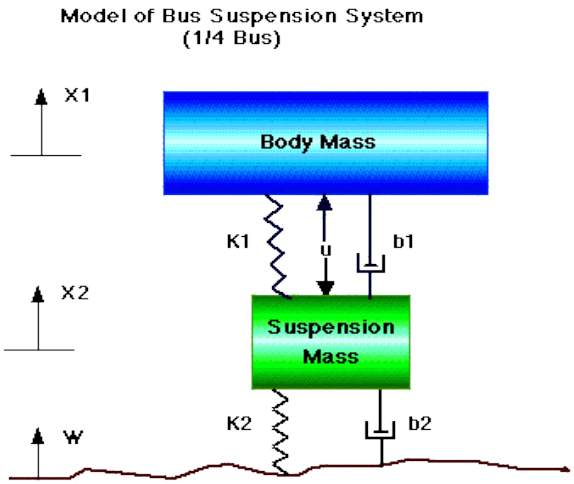
Example: Modeling a Bus Suspension System using Transfer Function

Zidarta - jimfackyou@hotmail.com

Physical setup

Photo courtesy: Eisenhower Center

Designing an automatic suspension system for a bus turns out to be an interesting control problem. When the suspension system is designed, a 1/4 bus model (one of the four wheels) is used to simplify the problem to a one dimensional spring-damper system. A diagram of this system is shown below:



Where:

* body mass (m1) = 2500 kg,
* suspension mass (m2) = 320 kg,
* spring constant of suspension system(k1) = 80,000 N/m,
* spring constant of wheel and tire(k2) = 500,000 N/m,
* damping constant of suspension system(b1) = 350 Ns/m.
* damping constant of wheel and tire(b2) = 15,020 Ns/m.
* control force (u) = force from the controller we are going to design.

Design requirements:

A good bus suspension system should have satisfactory road holding ability, while still providing comfort when riding over bumps and holes in the road. When the bus is experiencing any road disturbance (i.e. pot holes, cracks, and uneven pavement),the bus body should not have large oscillations, and the oscillations should dissipate quickly. Since the distance **X1-W** is very difficult to measure, and the deformation of the tire (**X2-W**) is negligible, we will use the distance **X1-X2** instead of **X1-W** as the output in our problem. Keep in mind that this is an estimation.

The road disturbance (**W**) in this problem will be simulated by a step input. This step could represent the bus coming out of a pothole. We want to design a feedback controller so that the output (**X1-X2**) has an overshoot less than 5% and a settling time shorter than 5 seconds. For example, when the bus runs onto a 10 cm high step, the bus body will oscillate within a range of +/- 5 mm and return to a smooth ride within 5 seconds.

Equations of motion:

From the picture above and Newton's law, we can obtain the dynamic equations as the following:

$$M_1\ddot{X}_1 = -b_1(\dot{X}_1 - \dot{X}_2) - K_1(X_1 - X_2) + U$$

$$M_2\ddot{X}_2 = b_1(\dot{X}_1 - \dot{X}_2) + K_1(X_1 - X_2) + b_2(\dot{W} - \dot{X}_2) + K_2(W - X_2) - U$$

Transfer Function Equation:

Assume that all of the initial condition are zeroes, so these equations represent the situation when the bus's wheel go up a bump. The dynamic equations above can be expressed in a form of transfer functions by taking

Laplace Transform of the above equations. The derivation from above equations of the Transfer Functions **G1(s)** and **G2(s)** of output,**X1-X2**, and two inputs,**U** and **W**, are as follows.

$$(M_1 s^2 + b_1 s + K_1)X_1(s) - (b_1 s + k_1)X_2(s) = U(s)$$

$$-(b_1 s + K_1)X_1(s) + (M_2 s^2 + (b_1 + b_2)s + (K_1 + K_2))X_2(s) = (b_2 s + K_2)W(s) - U(s)$$

$$\begin{bmatrix} (M_1 s^2 + b_1 s + K_1) & -(b_1 s + K_1) \\ -(b_1 s + K_1) & (M_2 s^2 + (b_1 + b_2)s + (K_1 + K_2)) \end{bmatrix} \begin{bmatrix} X_1(s)) \\ X_2(s) \end{bmatrix} = \begin{bmatrix} U(s) \\ (b_2 s + K_2)W(s) - U(s) \end{bmatrix}$$

$$A = \begin{bmatrix} (M_1 s^2 + b_1 s + K_1) & -(b_1 s + K_1) \\ -(b_1 s + K_1) & (M_2 s^2 + (b_1 + b_2)s + (K_1 + K_2)) \end{bmatrix}$$

$$\Delta = \det \begin{bmatrix} (M_1 s^2 + b_1 s + K_1) & -(b_1 s + K_1) \\ -(b_1 s + K_1) & (M_2 s^2 + (b_1 + b_2)s + (K_1 + K_2)) \end{bmatrix}$$

$$\text{or, } \Delta = (M_1 s^2 + b_1 s + K_1) \cdot (M_2 s^2 + (b_1 + b_2)s + (K_1 + K_2)) - (b_1 s + K_1) \cdot (b_1 s + K_1)$$

Find the inverse of matrix A and then multiple with inputs U(s)and W(s) on the right hand side as the following:

$$\begin{bmatrix} X_1(s) \\ X_2(s) \end{bmatrix} = \frac{1}{\Delta} \begin{bmatrix} (M_2 s^2 + (b_1 + b_2)s + (K_1 + K_2)) & (b_1 s + K_1) \\ (b_1 s + K_1) & (M_1 s^2 + b_1 s + K_1) \end{bmatrix} \begin{bmatrix} U(s) \\ (b_2 s + K_2)W(s) - U(s) \end{bmatrix}$$

$$\begin{bmatrix} X_1(s) \\ X_2(s) \end{bmatrix} = \frac{1}{\Delta} \begin{bmatrix} (M_2 s^2 + b_2 s + K_2) & \left(b_1 b_2 s^2 + (b_1 K_2 + b_2 K_1)s + K_1 K_2\right) \\ -M_1 s^2 & \left(M_1 b_2 s^3 + (M_1 K_2 + b_1 b_2)s^2 + (b_1 K_2 + b_2 K_1)s + K_1 K_2\right) \end{bmatrix} \begin{bmatrix} U(s) \\ W(s) \end{bmatrix}$$

When we want to consider input **U(s)** only, we set **W(s)** = 0. Thus we get the transfer function **G1(s)** as the following:

$$G_1(s) = \frac{X_1(s) - X_2(s)}{U(s)} = \frac{(M_1 + M_2)s^2 + b_2 s + K_2}{\Delta}$$

When we want to consider input **W(s)** only, we set **U(s)** = 0. Thus we get the transfer function **G2(s)** as the following:

$$G_2(s) = \frac{X_1(s) - X_2(s)}{W(s)} = \frac{-M_1 b_2 s^3 - M_1 K_2 s^2}{\Delta}$$

Also we can express and derive the above equations in state-space form. Even though this approach will express the first two equations above in the standard form of matrix, it will simplify the transfer function without going through any algebra, because we can use a function ss2tf to transform from state-space form to transfer function form for both inputs

Entering equations into Matlab

We can put the above Transfer Function equations into Matlab by defining the numerator and denominator of Transfer Functions in the form, **nump/denp** for actuated force input and **num1/den1** for disturbance input, of the standard transfer function **G1(s)** and **G2(s)**:

G1(s) = nump/denp
G2(s) = num1/den1

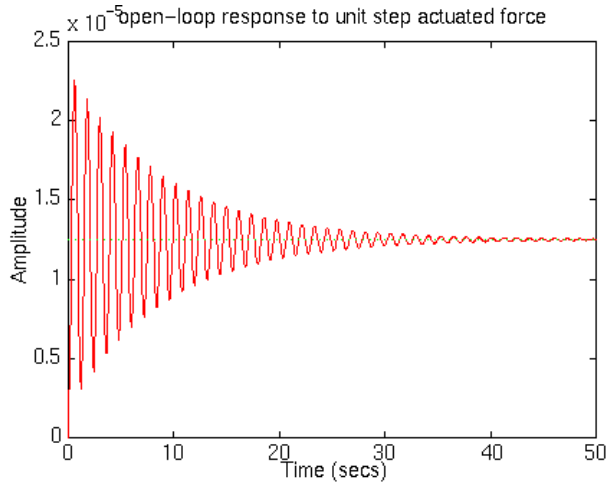
Now, let's create a new m-file and enter the following code:

```
m1=2500;
m2=320;
k1=80000;
k2=500000;
b1 = 350;
b2 = 15020;

nump=[(m1+m2) b2 k2];
denp=[(m1*m2) (m1*(b1+b2))+(m2*b1) (m1*(k1+k2))+(m2*k1)+(b1*b2) (b1*k2)+(b2*k1)
k1*k2];
'G(s)1'
printsys(nump,denp)

num1=[-(m1*b2) -(m1*k2) 0 0];
den1=[(m1*m2) (m1*(b1+b2))+(m2*b1) (m1*(k1+k2))+(m2*k1)+(b1*b2) (b1*k2)+(b2*k1)
k1*k2];
'G(s)2'
printsys(0.1*num1,den1)
```

