EECS 461 Fall 2009
Lab 6: Virtual Worlds with Time

1 Overview

In this lab, we will design and implement a number of simple virtual worlds. These worlds are various connections of static positions (walls), dampers (dashpots), and springs. Our interface to the virtual worlds is the haptic wheel (the puck), which we used previously. The haptic wheel is a rotational device that can apply torque to the user's hand. We draw the systems below as translational systems only for simplicity.

We will then design a virtual knob that gives feedback to the user. The fundamental difference between these virtual worlds and those of Lab 4 is that we now require a notion of time, which will be supplied by the interrupt timer (DEC), which we learned to use in Lab 5. We shall also use the functions for quadrature decoding and Pulse-Width Modulation (PWM) that we developed in earlier labs.

1.1 Virtual Wall-Damper

This system is similar to the virtual wall from Lab 4. We will add damping to the virtual wall to reduce the chatter felt with high spring constants (see notes on the Virtual Wall on the course website). There is an "ideal" value, derived in lecture; we will test the system with this ideal value and various other values.

1.2 Virtual Spring-Damper

This system is similar to the virtual spring from Lab 4, except that the puck is now connected to the reference point with a damper as well as a spring. The purpose of the damper is to providing a retarding force proportional to the speed of the puck: $F_d = -b\frac{dx}{dt}$. This force is in addition to the restoring force of the spring: $F_d = -kx$.

In our application, the motor must supply a torque equal to the sum of the virtual spring and virtual damper torques. To compute the latter, we need a measure of the velocity of the wheel. An effective way to determine velocity is through numerical differentiation. By setting the DEC so that an interrupt occurs every T_s seconds, velocity may be approximated by the first difference $\frac{\Delta x}{\Delta t}$, where $\Delta x = (x_{current} - x_{old})$ and $\Delta t = (t_{current} - t_{old}) = T_s$.

1.3 Virtual Spring-Mass

For the virtual spring system in Lab 4, the puck was attached by a virtual spring to an immovable reference point. For the virtual spring-mass system in Figure 2, the puck is attached by a virtual spring to a free virtual mass that is movable. As we shall see, the dynamics of a system with two objects (the puck and the free mass) that may move are much more complex than the dynamics of a system with only one moveable object.

Assume that the mass, when applying a force on the puck, does not cause the puck to move. The puck represents the user's hand, and because the user is holding the puck, it does not move.

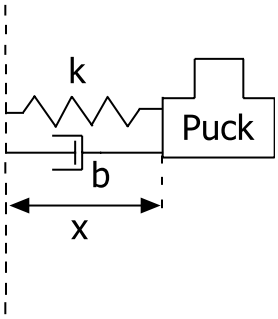

Figure 1: Spring-Damper System

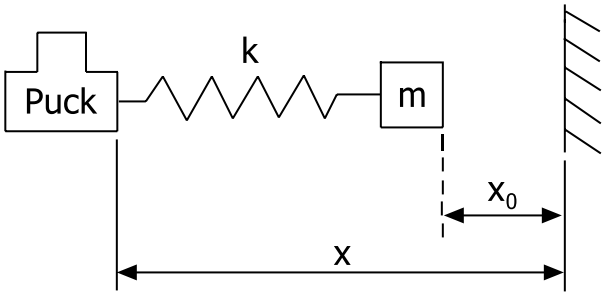

Figure 2: Spring-Mass System

1.4 Virtual Spring-Mass-Damper

The third virtual world is a combination of the previous two: a freely moving mass is connected to a puck by both a damper and a spring. This results in a coupled set of second-order differential equations. The system this models is shown in Figure 3.

1.5 Virtual Knob

A virtual knob is an implementation of a mechanical selector knob with detents that can be felt at regular intervals, as you might find on a car stereo or home theater receiver. Review the TouchSense [1] document provided on the course web page before coming to lab, to familiarize yourself with the uses of programmable haptic effects.

2 Design Specifications

2.1 Hardware

As we learned in lab 4, the haptic interface available in lab contains a power supply, an amplifier, an encoder, a motor, and a DB-25 connector for connection to the interface board. The power supply converts 120 VAC into 24 VDC to be used by the amplifier. The amplifier receives a PWM signal from 'MIOS0' and drives the motor. The amplifier receives the PWM signal as a current command and drives the motor with a commanded current that is proportionally related to the torque applied by the motor.

