Workshop: modeling and control of a vehicle suspension

Joan Vazquez Molina

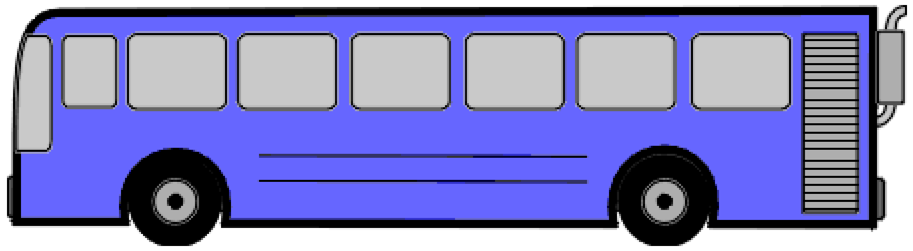
2019-10-21

Course: Mathematical Modeling in the Industry
MSc in Mathematical Research
Universitat Politècnica de València

JVazquezMolina@driv.com

Goal

- Design a feedback controller for an active suspension so that the suspension deflection has an overshoot of less than 5% and a settling time shorter than 5 seconds under a road step input.



Source: CONTROL TUTORIALS FOR MATLAB® & SIMULINK® http://ctms.engin.umich.edu/CTMS/index.php?example=Suspension&section=SystemModeling

Goal

- Design a feedback controller for an active suspension so that the suspension deflection has an overshoot of less than 5% and a settling time shorter than 5 seconds under a road step input.
  - Task
  - System
  - Input
  - Output
  - Specifications

- This is a toy example, but extensions of this mathematical framework are used daily to make business decisions with a huge financial and human impact.

Agenda

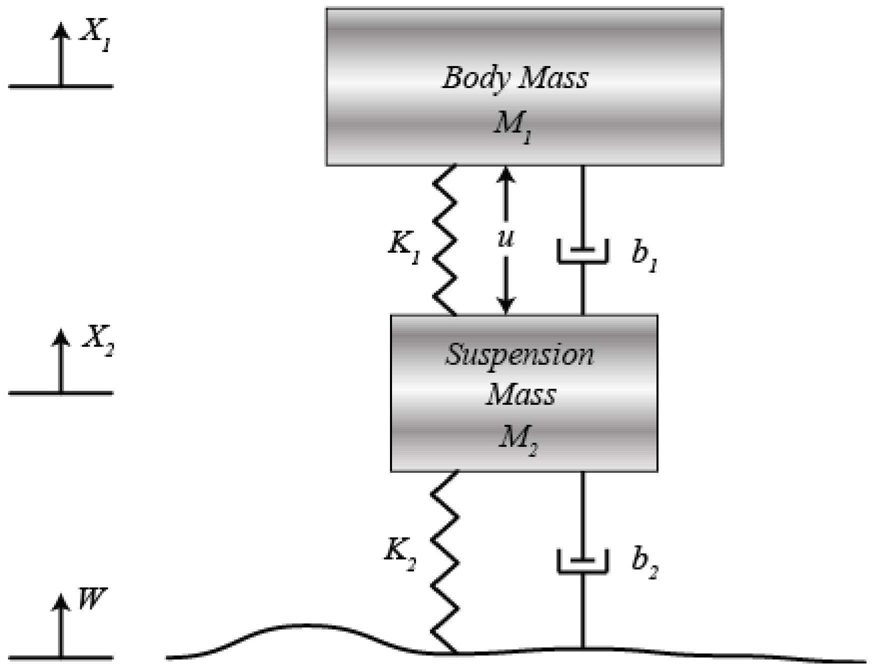
- System Modelling
  - Model the system with ODEs.
  - Calculate the transfer functions of interest.
- System Analysis
  - Analyze the open-loop response.
  - Analyze the frequency response with Bode plots.
- PID control
  - Create a PID controller that meets the requirements.
- Simulink
  - Simulate the open-loop response.

System Modeling

Equations of motion

- Simplest vehicle model.
  - The *quarter car* model. Only vertical dynamics.
  - Road displacement w.
  - Actuator force u.
- **Challenge**: find the equations of motion.
  - But first, find: number of Degrees of Freedom, State variables, Inputs, Parameters.
- Use physics:
  - Newton's law: $F = m \cdot \ddot{x}$
  - Constitutive equations:
    - Spring: $F = k \cdot x$
    - Damper: $F = b \cdot \dot{x}$

Model of Bus Suspension System (1/4 Bus)



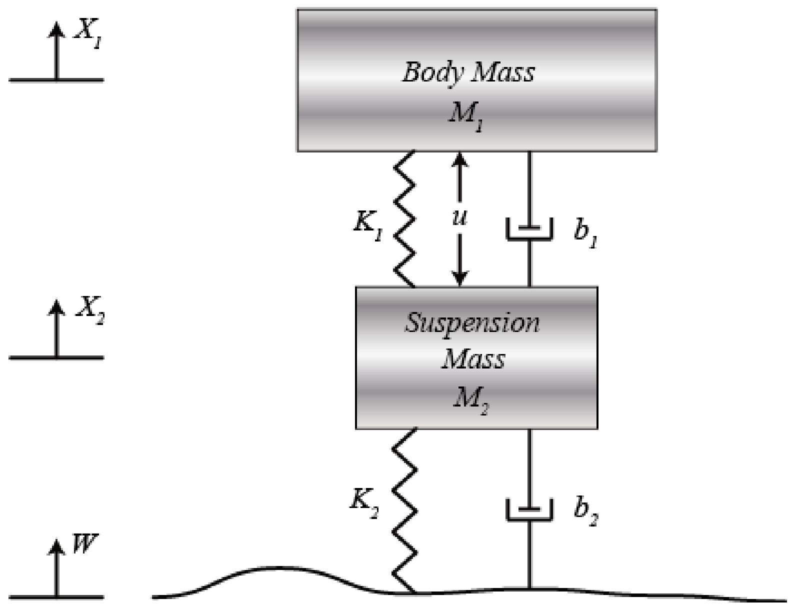
System Modeling

Equations of motion (SOLUTION)

- Degrees of Freedom: 2
- State variables: $x_1,\ x_2$
- Inputs: u (actuator force), w (road displacement)
- Output: $y = x_1 - x_2$ (suspension deflection)
- Parameters: $m_1, m_2, k_1, k_2, b_1, b_2,$

$$\boxed{\begin{gathered} m_1\ddot{x}_1 = -b_1(\dot{x}_1 - \dot{x}_2) - k_1(x_1 - x_2) + u \\ m_2\ddot{x}_2 = b_1(\dot{x}_1 - \dot{x}_2) + k_1(x_1 - x_2) + b_2(\dot{w} - \dot{x}_2) + k_1(w - x_2) - u \end{gathered}}$$

Model of Bus Suspension System (1/4 Bus)



System Modeling

Laplace transform

- The Laplace transform is used to transform differential equations into algebraic equations and obtain the transfer functions.
- **Challenge**: find the Laplace transform of the system equations:

$$m_1\ddot{x}_1 = -b_1(\dot{x}_1 - \dot{x}_2) - k_1(x_1 - x_2) + u$$
$$m_2\ddot{x}_2 = b_1(\dot{x}_1 - \dot{x}_2) + k_1(x_1 - x_2) + b_2(\dot{w} - \dot{x}_2) + k_1(w - x_2) - u$$

(assume static initial conditions)

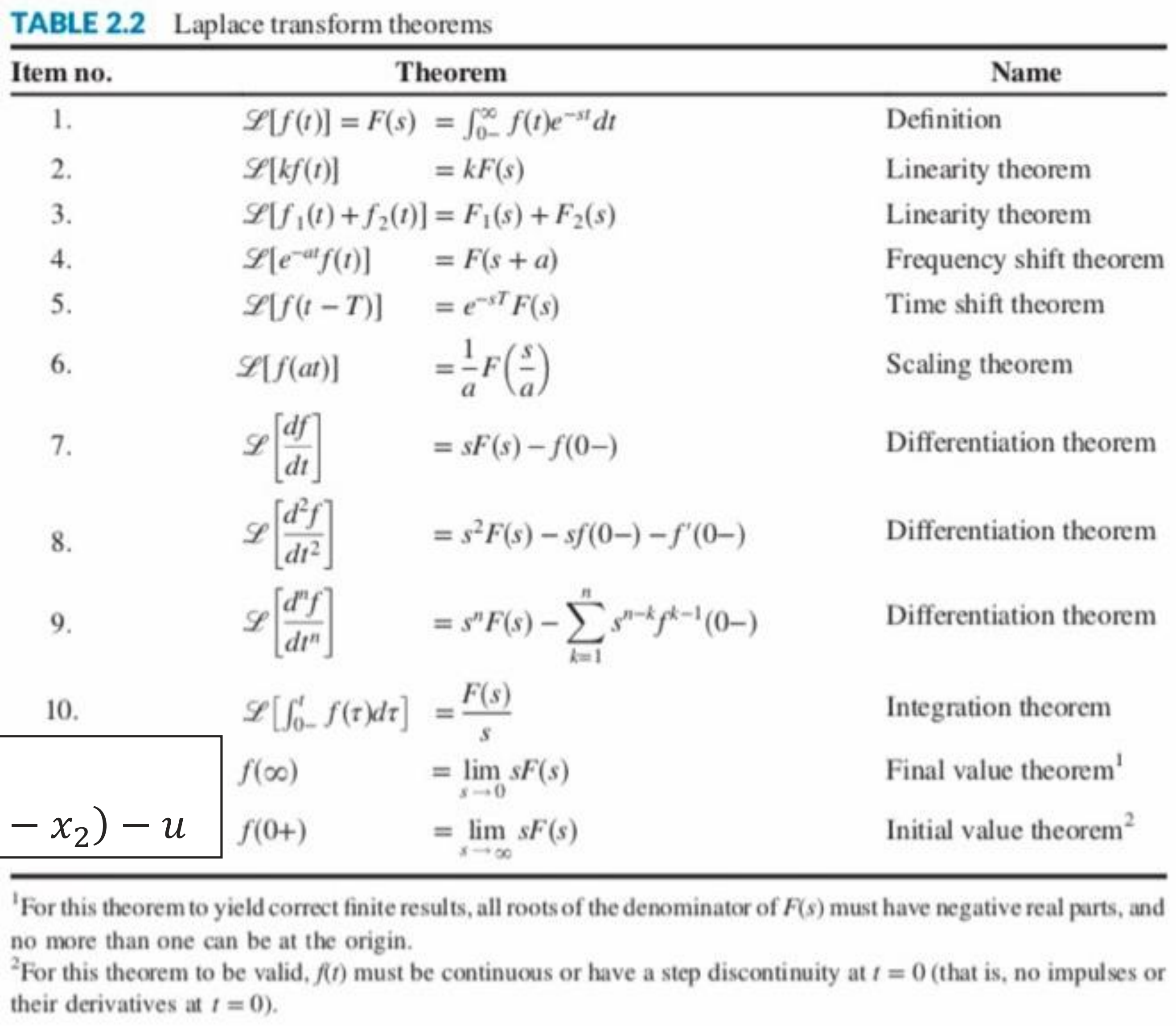


TABLE 2.2 Laplace transform theorems

Item no.	Theorem		Name
1.	$\mathscr{L}[f(t)] = F(s)$	$= \int_{0-}^{\infty} f(t)e^{-st}dt$	Definition
2.	$\mathscr{L}[kf(t)]$	$= kF(s)$	Linearity theorem
3.	$\mathscr{L}[f_1(t)+f_2(t)]$	$= F_1(s)+F_2(s)$	Linearity theorem
4.	$\mathscr{L}[e^{-at}f(t)]$	$= F(s+a)$	Frequency shift theorem
5.	$\mathscr{L}[f(t-T)]$	$= e^{-sT}F(s)$	Time shift theorem
6.	$\mathscr{L}[f(at)]$	$= \frac{1}{a}F\left(\frac{s}{a}\right)$	Scaling theorem
7.	$\mathscr{L}\left[\frac{df}{dt}\right]$	$= sF(s) - f(0-)$	Differentiation theorem
8.	$\mathscr{L}\left[\frac{d^2f}{dt^2}\right]$	$= s^2F(s) - sf(0-) - f'(0-)$	Differentiation theorem
9.	$\mathscr{L}\left[\frac{d^nf}{dt^n}\right]$	$= s^nF(s) - \sum_{k=1}^{n} s^{n-k}f^{k-1}(0-)$	Differentiation theorem
10.	$\mathscr{L}\left[\int_{0-}^{t} f(\tau)d\tau\right]$	$= \frac{F(s)}{s}$	Integration theorem
	$f(\infty)$	$= \lim_{s\to 0} sF(s)$	Final value theorem[1]
	$f(0+)$	$= \lim_{s\to\infty} sF(s)$	Initial value theorem[2]

[1]For this theorem to yield correct finite results, all roots of the denominator of $F(s)$ must have negative real parts, and no more than one can be at the origin.
[2]For this theorem to be valid, $f(t)$ must be continuous or have a step discontinuity at $t = 0$ (that is, no impulses or their derivatives at $t = 0$).