Open-loop response

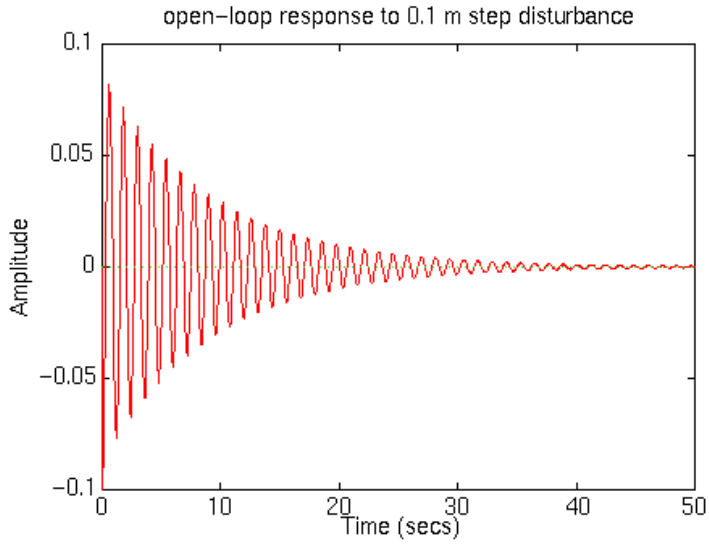
We can use Matlab to display how the original open-loop system performs (without any feedback control). Add the following commands into the m-file and run it in the Matlab command window to see the response of unit step actuated force input and unit step disturbance input.Note that the step command will generate the unit step inputs for each input.

```
step(nump,denp)
```



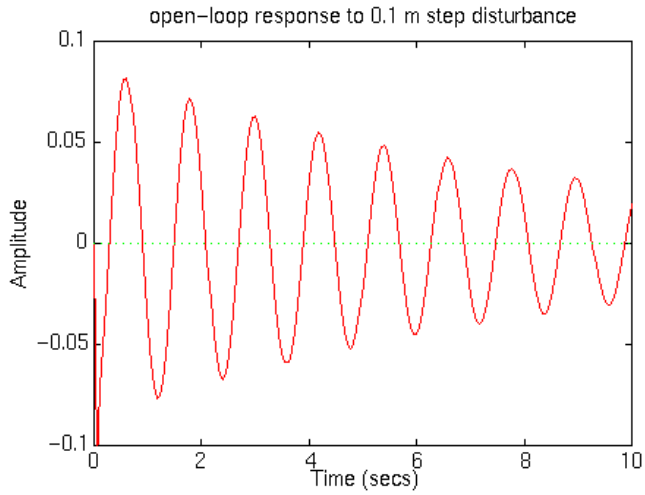
From this graph of the open-loop response for a unit step actuated force, we can see that the system is under-damped. People sitting in the bus will feel very small amount of oscillation and the steady-state error is about 0.013 mm. Moreover, the bus takes very unacceptably long time for it to reach the steady state or the settling time is very large. The solution to this problem is to add a feedback controller into the system's block diagram.

```
step(0.1*num1,den1)
```

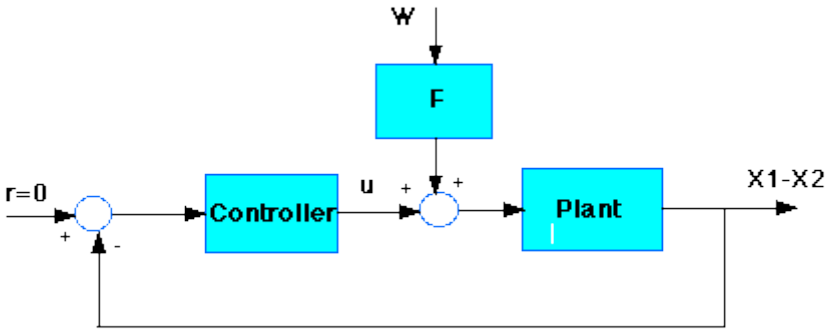


To see some details, you can change the axis:

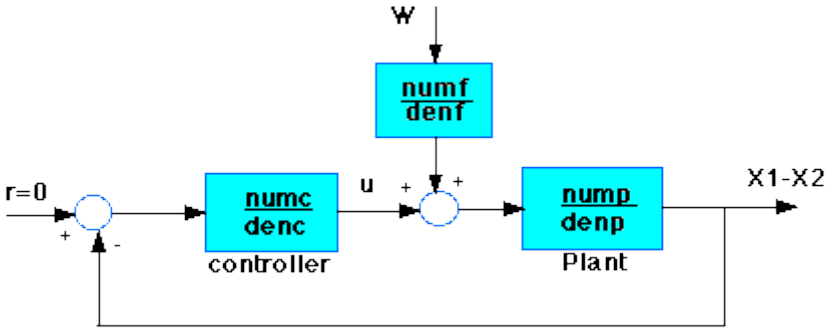
```
axis([0 10 -.1 .1])
```



From this graph of open-loop response for 0.1 m step disturbance, we can see that when the bus passes a 10 cm high bump on the road, the bus body will oscillate for an unacceptably long time(100 seconds) with larger amplitude, 13 cm, than the initial impact. People sitting in the bus will not be comfortable with such an oscillation. The big overshoot (from the impact itself) and the slow settling time will cause damage to the suspension system. The solution to this problem is to add a feedback controller into the system to improve the performance. The schematic of the closed-loop system is the following:



From the above transfer functions and schematic, we can draw the bus-system block diagram as the following:



From the schematic above we see that:

Plant = nump/denp

F * Plant=num1/den1

so that

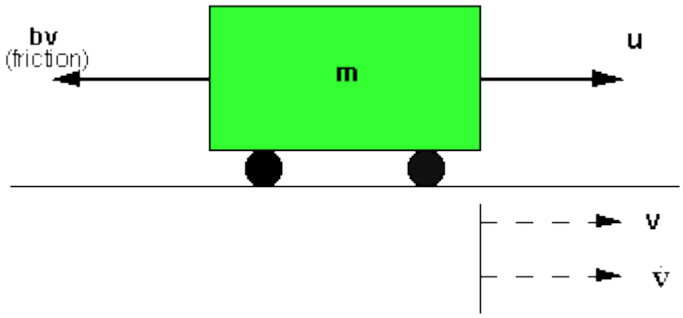
F=num1/(den1*Plant)

$$F = \frac{num1 \cdot denp}{den1 \cdot nump} = \frac{num1}{nump} = \frac{numf}{denf}$$

Example: Modeling a Cruise Control System

Physical setup and system equations

The model of the cruise control system is relatively simple. If the inertia of the wheels is neglected, and it is assumed that friction (which is proportional to the car's speed) is what is opposing the motion of the car, then the problem is reduced to the simple mass and damper system shown below.



Using Newton's law, modeling equations for this system becomes:

$$m\dot{v} + bv = u$$
$$y = v \qquad (1)$$

where u is the force from the engine. For this example, let's assume that

m = 1000kg
b = 50Nsec/m
u = 500N

Design requirements

The next step in modeling this system is to come up with some design criteria. When the engine gives a 500 Newton force, the car will reach a maximum velocity of 10 m/s (22 mph). An automobile should be able to accelerate up to that speed in less than 5 seconds. Since this is only a cruise control system, a 10% overshoot on the velocity will not do much damage. A 2% steady-state error is also acceptable for the same reason.

Keeping the above in mind, we have proposed the following design criteria for this problem:

Rise time < 5 sec
Overshoot < 10%
Steady state error < 2%

Matlab representation

1. Transfer Function

To find the transfer function of the above system, we need to take the Laplace transform of the modeling equations (1). **When finding the transfer function, zero initial conditions must be assumed.** Laplace transforms of the two equations are shown below.

$$msV(s) + bV(s) = U(s)$$
$$Y(s) = V(s)$$

Since our output is the velocity, let's substitute V(s) in terms of Y(s)

$$msY(s) + bY(s) = U(s)$$

The transfer function of the system becomes

$$\frac{Y(s)}{U(s)} = \frac{1}{ms+b}$$

To solve this problem using Matlab, copy the following commands into an new m-file:

```
m=1000;
b=50;
u=500;
num=[1];
den=[m b];
```

These commands will later be used to find the open-loop response of the system to a step input. But before getting into that, let's take a look at another representation, the state-space.

