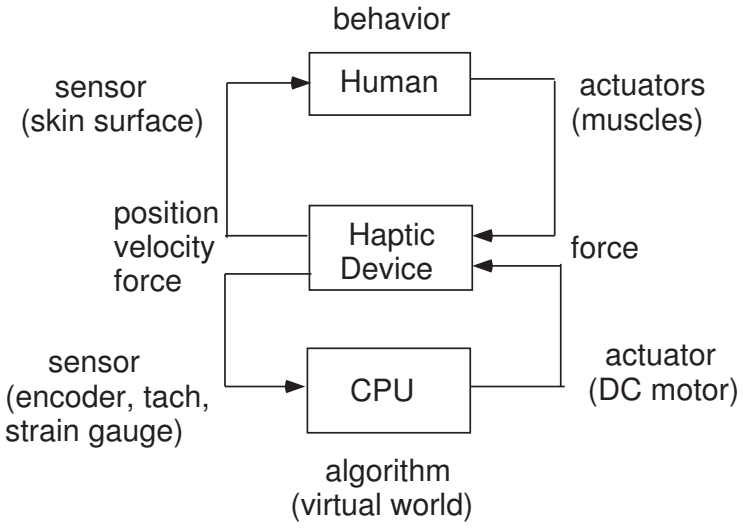
Haptic Interfaces and Virtual Environments

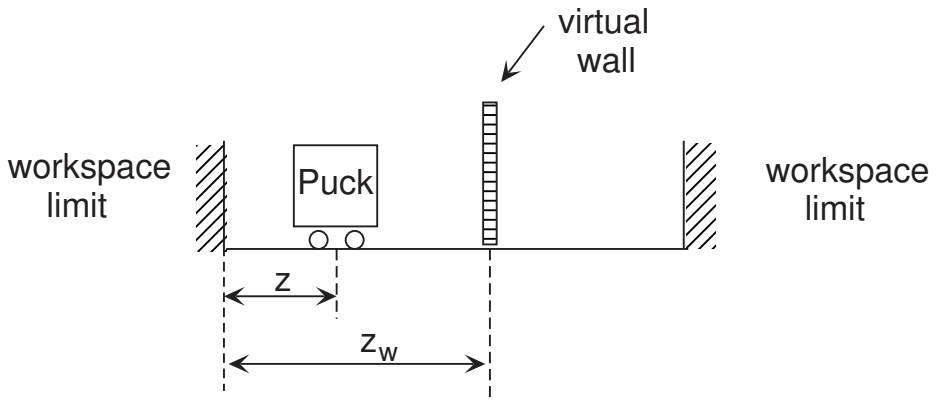
- A haptic interface allows for a human to interact with a computer through the sense of touch.
  - The user moves a "puck" (joystick, wheel) in the workspace.
  - The CPU senses this motion, and computes reaction forces that are transmitted to the human using the motor.
  - A "force feedback" system



- The algorithm used by the CPU to process the position and/or velocity measurements to produce the reaction forces is called a "virtual environment" or "haptic world"
  - virtual wall [1]
  - virtual sprung mass [1]
  - surgery simulator (virtual body tissue) [2]

Virtual Wall

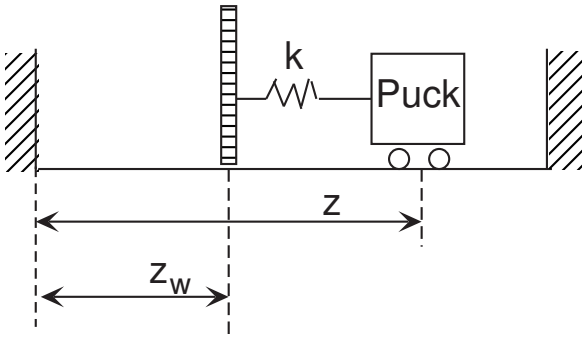
- Consider a puck in a linear workspace:



- Our goal is to create a *virtual wall* to prevent us from moving the puck all the way to the right of the workspace
- Idea
  - If the puck is to the left of the "virtual wall", do nothing, so the puck moves freely.
  - If the puck is to the right of the "virtual wall", have the motor exert a force to push it back to the left
- To implement a virtual wall, we require
  - a position measurement from an encoder
  - an algorithm to compute the force, implemented on a microprocessor
  - a motor to exert the force