System Modeling

Laplace transform (SOLUTION)

- The Laplace transform is:

$$(m_1 s^2 + b_1 s + k_1)X_1(s) - (b_1 s + k_1)X_2(s) = U(s)$$
$$-(b_1 s + k_1)X_1(s) + \big(m_2 s^2 + (b_1 + b_2)s + (k_1 + k_2)\big)X_2(s) = (b_2 s + k_2)W(s) - U(s)$$

- If you prefer it in matrix form:

$$\underbrace{\begin{bmatrix} m_1 s^2 + b_1 s + k_1 & -(b_1 s + k_1) \\ -(b_1 s + k_1) & m_2 s^2 + (b_1 + b_2)s + (k_1 + k_2) \end{bmatrix}}_{A} \begin{bmatrix} X_1(s) \\ X_2(s) \end{bmatrix} = \begin{bmatrix} U(s) \\ (b_2 s + k_2)W(s) - U(s) \end{bmatrix}$$

System Modeling

Transfer functions

- “A transfer function of an electronic or control system component is a mathematical function which theoretically models the device's output for each possible input.” (Wikipedia)

$$H(s) = \frac{Y(s)}{X(s)} = \frac{\mathcal{L}\{y(t)\}}{\mathcal{L}\{x(t)\}}$$

- **Challenge**: find the transfer functions between the inputs and the suspension deflection:

$$G_1(s) = \frac{X_1(s) - X_2(s)}{U(s)} \quad ; \quad G_2(s) = \frac{X_1(s) - X_2(s)}{W(s)}$$

- Hint: invert A using Δ, then put the system in standard form $\begin{bmatrix} X_1(s) \\ X_2(s) \end{bmatrix} = M \cdot \begin{bmatrix} U(s) \\ W(s) \end{bmatrix}$

$$\Delta = \det(A) = (m_1 s^2 + b_1 s + k_1)(m_2 s^2 + (b_1 + b_2)s + (k_1 + k_2)) - (b_1 s + k_1)(b_1 s + k_1)$$

System Modeling

Transfer functions (SOLUTION)

- Invert and rearrange

$$\begin{bmatrix} X_1(s) \\ X_2(s) \end{bmatrix} = \frac{1}{\Delta}\begin{bmatrix} m_2s^2 + (b_1+b_2)s + (k_1+k_2) & (b_1s+k_1) \\ (b_1s+k_1) & m_1s^2 + b_1s + k_1 \end{bmatrix}\begin{bmatrix} U(s) \\ (b_2s+k_2)W(s) - U(s) \end{bmatrix}$$

$$\begin{bmatrix} X_1(s) \\ X_2(s) \end{bmatrix} = \frac{1}{\Delta}\begin{bmatrix} m_2s^2 + b_2s + k_2 & b_1b_2s^2 + (b_1k_2 + b_2k_1)s + k_1k_2 \\ -m_1s^2 & m_1b_2s^3 + (m_1k_2 + b_1b_2)s^2 + (b_1k_2 + b_2k_1)s + k_1k_2 \end{bmatrix}\begin{bmatrix} U(s) \\ W(s) \end{bmatrix}$$

- Set $w = 0$ or $u = 0$ to get

$$G_1(s) = \frac{X_1(s) - X_2(s)}{U(s)} = \frac{(m_1+m_2)s^2 + b_2s + k_2}{\Delta} \quad ; \quad G_2(s) = \frac{X_1(s) - X_2(s)}{W(s)} = \frac{-m_1b_2s^3 - m_1k_2s^2}{\Delta}$$

System Modeling

Transfer functions in Matlab

- **Challenge**: define the transfer functions (TFs) in Matlab

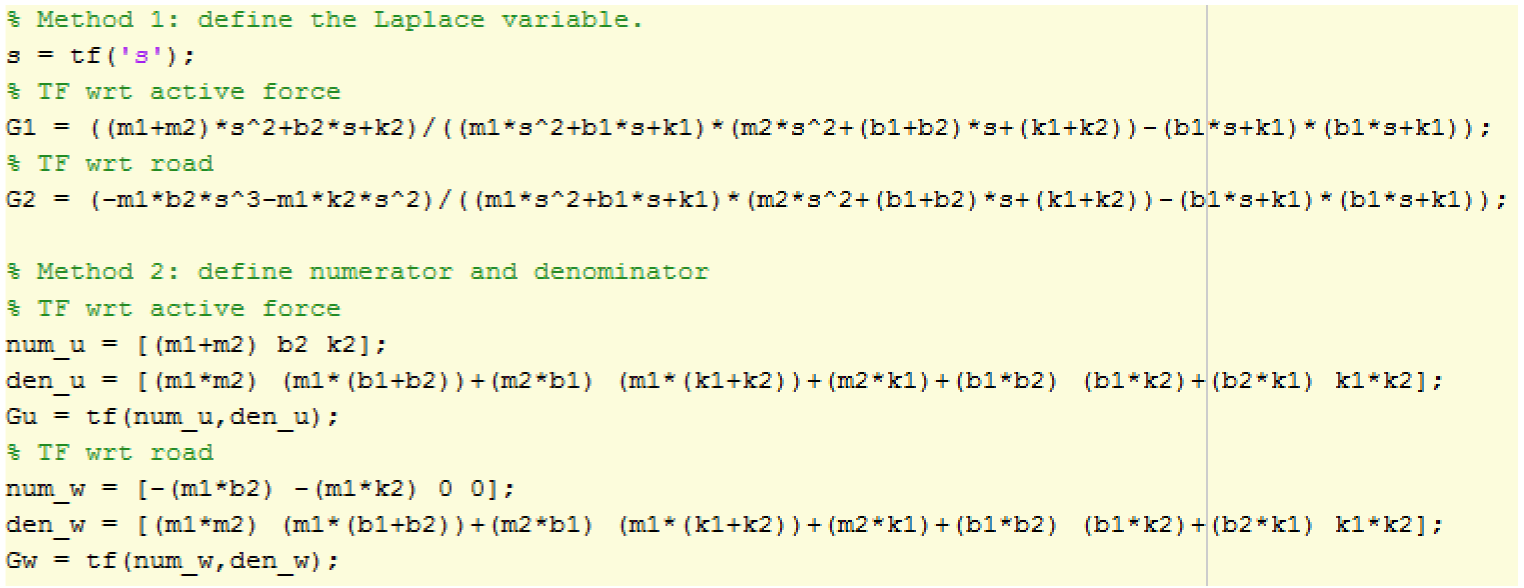
```
% Suggestion: help tf
```

- Use:

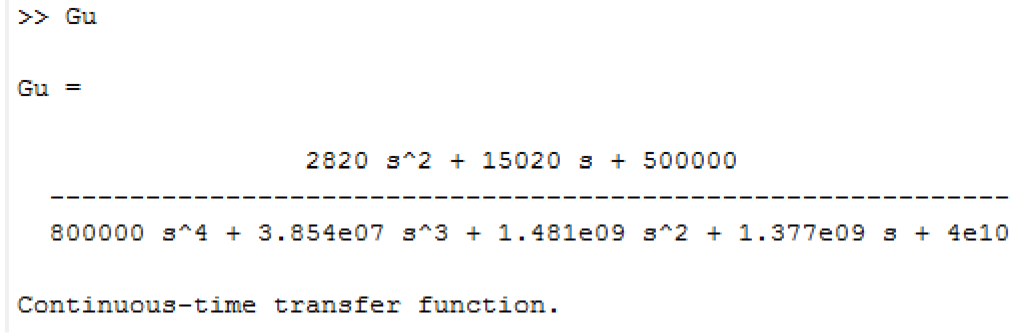
```
%% Parameters
m1 = 2500;              % kg, sprung mass
m2 = 320;               % kg, unsprung mass
k1 = 80000;             % N/m, spring constant of suspension
k2 = 500000;            % N/m, spring constant of tire
b1 = 350;               % N*s/m, damping constant of suspension
b2 = 15020;             % N*s/m, damping constant of tire
```

System Modeling

Transfer functions in Matlab [SOLUTION]



```
% Method 1: define the Laplace variable.
s = tf('s');
% TF wrt active force
G1 = ((m1+m2)*s^2+b2*s+k2)/((m1*s^2+b1*s+k1)*(m2*s^2+(b1+b2)*s+(k1+k2))-(b1*s+k1)*(b1*s+k1));
% TF wrt road
G2 = (-m1*b2*s^3-m1*k2*s^2)/((m1*s^2+b1*s+k1)*(m2*s^2+(b1+b2)*s+(k1+k2))-(b1*s+k1)*(b1*s+k1));

% Method 2: define numerator and denominator
% TF wrt active force
num_u = [(m1+m2) b2 k2];
den_u = [(m1*m2) (m1*(b1+b2))+(m2*b1) (m1*(k1+k2))+(m2*k1)+(b1*b2) (b1*k2)+(b2*k1) k1*k2];
Gu = tf(num_u,den_u);
% TF wrt road
num_w = [-(m1*b2) -(m1*k2) 0 0];
den_w = [(m1*m2) (m1*(b1+b2))+(m2*b1) (m1*(k1+k2))+(m2*k1)+(b1*b2) (b1*k2)+(b2*k1) k1*k2];
Gw = tf(num_w,den_w);
```



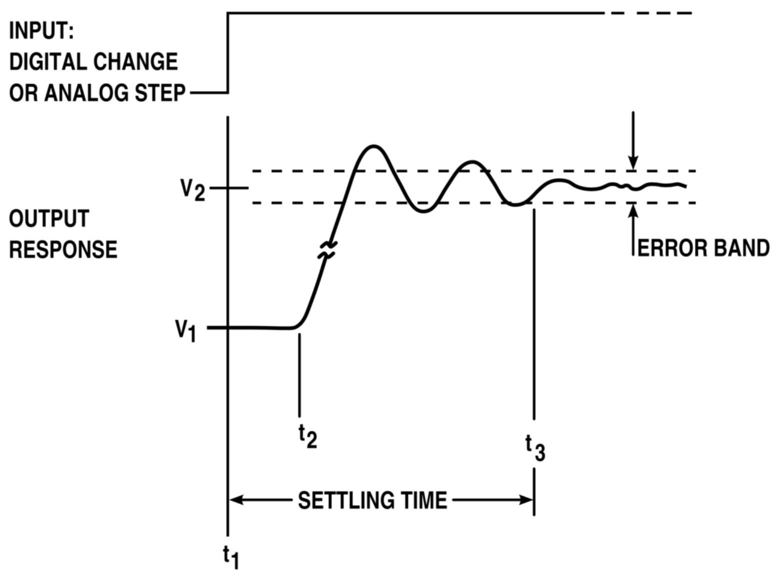
```
>> Gu

Gu =

                  2820 s^2 + 15020 s + 500000
  ------------------------------------------------------------
  800000 s^4 + 3.854e07 s^3 + 1.481e09 s^2 + 1.377e09 s + 4e10

Continuous-time transfer function.
```

System Analysis

Open-loop step response

- Open-loop = no feedback control
- “The step response of a system in a given initial state consists of the time evolution of its outputs when its control inputs are Heaviside step functions” (Wikipedia).
- **Challenge**: find the open-loop step response under single input
  - $u = 1\text{N}$
  - $w = 10\text{cm}$

