Notes on the Virtual Spring Mass System*

J. S. Freudenberg

EECS 461, Fall 2006

1 Human Computer Interaction

A force feedback system, such as the haptic wheel used in the EECS 461 lab, is capable of exhibiting a wide range of interesting phenomena. It is useful to remember that the system consists of a mechanical device (the wheel in our case), with two feedback loops wrapped around it. As shown in Figure 1, one feedback loop consists of (i) a sensor such as an encoder, tachometer, or strain gauge, (ii) a microprocessor, and (iii) an actuator such as a DC motor. The purpose of the sensor is to measure position, velocity, or force, the purpose of the microprocessor is to implement an algorithm (the virtual world) that takes the sensor input and computes a force output, and the purpose of the motor is to impose that force on the haptic device. The other feedback loop consists of the human sensing the output of the haptic device through the skin, executing some behavior in response, and exerting a force on the haptic device through the muscles.

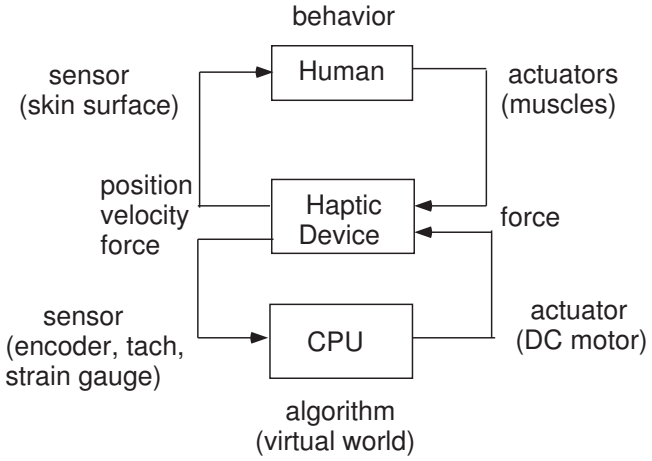

Figure 1: Human Computer Interaction through Force Feedback

The feedback loop involving the human is the most difficult to analyze because the human is unpredictable. Nevertheless, we can postulate several possible human behaviors that can be analyzed. We shall now examine three of these that may be explored in the lab.

2 A Step Change in Wheel Position

Suppose that we implement a virtual world consisting of a mass, m attached to the puck by a spring with constant k (Figure 2). The equations of motion of the system are

$$\ddot{w} + \frac{k}{m} w = \frac{k}{m} z. \tag{2.1}$$

*Revised October 9, 2006.

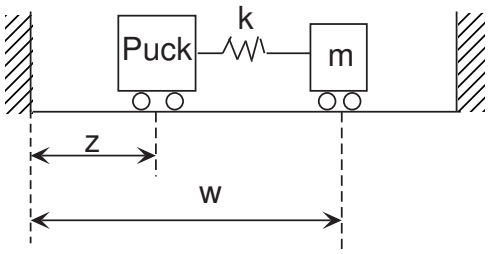

Figure 2: Virtual Spring Mass System

Note that this equation describes a harmonic oscillator with natural frequency $\omega_n = \sqrt{k/m}$ rad/sec. A block diagram representation is shown in Figure 3. The force exerted on the virtual mass is given by

$$F = k(z - w). \tag{2.2}$$

It is this force, which depends on both the spring constant k and the relative position of the virtual mass with respect to the puck, that the user will feel exerted on the haptic device by the DC motor. The solution to the differential equation in response to zero initial conditions and a constant input $z(t) = z_0$ starting at $t = t_0$ is given by

$$w(t) = z_0(1 - \cos(\omega_n(t - t_0)),\ t \geq t_0. \tag{2.3}$$

The resulting force exerted on the virtual mass is given by

$$F(t) = kz_0 \cos(\omega_n(t - t_0)),\ t \geq t_0. \tag{2.4}$$

Note several key points:

(i). The behavior of the virtual mass (position, velocity, acceleration, natural frequency) depends only on the ratio k/m.

