

# Control Tutorials for MATLAB® and Simulink®

## State Space Tutorial

[State-space equations](#)

[Control design using pole placement](#)

[Introducing the reference input](#)

[Observer design](#)

Key MATLAB commands used in this tutorial: **acker**, **lsim**, **place**, **plot**, **rscale**

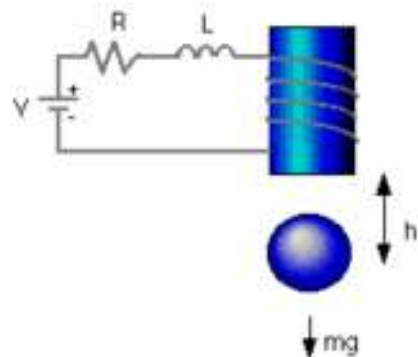
### State-space equations

There are several different ways to describe a system of linear differential equations. The **state-space representation** is given by the equations:

$$\begin{aligned}\frac{d\mathbf{x}}{dt} &= \mathbf{A}\mathbf{x} + \mathbf{B}u \\ \mathbf{y} &= \mathbf{C}\mathbf{x} + \mathbf{D}u\end{aligned}$$

where  $\mathbf{x}$  is an  $n$  by 1 vector representing the state (commonly position and velocity variables in mechanical systems),  $u$  is a scalar representing the input (commonly a force or torque in mechanical systems), and  $y$  is a scalar representing the output. The matrices  $\mathbf{A}$  ( $n$  by  $n$ ),  $\mathbf{B}$  ( $n$  by 1), and  $\mathbf{C}$  (1 by  $n$ ) determine the relationships between the state and input and output variables. Note that there are  $n$  first-order differential equations. State space representation can also be used for systems with multiple inputs and outputs (MIMO), but we will only use single-input, single-output (SISO) systems in these tutorials.

To introduce the state space design method, we will use the magnetically suspended ball as an example. The current through the coils induces a magnetic force which can balance the force of gravity and cause the ball (which is made of a magnetic material) to be suspended in midair. The modeling of this system has been established in many control text books (including *Automatic Control Systems* by B. C. Kuo, the seventh edition). The equations for the system are given by:



$$M \frac{d^2 h}{dt^2} = Mg - \frac{K i^2}{h}$$

$$V = L \frac{di}{dt} + iR$$

where  $h$  is the vertical position of the ball,  $i$  is the current through the electromagnet,  $V$  is the applied voltage,  $M$  is the mass of the ball,  $g$  is gravity,  $L$  is the inductance,  $R$  is the resistance, and  $K$  is a coefficient that determines the magnetic force exerted on the ball. For simplicity, we will choose values  $M = 0.05$  Kg,  $K = 0.0001$ ,  $L = 0.01$  H,  $R = 1$  Ohm,  $g = 9.81$  m/sec<sup>2</sup>. The system is at equilibrium (the ball is suspended in midair) whenever  $h = K i^2 / Mg$  (at which point  $dh/dt = 0$ ). We linearize the equations about the point  $h = 0.01$  m (where the nominal current is about 7 amp) and get the state space equations:

$$\frac{d\vec{x}}{dt} = A\vec{x} + Bu$$

$$y = C\vec{x} + Du$$

where:  $\vec{x} = \begin{bmatrix} \Delta h \\ \Delta \dot{h} \\ \Delta i \end{bmatrix}$  is the set of state variab for the system (a 3x1 vector),  $u$  is the input voltage (delta V), and  $y$  (the output), is delta h. Enter the system matrices into a [m-file](#).

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 980 & 0 & -2.8 \\ 0 & 0 & -100 \end{bmatrix};$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 100 \end{bmatrix};$$

$$C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix};$$

One of the first things you want to do with the state equations is find the poles of the system; these are the values of  $s$  where  $\det(sI - A) = 0$ , or the eigenvalues of the  $A$  matrix:

$$\text{poles} = \text{eig}(A)$$

You should get the following three poles:

$$\text{poles} =$$

$$\begin{matrix} 31.3050 \\ -31.3050 \\ -100.0000 \end{matrix}$$

One of the poles is in the right-half plane, which means that the system is unstable in open-loop.