2. State-Space

We can rewrite the first-order modeling equation (1) as the state-space model.

$$[\dot{v}] = \left[-b/m\right][v] + \left[1/m\right][u]$$
$$y = [1][v]$$

To use Matlab to solve this problem, create an new m-file and copy the following commands:

```
m = 1000;
b = 50;
u = 500;
A = [-b/m];
B = [1/m];
C = [1];
D = 0;
```

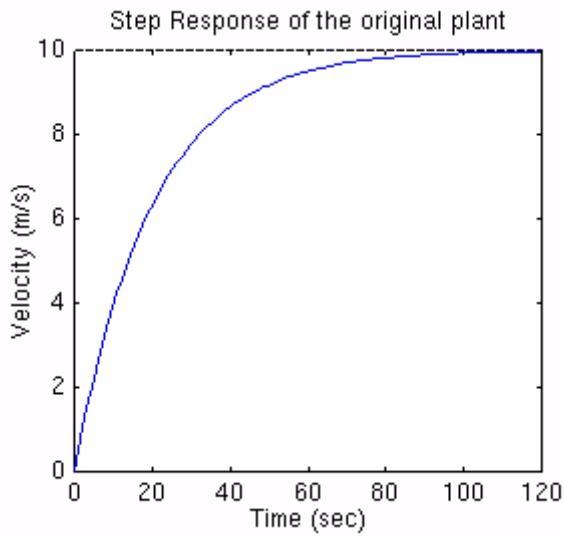
Note: It is possible to convert from the state-space representation to the transfer function or vise versa using Matlab. To learn more about the conversion, click Conversion

Open-loop response

Now let's see how the open-loop system responds to a step input. Add the following command onto the end of the m-file written for the tranfer function (the m-file with num and den matrices) and run it in the Matlab command window:

```
step (u*num,den)
```

You should get the following plot:



To use the m-file written for the state-space (the m-file with A, B, C, D matrices), add the following command at the end of the m-file and run it in the Matlab command window:

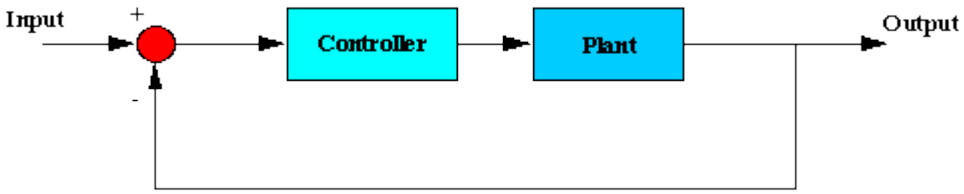
```
step (A,u*B,C,D)
```

You should get the same plot as the one shown above.

From the plot, we see that the vehicle takes more than 100 seconds to reach the steady-state speed of 10 m/s. This does not satisfy our rise time criterion of less than 5 seconds.

Closed-loop transfer function

To solve this problem, a unity feedback controller will be added to improve the system performance. The figure shown below is the block diagram of a typical unity feedback system.

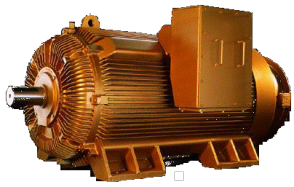


The transfer function in the plant is the transfer function derived above {Y(s)/U(s)=1/ms+b}. The controller will to be designed to satisfy all design criteria. Four different methods to design the controller are listed at the bottom of this page. You may choose on PID, Root-locus, Frequency response, or State-space.

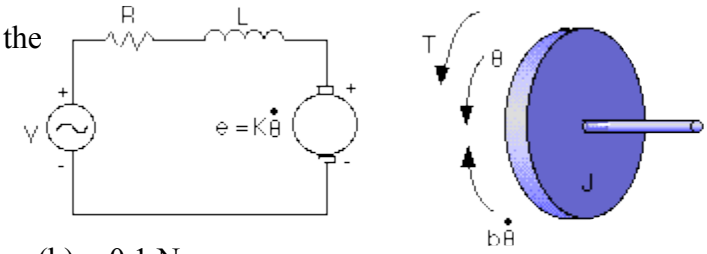
Example: DC Motor Speed Modeling

Physical setup and system equations

Photo courtesy: Pope Electric Motors Pty Limited



A common actuator in control systems is the DC motor. It directly provides rotary motion and, coupled with wheels or drums and cables, can provide transitional motion. The electric circuit of the armature and the free body diagram of the rotor are shown in the following figure:



For this example, we will assume the following values for the physical parameters. These values were derived by experiment from an actual motor in Carnegie Mellon's undergraduate controls lab.

* moment of inertia of the rotor (J) = 0.01 kg.m^2/s^2
* damping ratio of the mechanical system (b) = 0.1 Nms
* electromotive force constant (K=Ke=Kt) = 0.01 Nm/Amp
* electric resistance (R) = 1 ohm
* electric inductance (L) = 0.5 H
* input (V): Source Voltage
* output (theta): position of shaft
* The rotor and shaft are assumed to be rigid

The motor torque, **T**, is related to the armature current, **i**, by a constant factor **Kt**. The back emf, **e**, is related to the rotational velocity by the following equations:

$$T = K_t i$$
$$e = K_e \dot{\theta}$$

In SI units (which we will use), **Kt** (armature constant) is equal to **Ke** (motor constant).

From the figure above we can write the following equations based on Newton's law combined with Kirchhoff's law:

$$J\ddot{\theta} + b\dot{\theta} = Ki$$
$$L\frac{di}{dt} + Ri = V - K\dot{\theta}$$

1. Transfer Function

Using Laplace Transforms, the above modeling equations can be expressed in terms of s.

$$s(Js + b)\Theta(s) = KI(s)$$

$$(Ls + R)I(s) = V - Ks\Theta(s)$$

By eliminating I(s) we can get the following open-loop transfer function, where the rotational speed is the output and the voltage is the input.

$$\frac{\dot{\theta}}{V} = \frac{K}{(Js + b)(Ls + R) + K^2}$$

2. State-Space

In the state-space form, the equations above can be expressed by choosing the rotational speed and electric current as the state variables and the voltage as an input. The output is chosen to be the rotational speed.

$$\frac{d}{dt}\begin{bmatrix} \dot{\theta} \\ i \end{bmatrix} = \begin{bmatrix} -\frac{b}{J} & \frac{K}{J} \\ -\frac{K}{L} & -\frac{R}{L} \end{bmatrix}\begin{bmatrix} \dot{\theta} \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} V$$

$$\dot{\theta} = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} \dot{\theta} \\ i \end{bmatrix}$$

Design requirements

First, our uncompensated motor can only rotate at 0.1 rad/sec with an input voltage of 1 Volt (this will be demonstrated later when the open-loop response is simulated). Since the most basic requirement of a motor is that it should rotate at the desired speed, the steady-state error of the motor speed should be less than 1%. The other performance requirement is that the motor must accelerate to its steady-state speed as soon as it turns on. In this case, we want it to have a settling time of 2 seconds. Since a speed faster than the reference may damage the equipment, we want to have an overshoot of less than 5%.

If we simulate the reference input (r) by an unit step input, then the motor speed output should have:

- Settling time less than 2 seconds
- Overshoot less than 5%
- Steady-state error less than 1%

Matlab representation and open-loop response

1. Transfer Function

We can represent the above transfer function into Matlab by defining the numerator and denominator matrices as follows:

$$num = K$$
$$den = (Js + b)(Ls + R) + K^2$$

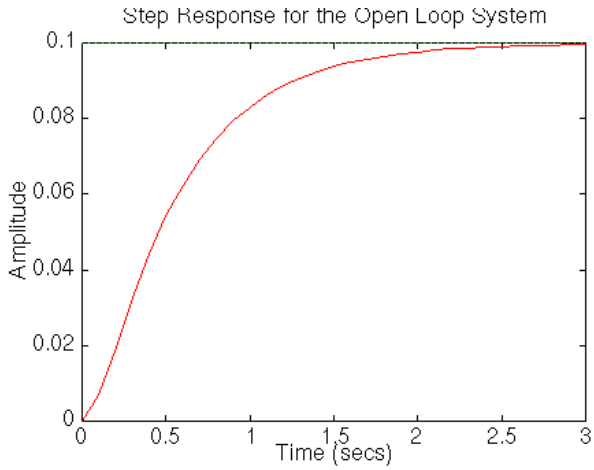
Create a new m-file and enter the following commands:

```
J=0.01;
b=0.1;
K=0.01;
R=1;
L=0.5;
num=K;
den=[(J*L) ((J*R)+(L*b)) ((b*R)+K^2)];
```

Now let's see how the original open-loop system performs. Add the following commands onto the end of the m-file and run it in the Matlab command window:

```
step(num,den,0:0.1:3)
title('Step Response for the Open Loop System')
```

You should get the following plot:



From the plot we see that when 1 volt is applied to the system, the motor can only achieve a maximum speed of 0.1 rad/sec, ten times smaller than our desired speed. Also, it takes the motor 3 seconds to reach its steady-state speed; this does not satisfy our 2 seconds settling time criterion.

