MECH 6v29.002 – Model Predictive Control

L16 – Active Suspension Application

Outline

- Summary of Mid-Semester Feedback
- Active Suspension Application
- Goal
- System Description
- Modeling
- System Identification
- Discretization
- Control Requirements
- Controller Formulation
- Tuning

Summary of Mid-Semester Feedback

10 out of 12 responses

Teaching effectiveness:
8 Very High
2 High

Course quality:
7 Very High
3 High

Hope stays the same:

- Communication/explanations
- Lectures
- Examples
- Homework
- Instructor engagement/responsiveness

Hope is changed:

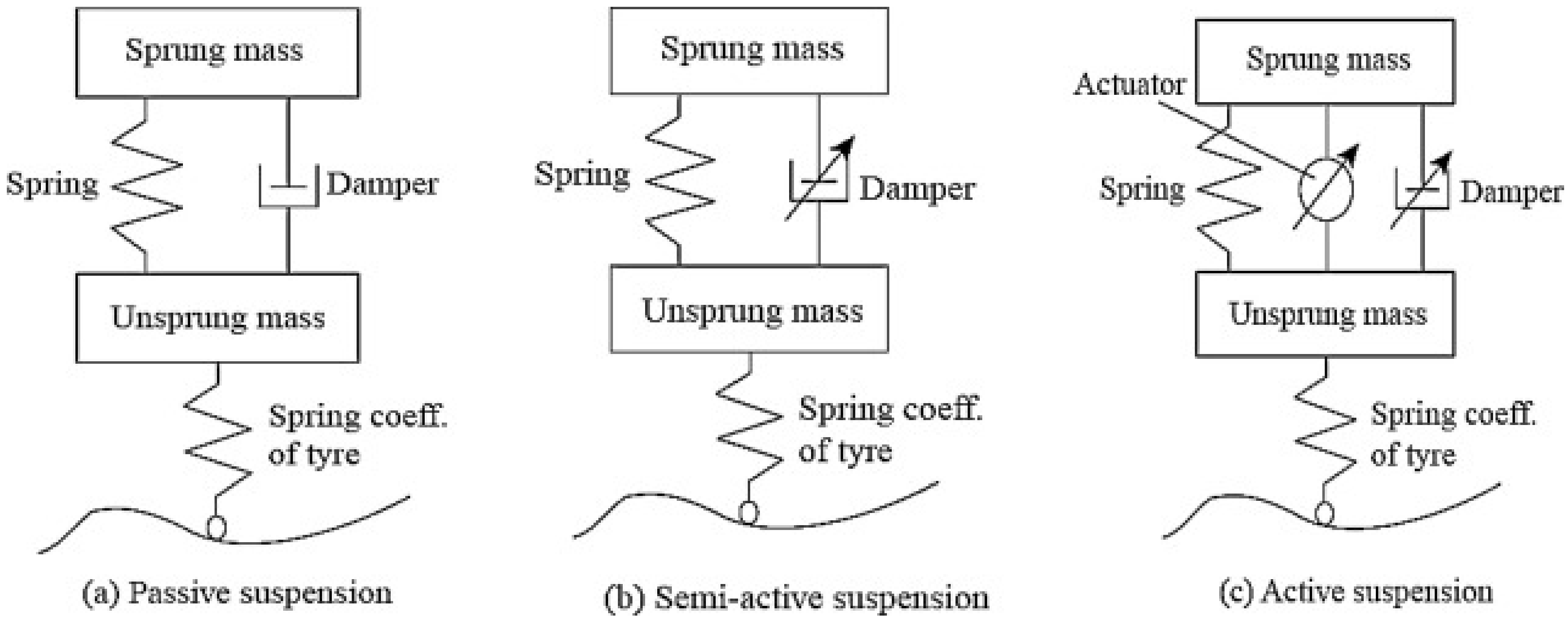
- Timing/pace of lecture (getting better)
- Practical/real examples
- More focus on theory
- Fixing errors on lectures slides posted to eLearning
- Time of lecture
- More MATLAB in slides
- Post slide animations/code

Goal

- Over the last ~5 weeks, we have focused on the mathematical formulation and analysis that allowed us to analyze the feasibility of MPC under nominal and robust cases
- In HW applications, you have been given many of the MPC design parameters
- In preparation for your projects (some of which will be application-oriented), this week we will explore the development process of an MPC controller for a particular application
- In HW #4, you will go through this process with less guidance on the specifics of the MPC design and parameter choice
- We will focus on the control of an active suspension system for a car

System Description

- Evolution of car suspension



- What happens with Bose makes a car suspension

https://youtu.be/3KPYIaks1UY?t=65

System Description (cont.)

- We will focus on the quarter-car model (only look at one wheel)

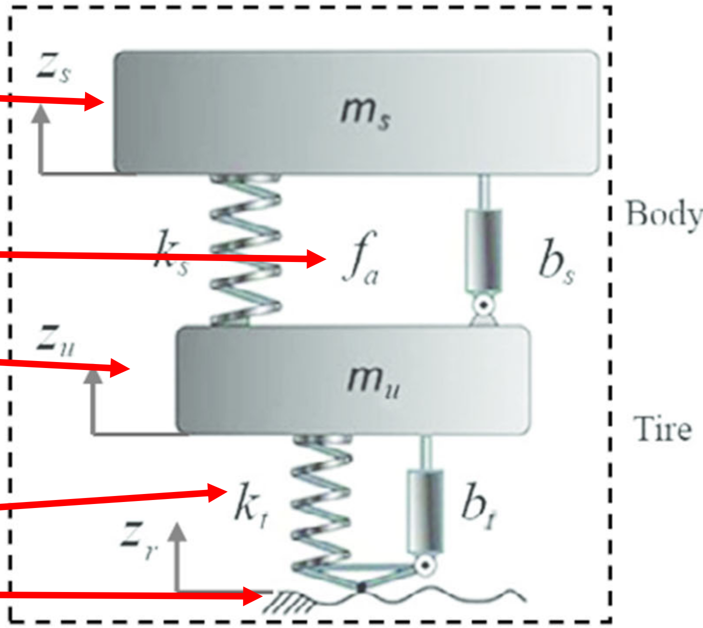
Sprung mass
(body of vehicle)

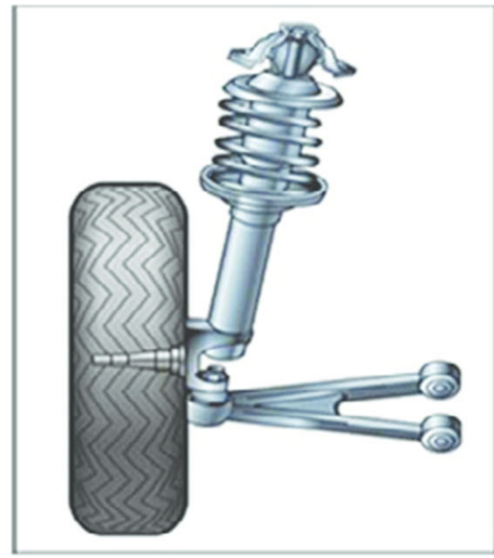
Active suspension

Unsprung mass
(wheel and tire)

Tire

Road

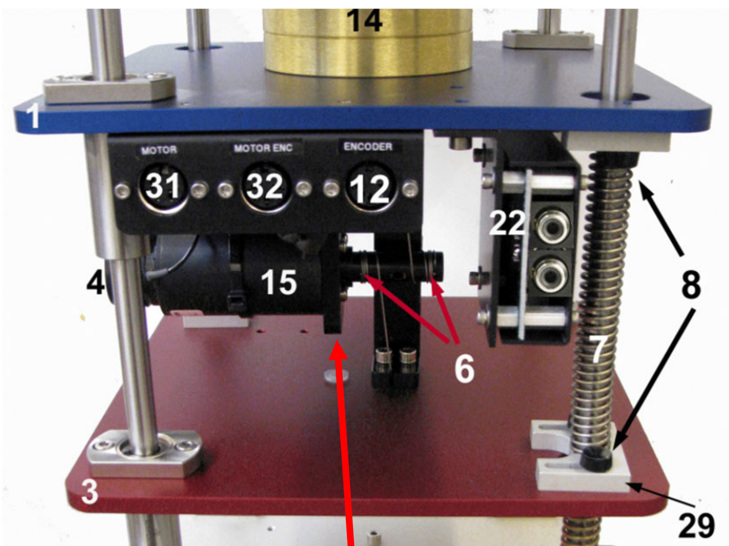




Alvarez Sanchez, Ervin. (2013). A Quarter-Car Suspension System: Car Body Mass Estimator and Sliding Mode Control. Procedia Technology. 7. 208-214. 10.1016/j.protcy.2013.04.026.