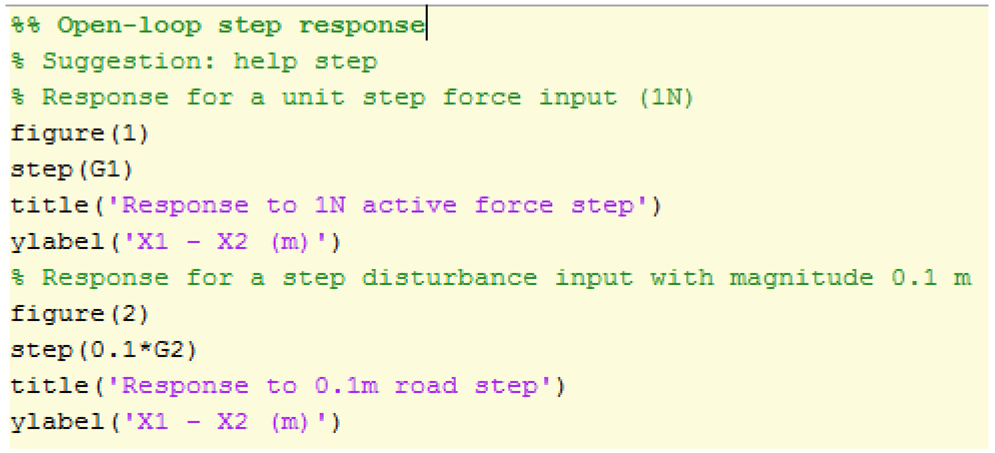
```
% Suggestion: help step
```



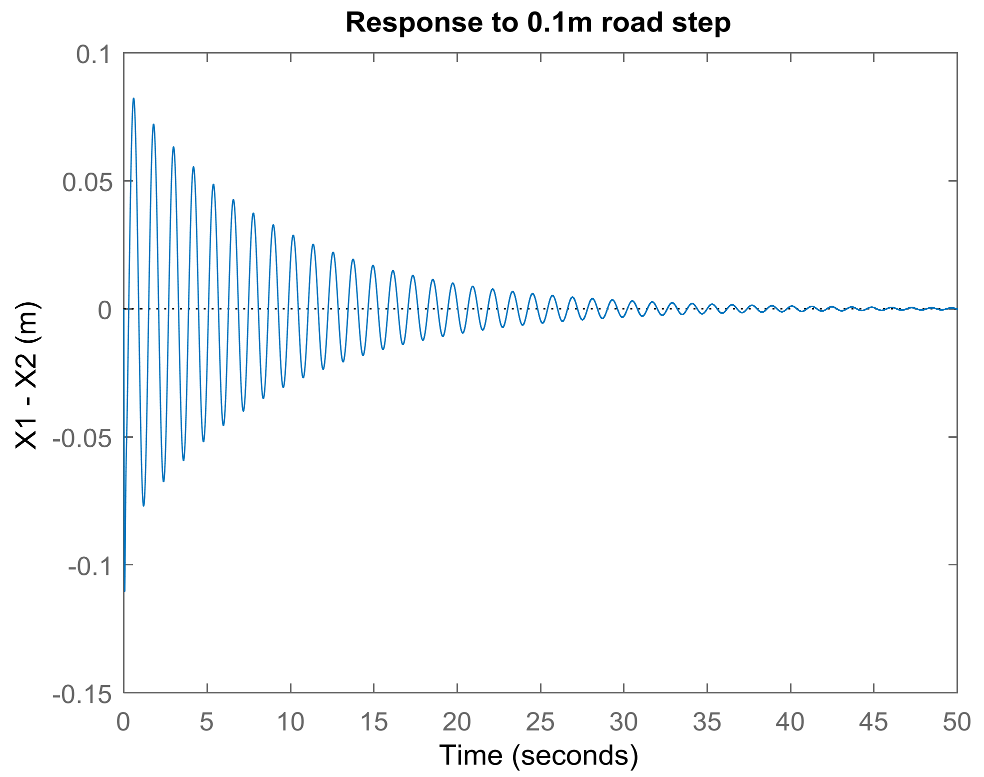
System Analysis

Open-loop step response [SOLUTION]

- Road step: large overshoot (8cm), long settling time (50s)



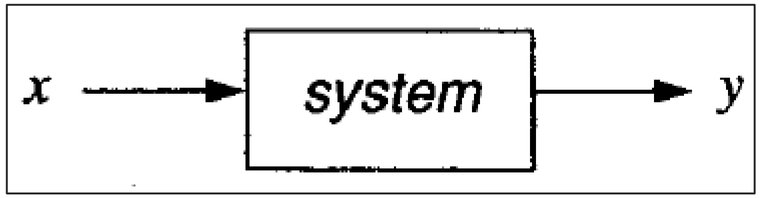
```
%% Open-loop step response
% Suggestion: help step
% Response for a unit step force input (1N)
figure(1)
step(G1)
title('Response to 1N active force step')
ylabel('X1 - X2 (m)')
% Response for a step disturbance input with magnitude 0.1 m
figure(2)
step(0.1*G2)
title('Response to 0.1m road step')
ylabel('X1 - X2 (m)')
```



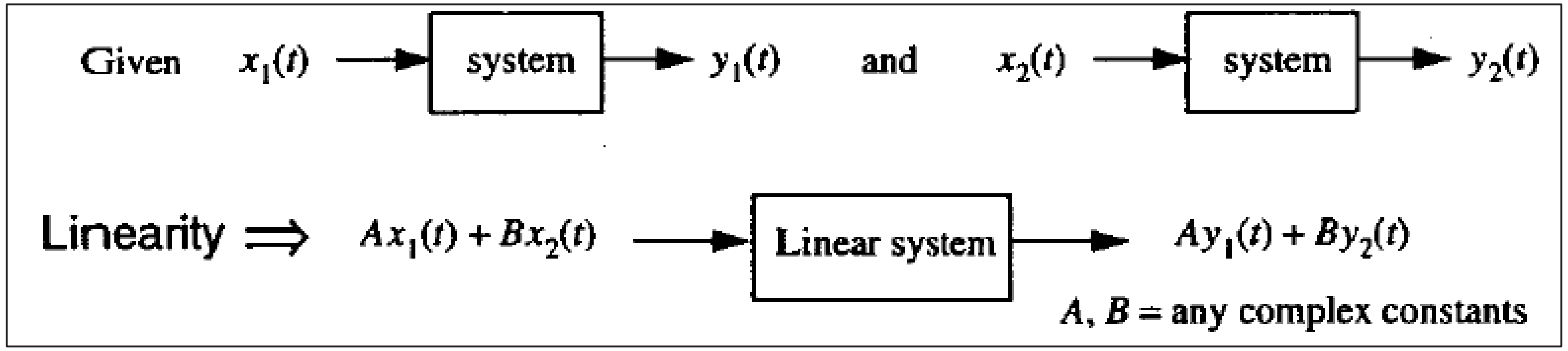
System Analysis

Linear Time-Invariant Systems [THEORY]

A *system* is an abstraction for anything that takes an input and produces an output.

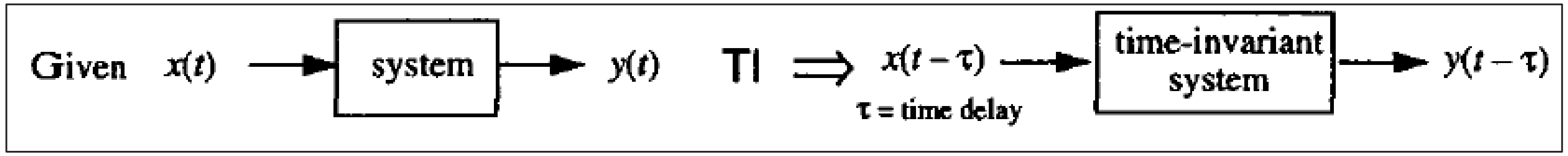


Linear



Given $x_1(t)$ system $y_1(t)$ and $x_2(t)$ system $y_2(t)$
Linearity $\Longrightarrow$ $Ax_1(t)+Bx_2(t)$ Linear system $Ay_1(t)+By_2(t)$
$A, B =$ any complex constants

Time-invariant



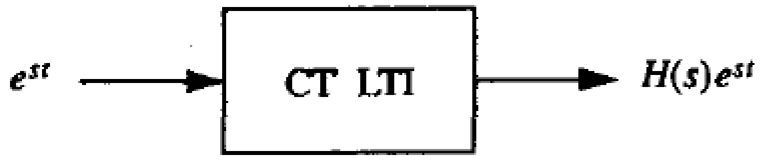
Given $x(t)$ system $y(t)$
TI $\Longrightarrow$ $x(t-\tau)$ time-invariant system $y(t-\tau)$
$\tau =$ time delay

System Analysis

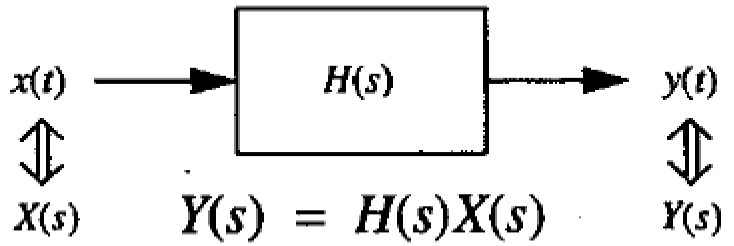
LTI systems, eigenfunctions and frequency preservation [THEORY]

Functions of the form e^{st} are known as *eigenfunctions* of continuous-time LTI systems. This means that when such a function is an input to a CT LTI system, the output is a function of the exact same complex frequency as the input, except it is multiplied by a scaling factor.

Output has same frequency as input!



H(*s*) is the System Function



$$Y(s) = H(s)X(s)$$

$$H(s) = \frac{Y(s)}{X(s)}$$

System Analysis

Bode Plots [THEORY]

$$\boxed{\text{Bode plot} = H(s) \text{ evaluated along } j\omega\text{-axis } (s=j\omega)}$$

$H(s)$ is a complex number: a Bode plot shows its magnitude (in decibels) and phase as a function of the frequency ω

Let's illustrate the mechanics behind graphing a frequency response with a simple example. Given the $H(s)$ shown below, plug in different values of ω and compute the magnitude and phase of $H(j\omega)$.

$$H(s) = \frac{1}{s+10} \qquad H(j\omega) = \frac{1}{j\omega+10}$$

$$\text{magnitude of } H(j\omega) = \frac{\text{magnitude of numerator}}{\text{magnitude of denominator}}$$

$$\text{phase of } H(j\omega) = \text{phase of numerator} - \text{phase of denominator}$$

$$\text{magnitude of } a+bj = \sqrt{a^2+b^2}$$

ω	$\lvert H(j\omega)\rvert$	$\angle H(j\omega)$	$20\log_{10}\lvert H(j\omega)\rvert$
0	0.1000	0.0 deg	-20.00 dB
1	0.0995	-5.71 deg	-20.04 dB
2	0.0981	-11.31 deg	-20.17 dB
5	0.0894	-26.57 deg	-20.97 dB
10	0.0707	-45.00 deg	-23.01 dB
20	0.0447	-63.43 deg	-26.99 dB
50	0.0196	-78.69 deg	-34.15 dB
100	0.0100	-84.29 deg	-40.04 dB

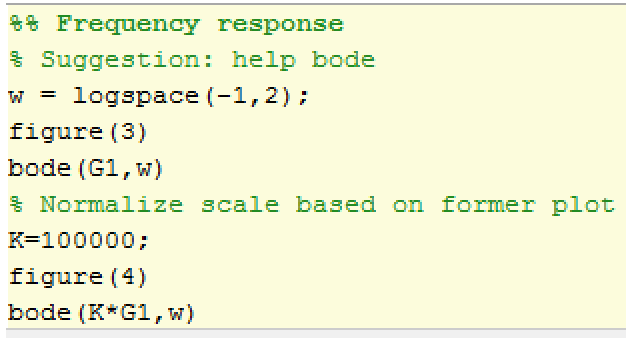
System Analysis
Bode Plots

- **Challenge**: draw the Bode plot for the transfer function G_1

```
% Suggestion: help bode
```

System Analysis

Bode Plots [SOLUTION]



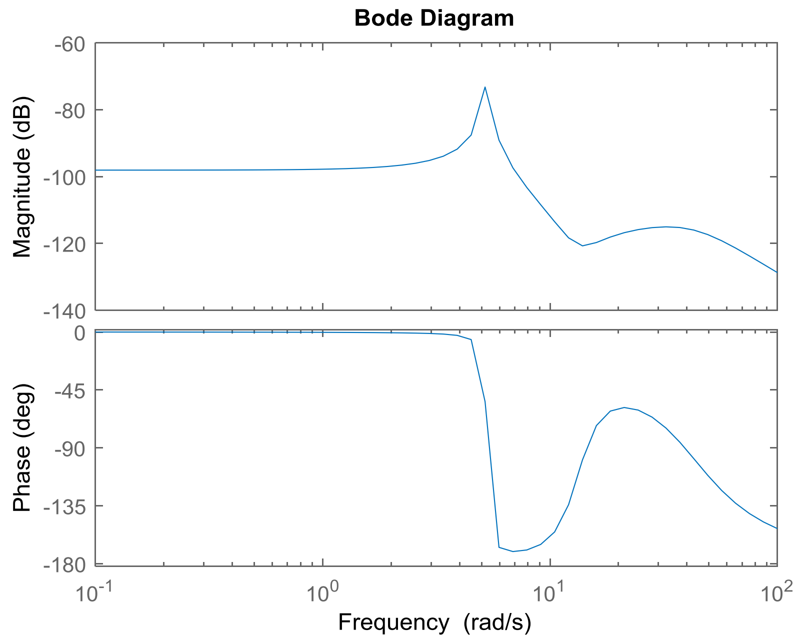
```
%% Frequency response
% Suggestion: help bode
w = logspace(-1,2);
figure(3)
bode(G1,w)
% Normalize scale based on former plot
K=100000;
figure(4)
bode(K*G1,w)
```

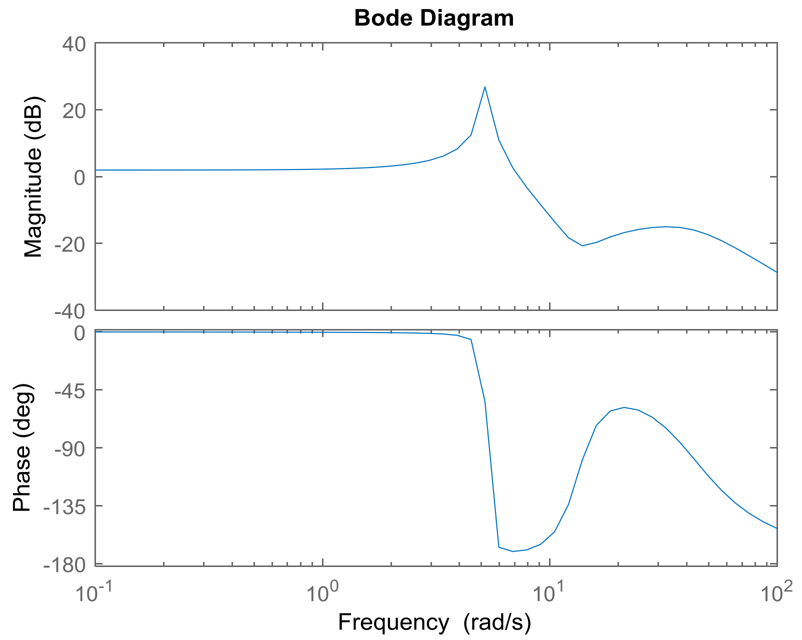
K is a normalization factor so that the left asymptote goes to 0.