Virtual Spring Algorithm

- To make a virtual wall, we suppose that when the puck is to the right of the wall, it is pulled back to the wall with a spring:



- Hence we need the motor to exert a force equal to the restoring force of a spring

$$F = -k(z - z_w)$$

- Pseudocode for virtual wall:

```
set  z_w
set spring constant k
while(1) {
    read puck position z from potentiometer (encoder)
    if   z > z_w
       compute force F = k*(z - z_w)
       output force to DAC (set PWM duty cycle)
    else
      do nothing
    end
}
```

Issues with Virtual Wall

- Rotary vs. linear workspace is analogous:
  - angular rather than linear displacement
  - torque rather than force

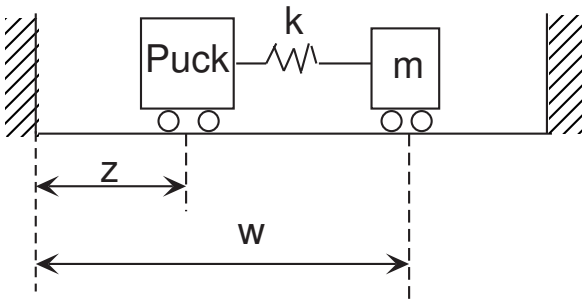
- Chatter: an "artifact" that distinguishes a computer generated wall from an actual wall. Occurs due to combination of
  - large restoring force
  - slow sample time
  - large quantization error

- If a tachometer is used to sense velocity, then the velocity must be numerically integrated to obtain position displacement

- For a more detailed schematic of linear and rotary haptic interfaces, see [1]

Virtual Sprung Mass

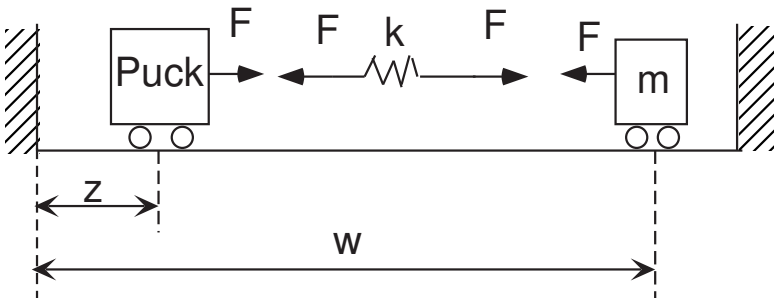
- Suppose we wish the puck to behave as though it had a mass attached to it with a spring:



- If we move the puck to a new position, z, then
  - the spring will exert a force $F = -k(w - z)$ on the mass
  - the force will accelerate the mass according to Newton's laws:
$$-k(w - z) = m\ddot{w}$$
  - the position of the mass will change acccording to

$$\ddot{w} + \frac{k}{m}w = \frac{k}{m}z \tag{1}$$

  - the puck experiences a *reaction force* equal in magnitude and opposite in direction to the force exerted on the mass:

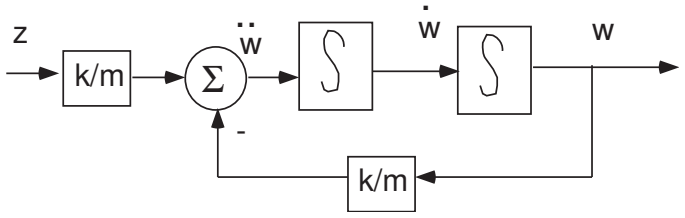


Simulink Model of Spring Mass System

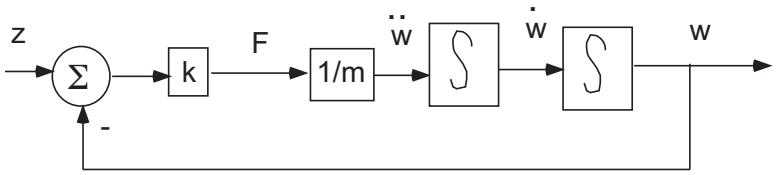
- The differential equation

$$\ddot{w} = \frac{k}{m}z - \frac{k}{m}w$$

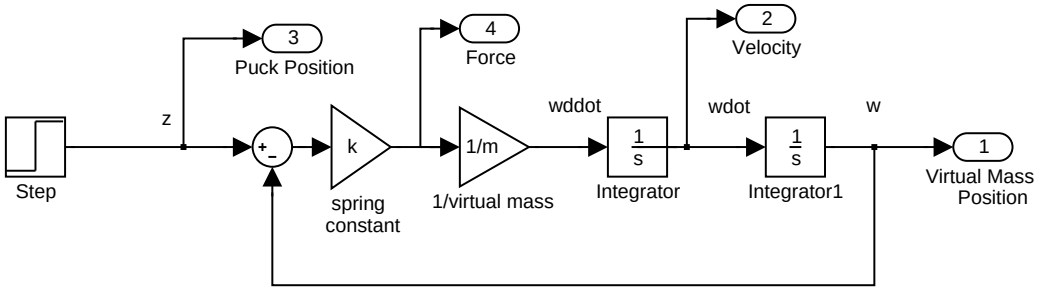
  has a block diagram (cf. the notes on second order systems from Lecture 7):



- It is easy to show that this diagram is equivalent to one from which the force acting on the mass, $F = -k(w - z)$, can be directly computed:

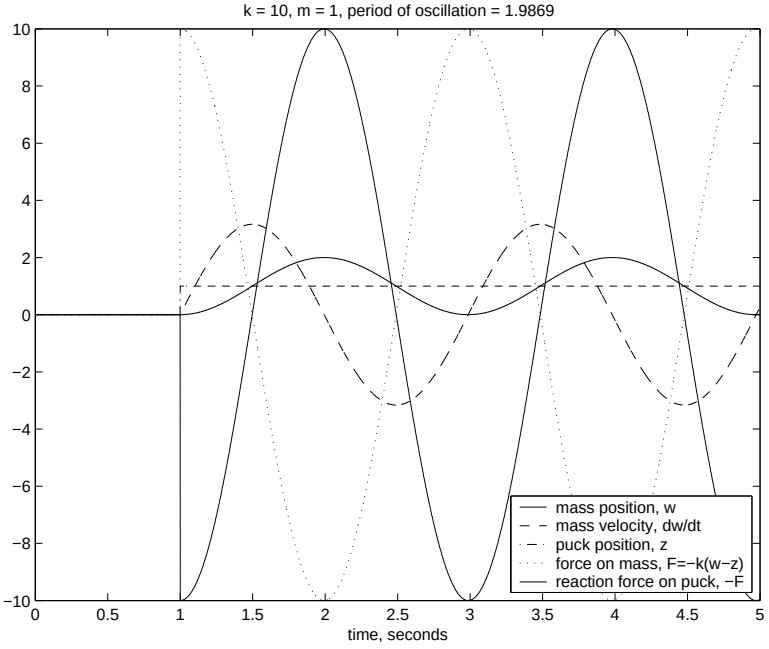


- an appropriate SIMULINK diagram:



Simulated Response

- Let us move the puck one unit to the right and hold it there.
  - The mass will respond according to (1)
  - A reaction force will be exerted on the puck
  - This force must be countered by the user to hold the puck stationary!
- Matlab simulation[1] of mass motion, velocity, and force exerted on the puck (assumes that the puck and mass have no physical dimension)



- Note: the spring-mass system is an undamped harmonic oscillator with
  - frequency in radians/second, $\omega_n = \sqrt{k/m}$
  - frequency in Hz, $f_n = \omega_n/2\pi$
  - period in seconds, $T_n = 1/f_n$

[1] Obtained from Matlab files `virtual_spring_mass.m` and `oscillator_force.mdl`.

Algorithm for Sprung Mass

- If puck is not held constant, then the force will depend on the puck position.
- To simulate the motion of the puck requires that we model
  - the response of the puck states (position, velocity) to the reaction force.
  - the effect of the human interacting with the puck
- To implement the haptic interface,
  - we *measure* the position of the puck.
  - we do not need to simulate puck motion or human interaction.