(ii). By varying the spring constant k while keeping the ratio k/m fixed, we can influence the reaction force required of the DC motor and experienced by the user.

(iii). The maximum force experienced by the user will have magnitude equal to kz_0.

It follows from these observations that we should be able to construct virtual spring mass systems with a variety of natural frequencies while keeping the force exerted by the motor in response to an initial movement of the puck limited to an achievable value.

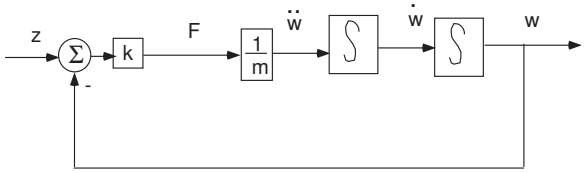

Figure 3: Block Diagram of the Virtual Spring Mass System

2.1 The Virtual Wheel/Torsional Spring System

In fact, our force feedback system does not display linear motion, as shown in Figure 1, but rather it displays rotary motion as shown in Figure 4. In this figure, the "puck" is our haptic wheel, and is shown connected to a virtual wheel by a shaft that is not rigid, and that thus behaves like a torsional spring. With the angles θ_z and θ_w given in radians, the rotational inertia of the virtual wheel is J_w N-m/(deg/sec^2) and the torsional spring constant is given by k N-m/deg. The equations of motion of the system are

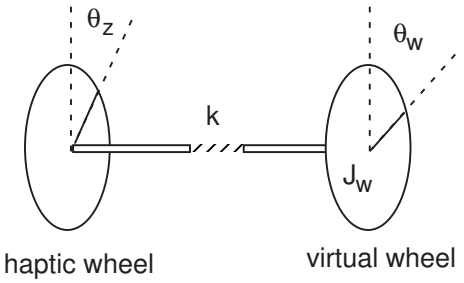

Figure 4: Virtual Wheel and Torsional Spring

$$\ddot{\theta}_w + \frac{k}{J_w}\theta_w = \frac{k}{J_w}\theta_z. \tag{2.5}$$

Note that (2.5) is identical to (2.1); both describe harmonic oscillators. The qualitative behavior of the two systems is thus identical. However, to actually write embedded software to implement the virtual worlds, one must account correctly for the actual values of the constants, and thus also for their units.

Suppose that we wish to move the wheel no more than a fixed amount (say 45°). The reaction torque generated in response to this motion should be no more than, say $800Nmm$, so that we do not hit the software limit on duty cycle. The natural frequency of the oscillator should be 1 Hz. We may simulate this system with the SIMULINK model found in Figure 5. A typical response is shown in Figure 6.

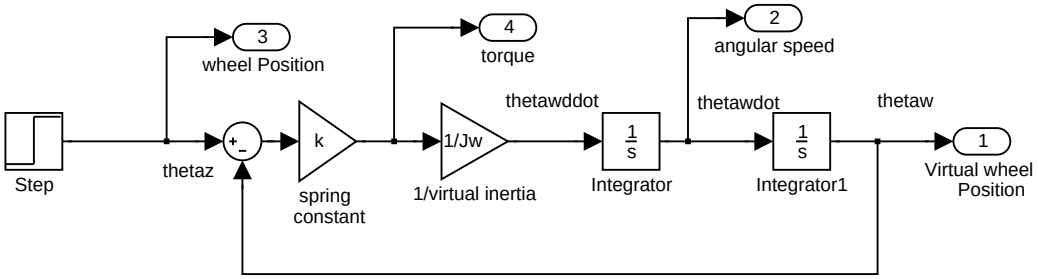

Figure 5: Simulation of Virtual Wheel and Torsional Spring

2.2 Numerically Unstable Implementation of the Virtual Wheel/Torsional Spring

As we discussed in class, to implement the virtual wheel/torsional spring system, it is necessary to numerically integrate the differential equation (2.1). We showed in class how to do this using the forward Euler method. Although forward Euler integration is numerically unstable, the fact that we are only interacting with the virtual spring-mass system for a finite time, together with the assumption of a relatively fast sample rate, suggests that this instability will not be a problem.