$K = 10^5 \rightarrow$ shift up by $20 \cdot \log_{10} 10^5 = 100$ dB

Interpretation

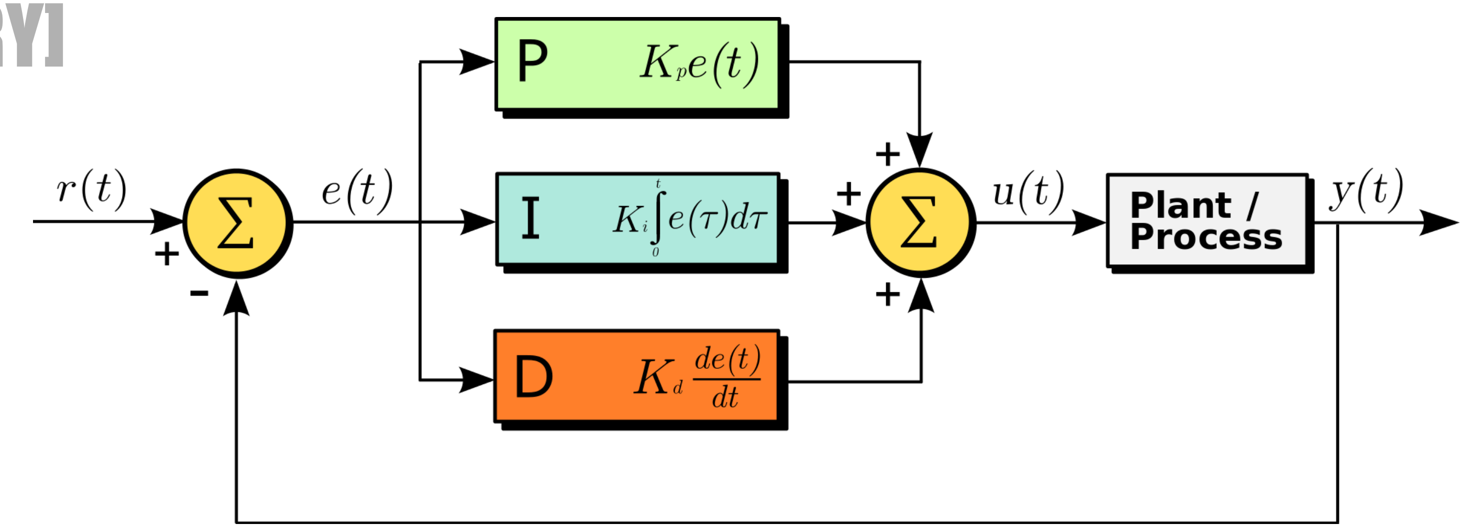
Up to 5 rad/s (~ 0.8Hz), output follows input





PID control

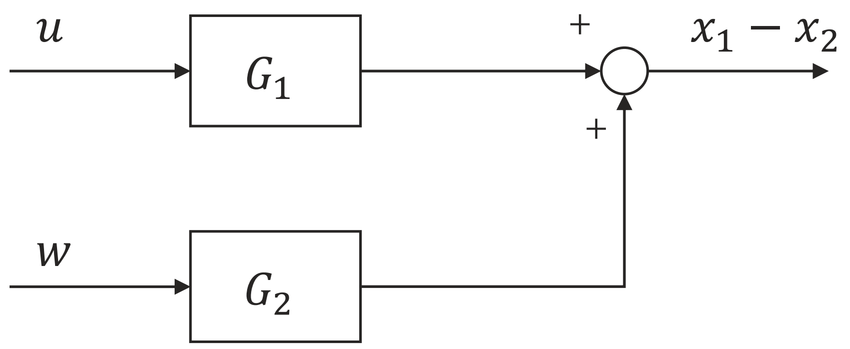
PID fundamentals [THEORY]



- P term: proportional to current error.
- I term: accounts for past values of the error, to eliminate residual error.
- D term: best estimate of the future error, to reduce overshoot.

$$u(t) = K_\mathrm{p} e(t) + K_\mathrm{i} \int_0^t e(t')\, dt' + K_\mathrm{d} \frac{de(t)}{dt}$$

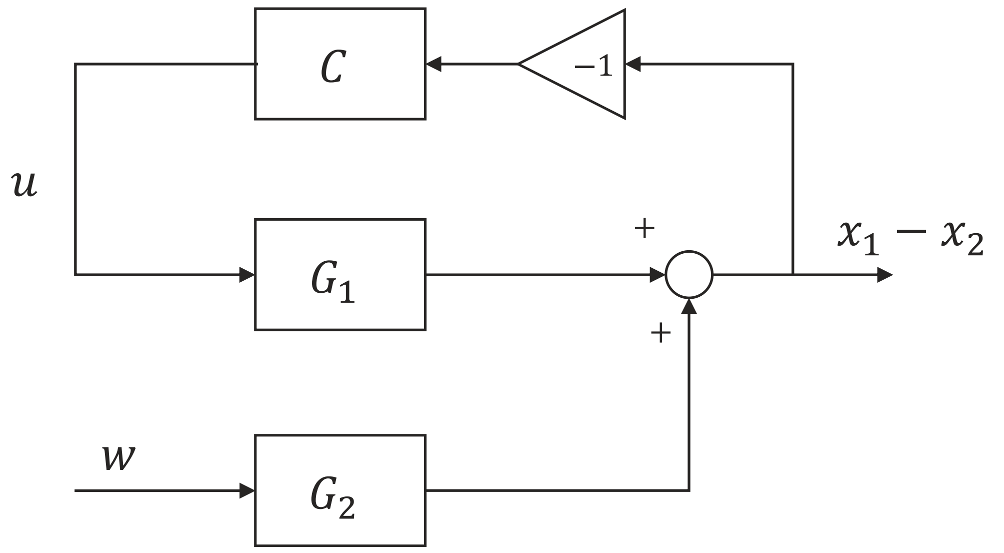
Controller structure
Open-loop



Controller structure

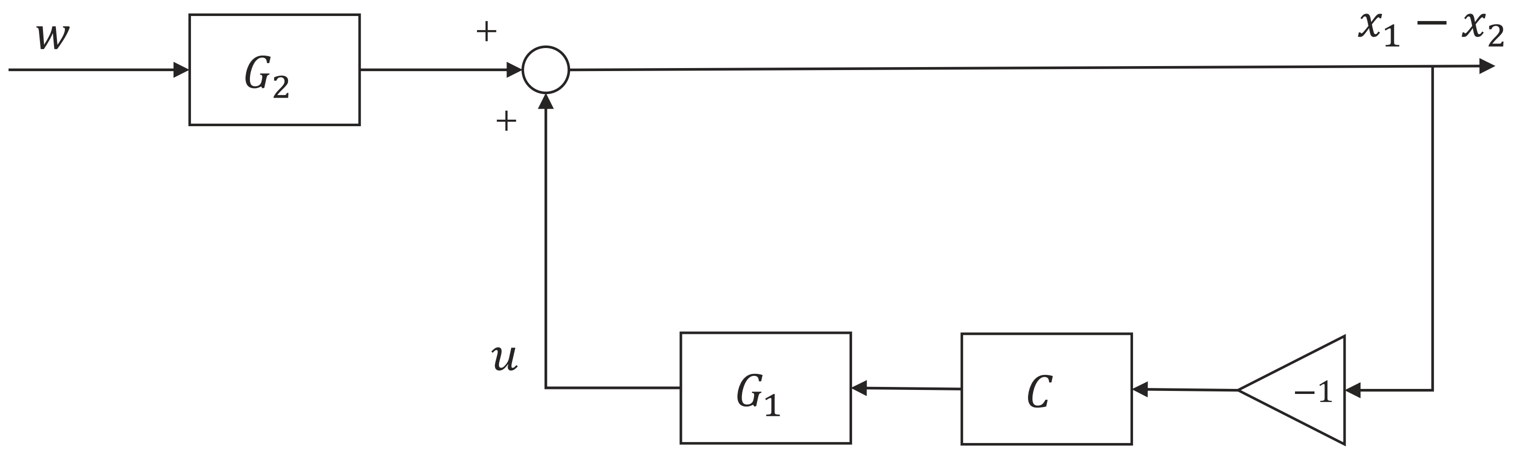
Controller location

We want a controller C to determine the control action u from the actuator deflection $x_1 - x_2$