System Description (cont.)

- At UTD, we have an active suspension experimental system built by Quanser
- We will develop an MPC controller for this system



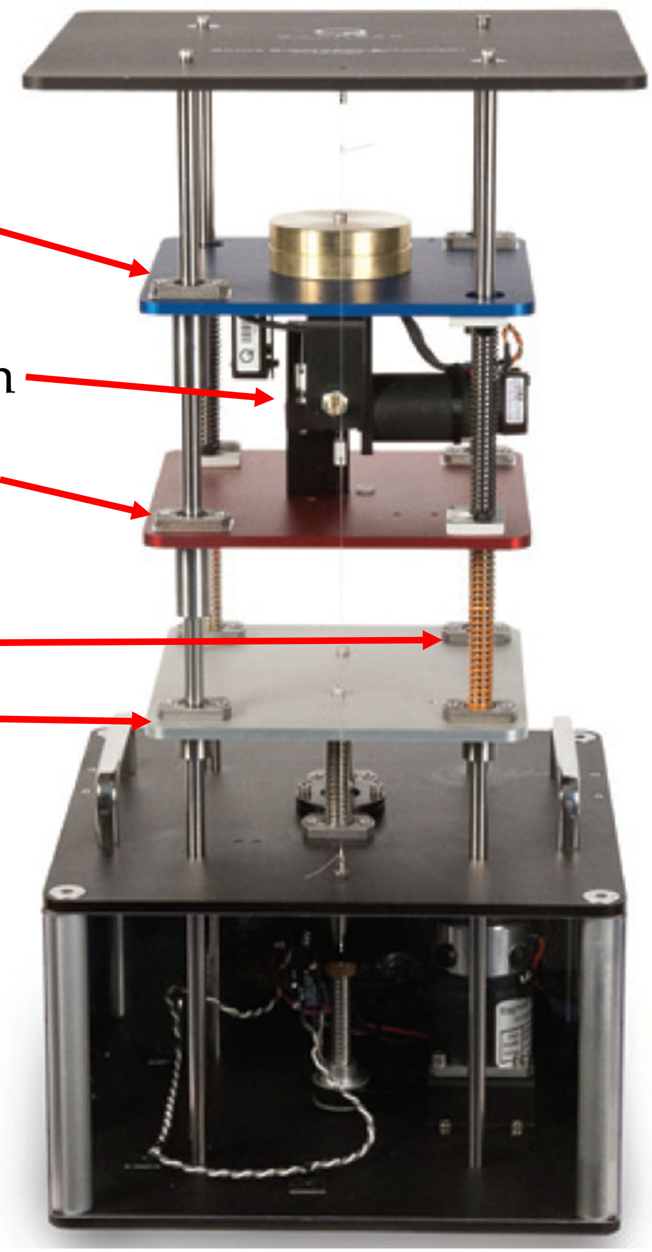
Sprung mass
(body of vehicle)

Active suspension

Unsprung mass
(wheel and tire)

Tire

Road

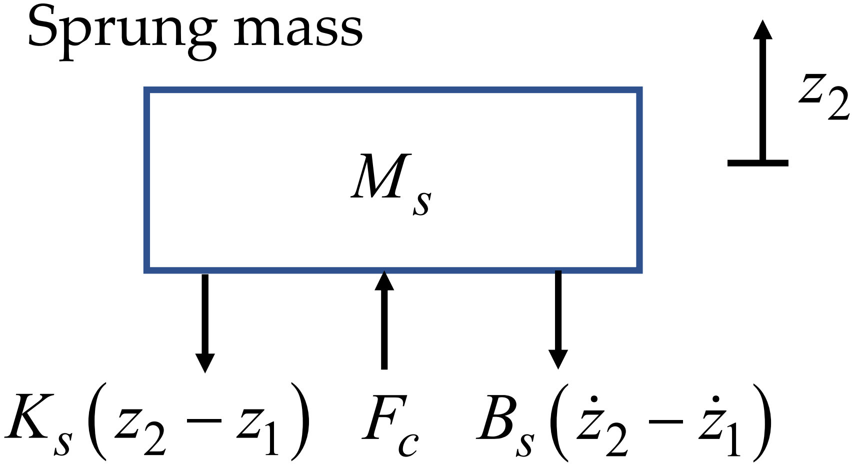


- DC motor creates force between sprung and unsprung masses

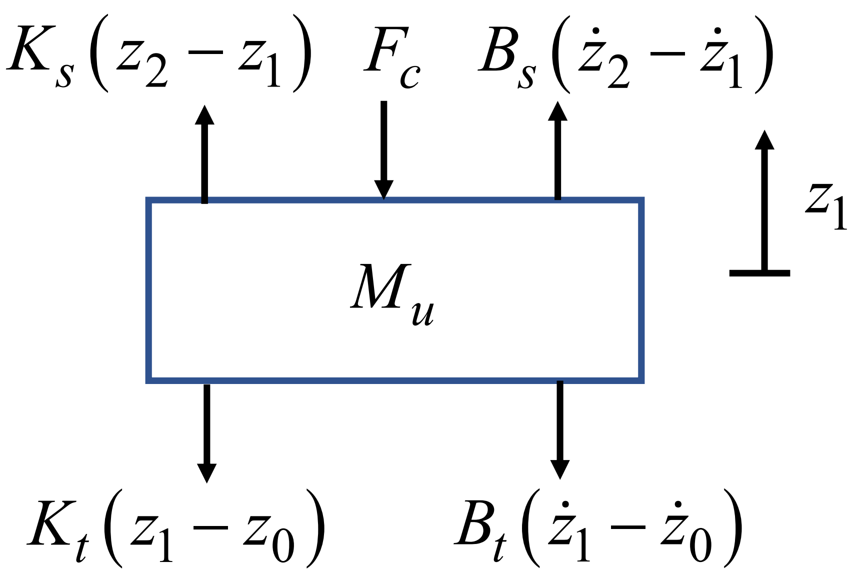
https://www.youtube.com/watch?v=NELQQgRyOjE&list=PLYw9s2m09EImDpjVxn-Qef12zklSRI6kC&ab_channel=gutierrezsj

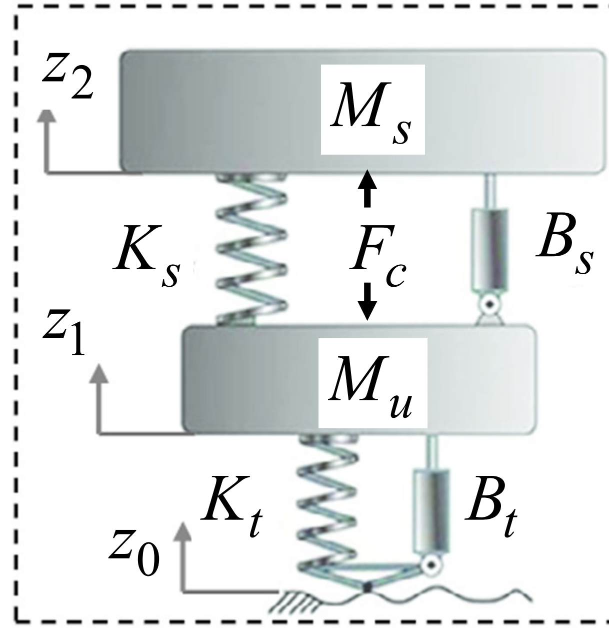
Modeling

- Free-body diagrams
- Assume $z_2 > z_1 > z_0 > 0 \quad \dot{z}_2 > \dot{z}_1 > \dot{z}_0 > 0$
- Sprung mass