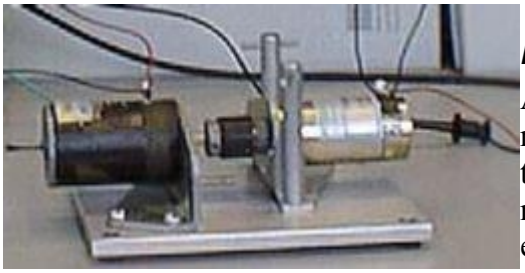
2. State-Space

We can also represent the system using the state-space equations. Try the following commands in a new m-file.

```
J=0.01;
b=0.1;
K=0.01;
R=1;
L=0.5;
A=[-b/J    K/J
   -K/L    -R/L];
B=[0
   1/L];
C=[1    0];
D=0;

step(A, B, C, D)
```

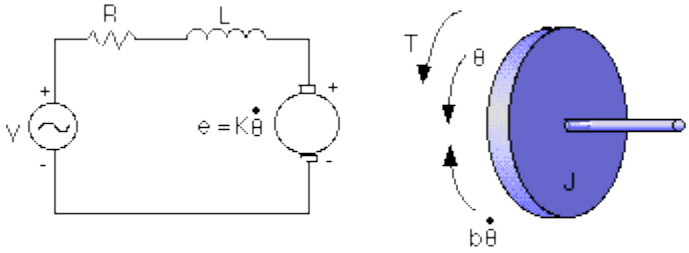
Run this m-file in the Matlab command window, and you should get the same output as the one shown above.

Example: Modeling DC Motor Position



Physical Setup

A common actuator in control systems is the DC motor. It directly provides rotary motion and, coupled with wheels or drums and cables, can provide transitional motion. The electric circuit of the armature and the free body diagram of the rotor are shown in the following figure:



For this example, we will assume the following values for the physical parameters. These values were derived by experiment from an actual motor in Carnegie Mellon's undergraduate controls lab.

* moment of inertia of the rotor (J) = 3.2284E-6 kg.m^2/s^2
* damping ratio of the mechanical system (b) = 3.5077E-6 Nms
* electromotive force constant (K=Ke=Kt) = 0.0274 Nm/Amp
* electric resistance (R) = 4 ohm
* electric inductance (L) = 2.75E-6 H
* input (V): Source Voltage
* output (theta): position of shaft
* The rotor and shaft are assumed to be rigid

System Equations

The motor torque, **T**, is related to the armature current, **i**, by a constant factor **Kt**. The back emf, **e**, is related to the rotational velocity by the following equations:

$$T = K_t i$$

$$e = K_e \dot{\theta}$$

In SI units (which we will use), **Kt** (armature constant) is equal to **Ke** (motor constant).

From the figure above we can write the following equations based on Newton's law combined with Kirchhoff's law:

$$J \ddot{\theta} + b \dot{\theta} = Ki$$

$$L \frac{di}{dt} + Ri = V - K \dot{\theta}$$

1. Transfer Function

Using Laplace Transforms the above equations can be expressed in terms of s.

$$s(Js + b)\Theta(s) = KI(s)$$

$$(Ls + R)I(s) = V - Ks\Theta(s)$$

By eliminating I(s) we can get the following transfer function, where the rotating speed is the output and the voltage is an input.

$$\frac{\dot{\theta}}{V} = \frac{K}{(Js + b)(Ls + R) + K^2}$$

However during this example we will be looking at the position, as being the output. We can obtain the position by integrating Theta Dot, therefore we just need to divide the transfer function by s.

$$\frac{\theta}{V} = \frac{K}{s((Js + b)(Ls + R) + K^2)}$$

2. State Space

These equations can also be represented in state-space form. If we choose motor position, motor speed, and armature current as our state variables, we can write the equations as follows:

$$\frac{d}{dt}\begin{bmatrix} \theta \\ \dot{\theta} \\ i \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{b}{J} & \frac{K}{J} \\ 0 & -\frac{K}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L} \end{bmatrix} V$$

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ i \end{bmatrix}$$

Design requirements

We will want to be able to position the motor very precisely, thus the steady-state error of the motor position should be zero. We will also want the steady-state error due to a disturbance, to be zero as well. The other performance requirement is that the motor reaches its final position very quickly. In this case, we want it to have a settling time of 40ms. We also want to have an overshoot smaller than 16%.

If we simulate the reference input (R) by a unit step input, then the motor speed output should have:

- Settling time less than 40 milliseconds
- Overshoot less than 16%
- No steady-state error
- No steady-state error due to a disturbance

Matlab representation and open-loop response

1. Transfer Function

We can put the transfer function into Matlab by defining the numerator and denominator as vectors:

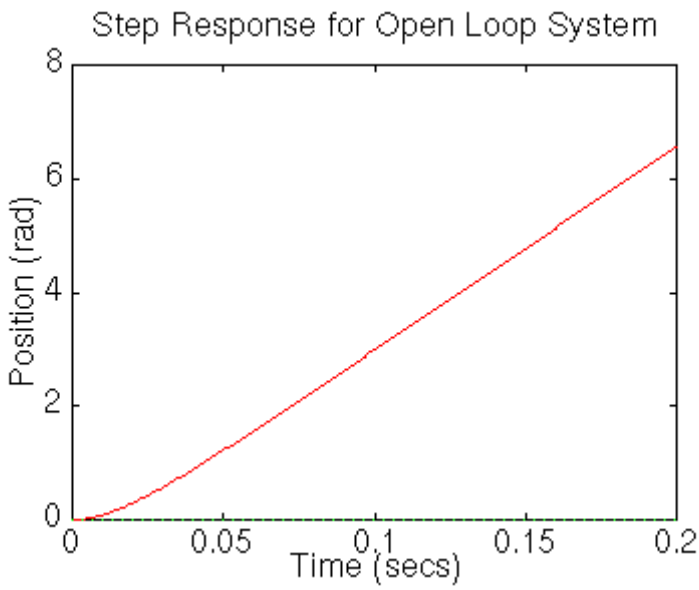
Create a new m-file and enter the following commands:

```
J=3.2284E-6;
b=3.5077E-6;
K=0.0274;
R=4;
L=2.75E-6;
num=K;
den=[(J*L) ((J*R)+(L*b)) ((b*R)
+K^2) 0];
```

Now let's see how the original open-loop system performs. Add the following command onto the end of the m-file and run it in the Matlab command window:

```
step(num,den,0:0.001:0.2)
```

You should get the following plot:



From the plot we see that when 1 volt is applied to the system, the motor position changes by 6 radians, six times greater than our desired position. For a 1 volt step input the motor should spin through 1 radian. Also, the motor doesn't reach a steady state which does not satisfy our design criteria

2. State Space

We can put the state space equations into Matlab by defining the system's matrices as follows:

```
J=3.2284E-6;
b=3.5077E-6;
K=0.0274;
R=4;
L=2.75E-6;

A=[0 1 0
   0 -b/J K/J
   0 -K/L -R/L];
B=[0 ; 0 ; 1/L];
C=[1 0 0];
D=[0];
```

The step response is obtained using the command

```
step(A,B,C,D)
```

Unfortunately, Matlab responds with

```
Warning: Divide by zero
???  Index exceeds matrix dimensions.

Error in ==> /usr/local/lib/matlab/toolbox/control/step.m
On line 84  ==> dt = t(2)-t(1);
```

There are numerical scaling problems with this representation of the dynamic equations. To fix the problem, we scale time by tscale = 1000. Now the output time will be in milliseconds rather than in seconds. The equations are given by

```
tscale = 1000;
J=3.2284E-6*tscale^2;
```