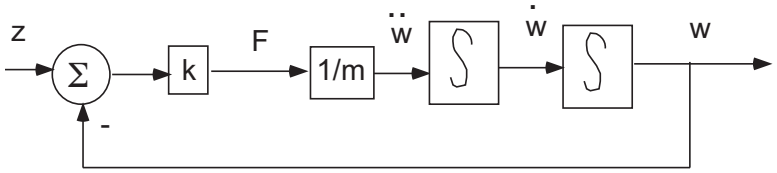
- Pseudocode for sprung mass:

```
set  k, m
while(1) {
    read puck position z from potentiometer (encoder)
    update w according to eqn.(1)
    compute reaction force F = k*(w-z)
    output force to DAC (set PWM duty cycle)
}
```

- Question: How to update w?
  - Must numerically integrate a second order differential equation

State Space Form of Differential Equations

- The differential equation governing the motion of the mass is a *second order* equation:



- It is convenient to represent this as *two first order equations*. Define *state variables* position and velocity:

$$x_1 \triangleq w$$

$$x_2 \triangleq \dot{w}$$

- Define the *input* as the puck position, $u = z$, and the *output* as the reaction force, $y = -F = k(w - z)$. Then the *state differential equations* are

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -\frac{k}{m}x_1 + \frac{k}{m}u$$

$$y = kx_1 - ku$$

Matrix Form of State Equations

- Define the *state vector*

$$x \triangleq \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} w \\ \dot{w} \end{bmatrix}$$

- Then the state equations can be written in matrix form as

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

  where

$$A \triangleq \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & 0 \end{bmatrix}, B \triangleq \begin{bmatrix} 0 \\ \frac{k}{m} \end{bmatrix}, C \triangleq \begin{bmatrix} k & 0 \end{bmatrix}, D \triangleq -k$$

- It is true in general that *any* n-th order differential equation may be written as a system of n first order differential equations, one for each of n state variables
- Example:
  - Suppose that, instead of holding the puck position fixed after an initial step change, we allow the puck to move freely in response to the reaction force exerted by the mass.
  - Need four state variables (position and velocity of both puck and mass)

Numerical Integration of State Equations

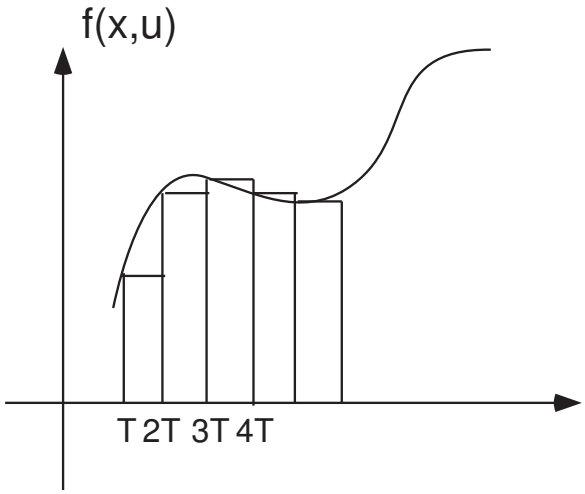
- Consider a differential equation $\dot{x} = f(x, u)$
- Question: How to find the value of

$$x(t) = x(0) + \int_0^t f(x, u)dt$$

  given some initial condition $x(0)$, and the input $u(t)$?
- Idea: Select a time interval T, and generate an approximating sequence $\tilde{x}(0), \tilde{x}(T), \tilde{x}(2T), \ldots$ using a difference equation

$$\tilde{x}((n+1)T) = \tilde{x}(nT) + Tf(\tilde{x}(nT), u(nT)),$$

  with initial condition $\tilde{x}(0) = x(0)$.
- The *forward Euler integration* algorithm



- The difference $\tilde{x}((n+1)T) - \tilde{x}(nT)$ is the area of one rectangle.
- We usually abbreviate $\tilde{x}(k) \triangleq \tilde{x}(kT)$ and suppress the $\tilde{}$ notation.

Pseudocode for Virtual Spring-Mass

- In our case, $f(x,u) = Ax + Bu$, where

$$A = \begin{bmatrix} 0 & 1 \\ -k/m & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ k/m \end{bmatrix}$$

- Hence

$$\begin{aligned} x(n+1) &= x(n) + T(Ax(n) + Bu(n)) \\ &= (I + TA)x(n) + TBu(n) \end{aligned}$$

- $x_1 =$ position, $x_2 =$ velocity

$$\begin{aligned} x_1(n+1) &= x_1(n) + Tx_2(n) \\ x_2(n+1) &= x_2(n) + T(k/m)(-x_1(n) + u(n)) \end{aligned}$$

- Pseudocode for sprung mass:

```
initialize parameters   k,m;
initialize states    x1prev = z, x2prev = 0;
set update period   T;
while(1) {
    read in puck position u = z;
    x1 = x1prev + T*x2prev;
    x2 = x2prev + T*(-(k/m)*x1prev + (k/m)*u);
    compute reaction force y = F = k*(x1-u)
    x1prev = x1;  x2prev = x2;
    output force value
}
```