- Unsprung mass



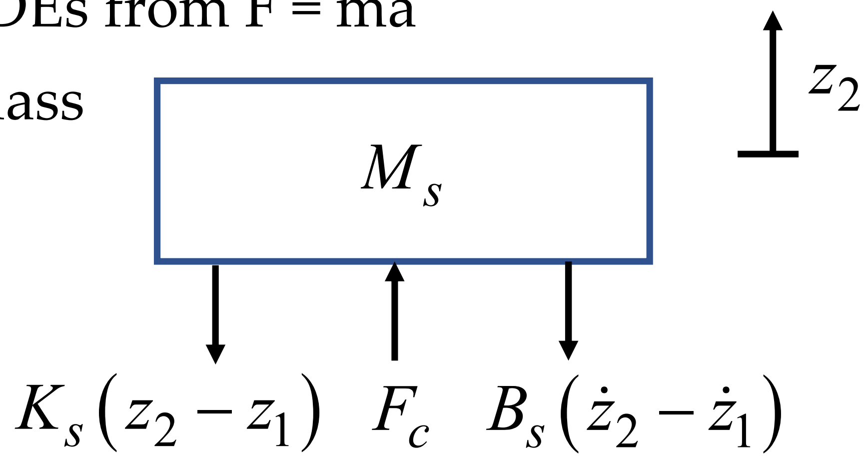


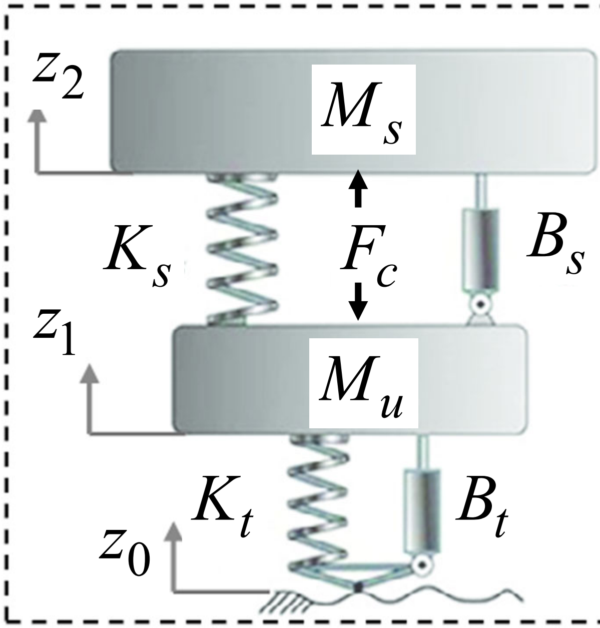
What about gravity?

- Assume z_1 and z_2 are deviations from equilibrium
- Gravity affects this equilibrium but not the dynamics of the deviations

Modeling (cont.)

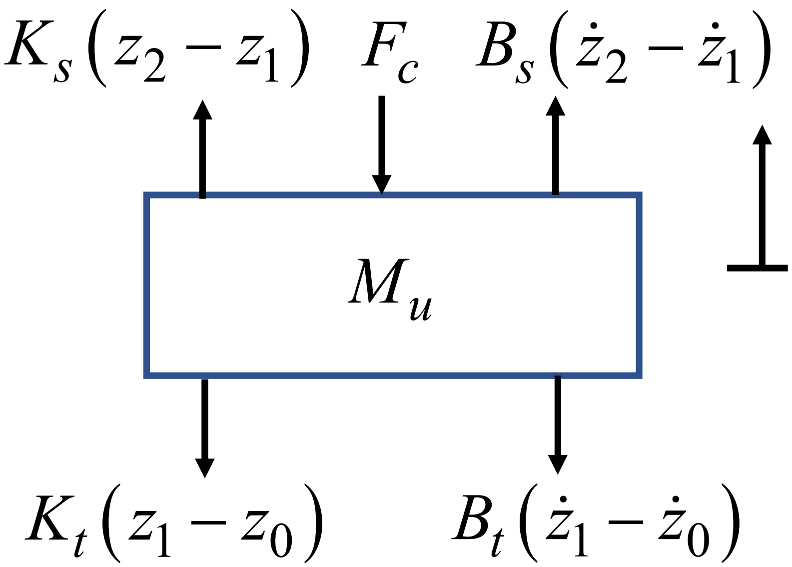
- Derive ODEs from F = ma
- Sprung mass





$$M_s \ddot{z}_2 = F_c - K_s(z_2 - z_1) - B_s(\dot{z}_2 - \dot{z}_1)$$

- Unsprung mass

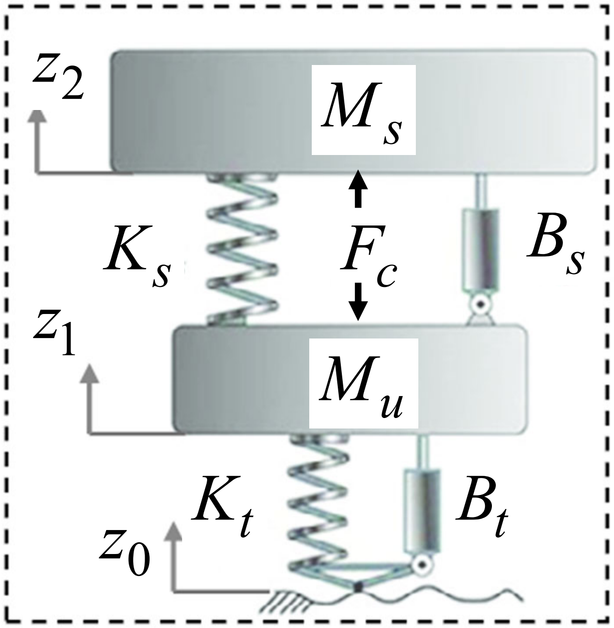


$$M_u \ddot{z}_1 = -F_c + K_s(z_2 - z_1) + B_s(\dot{z}_2 - \dot{z}_1) - K_t(z_1 - z_0) - B_t(\dot{z}_1 - \dot{z}_0)$$

Modeling (cont.)

$$M_u \ddot{z}_1 = -F_c + K_s\left(z_2 - z_1\right) + B_s\left(\dot{z}_2 - \dot{z}_1\right) - K_t\left(z_1 - z_0\right) - B_t\left(\dot{z}_1 - \dot{z}_0\right)$$

$$M_s \ddot{z}_2 = F_c - K_s\left(z_2 - z_1\right) - B_s\left(\dot{z}_2 - \dot{z}_1\right)$$