As it happens, the sample rate we use ($T = 1$ msec) is not sufficiently fast, and we do see the effect of the instability in the lab, as well as in Matlab simulations. To illustrate, consider the discrete implementation pictured in Figure 7. The resulting response is shown in Figure 8. Note that the virtual torque generated by the numerical integration increases until it hits the software limit on duty cycle, and thus on torque.

2.3 Analysis of Numerical Instability

How may the instability noted above be resolved? One way would be to use a numerically stable numerical integration technique, such as the trapezoidal method. Another way would be to add some damping to the model. We now explore the latter method.

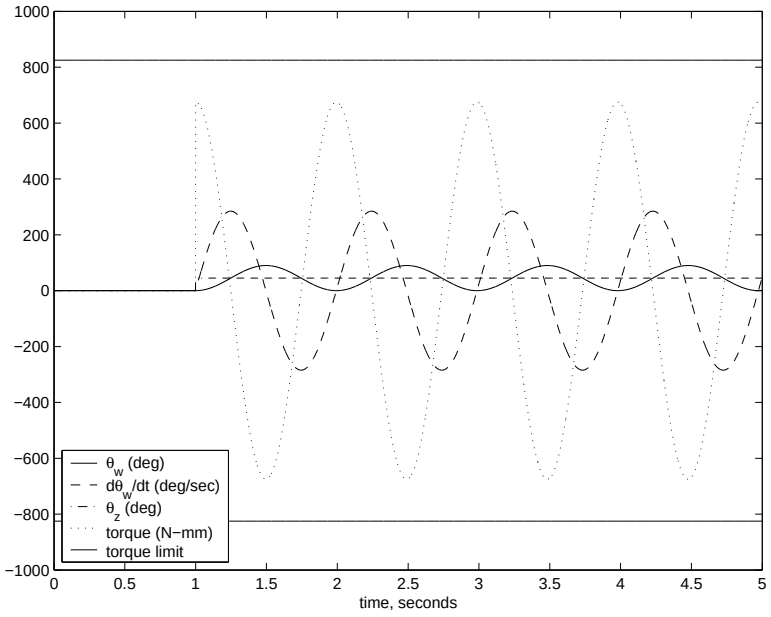

Figure 6: Response of Virtual Wheel

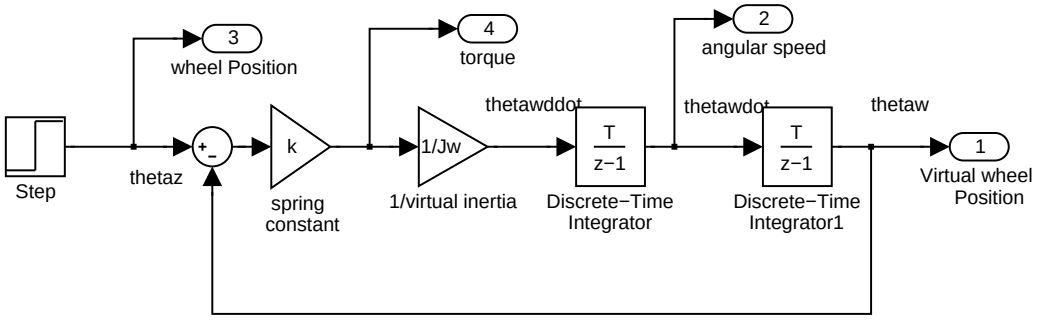

Figure 7: Discrete Simulation of Virtual Wheel and Torsional Spring

Recall from the class notes that stability of a continuous time system is governed by the characteristic roots. For a continuous time system with state space description