Difference Equations and the z-Transform

- Analogous to Laplace transform for continuous time systems and differential equations
- Given a discrete sequence

$$\{y(k)\} = \ldots, y(-2), y(-1), y(0), y(1), y(2), \ldots,$$

the z-transform is

$$Y(z) = \sum_{k=-\infty}^{\infty} y(k) z^{-k}$$

- The z-transform of the *time-shifted* sequence $\{\hat{y}(k)\} = \{y(k+n)\}$ is

$$\hat{Y}(z) = z^n Y(z)$$

- The transfer function for the system with difference equation

$$y(k+2) + a_1 y(k+1) + a_2 y(k) = b_1 u(k+1) + b_2 u(k)$$

is found from

$$Y(z) = G(z)U(z), \qquad G(z) = \frac{(b_1 z + b_2)}{(z^2 + a_1 z + a_2)}$$

Discrete Time Simulation

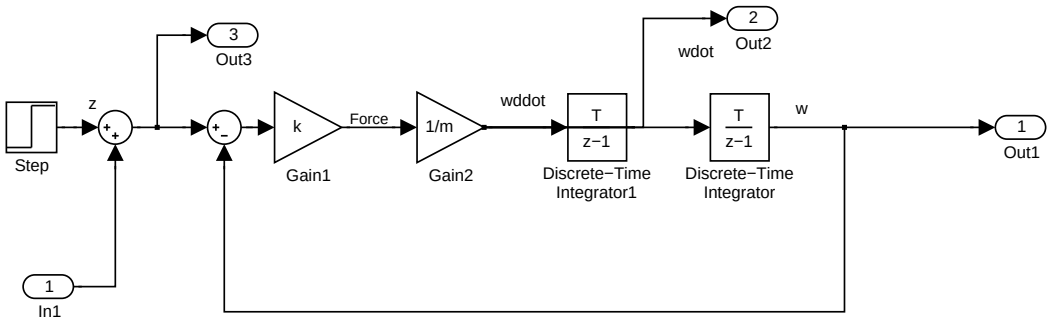
- Difference equations

$$x_1(n+1) = x_1(n) + Tx_2(n)$$
$$x_2(n+1) = x_2(n) + T(k/m)(-x_1(n) + u(n))$$

- Transfer functions

$$X_1(z) = \frac{T}{z-1}X_2(z)$$
$$X_2(z) = \frac{T}{z-1}(k/m)(-X_1(z) + U(z))$$

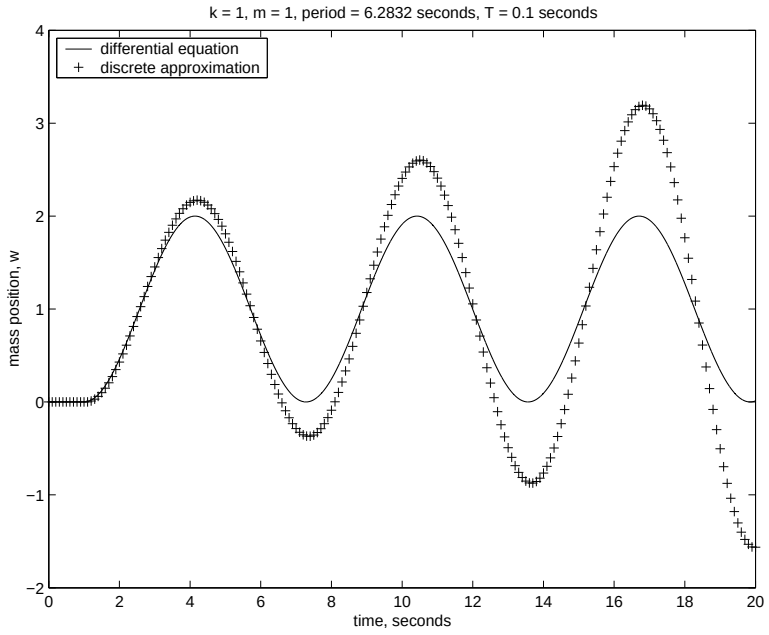
- For us, $x_1 = w$, $x_2 = \dot{w}$, $u = z$
- SIMULINK model[2]



[2] `discrete_oscillator_force.mdl`

Issues with Sprung Mass

- How well does the solution to the difference equation match the solution to the differential equation?
- Intuitively, if T is "small enough", approximation should be good
- Matlab simulation of the response of the system with $k = 1$, $m = 1$ to a step change in puck position[3]:
- If T is "too large"(in this case $T = 0.1$), response becomes unbounded:



- What goes wrong?