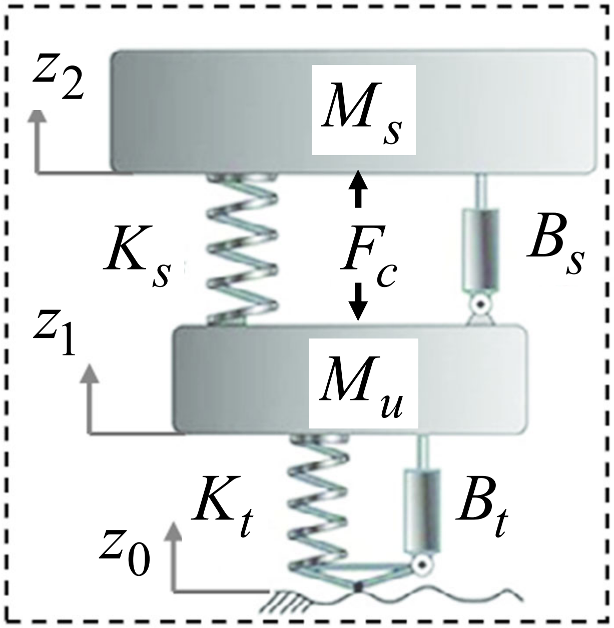
- Derive state-space model
- States
  $x_1 = z_1 - z_0 \quad (\text{tire deflection})$
  $x_2 = \dot{z}_1 \quad (\text{unsprung mass velocity})$
  $x_3 = z_2 - z_1 \quad (\text{suspension deflection})$
  $x_4 = \dot{z}_2 \quad (\text{sprung mass velocity})$
- Inputs
  $u_1 = F_c \quad (\text{active suspension force})$
- Disturbances
  $d_1 = \dot{z}_0 \quad (\text{rate of change of road height})$
  $d_2 = z_0 \quad (\text{road height})$
- Outputs
  $y_1 = z_1 \quad (\text{unsprung mass height})$
  $y_2 = z_2 \quad (\text{sprung mass height})$
  $y_3 = \ddot{z}_2 \quad (\text{sprung mass acceleration})$

Modeling (cont.)

$$M_u\ddot{z}_1 = -F_c + K_s(z_2 - z_1) + B_s(\dot{z}_2 - \dot{z}_1) - K_t(z_1 - z_0) - B_t(\dot{z}_1 - \dot{z}_0)$$

$$M_s\ddot{z}_2 = F_c - K_s(z_2 - z_1) - B_s(\dot{z}_2 - \dot{z}_1)$$

- Derive state-space model
- Need 4 equations
  (one defining the derivative of each state)



$$\dot{x}_1 = x_2 - d_1$$

$$M_u\dot{x}_2 = -u_1 + K_s x_3 + B_s(x_4 - x_2) - K_t x_1 - B_t(x_2 - d_1)$$

$$\dot{x}_3 = x_4 - x_2$$

$$M_s\dot{x}_4 = u_1 - K_s x_3 - B_s(x_4 - x_2)$$

$$y_1 = x_1 + d_2$$

$$y_2 = x_3 + x_1 + d_2$$

$$y_3 = \frac{1}{M_s}\left(u_1 - K_s x_3 - B_s(x_4 - x_2)\right)$$

$y_1 = z_1$ (unsprung mass height)
$y_2 = z_2$ (sprung mass height)
$y_3 = \ddot{z}_2$ (sprung mass acceleration)

$x_1 = z_1 - z_0$ (tire deflection)
$x_2 = \dot{z}_1$ (unsprung mass velocity)
$x_3 = z_2 - z_1$ (suspension deflection)
$x_4 = \dot{z}_2$ (sprung mass velocity)
$u_1 = F_c$ (active suspension force)
$d_1 = \dot{z}_0$ (rate of change of road height)
$d_2 = z_0$ (road height)

Modeling (cont.)

- Derive state-space model
- Need 4 equations
  (one defining the derivative of each state)

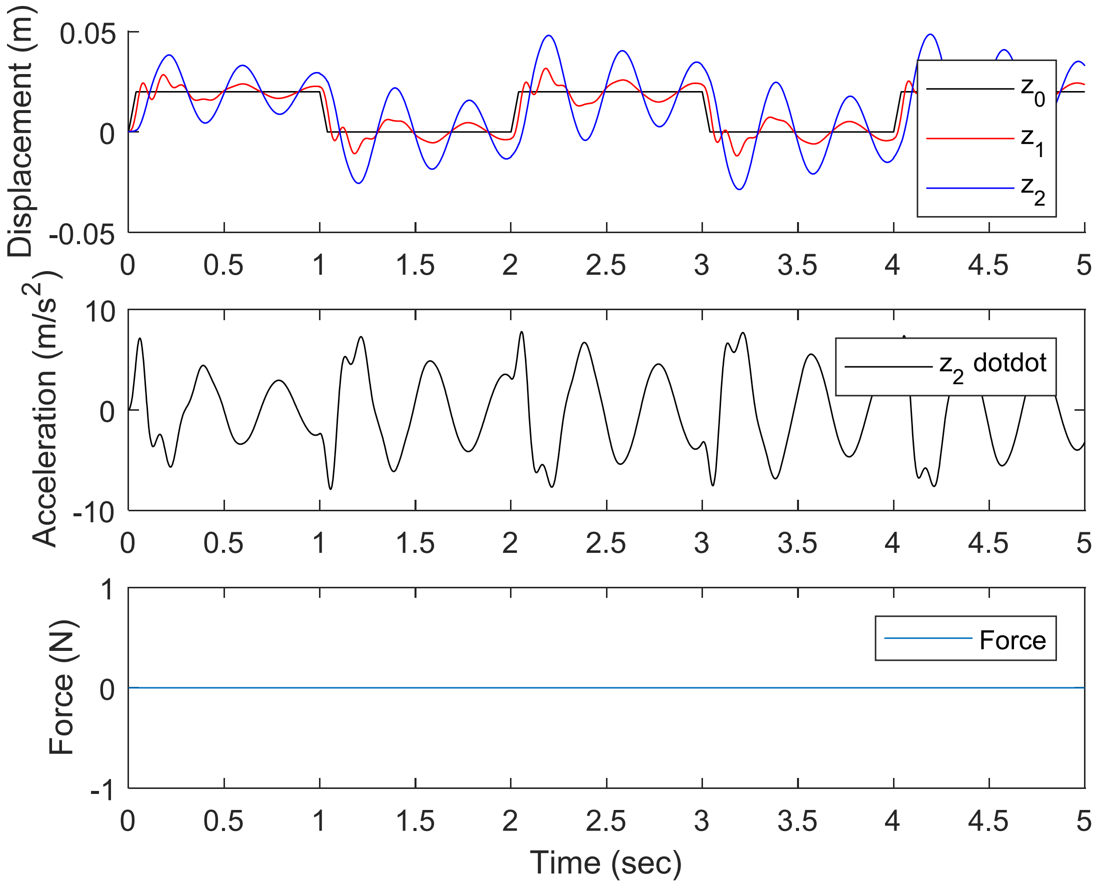
$$
\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\dfrac{K_t}{M_u} & -\dfrac{B_s + B_t}{M_u} & \dfrac{K_s}{M_u} & \dfrac{B_s}{M_u} \\ 0 & -1 & 0 & 1 \\ 0 & \dfrac{B_s}{M_s} & -\dfrac{K_s}{M_s} & -\dfrac{B_s}{M_s} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ -\dfrac{1}{M_u} \\ 0 \\ \dfrac{1}{M_s} \end{bmatrix} u_1 + \begin{bmatrix} -1 & 0 \\ \dfrac{B_t}{M_u} & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}
$$