Controller structure

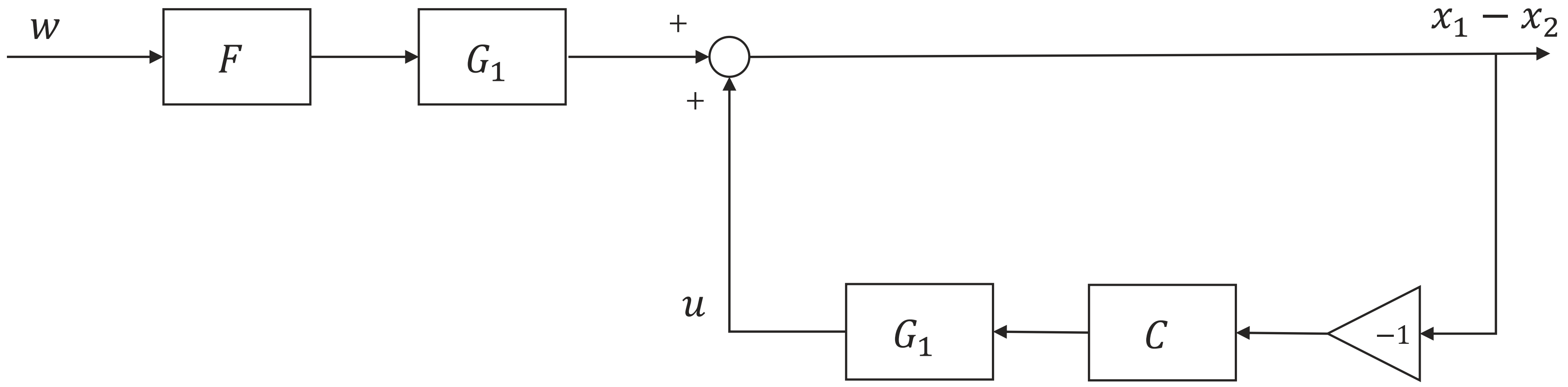
Rearranging



Controller structure

Redefine

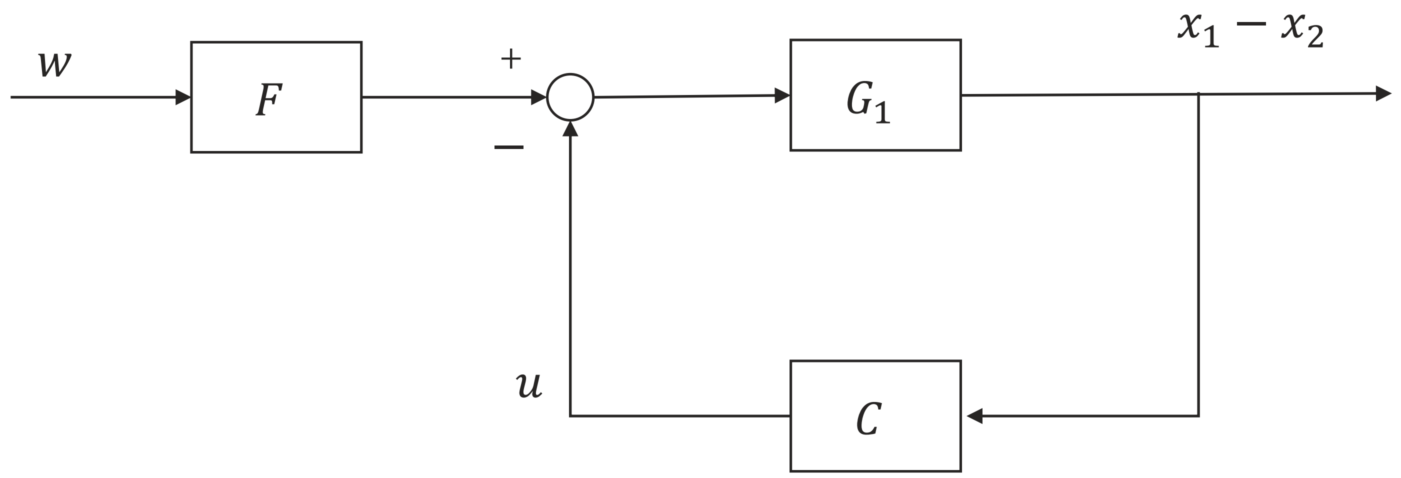
$$G_2 = F * G_1$$



Controller structure

Rearrange, absorb the -1

$F = G_2/G_1$



PID control

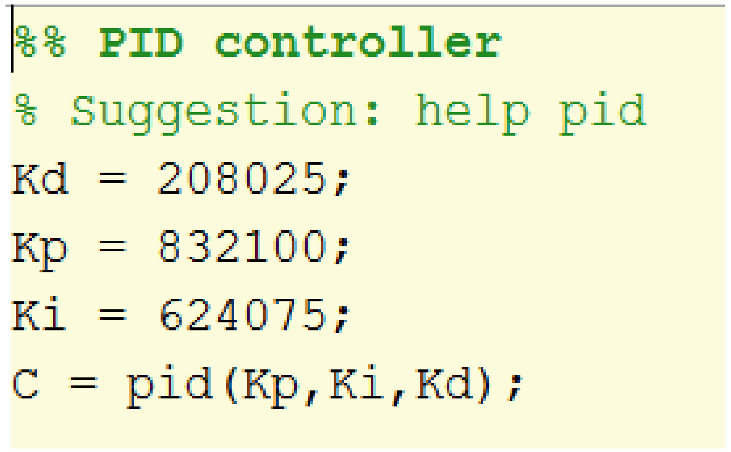
PID controller in Matlab

- **Challenge**: create a PID controller block with the following parameters

```
% Suggestion: help pid

Kd = 208025;

Kp = 832100;

Ki = 624075;
```

PID control

PID controller in Matlab [SOLUTION]

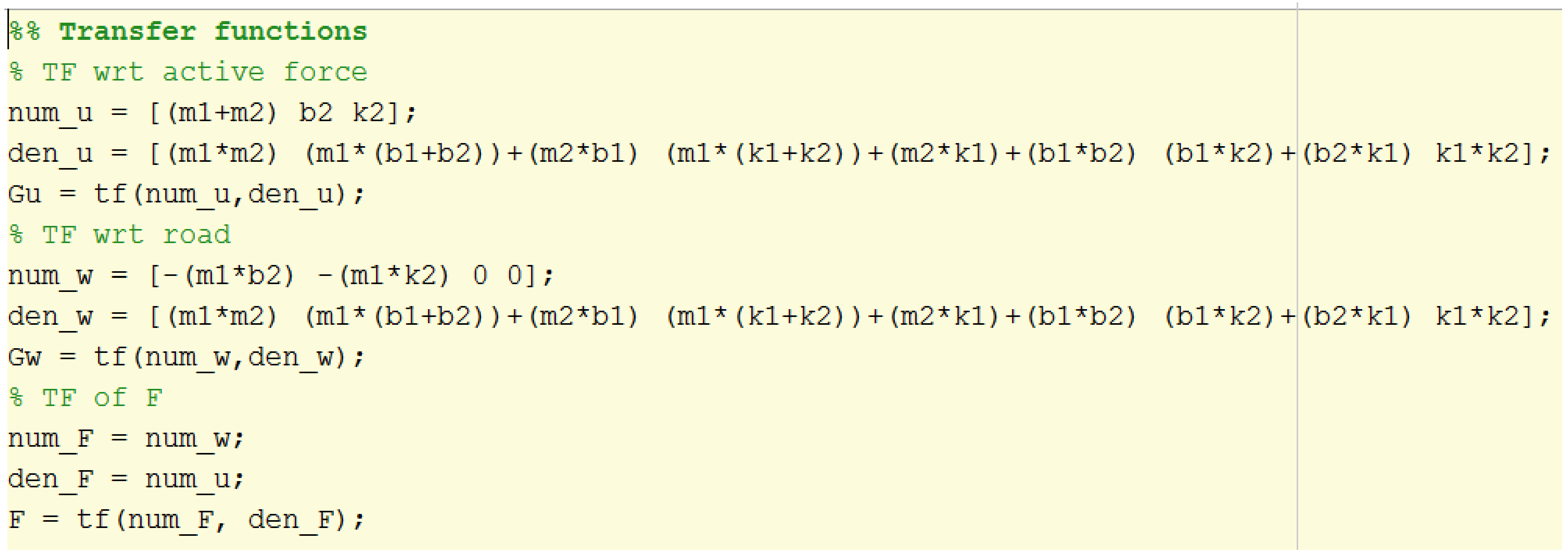


```
%% PID controller
% Suggestion: help pid
Kd = 208025;
Kp = 832100;
Ki = 624075;
C = pid(Kp,Ki,Kd);
```

PID control

Define F

- **Challenge [SOLVED]**: define the transfer function of F



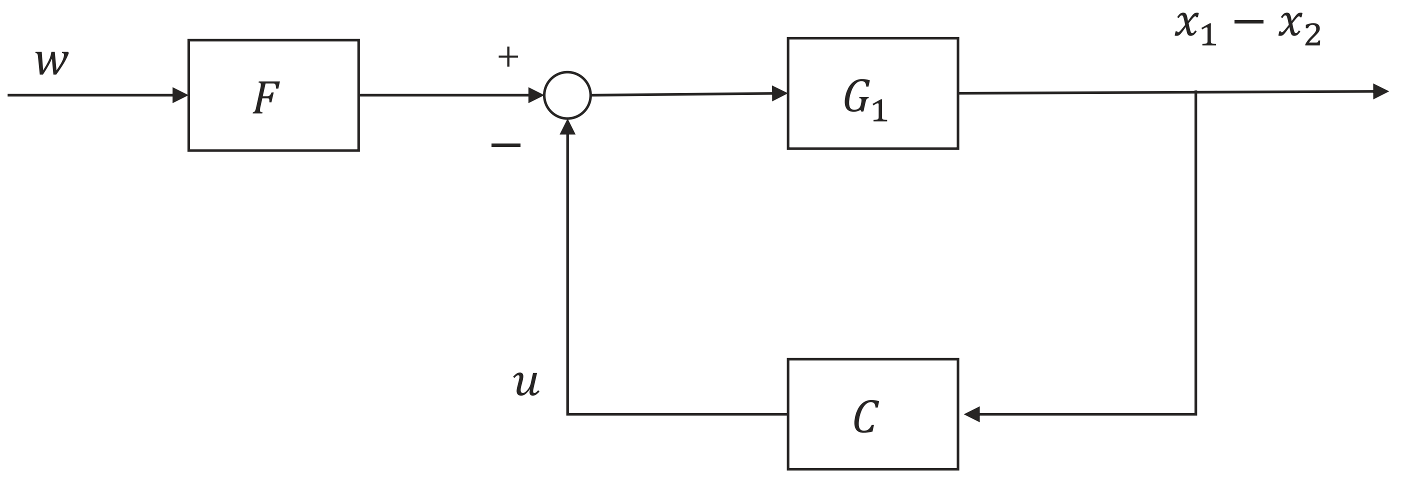
```
%% Transfer functions
% TF wrt active force
num_u = [(m1+m2) b2 k2];
den_u = [(m1*m2) (m1*(b1+b2))+(m2*b1) (m1*(k1+k2))+(m2*k1)+(b1*b2) (b1*k2)+(b2*k1) k1*k2];
Gu = tf(num_u,den_u);
% TF wrt road
num_w = [-(m1*b2) -(m1*k2) 0 0];
den_w = [(m1*m2) (m1*(b1+b2))+(m2*b1) (m1*(k1+k2))+(m2*k1)+(b1*b2) (b1*k2)+(b2*k1) k1*k2];
Gw = tf(num_w,den_w);
% TF of F
num_F = num_w;
den_F = num_u;
F = tf(num_F, den_F);
```

PID control

The feedback loop

- **Challenge**: define the closed-loop system

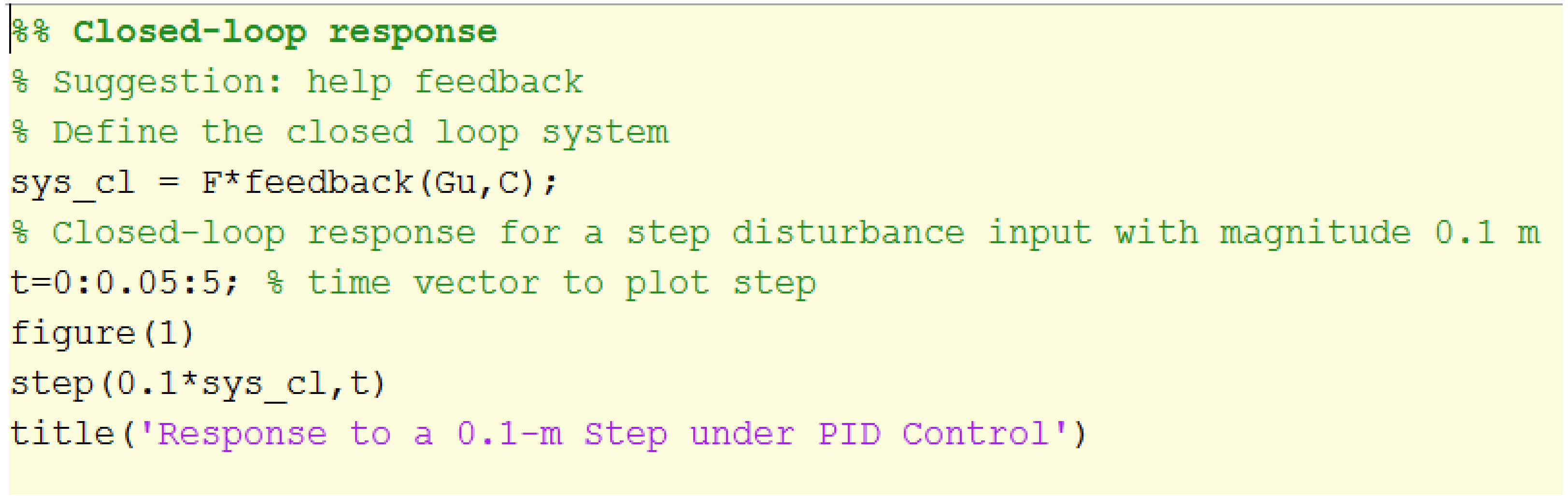
```
% Suggestion: help feedback
```



PID control

The feedback loop [SOLUTION]

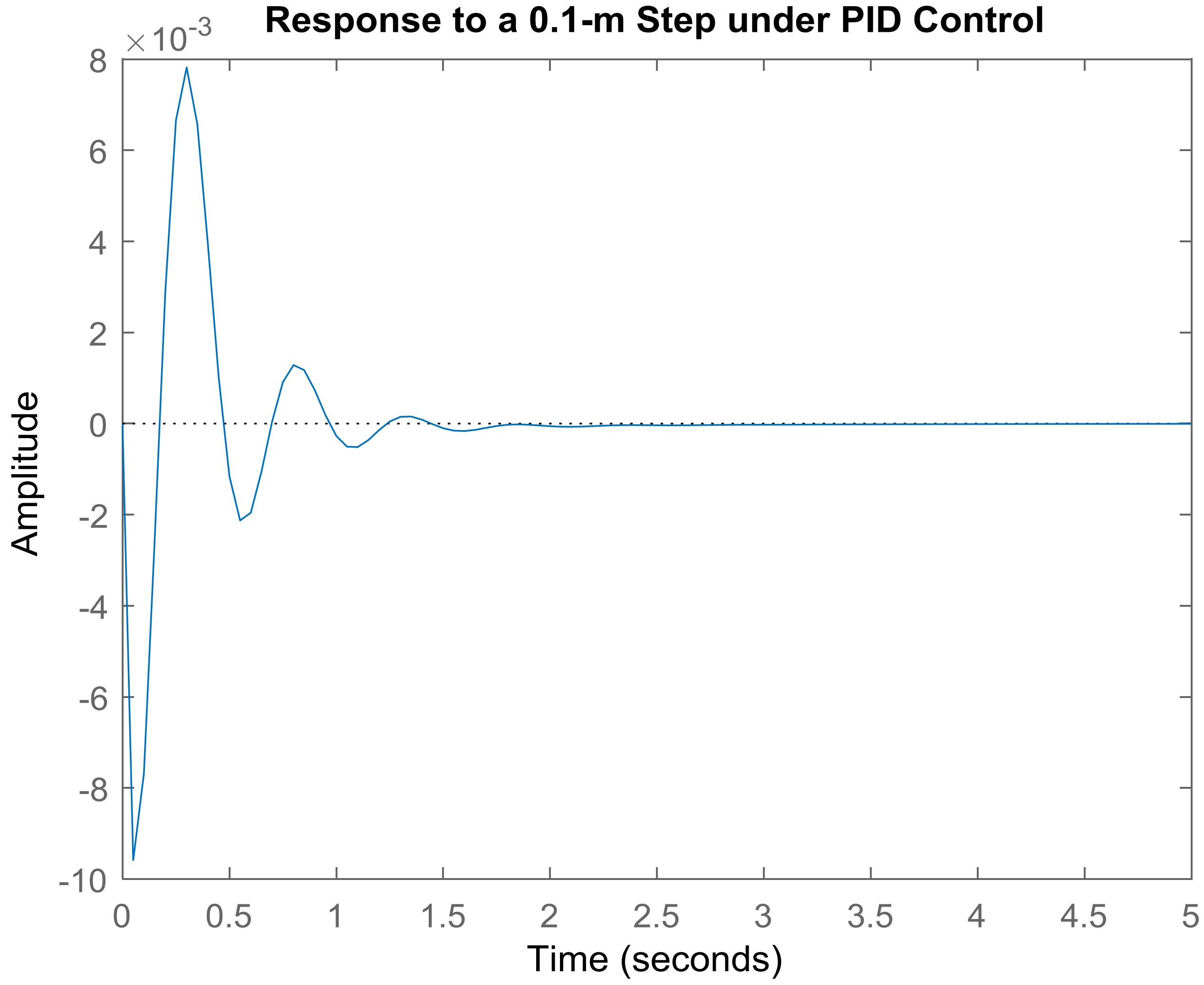
- **Challenge**: define the closed-loop system



```
%% Closed-loop response
% Suggestion: help feedback
% Define the closed loop system
sys_cl = F*feedback(Gu,C);
% Closed-loop response for a step disturbance input with magnitude 0.1 m
t=0:0.05:5; % time vector to plot step
figure(1)
step(0.1*sys_cl,t)
title('Response to a 0.1-m Step under PID Control')
```