The DB-25 connector connects the PWM motor control signal, and two quadrature encoder signals on 'TPU0' and 'TPU1'. By connecting the parallel cable between the haptic interface and the interface board,

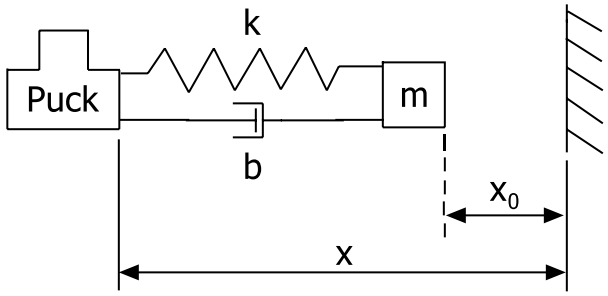

Figure 3: Spring-Mass-Damper System

no further connections are required to control the haptic interface.

2.2 Software

You will need to use your FQD, PWM, and ISR software from the previous labs to control the haptic interface. Note that the PWM output needs to be **limited to 35 to 65 percent duty cycle** to prevent damage to the haptic devices. These values can be set as constants within *mios.h*.

Your *lab6.c* file should have your main function and call the necessary initialization functions to setup your virtual world. A majority of numerical processing will take place in the ISR called by the DEC, not in the main function.

The DEC ISR is called at a fixed periodic rate. It should determine the position and velocity of the haptic wheel, call the virtual world function with this information, and then output the torque to the motor. The virtual world function, then, is a transfer function from haptic position and velocity to torque, or from haptic input to haptic output.

For both the spring-damper, spring-mass, and spring-mass-damper systems, you will need to use the DEC to keep track of time. Note that to compile the code that uses the DEC, you can use the makefile provided.

3 Pre-Lab Assignment

*Pre-lab questions must be done **individually** and handed in at the start of your lab section. You must also, **with your partner**, design an initial version of your software before you come to your lab section. Your software does not need to be completely debugged, but obviously more debugging before coming to lab means less time you will spend in the lab.*

1. [**5**] **Create a function to control the virtual spring-damper system.** This function will be called periodically by the DEC ISR to calculate and apply the appropriate torque response to the system at that time. The DEC will need to be used in order to develop a concept of time in the microcontroller; the velocity calculation will be done in the ISR, not the virtual world function.

   Follow the function prototype for `virtualSpringDamper`, as provided in the *worlds.h* file, and implement the function in the *worlds.c* file. The two input arguments represent the current angle of the wheel and the current velocity of the wheel.

2. [**2**] **Add damping to the virtual wall to reduce chatter due to the half-sample delay of the ZOH.**

3. **Create a function to control the virtual spring-mass system.** Use the equations developed in class for state space approximations of the differential equations. Note that unlike in the above example our system has a rotational inertia instead of a linear inertia and thus for the remainder of the lab we will use J to denote rotational inertia (angular mass).

   (a) **[2+2] First, calculate the spring constant k and angular mass J so that the following property is satisfied:** Suppose that you turn the wheel 45 degrees and hold its position constant. Then the virtual mass should oscillate with a 1 Hz frequency, and the maximum torque generated should be 800 N-mm.

   (b) **[1] Verify with a Matlab simulation that the values you chose give the correct behavior, and include a response graph in your pre-lab.** You may use the provided Matlab files as a starting point.

   (c) **[1] Second, model a discrete implementation of the virtual spring mass system, using Forward Euler integration to approximate the analog integrators. Use a 1 millisecond integration step size. Where do the closed loop characteristics roots lie?**

   (d) **[1] Third, compute the precise amount of damping required to yield a stable discrete oscillator.**

4. **[1] Open *worlds.h*, create the prototype function for virtual knob. Include this file in your Pre-lab report.**

4 In-Lab Assignment

1. Connect a haptic device to the interface board.

2. Implement the virtual wall with damping (the **wall-damper** system). Vary the amount of damping to see how the level of chatter varies, but be sure to try the value calculated in lecture to give ideal results. For this experiment, it is useful to use a dipswitch to turn the damping on and off. In this way, the amount of chatter with and without damping can be easily observed.