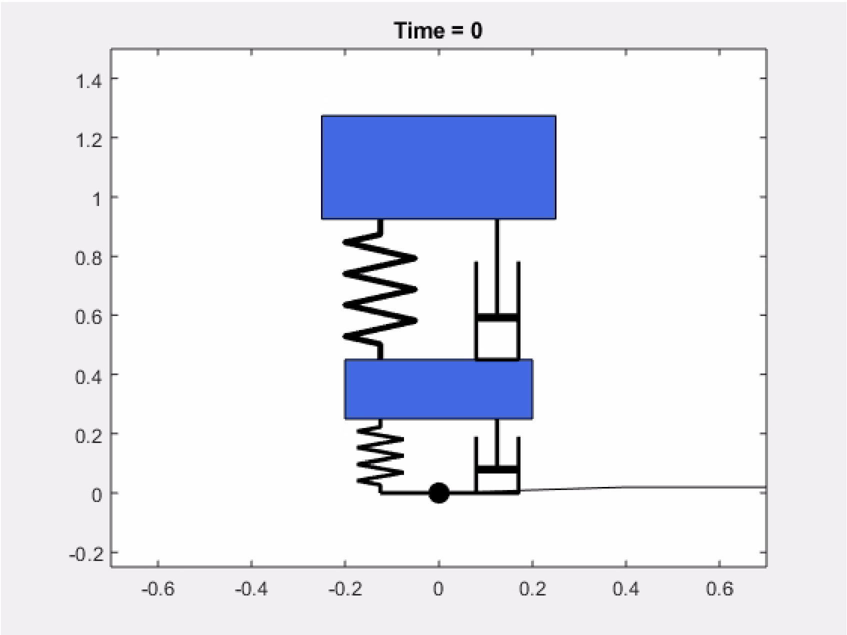
$$
\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & \dfrac{B_s}{M_s} & -\dfrac{K_s}{M_s} & -\dfrac{B_s}{M_s} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dfrac{1}{M_s} \end{bmatrix} u_1 + \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}
$$

Open-Loop Simulation

- No active suspension $F_c = u_1 = 0$
- Parameters from Quanser system

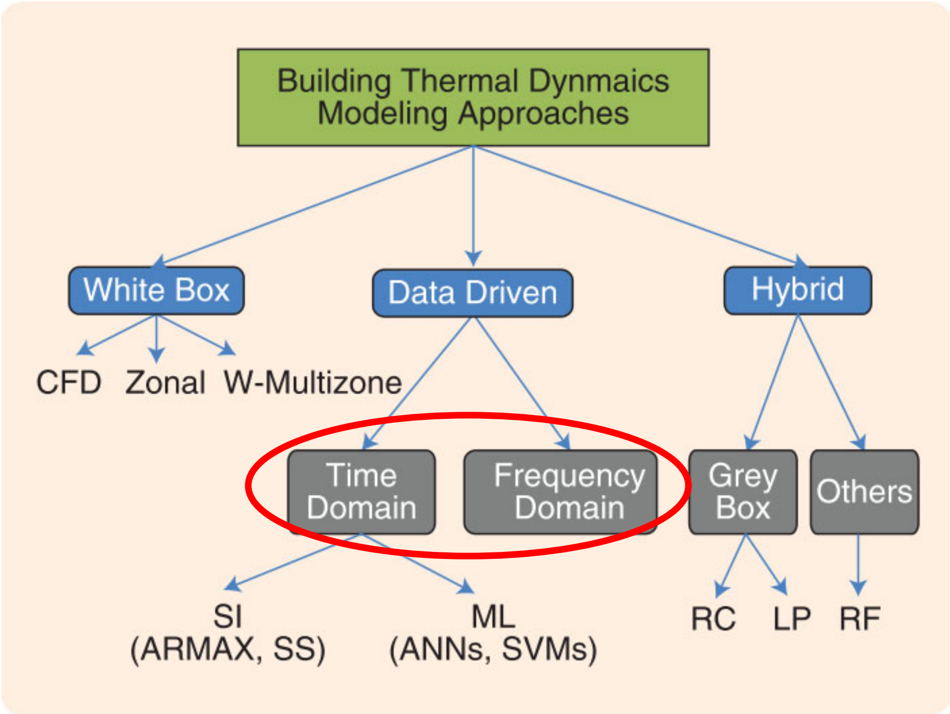
$$M_s = 2.45\ kg$$
$$M_u = 1\ kg$$
$$K_s = 900\ N/m$$
$$K_t = 2500\ N/m$$
$$B_s = 7.5\ \sec/m$$
$$B_t = 5\ \sec/m$$





System Identification

- 3 different types of modeling/system identification
  - First-principles (from fundamentals)
  - Empirical (from data)
  - Parameter identification (combination/both)
- This is a huge field with a rich theory[1]
  - We will only cover some of the very basics



Atam and Helsen, *Control-Oriented Thermal Modeling of Multizone Buildings: Methods and Issues*, CSM, 2016.

[1] System Identification by Lennart Ljung

System Identification (cont.)

- Empirical modeling – data driven

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

System

https://en.wikipedia.org/wiki/Mathematical_model

$u(t)$

$y(t)$

System Identification Algorithm

n4sid in Matlab

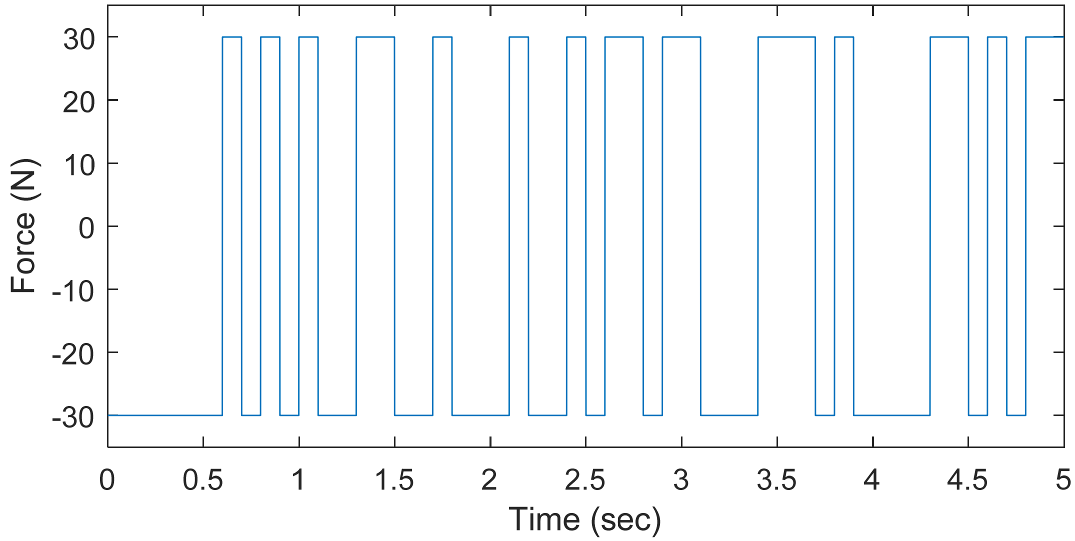
$$\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D}$$

System Identification (cont.)

- Generally trying to identify a linear model about an known equilibrium condition
- Design input trajectories to cause the system to deviate from this equilibrium.
- What are some of the considerations we should make when designing these input trajectories to identify the system dynamics?
  - What shape should these inputs be? (steps, sinusoids, random)
  - How big should these deviations be?
  - How fast should the inputs be changing?
  - How do I 'excite' all the important modes/features of the system?
  - Could my inputs cause unsafe system operation?
- We will try two different data-driven approaches
  - Time-domain – step inputs
  - Frequency-domain – sinusoidal inputs (next class?)

System Identification (cont.)