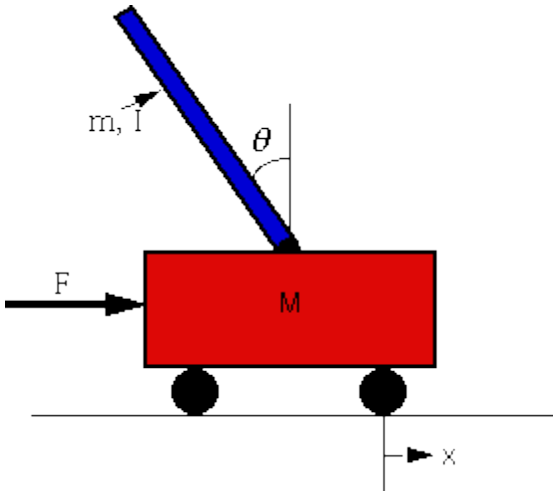
```
b=3.5077E-6*tscale;
K=0.0274*tscale;
R=4*tscale;
L=2.75E-6*tscale^2;

A=[0 1 0
   0 -b/J K/J
   0 -K/L -R/L];
B=[0 ; 0 ; 1/L];
C=[1 0 0];
D=[0];
```

The output appears the same as when obtained through the transfer function, but the time vector must be divided by tscale.

```
[y,x,t]=step(A,B,C,D);
plot(t/tscale,y)
ylabel('Amplitude')
xlabel('Time (sec)')
```

Example: Modeling an Inverted Pendulum

Problem setup and design requirements



The cart with an inverted pendulum, shown below, is "bumped" with an impulse force, F. Determine the dynamic equations of motion for the system, and linearize about the pendulum's angle, theta = Pi (in other words, assume that pendulum does not move more than a few degrees away from the vertical, chosen to be at an angle of Pi). Find a controller to satisfy all of the design requirements given below.

For this example, let's assume that

M	mass of the cart	0.5 kg
m	mass of the pendulum	0.5 kg
b	friction of the cart	0.1 N/m/sec
l	length to pendulum center of mass	0.3 m
I	inertia of the pendulum	0.006 kg*m^2
F	force applied to the cart	
x	cart position coordinate	
theta	pendulum angle from vertical	

For the PID, root locus, and frequency response sections of this problem we will be only interested in the control of the pendulums position. This is because the techniques used in these tutorials can only be applied for a single-input-single-output (SISO) system. Therefore, none of the design criteria deal with the cart's position. For these sections we will assume that the system starts at equilibrium, and experiences an impulse force of 1N. The pendulum should return to its upright position within 5 seconds, and never move more than 0.05 radians away from the vertical.

The design requirements for this system are:

- Settling time of less than 5 seconds.
- Pendulum angle never more than 0.05 radians from the vertical.

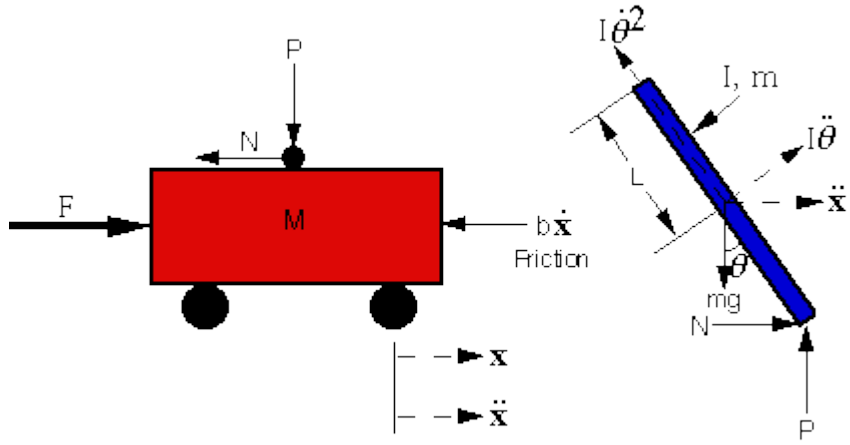
However, with the state-space method we are more readily able to deal with a multi-output system. Therefore, for this section of the Inverted Pendulum example we will attempt to control both the pendulum's angle and the cart's position. To make the design more challenging we will be applying a step input to the cart. The cart should achieve it's desired position within 5 seconds and have a rise time under 0.5 seconds. We will also limit the pendulum's overshoot to 20 degrees (0.35 radians), and it should also settle in under 5 seconds.

The design requirements for the Inverted Pendulum state-space example are:

- Settling time for x **and** theta of less than 5 seconds.
- Rise time for x of less than 0.5 seconds.
- Overshoot of theta less than 20 degrees (0.35 radians).

Force analysis and system equations

Below are the two Free Body Diagrams of the system.



Summing the forces in the Free Body Diagram of the cart in the horizontal direction, you get the following equation of motion:

$$M\ddot{x}+b\dot{x}+N=F$$

Note that you could also sum the forces in the vertical direction, but no useful information would be gained.

Summing the forces in the Free Body Diagram of the pendulum in the horizontal direction, you can get an equation for N:

$$N=m\ddot{x}+ml\ddot{\theta}\cos\theta-ml\dot{\theta}^2\sin\theta$$

If you substitute this equation into the first equation, you get the first equation of motion for this system:

$$\textbf{(1)}\ (M+m)\ddot{x}+b\dot{x}+ml\ddot{\theta}\cos\theta-ml\dot{\theta}^2\sin\theta=F$$

To get the second equation of motion, sum the forces perpendicular to the pendulum. Solving the system along this axis ends up saving you a lot of algebra. You should get the following equation:

$$P\sin\theta+N\cos\theta-mg\sin\theta=ml\ddot{\theta}+m\ddot{x}\cos\theta$$

To get rid of the P and N terms in the equation above, sum the moments around the centroid of the pendulum to get the following equation:

$$-Pl\sin\theta-Nl\cos\theta=I\ddot{\theta}$$

Combining these last two equations, you get the second dynamic equation:

$$\textbf{(2)}\ (I+ml^2)\ddot{\theta}+mgl\sin\theta=-ml\ddot{x}\cos\theta$$

Since Matlab can only work with linear functions, this set of equations should be linearized about theta = Pi. Assume that theta = Pi + ø (ø represents a small angle from the vertical upward direction). Therefore, cos(theta) = -1, sin(theta) = -ø, and (d(theta)/dt)^2 = 0. After linearization the two equations of motion become (where u represents the input):

$$\left(I+ml^2\right)\ddot{\phi}-mgl\phi=ml\ddot{x}$$
$$(M+m)\ddot{x}+b\dot{x}-ml\ddot{\phi}=u$$

1. Transfer Function

To obtain the transfer function of the linearized system equations analytically, we must first take the Laplace transform of the system equations. The Laplace transforms are:

$$(\mathbf{I}+\mathbf{ml}^2)\Phi(s)s^2 - \mathbf{mgl}\Phi(s) = \mathbf{ml}X(s)s^2$$
$$(\mathbf{M}+\mathbf{m})X(s)s^2 + \mathbf{b}X(s)s - \mathbf{ml}\Phi(s)s^2 = U(s)$$

NOTE: When finding the transfer function initial conditions are assumed to be zero.

Since we will be looking at the angle Phi as the output of interest, solve the first equation for X(s),

$$X(s) = \left[\frac{(\mathbf{I}+\mathbf{ml}^2)}{\mathbf{ml}} - \frac{\mathbf{g}}{\mathbf{s}^2}\right]\Phi(s)$$

then substituting into the second equation:

$$(\mathbf{M}+\mathbf{m})\left[\frac{(\mathbf{I}+\mathbf{ml}^2)}{\mathbf{ml}} + \frac{\mathbf{g}}{\mathbf{s}}\right]\Phi(s)s^2 + \mathbf{b}\left[\frac{(\mathbf{I}+\mathbf{ml}^2)}{\mathbf{ml}} + \frac{\mathbf{g}}{\mathbf{s}}\right]\Phi(s)s - \mathbf{ml}\Phi(s)s^2 = U(s)$$

Re-arranging, the transfer function is:

$$\frac{\Phi(s)}{U(s)} = \frac{\frac{\mathbf{ml}}{\mathbf{q}}s^2}{s^4 + \frac{\mathbf{b}(\mathbf{I}+\mathbf{ml}^2)}{\mathbf{q}}s^3 - \frac{(\mathbf{M}+\mathbf{m})\mathbf{mgl}}{\mathbf{q}}s^2 - \frac{\mathbf{bmgl}}{\mathbf{q}}s}$$

where,

$$\mathbf{q} = \left[(\mathbf{M}+\mathbf{m})(\mathbf{I}+\mathbf{ml}^2) - (\mathbf{ml})^2\right]$$

From the transfer function above it can be seen that there is both a pole and a zero at the origin. These can be canceled and the transfer function becomes:

$$\frac{\Phi(s)}{U(s)} = \frac{\frac{\mathbf{ml}}{\mathbf{q}}s}{s^3 + \frac{\mathbf{b}(\mathbf{I}+\mathbf{ml}^2)}{\mathbf{q}}s^2 - \frac{(\mathbf{M}+\mathbf{m})\mathbf{mgl}}{\mathbf{q}}s - \frac{\mathbf{bmgl}}{\mathbf{q}}}$$

2. State-Space

After a little algebra, the linearized system equations equations can also be represented in state-space form:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\Phi} \\ \ddot{\Phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I+ml^2)b}{I(M+m)+Mml^2} & \frac{m^2gl^2}{I(M+m)+Mml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-mlb}{I(M+m)+Mml^2} & \frac{mgl(M+m)}{I(M+m)+Mml^2} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \Phi \\ \dot{\Phi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{I+ml^2}{I(M+m)+Mml^2} \\ 0 \\ \frac{ml}{I(M+m)+Mml^2} \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \Phi \\ \dot{\Phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u$$

The C matrix is 2 by 4, because both the cart's position and the pendulum's position are part of the output. For the state-space design problem we will be controlling a multi-output system so we will be observing the cart's position from the first row of output and the pendulum's with the second row.