PID control
Step response

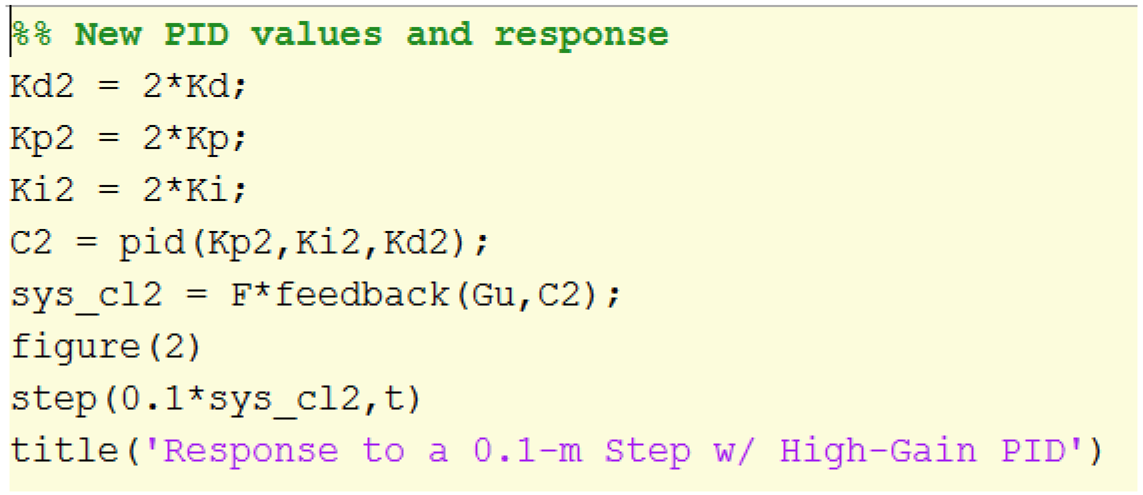
- Requirements:
  - Settling time < 10 s
  - Overshoot < 5mm



PID control

New controller

- Create a new PID block with all coefficients doubled:

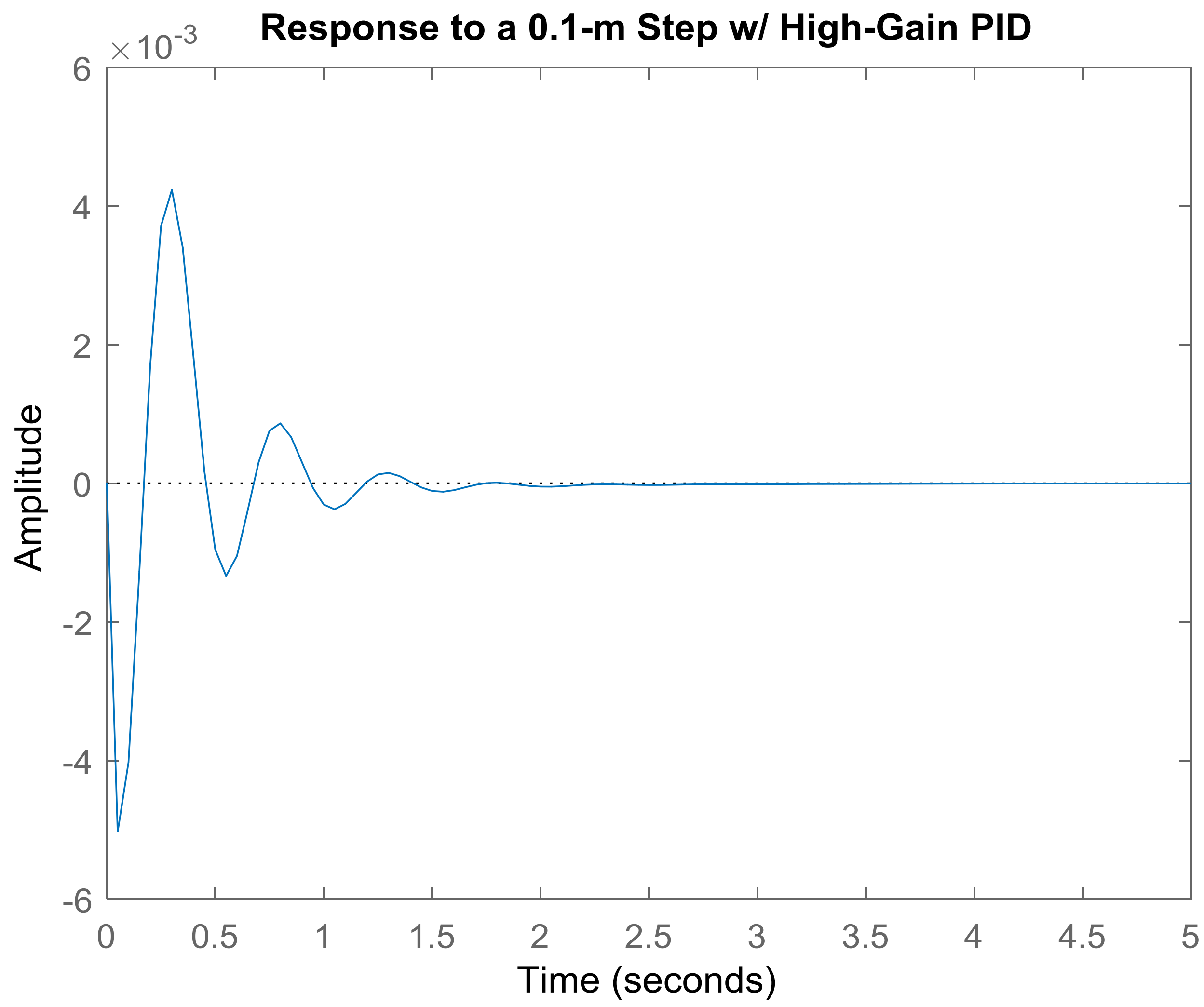


```
%% New PID values and response
Kd2 = 2*Kd;
Kp2 = 2*Kp;
Ki2 = 2*Ki;
C2 = pid(Kp2,Ki2,Kd2);
sys_cl2 = F*feedback(Gu,C2);
figure(2)
step(0.1*sys_cl2,t)
title('Response to a 0.1-m Step w/ High-Gain PID')
```

PID control

Step response

- Requirements:
  - **Settling time < 10 s**
  - **Overshoot < 5mm**



Simulink

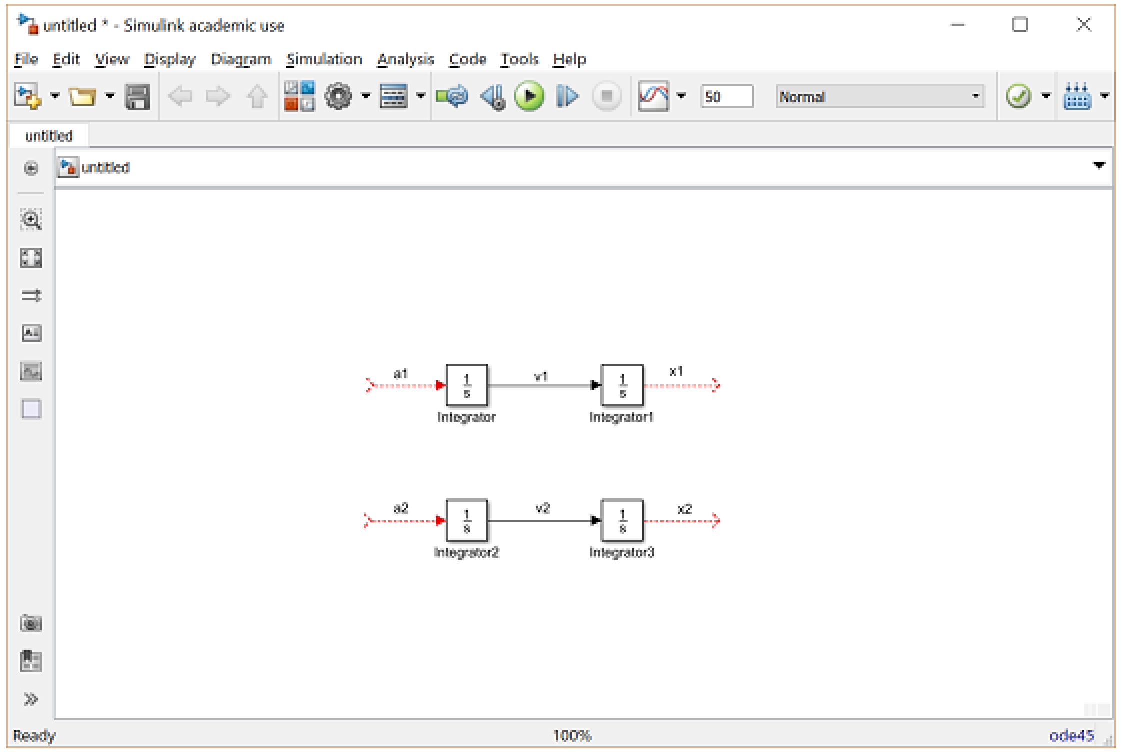
- Graphical tool to model and simulate dynamical systems.
- Advantages w.r.t transfer functions in Matlab:
  - No need to do analytical work, only ODE modeling.
  - Easier to simulate nonlinearities.
  - Easier to include initial conditions.
  - Easier to work with signals and control strategies.
- Some advise:
  - Sketch the design with pen and paper for a nice layout.
  - Start with the maximum derivatives.
- Easier to learn with this example!