- Focus on designing active suspension force input
  - Would also want to vary road height
- Step changes are widely used for system identification
  - Pseudo-Random Binary Signals (PRBS)
  - Looks like random steps between a low and high value
  - Great way to get all the combinations if you have multiple inputs
  - Magnitude should be maximized to maximize the signal-to-noise ratio of the measured outputs
  - Magnitude should be small if the system is nonlinear and you are trying to estimate a linear model



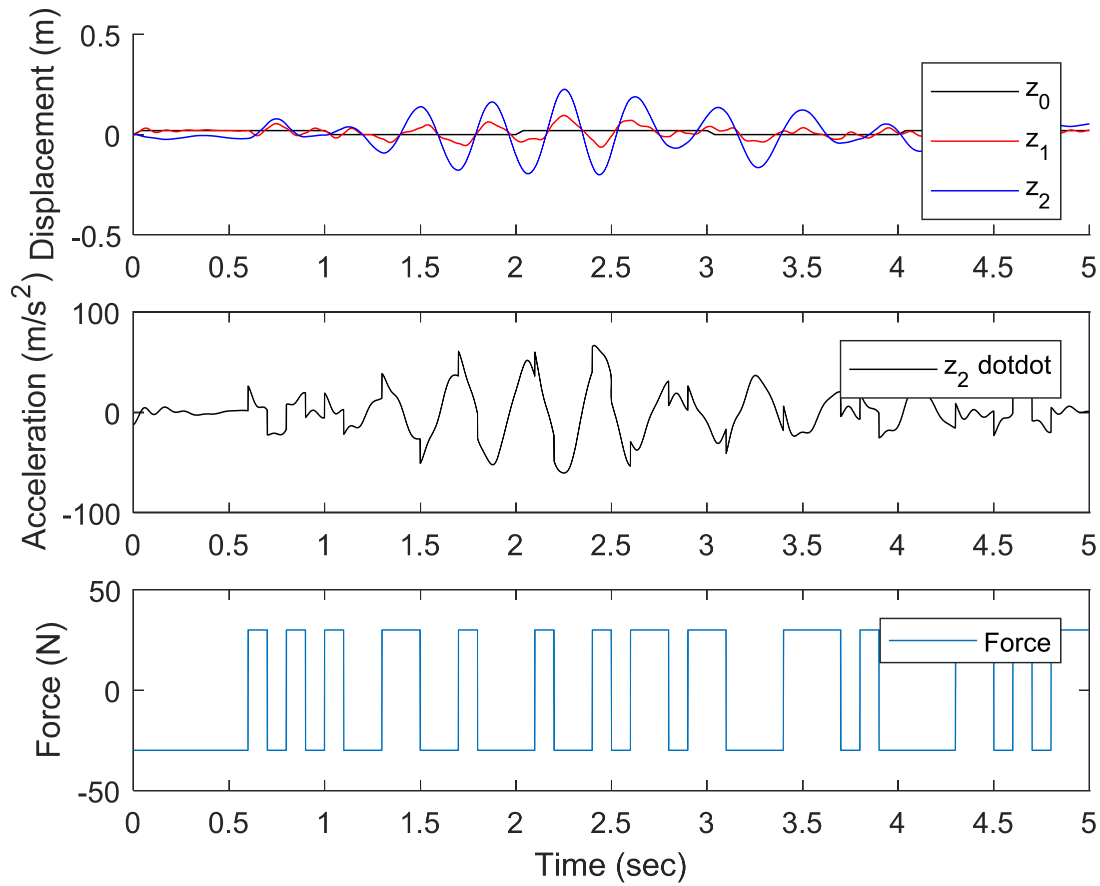
System Identification (cont.)

- Use Matlab function **idinput** to generate input trajectory
  - https://www.mathworks.com/help/ident/ref/idinput.html?s_tid=srchtitle
- Simulate system to collect output data

```
%% System identification Data
Range = [-30,30]; % Max force is 38 Newtons
Band = [0 1e-2];
F_ID = idinput(length(z0),'prbs',Band,Range);
figure;
plot(t,F_ID)
ylabel('Force (N)')
xlabel('Time (sec)')

% Collect inputs/disturbances
u_ID = zeros(length(t),nu+nd);
u_ID(:,1) = F_ID;
u_ID(:,2) = z0dot;
u_ID(:,3) = z0;

% Simulate open-loop system
[y_ID,~,x_ID] = lsim(sys_c,u_ID,t,x0);
```



System Identification (cont.)

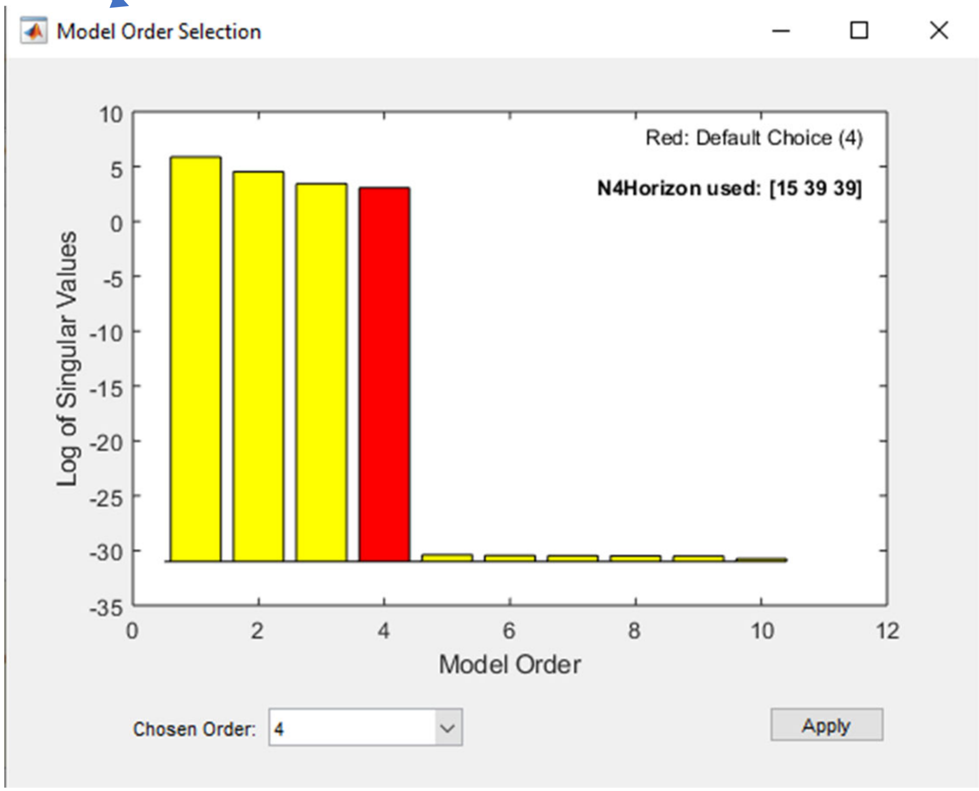
- Collect input/output data using **iddata**
  - https://www.mathworks.com/help/ident/ref/iddata.html?s_tid=srchtitle
- Generally, you would use output data (since the outputs are what you can measure)
- Here we will use the state trajectories to show that we can exactly identify the correct state-space model
- Identify the system model using **n4sid** by providing:
  - the input/output data
  - the range of system orders you would like to try
  - the time step of the model (0 for continuous-time)
  - the form of the model
    - Modal
    - Companion
    - Canonical
  - https://www.mathworks.com/help/ident/ref/n4sid.html?s_tid=srchtitle

```
%% System identification (States, no noise)
data = iddata(x_ID,u_ID,dt);
sys_ID = n4sid(data,[1:10],'Ts',0,'Form','canonical');
figure;
compare(data,sys_ID)
```

System Identification (cont.)

```
%% System identification (States, no noise)
data = iddata(x_ID,u_ID,dt);
sys_ID = n4sid(data,[1:10],'Ts',0,'Form','canonical');
figure;
compare(data,sys_ID)
```