$$\dot{x} = Ax + Bu, \tag{2.6}$$

the characteristic roots are equal to the eigenvalues of A. For a discrete time system with state space description

$$x(k+1) = A_d x(k) + B_d u(k), \tag{2.7}$$

the characteristic roots are equal to the eigenvalues of A_d. If the discrete time system is obtained by approximating the continuous time system using the forward Euler method with sample period T, then

$$A_d = I + TA, \quad B_d = TB, \tag{2.8}$$

where I denotes the identity matrix. Facts from linear algebra imply that if λ is an eigenvalue of A, then $\lambda_d = 1 + T\lambda$ is an eigenvalue of A_d.

For a second order system (i.e., a system with two state variables), the characteristic equation may be written as

$$s^2 + \alpha_1 s + \alpha_2 = 0. \tag{2.9}$$

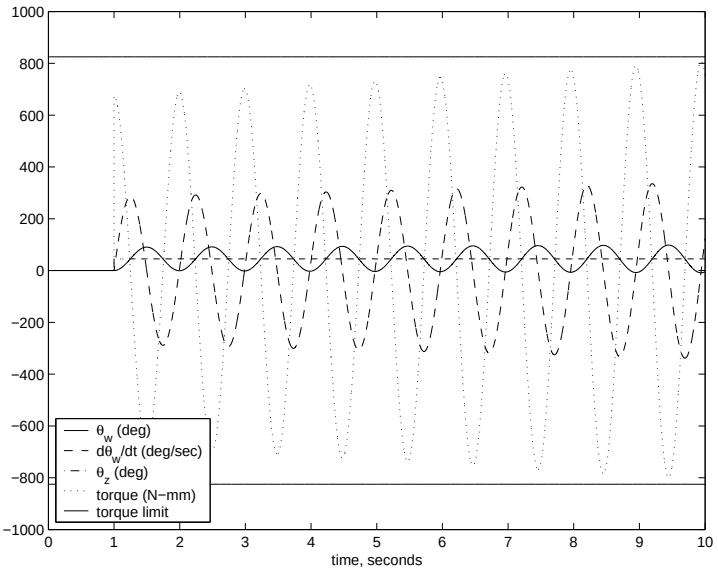

Figure 8: Instability Due to Numerical Integration

If the roots are complex, then they lie on a circle of radius ω_n, and (2.9) may be parameterized as

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0, \tag{2.10}$$

where the damping coefficient ζ must satisfy $|\zeta| < 1$ in order for the eigenvalues to be complex. Expressed in terms of natural frequency and damping, the eigenvalues of A are given by

$$\lambda^{\pm} = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2}. \tag{2.11}$$

Hence the eigenvalues of A_d in this case are

$$\lambda_d^{\pm} = 1 - T\zeta\omega_n \pm jT\omega_n\sqrt{1-\zeta^2}. \tag{2.12}$$

It is easy to show that the discrete eigenvalues from (2.12) have magnitude given by

$$|\lambda_d^{\pm}|^2 = |1 - 2T\zeta\omega_n + T^2\omega_n^2|. \tag{2.13}$$

It follows that if the continuous time system has no damping ($\zeta = 0$), then the discrete time eigenvalues will satisfy $|\lambda_d^{\pm}|^2 = |1 + T^2\omega_n^2|$, lie outside the unit circle, and thus be unstable.

By adding damping to the continuous time system, the discrete time system will be stable for sufficiently small values of T:

$$|\lambda_d^{\pm}| \begin{cases} < 1, & T < 2\zeta/\omega_n, \\ = 1, & T = 2\zeta/\omega_n, \\ > 1, & T > 2\zeta/\omega_n. \end{cases} \tag{2.14}$$

It follows that if we want to implement a discrete time harmonic oscillator, then adding damping $\zeta = T\omega_n/2$ to the continuous time system will exactly balance the destabilizing effects of the forward Euler numerical integration.