3. Implement and test the virtual **spring-damper** system.

4. Increase the damping coefficient b from zero until the motion of the wheel becomes critically damped. That is, the wheel should not oscillate about its final position. Be sure to record this value after you test it several times.

5. Implement and test the virtual **spring-mass** system. Show your instructor the results of your Matlab simulations before testing on the wheel.

6. The inherent dynamic friction in the system means that for low spring constants, the spring-mass system is stable despite using the Forward Euler integration method. By choosing a large enough value for the spring constant, however, the system will become unstable. Increase the spring constant, k, to a large value in order to make the system unstable.

7. Implement and test the **spring-mass-damper** system. Show your instructor the results of your Matlab simulations before testing on the wheel.

8. Using the header files given in previous labs, develop a terminal interface to the spring-mass-damper system. This interface should run as task code in the main program while the spring-mass calculations will run as an interrupt routine. With the terminal interface, provide a way to change the spring constant and the mass of the system using the '+' and '-' keys. Adjust the parameters and notice how the system changes. As you change your spring constant and mass values, your damping should also change such that the resulting system is stable.

   Note that the virtual spring constant and mass are shared between the main program and the interrupt routine. Hence the interrupt routine should not be allowed to read these parameter values while the

main program is changing them. (Otherwise, the interrupt routine might use corrupted values of the parameters.) A section of code that must execute without being interrupted is reffered to as a *critical section*. Define the commands `ENTER_CRITICAL()` and `LEAVE_CRITICAL()` to disable and reenable interrupts in your main routine while the new values of the spring and mass constants are being calculated.

Example code:

```
#define ENTER_CRITICAL() asm (" wrteei 0"); /* disable all interrupts */
#define LEAVE_CRITICAL() asm (" wrteei 1"); /* enable all interrupts */

void main()
{
    while(1)
    {
        ...
        ENTER_CRITICAL();
        K = K*1.1;                          /* while main manipulates K
                                                  /* isr must not read K */
        LEAVE_CRITICAL();
        ...
    }
}
```

9. Create a new virtual world in *worlds.c* called **virtualKnob** that, like the other virtual worlds, takes the angle and velocity of the haptic wheel and returns a torque. Create a world in which a resistive force is felt when turning the haptic wheel, to simulate friction. Because this force is only present when the wheel is moving, should you use a spring force or should you use damping? Use your virtualKnob world in *lab6.c* and test your system to find settings that provide smooth resistance to motion, like you might feel on a volume knob of a stereo receiver. Be sure to insert the appropriate function prototype in *worlds.h*.

10. Add virtual detents to your **virtualKnob** system. Use the constant friction you developed earlier, but when the wheel reaches certain angles, alter the force such that the user feels a bump or a tick. Configure the detents (the bumps felt at the wheel) to be 10 degrees apart, and demonstrate the system to your GSI. Because you have full control over the severity and spacing of the detents, configurable tactile feedback is useful when interfaced with many software applications, as described in [1].

5 Post-Lab Assignment

1. **[0+3] How much damping is required in order to prevent the virtual spring-damper system from oscillating about its final position (In-Lab Step 4)? This is the critical damping value. What information would we need to have in order to be able to compute this damping value?**

2. **[3] You should have noticed that the chatter did not completely disappear when you added damping to the virtual wall. The addition of damping addresses chatter caused by hysteresis in the torque calculation. What factor that causes chatter is not eliminated by damping? Hint: How much does the torque change per degree? Per encoder tick?**

3. **[3] Suppose that you wanted a stable numerical integration algorithm without adding damping. What other integration algorithm could you choose; what other effects would this have?**

4. **[1] Briefly describe a theoretical application for your `virtualKnob` system.**

5. **[25 IN] Include well-documented copies of your files controlling each virtual world.**

If you have comments that you feel would help improve the clarity or direction of this assignment, please include them following your postlab solutions.

[15 pre, 25 in, 10 post = 50 total]

References

[1] *TouchSense Programmable Rotary Modules*, Immersion Corporation, www.immersion.com, Lit#BR.TSRotary.1004.1000.v3 (2004)