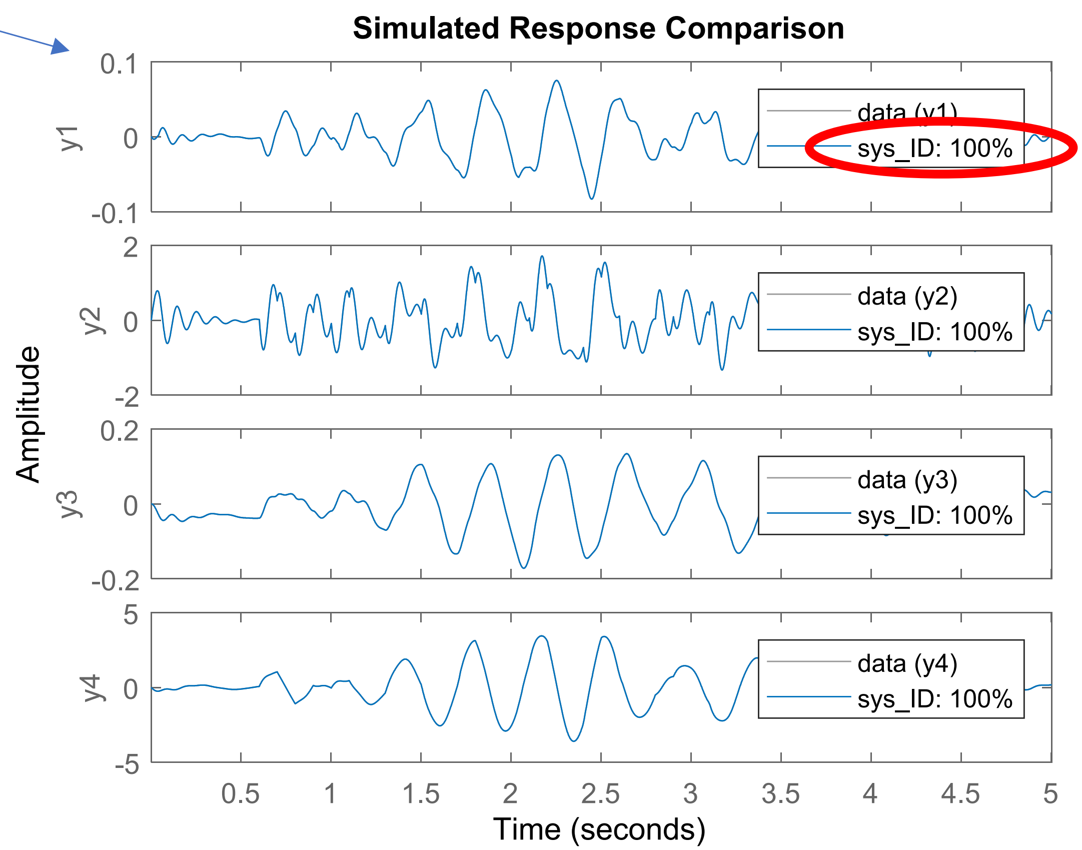
- When you give a range of system orders, n4sid will provide a plot of Hankel singular values
- Want to choose the smallest model order that accurately captures the data behavior
- Pick the order where there is a large change in singular values



System Identification (cont.)

```
%% System identification (States, no noise)
data = iddata(x_ID,u_ID,dt);
sys_ID = n4sid(data,[1:10],'Ts',0,'Form','canonical');
figure;
compare(data,sys_ID)
```

- By providing state measurements with no measurement noise, identified model provides perfect prediction of measured data
- 100% goodness-of-fit



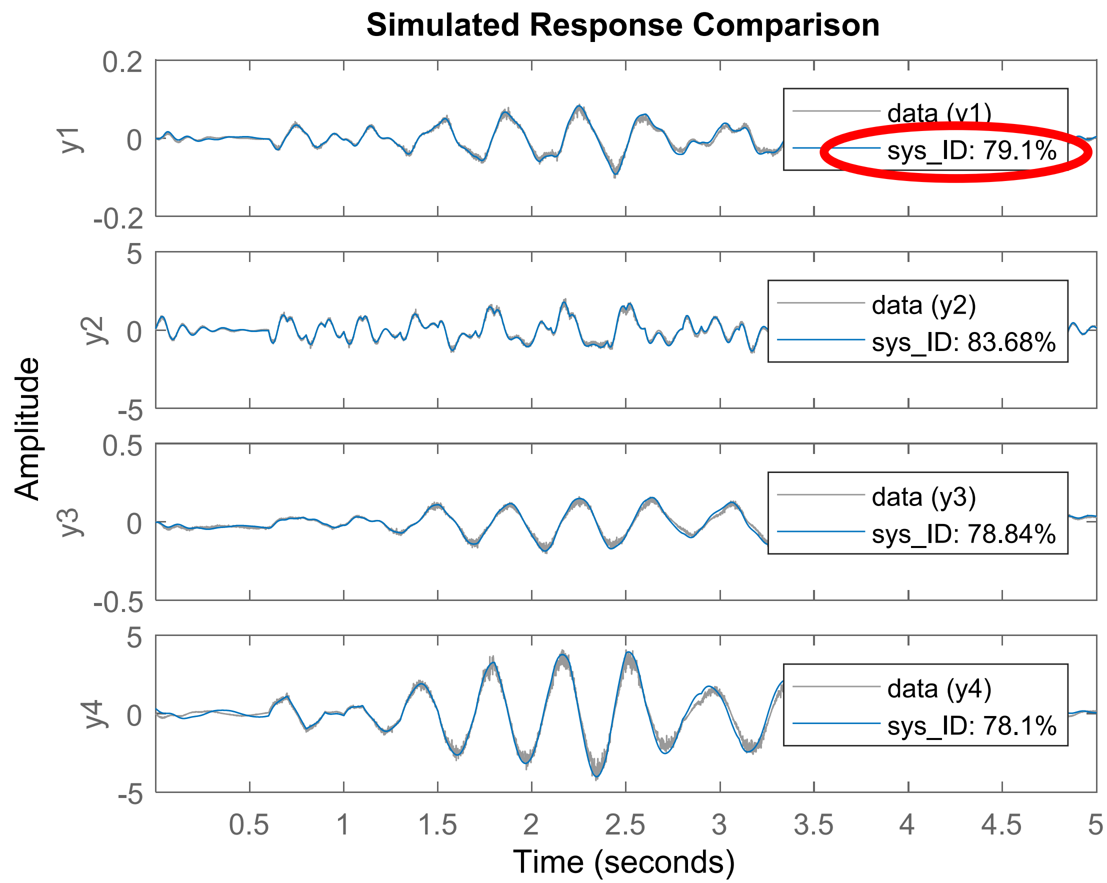
measured   model

$$\text{fit} = 100\left(1 - \frac{\left\| y - \hat{y} \right\|}{\left\| y - \text{mean}(y) \right\|}\right)$$

System Identification (cont.)

```
%% System identification (States, noise)
rng(1)
data = iddata(x_ID.*(1+1e-1*randn(size(x_ID))),u_ID,dt);
sys_ID = n4sid(data,[1:10],'Ts',0,'Form','canonical');
figure;
compare(data,sys_ID)
```

- Adding ~10% measurement noise degrades goodness-of-fit



Discretization

- Most of the time, our model will be in the continuous-time domain

$$\dot{x} = f_c(x,u,d) \qquad \dot{x} = A_c x + B_c u + V_c d$$
$$y = h(x,u,d) \qquad y = Cx + Du + Wd$$

- Need to convert to discrete-time

$$x_{k+1} = f(x_k, u_k, d_k) \qquad x_{k+1} = Ax_k + Bu_k + Vd_k$$
$$y_k = h(x_k, u_k, d_k) \qquad y_k = Cx_k + Du_k + Wd_k$$