In terms of the coefficients of the characteristic polynomial (2.9), it follows that

$$2\zeta/\omega_n = T \;\Leftrightarrow\; \alpha_1 = T\alpha_2. \tag{2.15}$$

2.4 Application to Virtual Wheel/Torsional Spring

The state variable description of the system described in (2.5) and Figures 4-5 is given by

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -k/J_w & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ k/J_w \end{bmatrix} \theta_z \tag{2.16}$$

$$\theta_w = \begin{bmatrix} 1 & 0 \end{bmatrix} x \tag{2.17}$$

In the state variable model (2.16)-(2.17) the states are the outputs of the integrators: $x_1 = \theta_w$ and $x_2 = \dot{\theta}_w$. The reaction torque felt by the user is equal in magnitude and opposite in sign to the torque exerted on the virtual wheel:

$$T_q = -k(\theta_w - \theta_z). \tag{2.18}$$

Suppose we wish to add damping to this model. Then (2.5) becomes

$$\ddot{\theta}_w + \frac{b}{J_w}\dot{\theta}_w + \frac{k}{J_w}\theta_w = \frac{k}{J_w}\theta_z + \frac{b}{J_w}\dot{\theta}_z. \tag{2.19}$$

Treating $\dot{\theta}_z$ as another input, we have the state equations

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -k/J_w & -b/J_w \end{bmatrix} x + \begin{bmatrix} 0 \\ k/J_w \end{bmatrix} \theta_z + \begin{bmatrix} 0 \\ b/J_w \end{bmatrix} \dot{\theta}_z \tag{2.20}$$

$$\theta_w = \begin{bmatrix} 1 & 0 \end{bmatrix} x \tag{2.21}$$

The reaction torque is now

$$T_q = -k(\theta_w - \theta_z) - b(\dot{\theta}_w - \dot{\theta}_z). \tag{2.22}$$

Note that to compute the reaction torque we need to measure the angular velocity of the haptic wheel[1]. If we are only interested in cases for which we hold this wheel constant (or move it slowly) then we can ignore this term[2], and simulate the virtual wheel with damping as shown in Figure 9.

The characteristic equation of (2.19)- (2.20) is given by

$$s^2 + \frac{b}{J_w}s + \frac{k}{J_w} = 0. \tag{2.23}$$

Equating (2.23) with (2.9), and choosing the coefficients according to the rule (2.15), shows that we should add damping

$$\frac{b}{J_w} = T\frac{k}{J_w} \;\Rightarrow\; b = Tk. \tag{2.24}$$

Implementing the virtual world as shown in Figure 10, with b chosen as in (2.24), results in the discrete system being a harmonic oscillator, as desired. The response of the simulation in Figure 10 to a 45° step change in wheel position is plotted in Figure 11. Note that, due to the damping added into the model, the oscillations no longer become unbounded.

[1] To do so, we must numerically differentiate the wheel position to obtain its angular velocity.

[2] Alternately, we can consider a source of damping that depends on the motion of the virtual wheel with respect to a fixed coordinate system, instead of motion relative to the physical wheel.