[3]Obtained with Matlab files `num_integration.m`, `oscillator.mdl`, and `discrete_oscillator.mdl`.

Numerical Stability

- System of differential equations:

$$\dot{x} = Ax + Bu \tag{2}$$
$$y = Cx + Du$$

- Many different ways to obtain a discrete approximation
- Simplest is the (forward) Euler scheme:

$$\dot{x}(nT) \approx \frac{x((n+1)T) - x(nT)}{T} \approx Ax(nT) + Bu(nT)$$

- Difference equations:

$$x((n+1)T) = A_d x(nT) + B_d u(nT) \tag{3}$$
$$y(nT) = C_d x(nT) + D_d u(nT)$$

  where

$$A_d = I + TA, \quad B_d = TB$$
$$C_d = C, \quad D_d = D$$

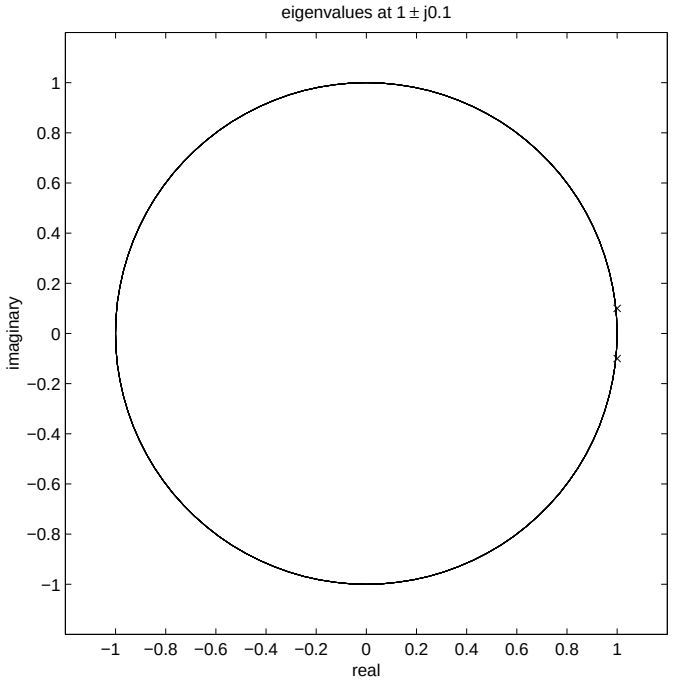
- Both the continuous system and the discrete appoximation are stable if
  - any bounded input results in a bounded output
  - the response to initial conditions decays to zero
- Question: Does stability of (2) imply stability of (3)?

Numerical Stability, II

- The difference equation

$$x(n+1) = (I + TA)x(n) + TBu(n)$$

  is stable if the eigenvalues of the matrix $I + TA$ have magnitude less than one.
- The discrete transfer function $G(z) = N(z)/D(z)$ is stable if all the roots of $D(z) = 0$ lie within the unit circle
- In our case, eigenvalues[4] are at $\lambda = 1 \pm jT\sqrt{k/m}$



- If $T\sqrt{k/m}$ is too large, the algorithm will diverge rapidly
- Other (more complicated) techniques for numerical integration do preserve stability – but does it matter?

[4] `num_integration.m`

Discrete Approximation to an Integrator

- Consider the continuous time simulation of the harmonic oscillator in Figure 1:

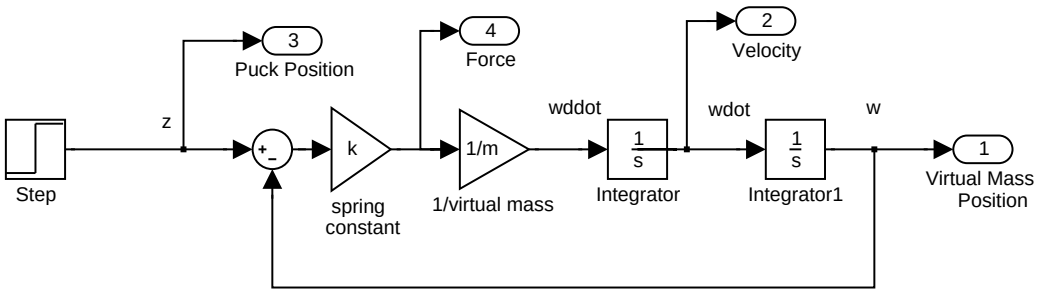


Figure 1: Analog Harmonic Oscillator

- The forward Euler approximation of the harmonic oscillator in Figure 2 is obtained by replacing each analog integrator by a discrete approximation, $1/s \longrightarrow T/(z-1)$:

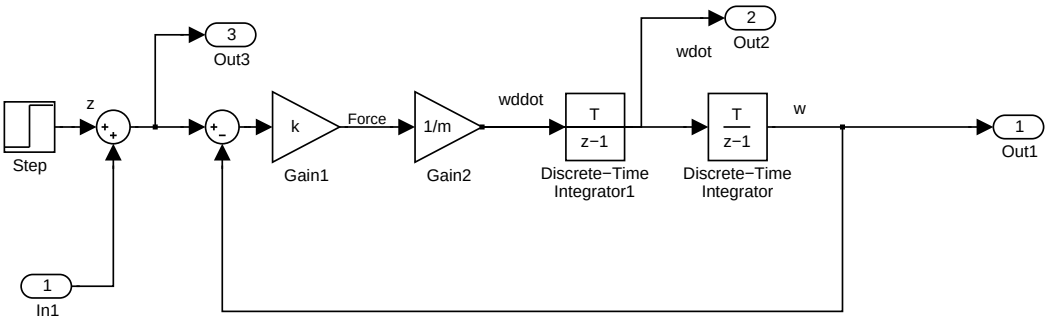


Figure 2: Discrete Harmonic Oscillator, Forward Euler

- Other discrete approximations to the harmonic oscillator may be obtained by replacing the analog integrators by other discrete approximations.

Discrete Approximation to an Integrator, II

Suppose we wish to approximate an integrator,

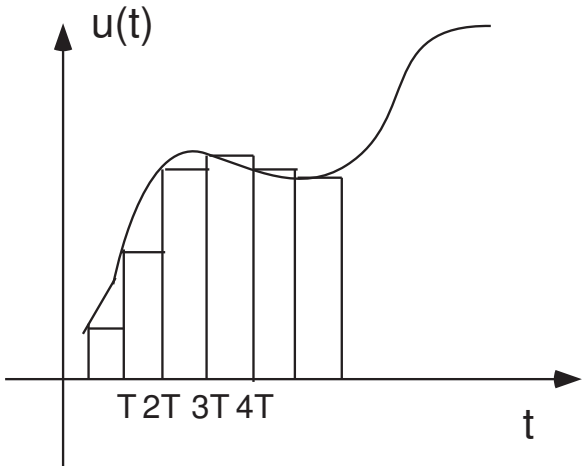
$$y(t) = \int_0^t u(t)dt$$

by a discrete sequence $y(0), y(T), y(2T), \ldots$

- Forward Euler:

$$y((k+1)T) = y(kT) + Tu(kT)$$

$$\frac{1}{s} \rightarrow \frac{T}{(z-1)}$$

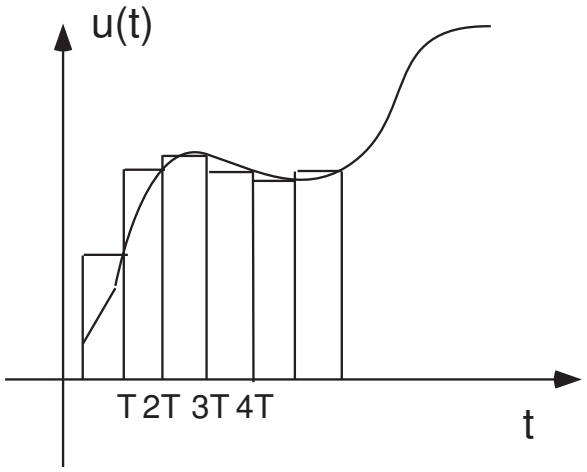


Discrete Approximation to an Integrator, II

- Backward Euler:

$$y((k+1)T) = y(kT) + Tu((k+1)T)$$

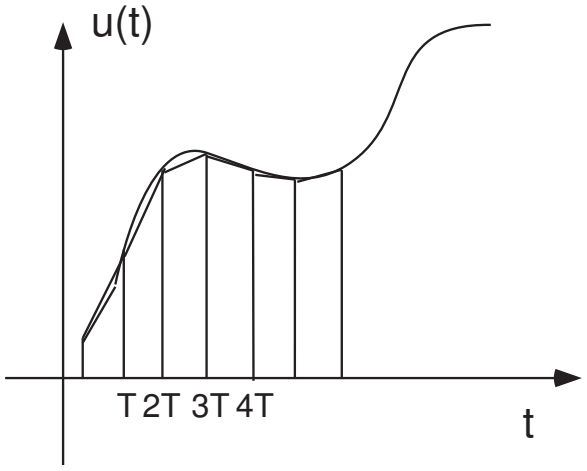
$$\frac{1}{s} \rightarrow \frac{Tz}{(z-1)}$$



- Trapezoidal:

$$y((k+1)T) = y(kT) + \frac{T}{2}\left(u(kT) + u((k+1)T)\right)$$

$$\frac{1}{s} \rightarrow \frac{T(z+1)}{2(z-1)}$$

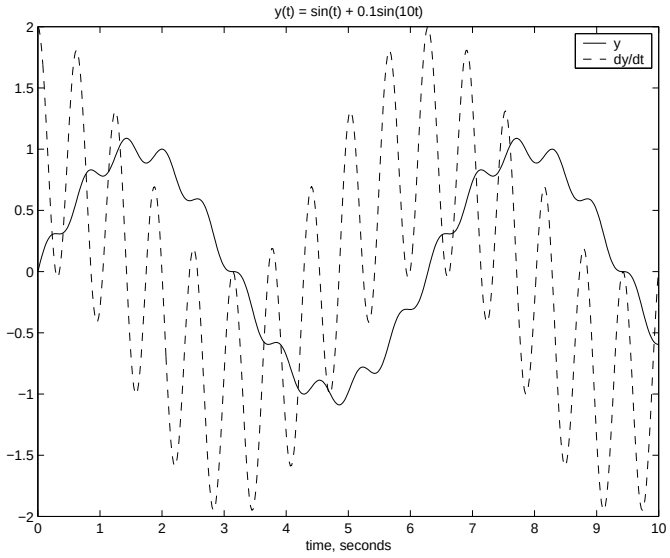


Signal Differentiation

- Used to obtain a velocity estimate from position measurements
- Problem: Differentiation amplifies noise. Consider

$$y(t) = \underbrace{\sin(t)}_{\text{signal}} + \underbrace{0.1\sin(10t)}_{\text{noise}}$$

- The high frequency noise is amplified[5]:



- A problem even with analog differentiation! The transfer function of a differentiator is s, and thus the frequency response is unbounded at high frequencies!
- A pure differentiator is never implemented. Instead, an approximate differentiator, $s/(\tau s + 1)$, is used to lowpass filter any noise

[5] Matlab m-file `diff_noise.m`

Numerical Differentiation

- A digital approximation to velocity can be obtained by approximating the derivative as

$$\dot{x}(nT) \approx \frac{x(nT) - x((n-1)T)}{T}$$

- Hence, we can define a sequence of approximations ot the derivative:

$$v(nT) = \frac{x(nT) - x((n-1)T)}{T}$$

- This procedure will still tend to amplify noise, which may still need to be removed with a filter:
  - analog filter (e.g., op amp circuit)
  - digital FIR filter (e.g., moving average filter)
  - digital IIR filter (e.g., lowpass filter)

Numerical Integration

- How to obtain a position estimate from velocity measurements?
- Assume that the velocity over a time interval is the average of the velocity at the endpoints of the interval:

$$x(nT) = x((n-1)T) + \frac{T(v(nT) + v((n-1)T))}{2}$$

- Issues:
  - small step size needed for rapidly varying signals, but can cause quantization error to accumulate for slowly varying signals
  - long step size better for slowly varying signals, but can "miss" high frequency variations.

References

[1] B. Gillespie. Tutorial on virtual environments for haptic display. University of Michigan, April 2000.

[2] D. Sorid and S. K. Moore. The virtual surgeon. *IEEE Spectrum*, pages 26–31, July 2000.