Matlab representation and the open-loop response

1. Transfer Function

The transfer function found from the Laplace transforms can be set up using Matlab by inputting the numerator and denominator as vectors. Create an m-file and copy the following text to model the transfer function:

```
M = .5;
m = 0.2;
b = 0.1;
i = 0.006;
g = 9.8;
l = 0.3;

q = (M+m)*(i+m*l^2)-(m*l)^2;    %simplifies input

num = [m*l/q  0]
den = [1  b*(i+m*l^2)/q  -(M+m)*m*g*l/q  -b*m*g*l/q]
```

Your output should be:

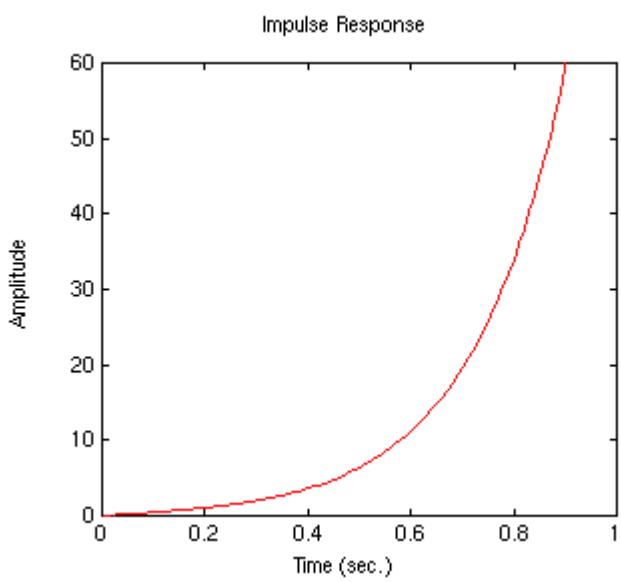
```
num =
    4.5455          0

den =
    1.0000    0.1818   -31.1818    -4.4545
```

To observe the system's velocity response to an impulse force applied to the cart add the following lines at the end of your m-file:

```
t=0:0.01:5;
impulse(num,den,t)
axis([0 1 0 60])
```

Note: Matlab commands from the control system toolbox are highlighted in red.

You should get the following velocity response plot:



As you can see from the plot, the response is entirely unsatisfactory. It is not stable in open loop. You can change the axis to see more of the response if you need to convince yourself that the system is unstable.

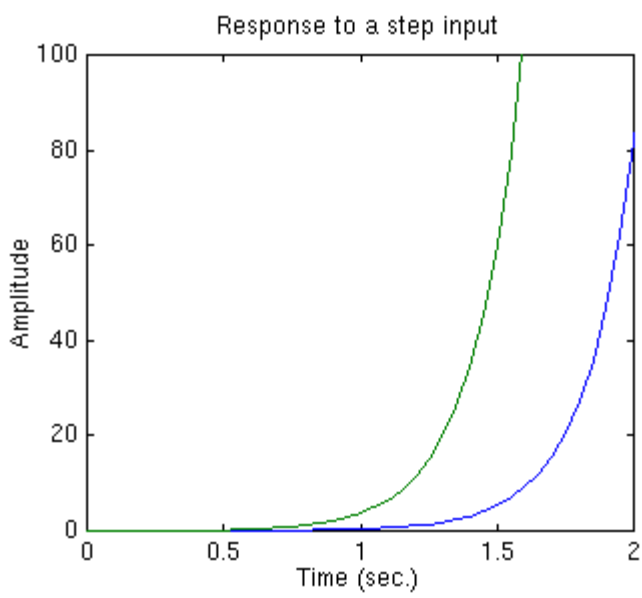
1. State-Space

Below, we show how the problem would be set up using Matlab for the state-space model. If you copy the following text into a m-file (or into a '.m' file located in the same directory as Matlab) and run it, Matlab will give you the A, B, C, and D matrices for the state-space model and a plot of the response of the cart's position and pendulum angle to a step input of 0.2 m applied to the cart.

```
M = .5;
m = 0.2;
b = 0.1;
i = 0.006;
g = 9.8;
l = 0.3;

p = i*(M+m)+M*m*l^2; %denominator for the A and B matricies
A = [0      1              0           0;
     0 -(i+m*l^2)*b/p   (m^2*g*l^2)/p   0;
     0      0              0           1;
     0 -(m*l*b)/p       m*g*l*(M+m)/p   0]
B = [     0;
     (i+m*l^2)/p;
          0;
        m*l/p]
C = [1 0 0 0;
     0 0 1 0]
D = [0;
     0]

T=0:0.05:10;
U=0.2*ones(size(T));
[Y,X]=lsim(A,B,C,D,U,T);
plot(T,Y)
axis([0 2 0 100])
```

You should see the following output after running the m-file:

```
A =
         0    1.0000         0         0
         0   -0.1818    2.6727         0
         0         0         0    1.0000
         0   -0.4545   31.1818         0

B =
         0
    1.8182
         0
    4.5455

C =
     1     0     0     0
     0     0     1     0

D =
     0
     0
```



The blue line represents the cart's position and the green line represents the pendulum's angle. It is obvious from this plot and the one above that some sort of control will have to be designed to improve the dynamics of the system. Four example controllers are included with these tutorials: PID, root locus, frequency response, and state space. Select from below the one you would like to use.

Note: The solutions shown in the **PID**, **root locus** and **frequency response** examples may not yield a workable controller for the inverted pendulum problem. As stated previously, when we put this problem into the single-input, single-output framework, we ignored the x position of the cart. The pendulum can be stabilized in an inverted position if the x position is constant or if the cart moves at a constant velocity (no acceleration). Where possible in these examples, we will show what happens to the cart's position when our controller is implemented on the system. We emphasize that the purpose of these examples is to demonstrate design and analysis techniques using Matlab; not to actually control an inverted pendulum.

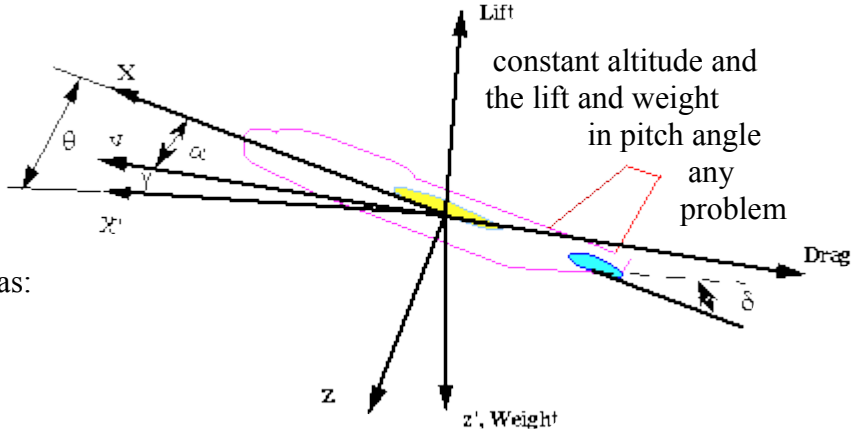
Example: Modeling a Pitch Controller

Photo courtesy: Boeing

Physical setup and system equations

The equations governing the motion of an aircraft are a very complicated set of six non-linear coupled differential equations. However, under certain assumptions, they can be decoupled and linearized into the longitudinal and lateral equations. Pitch control is a longitudinal problem, and in this example, we will design an autopilot that controls the pitch of an aircraft.