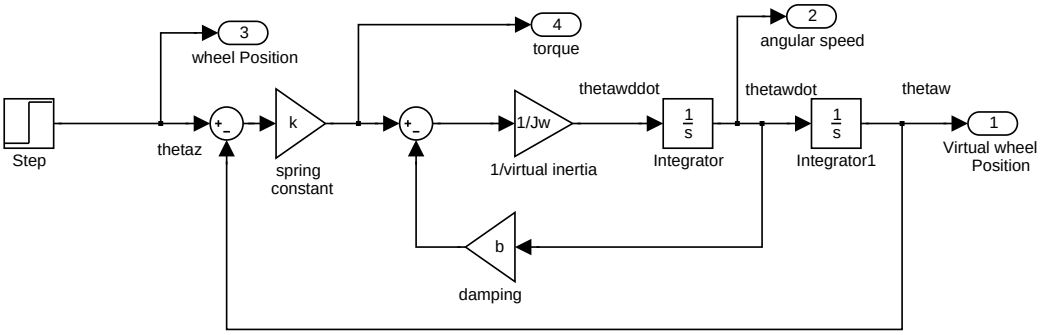

Figure 9: Simulation of Virtual Wheel and Torsional Spring with Damping

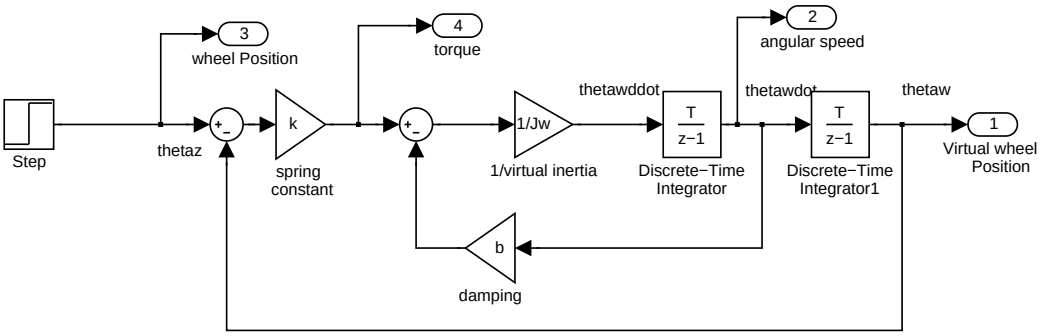

Figure 10: Discrete Simulation of Virtual Wheel and Torsional Spring with Damping

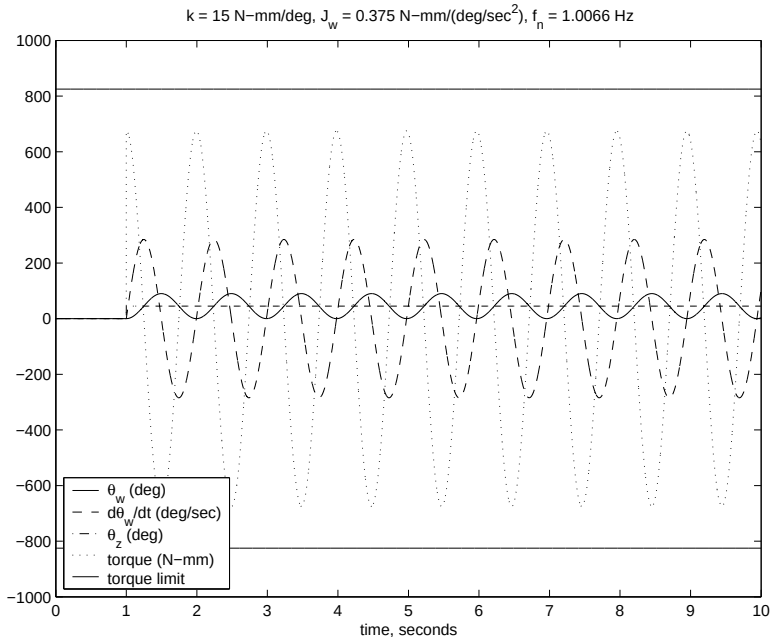

Figure 11: Stable Numerical Integration

3 Free Response of the Puck/Virtual Spring Mass System

Consider again the system in Figure 2. Suppose that we set the virtual mass in motion by moving the puck a fixed amount and then holding it constant. Then the virtual mass will be set in motion, and will oscillate with a period $\sqrt{k/m}$ radians/second.

Suppose next that the user releases the puck. Then the resulting motion will be that of two masses (one real, one virtual) coupled by a spring. This situation is depicted in Figure 12. Note that m is the virtual mass, and M is the mass of the physical device.