Simulink

$$\int\int \frac{d^2x_1}{dt^2}\,dt = \int \frac{dx_1}{dt}\,dt = x_1$$

$$\int\int \frac{d^2x_2}{dt^2}\,dt = \int \frac{dx_2}{dt}\,dt = x_2$$

- Insert an Integrator block (from the Continuous library) and draw lines to and from its input and output terminals.
- Label the input line "a1" (for acceleration) and the output line "v1" (for velocity) To add such a label, double click in the empty space just above the line.
- Insert another Integrator block and connect it to the output of the first.
- Draw a line from its output and label it "x1" (for position).
- Insert a second pair of Integrators below the first with lines labeled "a2", "v2", and "x2".


untitled * - Simulink academic use
File Edit View Display Diagram Simulation Analysis Code Tools Help
50
Normal
untitled
untitled
a1
v1
x1
Integrator
Integrator1
a2
v2
x2
Integrator2
Integrator3
Ready
100%
ode45

Simulink

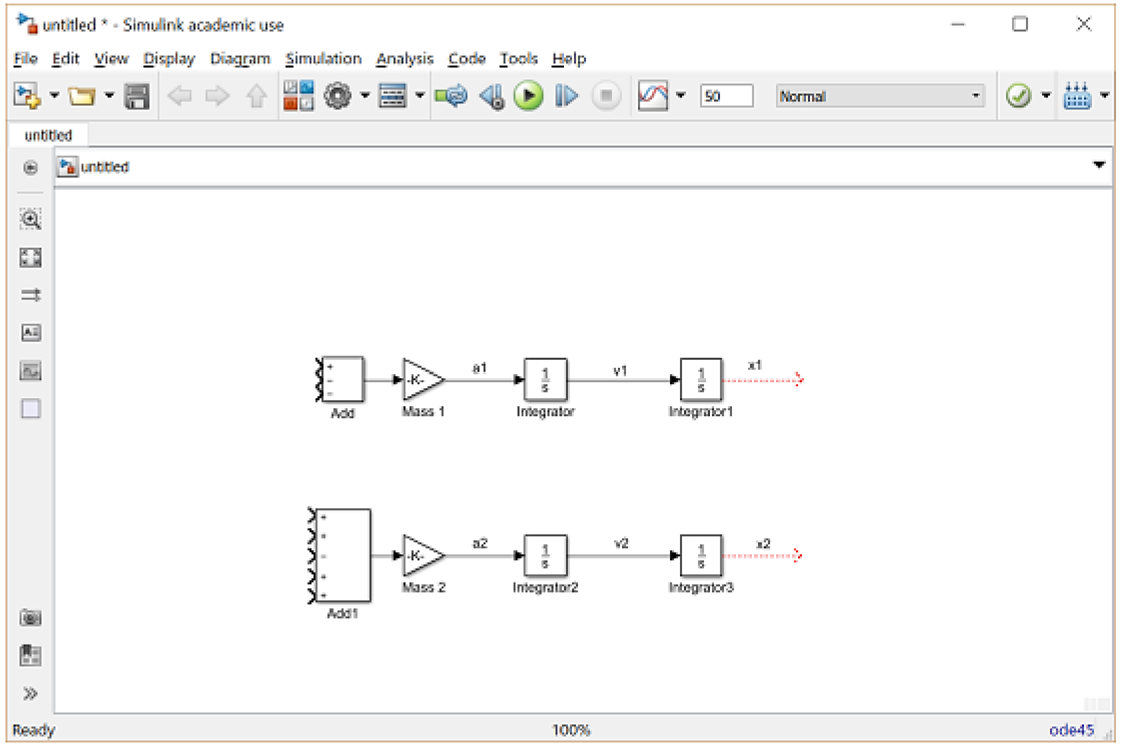
$$\frac{1}{M_1}\Sigma_1 F = \frac{d^2x_1}{dt^2}$$

$$\frac{1}{M_2}\Sigma_2 F = \frac{d^2x_2}{dt^2}$$

- Insert two Gain blocks, (from the Math Operations library) one attached to the inputs of each of the integrator pairs.
- Edit the gain block corresponding to M1 by double-clicking it and changing its value to "1/m1".
- Change the label of this Gain block to "Mass 1" by clicking on the word "Gain" underneath the block.
- Similarly, edit the other Gain's value to "1/m2" and it's label to "Mass 2". (You may want to resize the gain blocks to view the contents. To do this, single click on the block to highlight it, and drag one of the corners to the desired size.)

There are three forces acting on M1 (one spring, one damper, and the input, u) and five forces acting on M2 (two springs, two dampers, and the input, u).

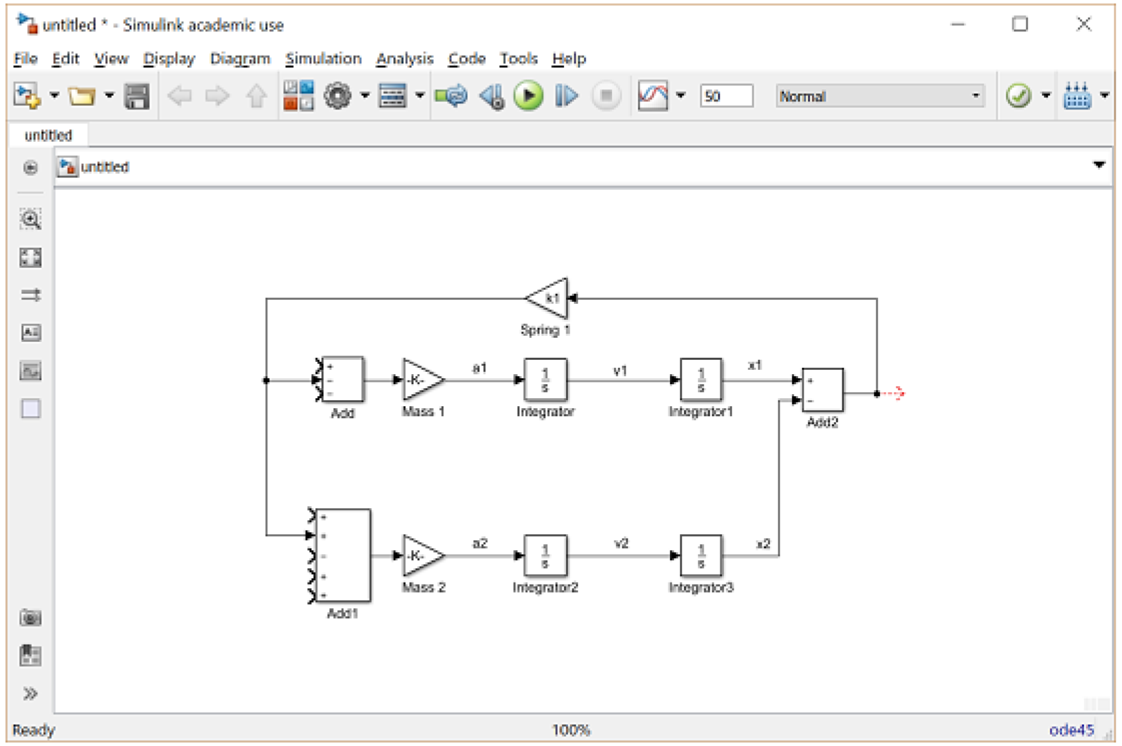
- Insert two Add blocks (from the Math Operations library), one attached by a line to each of the Gain blocks.
- Edit the signs of the Add block corresponding to M1 to "+--" to represent the three forces (two of which will be negative).
- Edit the signs of the other Add block to "++-++" to represent the five forces, one of which will be negative.



Simulink

Now, we will add in the forces acting on each mass. First, we will add in the force from Spring 1. This force is equal to a constant, k1 times the difference X1-X2.

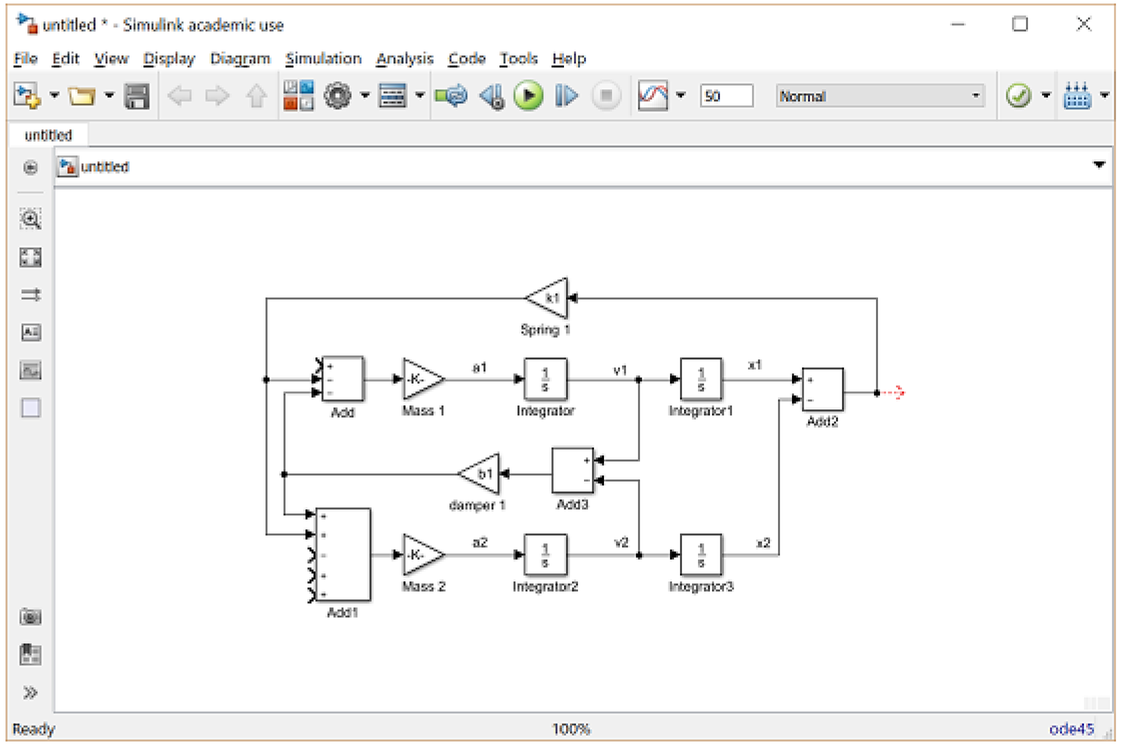
- Insert an Add block after the upper pair of integrators.
- Edit its signs to "+-" and connect the "x1" signal to the positive input and the "x2" signal to the negative input.
- Draw a line leading from the output of the Add block.
- Insert a Gain block above the "Mass 1" block.
- Flip it left-to-right by single-clicking on it and selecting **Flip Block** from the **Rotate & Flip** menu (or hit **Ctrl-I**).
- Edit the value of this gain to "k1" and label the block "Spring 1".
- Tap a line off the output of the last Add block and connect it to the input of this Gain block.
- Connect the output of this Gain block (the spring force) to the second input of the Mass 1 Add block. This input should be negative since the Spring 1 pulls down on Mass 1 when X1 > X2.
- Tap a line off the spring force line and connect it to the second input of the Mass 2 Add block. This input is positive since Spring 1 pulls up on Mass 2.



Simulink

Now, we will add in the force from Damper 1. This force is equal to b1 times V1-V2.

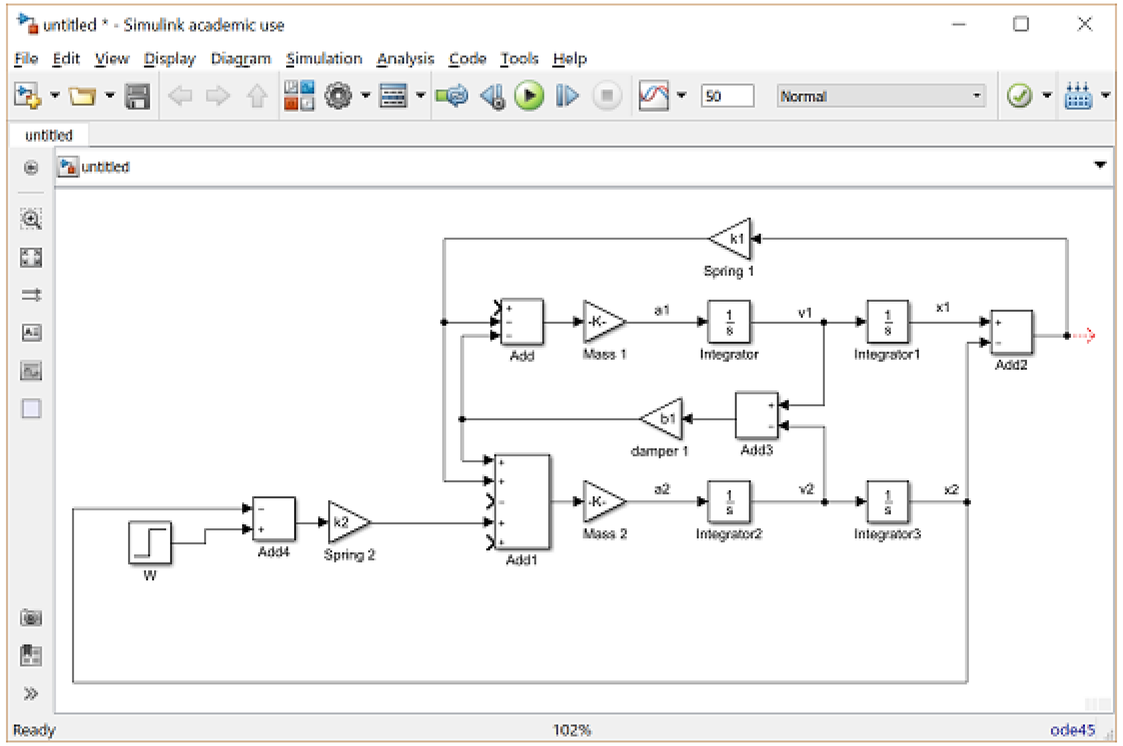
- Insert an Add block below the Mass 1's first integrator.
- Flip it left-to-right, and edit its signs to "+-".
- Tap a line off the "v1" line and connect it to the positive input of this Add block.
- Tap a line off the "v2" line and connect it to the negative input of this Add block.
- Insert a Gain block to the left of this Add block and flip it left-to-right.
- Edit its value to "b1" and label it "Damper 1".
- Connect the output of the new Add block to the input of this gain block.
- Connect the output of this gain block (the damper force) to the third input of the Mass 1 Add block. This input is negative, similar to Spring 1's force on Mass 1.
- Tap a line off Damper 1's force line and connect it to the first input (which is positive) of Mass 2's Add block.