The basic coordinate axes and forces acting on an aircraft are shown in the figure below:



Assume that the aircraft is in steady-cruise at constant altitude and velocity; thus, the thrust and drag cancel out and the lift and weight balance out each other. Also, assume that change in pitch angle does not change the speed of an aircraft under any circumstance (unrealistic but simplifies the problem a bit). Under these assumptions, the longitudinal equations of motion of an aircraft can be written as:

(1)

$$\dot{\alpha} = \mu\Omega\sigma[-(C_{L\alpha} + C_{D\alpha})\alpha + (1/\mu - C_{L\dot{\alpha}})q - (C_{W\alpha}\mathrm{Sin}\gamma_e)\theta + C_{L\delta_e}]$$

$$\dot{q} = \frac{\mu\Omega}{2i_{yy}}\{[C_{M\alpha} - \eta(C_{L\alpha} + C_{D\alpha})]\alpha + [C_{Mq} + \sigma C_{M\dot{\alpha}}(1 - \mu C_{L\dot{\alpha}})]q + (\eta C_{W\alpha}\mathrm{Sin}\gamma_e)\delta_e\}$$

$$\dot{\theta} = \Omega q$$

Please refer to any aircraft-related textbooks for the explanation of how to derive these equations. Also, click Variables to see what each variable represents.

For this system, the input will be the elevator deflection angle, and the output will be the pitch angle.

Design requirements

The next step is to set some design criteria. We want to design a feedback controller so that the output has an overshoot of less than 10%, rise time of less than 2 seconds, settling time of less than 10 seconds, and the steady-state error of less than 2%. For example, if the input is 0.2 rad (11 degress), then the pitch angle will not

exceed 0.22 rad, reaches 0.2 rad within 2 seconds, settles 2% of the steady-state within 10 seconds, and stays within 0.196 to 0.204 rad at the steady-state.

- Overshoot: Less than 10%
- Rise time: Less than 2 seconds
- Settling time: Less than 10 seconds
- Steady-state error: Less than 2%

Transfer function and the state-space

Before finding transfer function and the state-space model, let's plug in some numerical values to simplify the modeling equations (1) shown above.

$$\begin{aligned} \dot{\alpha} &= -0.313\alpha + 56.7q + 0.232\delta_e \\ \dot{q} &= -0.0139\alpha - 0.426q + 0.0203\delta_e \\ \dot{\theta} &= 56.7q \end{aligned} \qquad (2)$$

These values are taken from the data from one of the Boeing's commercial aircraft.

1. Transfer function

To find the transfer function of the above system, we need to take the Laplace transform of the above modeling equations (2). Recall from your control textbook, **when finding a transfer function, zero initial conditions must be assumed**. The Laplace transform of the above equations are shown below.

$$\begin{aligned} s\alpha(s) &= -0.313\alpha(s) + 56.7q(s) + 0.232\delta_e(s) \\ sq(s) &= -0.0139\alpha(s) - 0.426q(s) + 0.0203\delta_e(s \\ s\theta(s) &= 56.7q(s) \end{aligned}$$

After few steps of algebra, you should obtain the following transfer function.

$$\frac{\theta(s)}{\delta_e(s)} = \frac{1.151s + 0.1774}{s^3 + 0.739s^2 + 0.921s}$$

2. State-space

Knowing the fact that the modeling equations (2) are already in the state-variable form, we can rewrite them into the state-space model.

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} -0.313 & 56.7 & 0 \\ -0.0139 & -0.426 & 0 \\ 0 & 56.7 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} 0.232 \\ 0.0203 \\ 0 \end{bmatrix} [\delta_e]$$

Since our output is the pitch angle, the output equation is:

$$y = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \theta \end{bmatrix} + [0][\delta_e]$$

Matlab representation and open-loop response

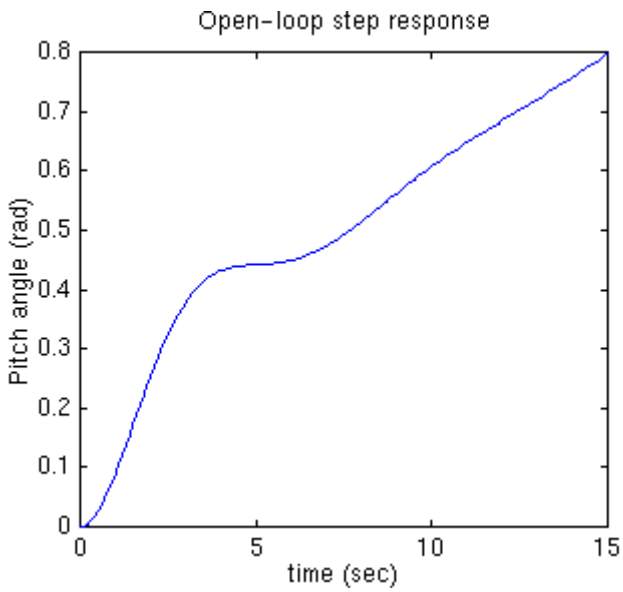
Now, we are ready to observe the system characteristics using Matlab.
First, let's obtain an open-loop system to a step input and determine which system characteristics need improvement. Let the input (delta e) be 0.2 rad (11 degrees). Create an new m-file and enter the following commands.

```
de=0.2;
```

```
num=[1.151 0.1774];
den=[1 0.739 0.921 0];

step (de*num,den)
```

Running this m-file in the Matlab command window should give you the following plot.

From the plot, we see that the open-loop response does not satisfy the design criteria at all. In fact the open-loop response is unstable.



If you noticed, the above m-file uses the numerical values from the transfer function. To use the state-space model, enter the following commands into a new m-file (instead of the one shown above) and run it in the command window.

```
de=0.2;

A=[-0.313 56.7 0; -0.0139 -0.426 0; 0 56.7 0];
B=[0.232; 0.0203; 0];
C=[0 0 1];
D=[0];

step(A,B*de,C,D)
```

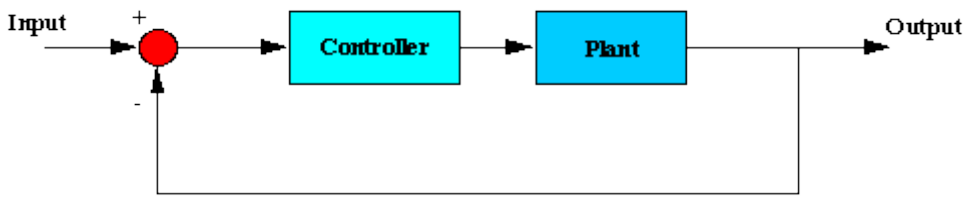
You should get the same response as the one shown above.

Note: It is possible to convert from the state-space to transfer function, or vice versa using Matlab. To learn more about conversions, see Conversions

---

Closed-loop transfer function

To solve this problem, a feedback controller will be added to improve the system performance. The figure shown below is the block diagram of a typical unity feedback system.

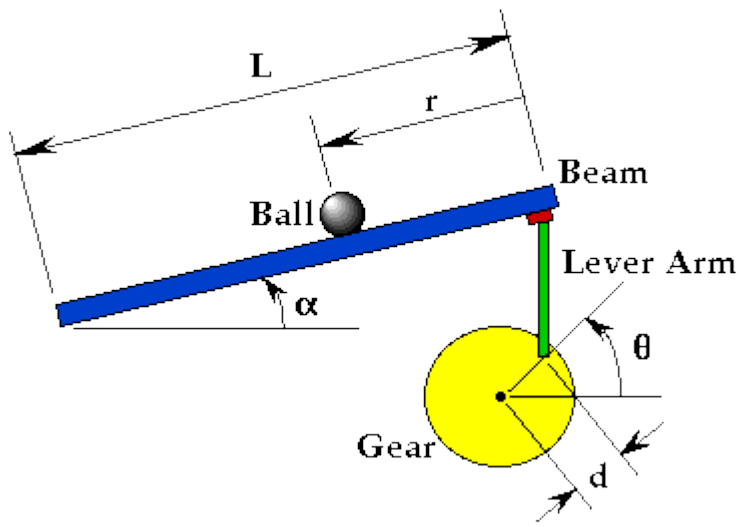


A controller needs to be designed so that the step response satisfies all design requirements. Four different methods to design a controller are listed at the bottom of this page. You may choose: PID, Root-locus, Frequency response, or State-space.