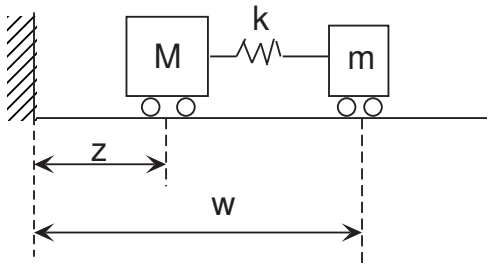

Figure 12: Coupled Virtual/Physical Masses

Define state variables

$$x_1 = z, \quad x_2 = \dot{z}, \quad x_3 = w, \quad x_4 = \dot{w} \tag{3.1}$$

The dynamics of the system with no external forces applied are given by

$$\dot{x}_1 = x_2, \qquad x_1(0) = x_{10} \tag{3.2}$$

$$\dot{x}_2 = \frac{k}{M}(w - z) = -\frac{k}{M}(x_1 - x_3), \quad x_2(0) = x_{20} \tag{3.3}$$

$$\dot{x}_3 = x_4, \qquad x_3(0) = x_{30} \tag{3.4}$$

$$\dot{x}_4 = \frac{k}{m}(z - w) = \frac{k}{m}(x_1 - x_3), \qquad x_4(0) = x_{40}. \tag{3.5}$$

In matrix form, these equations become

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{-k}{M} & 0 & \frac{k}{M} & 0 \\ 0 & 0 & 1 & 0 \\ \frac{k}{m} & 0 & \frac{-k}{m} & 0 \end{bmatrix} x, \quad x(0) = x_0. \tag{3.6}$$

It is difficult to analyze the response of the system in these coordinates. Instead, we define a new coordinate system, with position coordinates given by the center of mass and the distance between the two masses:

$$z_1 = \frac{Mx_1 + mx_3}{M + m}, \quad z_2 = \dot{z}_1, \quad z_3 = x_1 - x_3, \quad z_4 = \dot{z}_3 \tag{3.7}$$

Substituting (3.2)-(3.5) into (3.7), we obtain the state equations

$$\dot{z} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -k\left(\frac{1}{M} + \frac{1}{m}\right) & 0 \end{bmatrix} z, \quad z(0) = z_0. \tag{3.8}$$

It follows from (3.8) that the distance between the masses satisfies the equations of motion of a harmonic oscillator with natural frequency $\sqrt{k\,(1/M + 1/m)}$ radians/second. The center of mass, on the other hand, satisfies the equations of motion of a double integrator with input equal to the distance between the two masses.

When the virtual mass is set into motion and then the puck is released, the virtual mass will almost certainly have some nonzero initial condition, which will translate into a nonzero initial condition for the distance between the masses[3]. The resulting behavior is depicted in Figure 13.

