds for control design and implementation. Real-time visualization gives a picture of the results obtained, without actual hardware being needed. In contrast to the usual experiments, users can "play" with the systems through a joystick interface, enabling them to gain an appreciation for the challenges of control. Although no match to the latest video games on the market, the experiments are great fun, and it can be an exciting experience to rapidly test and modify control laws. The experiments may be used to illustrate fundamental issues in control systems, including stabilization of unstable systems, pole placement, frequency-domain compensation, effect of actuator saturation, discrete-time implementation, and others. The code may also be modified easily to implement more complex models (including nonlinearities, noise, or other effects), or to simulate more elaborate systems (such as a double inverted pendulum, for example). The minimal cost and the safety associated with the experiments make them particularly attractive for teaching and for other demonstrations.

7. References

[1] M. Bodson, "An Adaptive Algorithm for the Tuning of Two Input Shaping Methods," *Automatica*, vol. 34, no. 6, 1998, pp. 771-776.

[2] R.H. Cannon & E. Schmitz, "Initial Experiments on the End-Point Control of a Flexible One-Link Robot," *Int. Journal of Robotics Research*, vol. 3, no. 3, 1984, pp. 62-74.

[3] P.-M. Chang & S. Jayasuriya, "An Evaluation of Several Controller Synthesis Methodologies Using a Rotating Flexible Beam as a Test Bed," *Trans. of the ASME*, vol. 117, 1995, pp. 360-373.

[4] R.C. Dorf & R.H. Bishop, *Modern Control Systems*, 9th edition, Addison-Wesley, Menlo Park, CA, 2001.

[5] J. Hauser, S. Sastry, & P. Kokotovic, "Nonlinear Control via Approximate Input-Output Linearization: The Ball and Beam Example," *IEEE Trans. on Automatic Control*, vol. 37, no. 3, 1992, pp. 392-398.

[6] H. Khalil, *Nonlinear Systems,* 2nd edition, Prentice Hall, Upper Saddle River, NJ, 1996.

[7] H. Krishnan & M. Vidyasagar, "Control of a Single-Link Flexible Beam Using Hankel-Norm-Based Reduced-Order Model," *IEE Proc.-Control Theory Appl.*, vol. 145, no. 2, 1998, pp. 151-158.

[8] B. Kuo, *Automatic Control Systems*, 7th edition, Prentice Hall, Englewood Cliffs, NJ, 1995.

[9] M. Bodson, J. Lehoczky, R. Rajkumar, L. Sha, J. Stephan, & M. Smith, "Control Reconfiguration in the Presence of Software Failures," *Proc. of the Conference on Decision and Control*, San Antonio, TX, 1993, pp. 2284-2289.

[10] E. Laukonen & S. Yurkovich, "A Ball and Beam Testbed for Fuzzy Identification and Control Design, *Proc. of the American Control Conference*, San Francisco, CA, 1993.

[11] R. Olfati-Saber & A. Megretski, "Controller Design for the Beam-and-Ball System," *Proc. of the 37th IEEE Conference on Decision and Control*, Tampa, FL, 1998, pp. 4555-4560.

[12] M. Sobhani, B. Neisius, S. Jayasuriya, E. Rumler, & M.J. Rabins, "Some New Insights in the Classical Beam and Ball Balancing Experiment," *Proc. of the American Control Conference*, 1992, pp. 450-454.

[13] A.P. Tzes & S. Yurkovich, "Application and Comparison of On-Line Identification Methods for Flexible Manipulator Control," *Int. Journal of Robotics Research*, vol. 10, no. 5, 1991, pp. 515-526.