Example: Modeling the Ball and Beam Experiment

Problem Setup

A ball is placed on a beam, see figure below, where it is allowed to roll with 1 degree of freedom along the length of the beam. A lever arm is attached to the beam at one end and a servo gear at the other. As the servo gear turns by an angle theta, the lever changes the angle of the beam by alpha. When the angle is changed from the vertical position, gravity causes the ball to roll along the beam. A controller will be designed for this system so that the ball's position can be manipulated.



For this problem, we will assume that the ball rolls without slipping and friction between the beam and ball is negligible. The constants and variables for this example are defined as follows:

M	mass of the ball	0.11 kg
R	radius of the ball	0.015 m
d	lever arm offset	0.03 m
g	gravitational acceleration	9.8 m/s^2
L	length of the beam	1.0 m
J	ball's moment of inertia	9.99e-6 kgm^2
r	ball position coordinate	
alpha	beam angle coordinate	
theta	servo gear angle	

The design criteria for this problem are:

- Settling time less than 3 seconds
- Overshoot less than 5%

System Equations

The Lagrangian equation of motion for the ball is given by the following:

$$0 = \left(\frac{J}{R^2} + m\right)\ddot{r} + mg\sin\alpha - mr(\dot{\alpha})^2$$

Linearization of this equation about the beam angle, alpha = 0, gives us the following linear approximation of the system:

$$\left(\frac{J}{R^2} + m\right)\ddot{r} = -mg\,\alpha$$

The equation which relates the beam angle to the angle of the gear can be approximated as linear by the equation below:

$$\alpha = \frac{d}{L}\theta$$

$$\left(\frac{J}{R^2} + m\right)\ddot{r} = -mg\frac{d}{L}\theta$$

Substituting this into the previous equation, we get:

1. Transfer Function

Taking the Laplace transform of the equation above, the following equation is found:

$$\left(\frac{J}{R^2}+m\right)R(s)s^2 = -\frac{mgd}{L}\Theta(s)$$

NOTE: When taking the Laplace transform to find the transfer function initial conditions are assumed to be zero.

Rearranging we find the transfer function from the gear angle (theta(s)) to the ball position (*R(s)*).

$$\frac{R(s)}{\Theta(s)} = -\frac{mgd}{L\left(\frac{J}{R^2}+m\right)}\frac{1}{s^2}$$

It should be noted that the above plant transfer function is a double integrator. As such it is marginally stable and will provide a challenging control problem.

2. State-Space

The linearized system equations can also be represented in state-space form. This can be done by selecting the ball's position (r) and velocity (rdot) as the state variables and the gear angle (theta) as the input. The state-space representation is shown below:

$$\begin{bmatrix}\dot{r}\\ \ddot{r}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}r\\ \dot{r}\end{bmatrix} + \begin{bmatrix}0\\ \dfrac{mgd}{L\left(\frac{J}{R^2}+m\right)}\end{bmatrix}\theta$$

However, for our state-space example we will be using a slightly different model. The same equation for the ball still applies but instead of controlling the position through the gear angle, theta, we will control alpha-doubledot. This is essentially controlling the torque of the beam. Below is the representation of this system:

$$\begin{bmatrix}\dot{r}\\ \ddot{r}\\ \dot{\alpha}\\ \ddot{\alpha}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & \dfrac{-mg}{\left(\frac{J}{R^2}+m\right)} & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}r\\ \dot{r}\\ \alpha\\ \dot{\alpha}\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 0\\ 1\end{bmatrix}u$$

Note: For this system the gear and lever arm would not be used, instead a motor at the center of the beam will apply torque to the beam, to control the ball's position.

$$y = \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}r\\ \dot{r}\\ \alpha\\ \dot{\alpha}\end{bmatrix}$$

Matlab Representation and Open-Loop Response

1. Transfer Function

The transfer function found from the Laplace transform can be implemented in Matlab by inputting the numerator and denominator as vectors. To do this we must create an m-file and copy the following text into it:

```
m = 0.111;
R = 0.015;
g = -9.8;
L = 1.0;
d = 0.03;
J = 9.99e-6;
```

```
K = (m*g*d)/(L*(J/R^2+m));    %simplifies input

num = [-K];
den = [1 0 0];
printsys(num,den)
```

Your output should be:

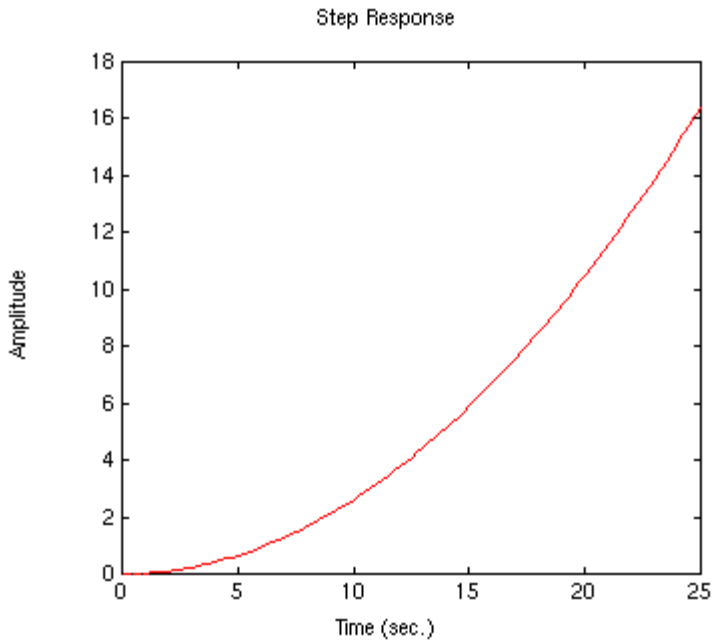
```
num/den =

            0.21
        ----------
            s^2
```

Now, we would like to observe the ball's response to a step input of 0.25 m. To do this you will need to add the following line to your m-file:

```
step(0.25*num,den)
```

NOTE: Matlab commands from the control system toolbox are highlighted in red.

You should see the following plot showing the balls position as a function of time:



From this plot it is clear that the system is unstable in open-loop causing the ball to roll right off the end of the beam. Therefore, some method of controlling the ball's position in this system is required. Three examples of controller design are listed below for the transfer function problem. You may select from PID, Root Locus, and Frequency Response.

2. State-Space

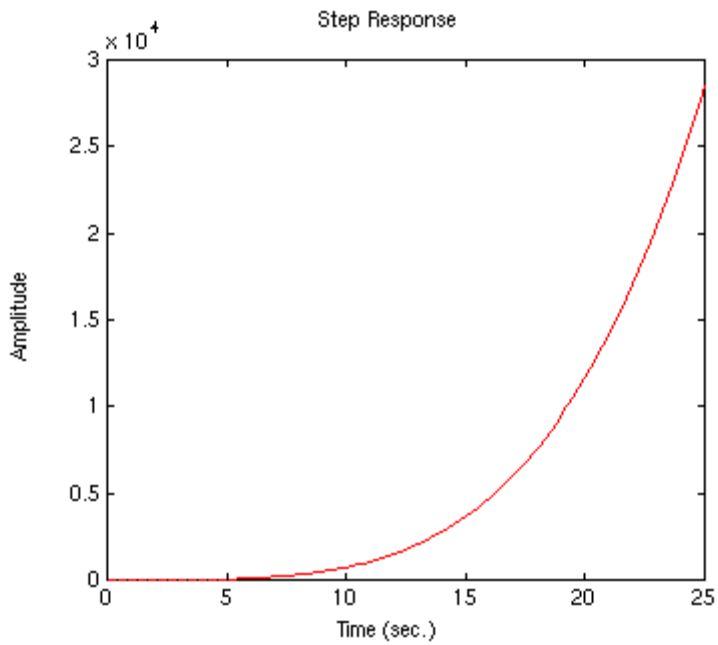
The state-space equations can be represented in Matlab with the following commands (these equations are for the torque control model).

```
m = 0.111;
R = 0.015;
g = -9.8;
J = 9.99e-6;

H = -m*g/(J/(R^2)+m);

A=[0 1 0 0
   0 0 H 0
   0 0 0 1
   0 0 0 0];
B=[0;0;0;1];
C=[1 0 0 0];
D=[0];
```

The step response to a 0.25m desired position can be viewed by running the command below:

```
step(A,B*.25,C,D)
```

Your output should look like the following:



Like the plot for the transfer function this plot shows that the system is unstable and the ball will roll right off the end of the beam.

Therefore, we will require some method of controlling the ball's position in this system. The State-Space example below shows how to implement a controller for this type of system.

If you are interested in knowing how to convert state-space representations to transfer function representations, and vice versa, please refer to the Conversions page.

Autor:

Zidarta

jimfackyou@hotmail.com

Perú

02/06/2008