Simulink

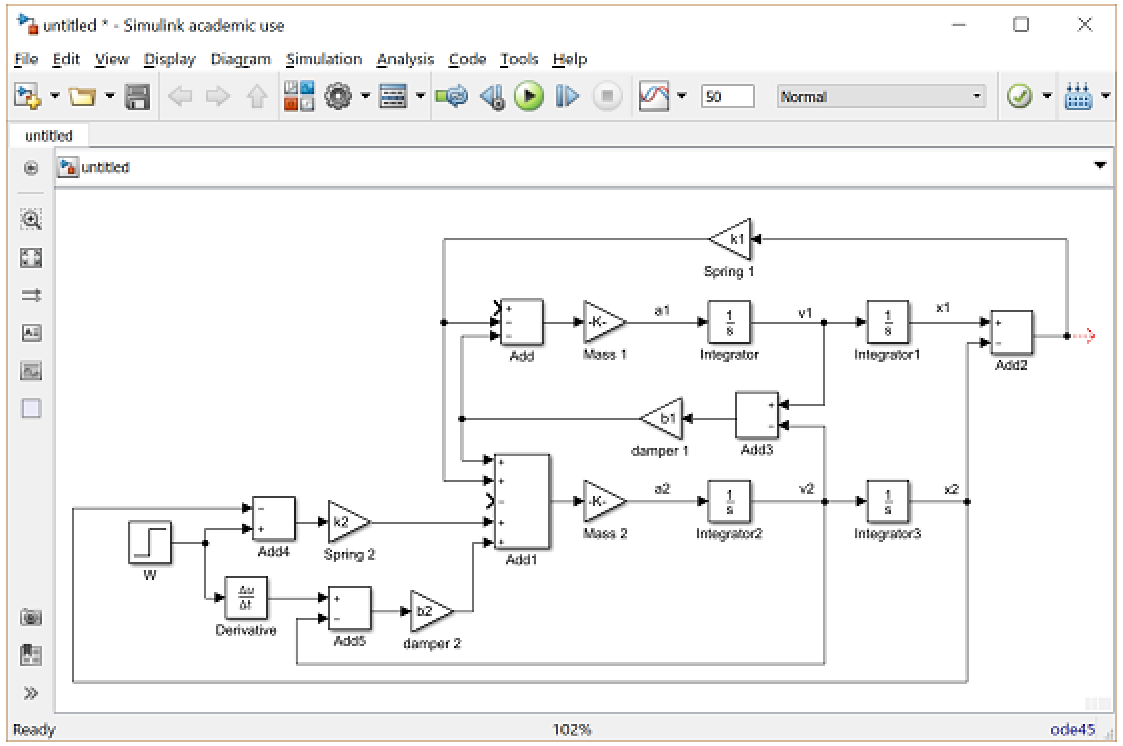
Now we will add in the force from Spring 2. This force acts only on Mass 2, but depends on the ground profile, W. Spring 2's force is equal to X2-W.

- Insert a Step block in the lower left area of your model window. Label it "W".
- Edit it's Step Time to "0" and it's Final Value to "0". (We will assume a flat road surface for now).
- Insert an Add block to the right of the W Step block and edit its signs to "-+".
- Connect the output of the Step block to the positive input of this Add block.
- Tap a line off the "x2" signal and connect it to the negative input of the new Add block.
- Insert a Gain block to the right of this Add block and connect the Add's output to the new Gain's input.
- Change the value of the gain to "k2" and label it "Spring 2".
- Connect the output of this block (Spring 2's force) to the fourth input of Mass 2's Add block. This force adds in in the positive sense.



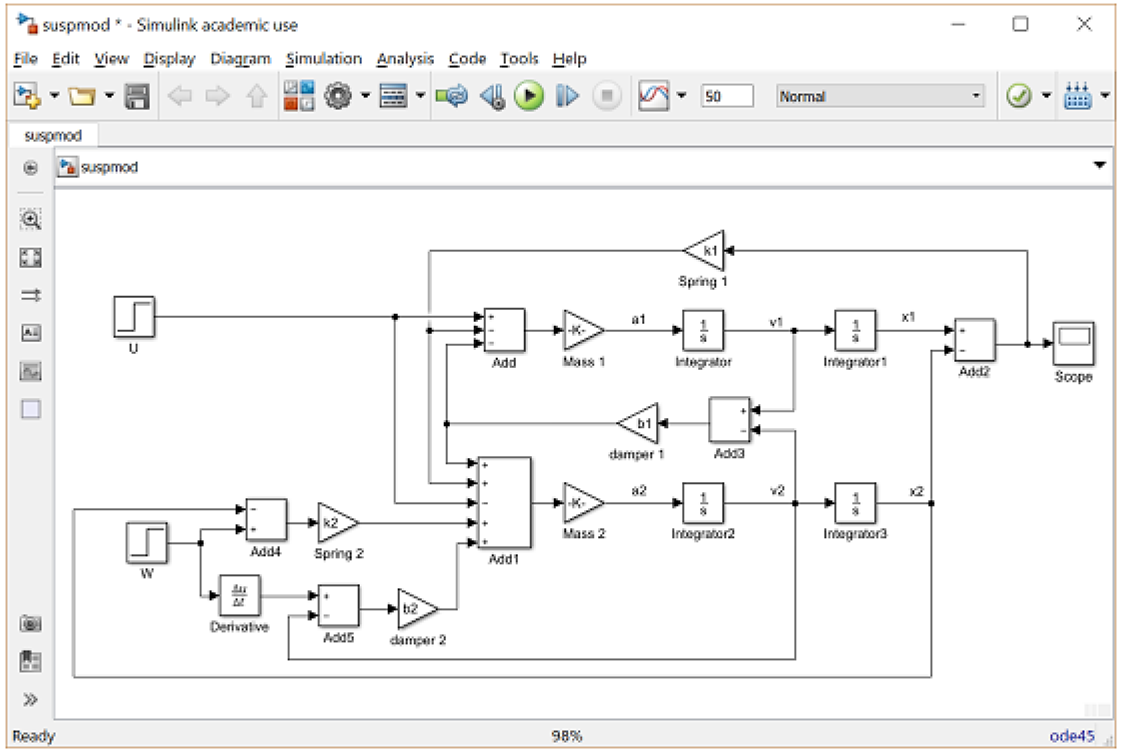
Simulink

- Next, we will add in the force from Damper 2. This force is equal to b2 times V2-d/dt(W). Since there is no existing signal representing the derivative of W we will need to generate this signal.
- Insert a Derivative block (from the Continuous library) to the right of the W step block.
- Tap a line of the Step's output and connect it to the input of the Derivative block.
- Insert an Add block after the Derivative block and edit it's signs to "+-".
- Connect the Derivative's output to the positive input of the new Add block.
- Tap a line off the "v2" line and connect it to the negative input of this Add block.
- Connect the output of this Add block (Damper 2's force) to the fifth input of Mass 2's Add block. This force also adds in with positive sign.



Simulink

- The last force is the input U acting between the two masses.
- Insert a Step block (from the Sources library) in the upper left of the model window.
- Connect its output to the remaining input of Mass 1's Add block (with positive sign).
- Tap a line off this signal and connect it to the remaining input of Mass 2's Add block (with negative sign).
- Edit this Step block's Step Time to "0" and leave its Final Value "1".
- Label this Step block "U".
- Finally, to view the output (X1-X2) insert a Scope block (from the Sinks library) and connect it to the output of the rightmost Add block.

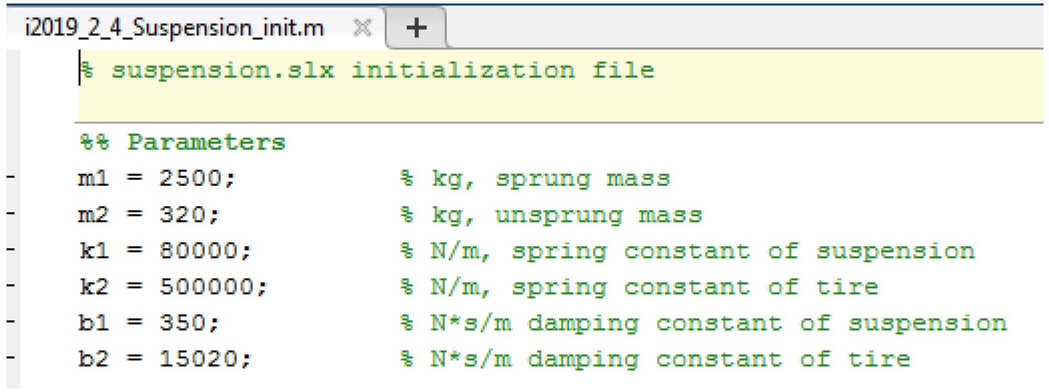


Simulink

Configuration parameters and initialization file

- Change configuration parameters
  - Simulation → Model Configuration Parameters → Stop Time = 50

- Create and run initialization file to define parameters in the workspace

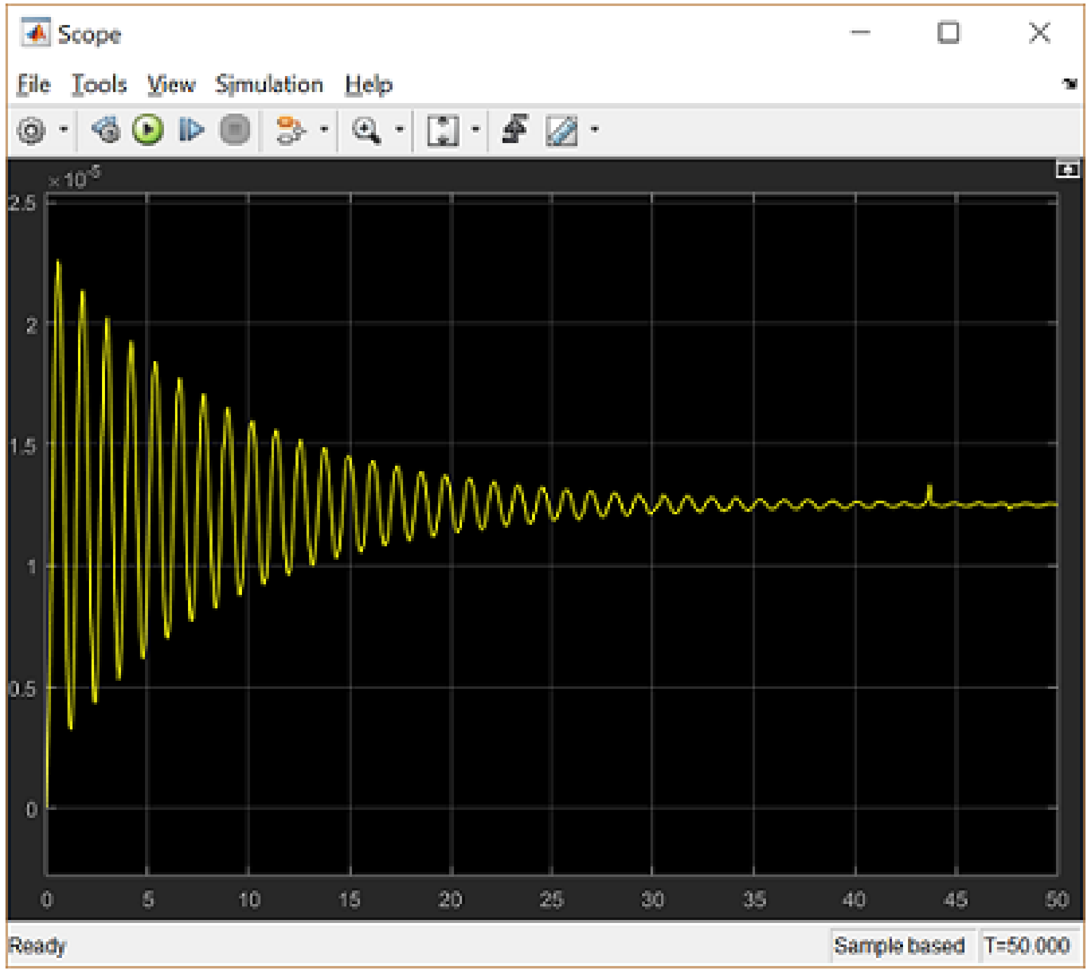


i2019_2_4_Suspension_init.m

```
% suspension.slx initialization file

%% Parameters
m1 = 2500;          % kg, sprung mass
m2 = 320;           % kg, unsprung mass
k1 = 80000;         % N/m, spring constant of suspension
k2 = 500000;        % N/m, spring constant of tire
b1 = 350;           % N*s/m damping constant of suspension
b2 = 15020;         % N*s/m damping constant of tire
```

Simulink

Open-loop response


Scope
File Tools View Simulation Help
×10^-5
Ready
Sample based
T=50.000