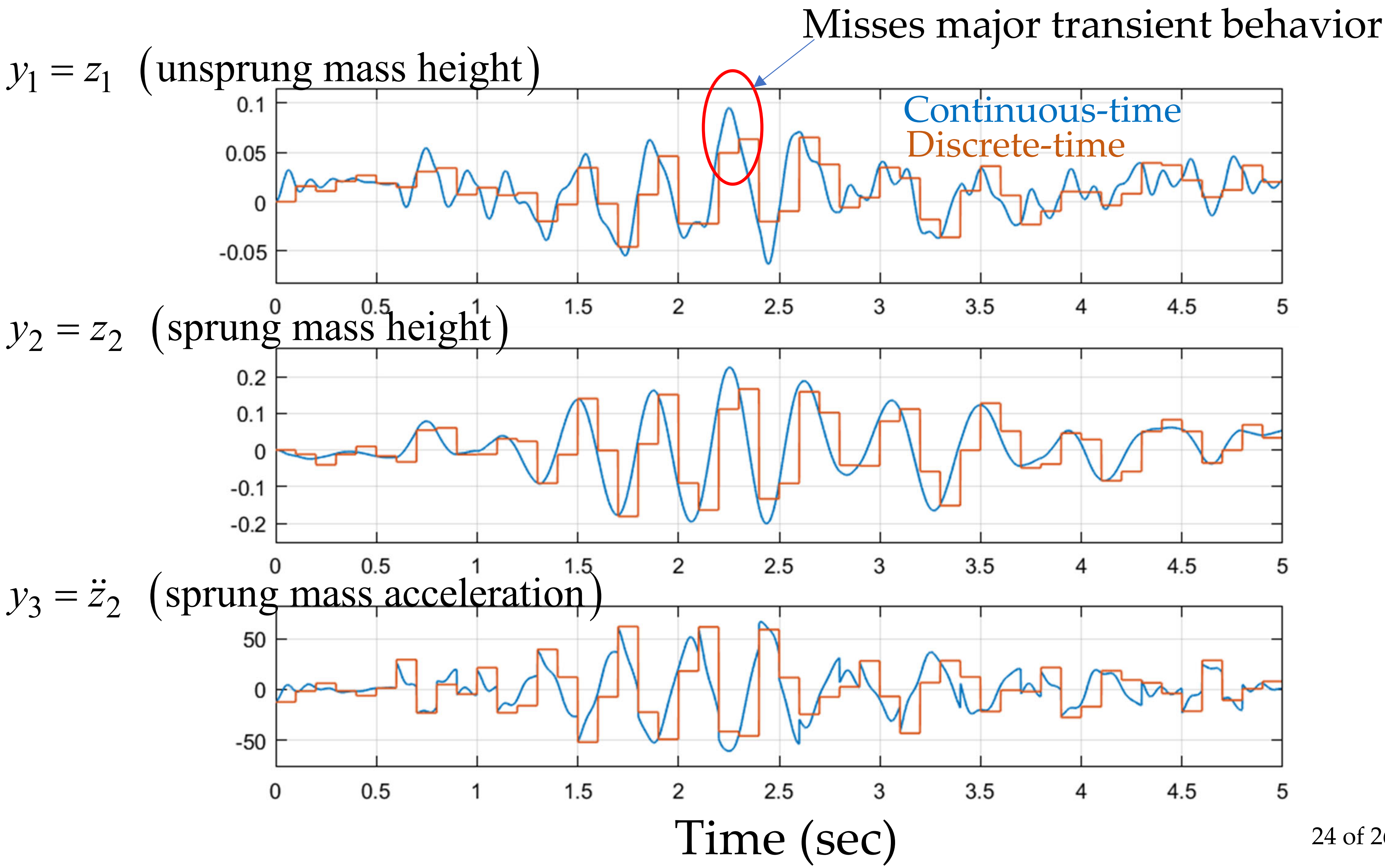
- We have discussed this in detail in Lecture 4 (Slides 6-15)
- For linear systems, use the **c2d** command in Matlab
  - https://www.mathworks.com/help/control/ref/c2d.html?s_tid=srchtitle

```
sys_c = ss(Ac,[Bc Vc],Cc,[Dc Wc]);
sys = c2d(sys_c,dt);
```

Want a large, time step that still captures the fast dynamics

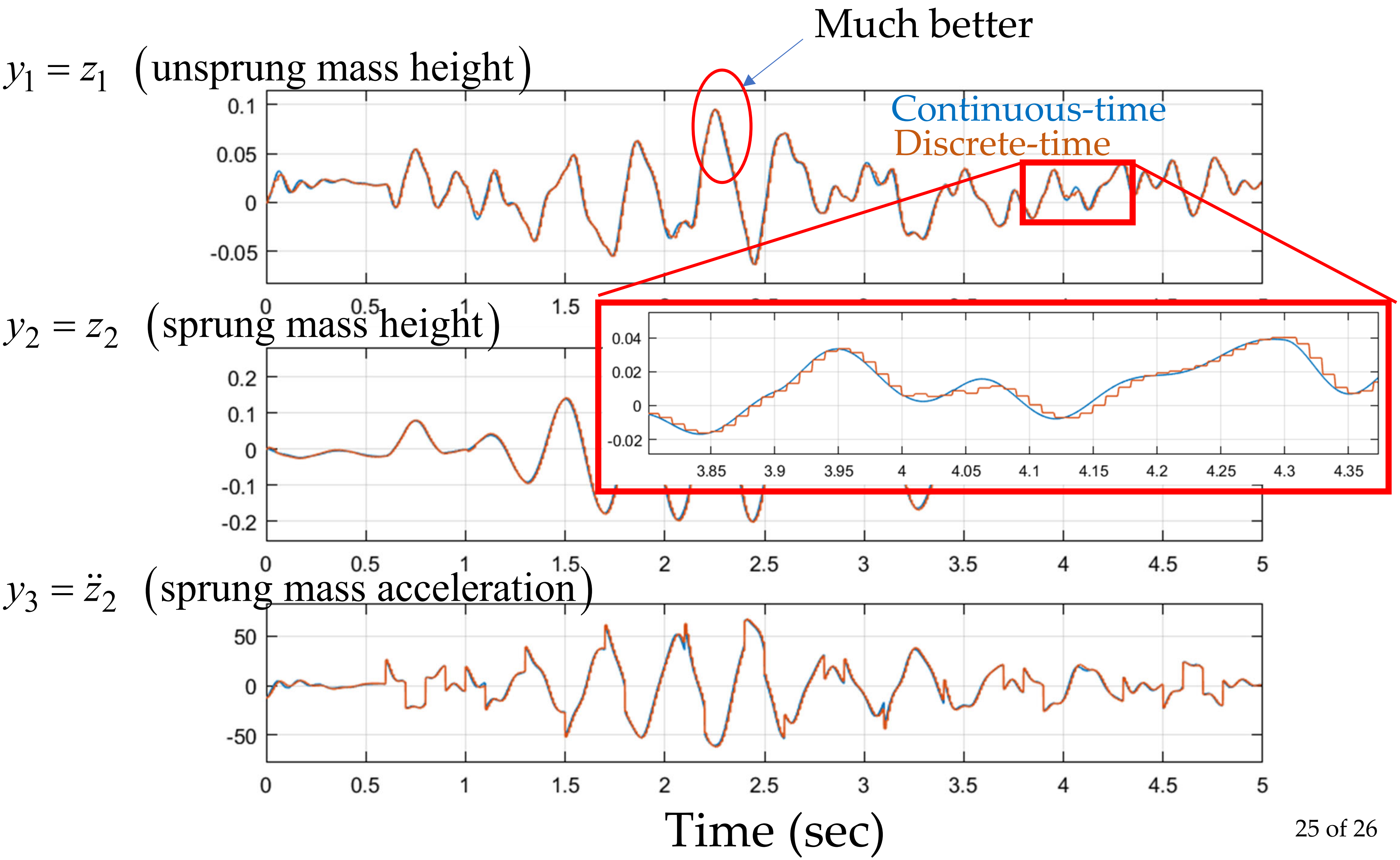
Discretization (cont.)

- Try a discrete-time step of 0.1 seconds



Discretization (cont.)

- Try a discrete-time step of 0.01 seconds



Discretization (cont.)

- Try a discrete-time step of 0.05 seconds
- This may be a better trade-off between accuracy and time step size