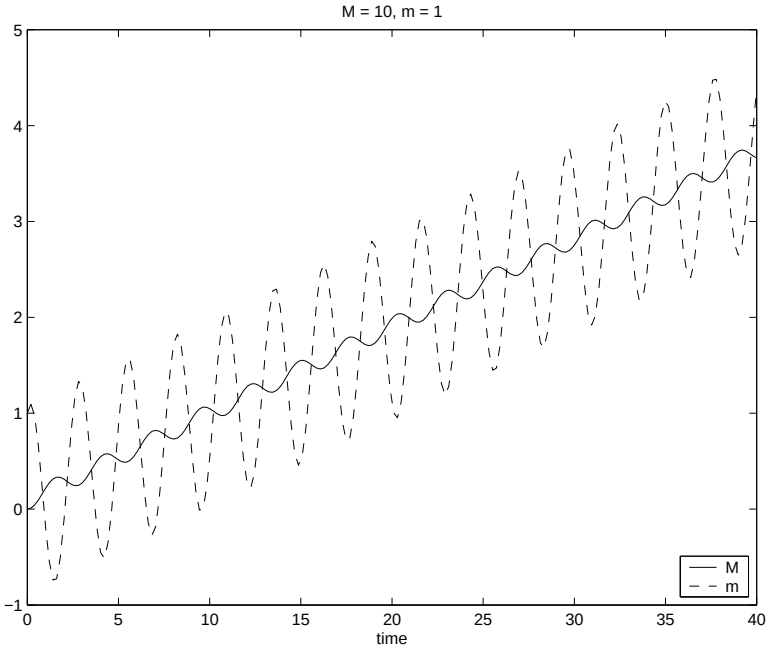

Figure 13: Unforced Motion of Physical and Virtual Masses

3.1 Application to Virtual Wheel/Torsional Spring

To apply the above results to our setup in the lab, we use state variables

$$x_1 = \theta_z, \quad x_2 = \dot{\theta}_z, \quad x_3 = \theta_w, \quad x_4 = \dot{\theta_w}. \tag{3.9}$$

The equations above continue to hold; we simply replace M with J_z and m with J_w. The spring constant k is simply replaced by the torsional spring constant.

The qualitative behavior one sees when setting the virtual wheel in motion by moving the physical wheel, and then releasing the physical wheel, will remain. One should see the wheel on the haptic device spin off in one direction, with a small oscillation as it spins.

To correctly model this behavior, we need to determine J_z, the moment of rotational inertia, for the wheel we use on the haptic device in the lab. Any volunteers?

4 Sinusoidal Excitation

We can also model the response of the virtual wheel when the human drives it by turning the physical wheel back and forth at a certain frequency. If this frequency is the same as the resonant frequency, $\sqrt{k/J_w}$, then this motion can be used to either dampen the motion of the virtual wheel, or to pump energy into it, depending on the phase difference between the two wheels. The simulation in Figure 14 shows what happens when the virtual wheel starts at rest, and is excited by a 1 Hz oscillation of amplitude 5°.

[3]As the virtual mass occupies no space, it is of course possible for the virtual mass to pass through the physical mass; i.e., it is possible that $w < z$.

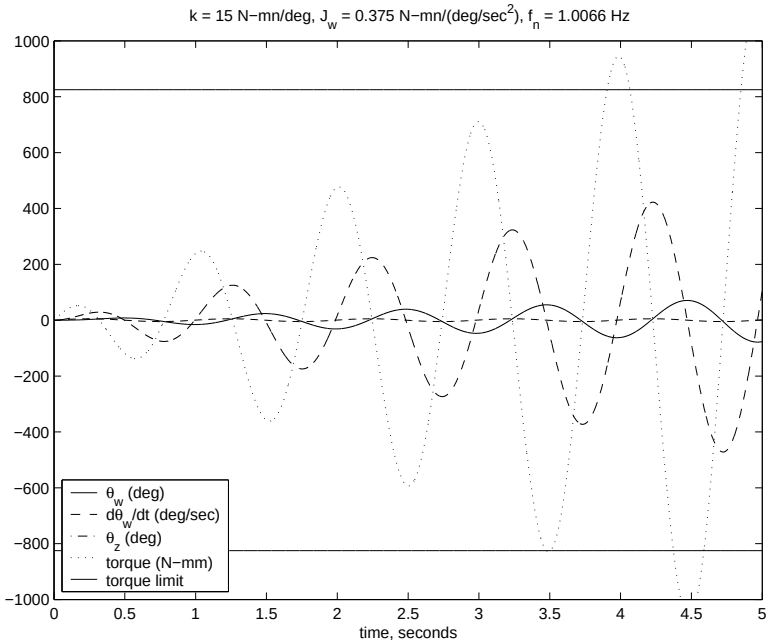

Figure 14: Response to Sinusoidal Input at $\sqrt{k/J_w}$ radians/second