To check out what happens to this unstable system when there is a nonzero initial condition, add the following lines to your m-file,

$$t = 0:0.01:2;$$

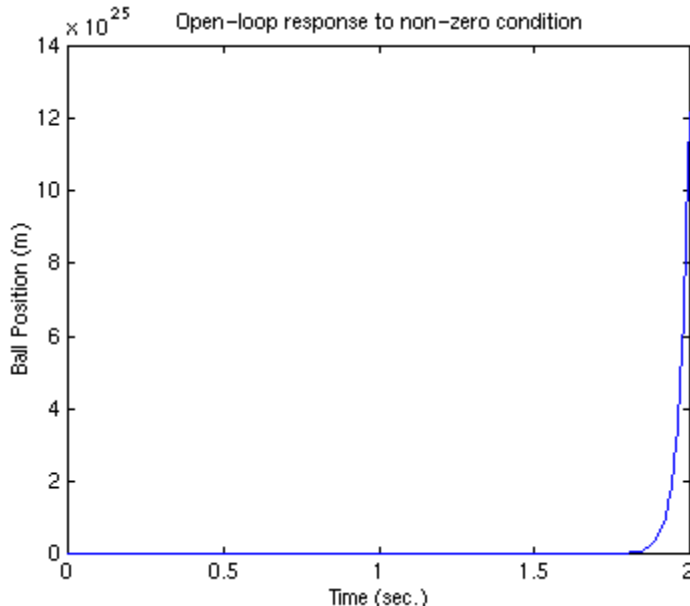
$$u = 0*t;$$

$$x0 = [0.005 \ 0 \ 0];$$

$$\text{sys} = \text{ss}(A, B, C, 0);$$

```
[y,t,x] = lsim(sys,u,t,x0);
plot(t,y)
```

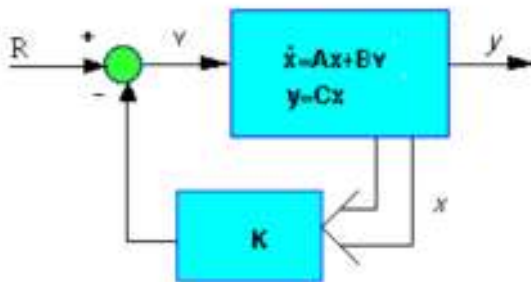
and run the file again.



It looks like the distance between the ball and the electromagnet will go to infinity, but probably the ball hits the table or the floor first (and also probably goes out of the range where our linearization is valid).

## Control design using pole placement

Let's build a controller for this system. The schematic of a full-state feedback system is the following:

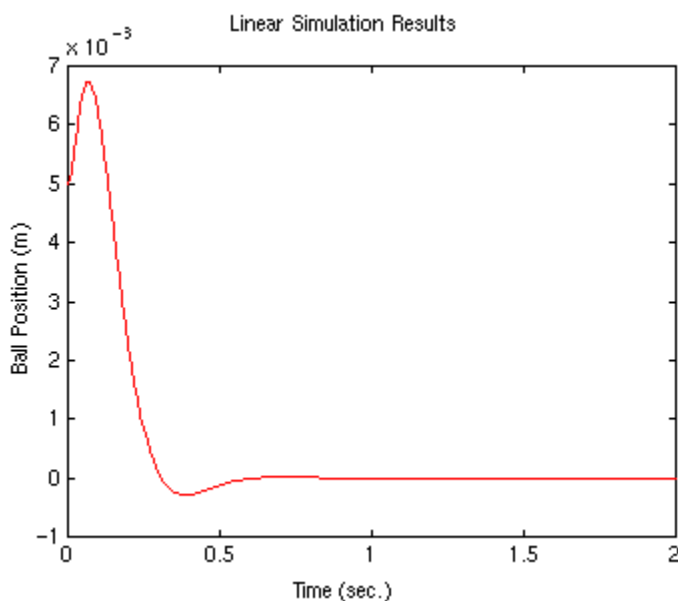


Recall that the characteristic polynomial for this closed-loop system is the determinant of  $(sI - (A - BK))$ . Since the matrices  $A$  and  $B \cdot K$  are both 3 by 3 matrices, there will be 3 poles for the system. By using full-state feedback we can place the poles anywhere we want. We could use the MATLAB function `place` to find the control matrix,  $K$ , which will give the desired poles.

Before attempting this method, we have to decide where we want the closed-loop poles to be. Suppose the criteria for the controller were settling time  $< 0.5$  sec and overshoot  $< 5\%$ , then we might try to place the two dominant poles at  $-10 \pm 10i$  (at  $\zeta = 0.7$  or  $45^\circ$  degrees with  $\sigma = 10 > 4.6 \cdot 2$ ). The third pole we might place at  $-50$  to start, and we can change it later depending on what the closed-loop behavior is. Remove the `lsim` command from your m-file and everything after it, then add the following lines to your m-file,

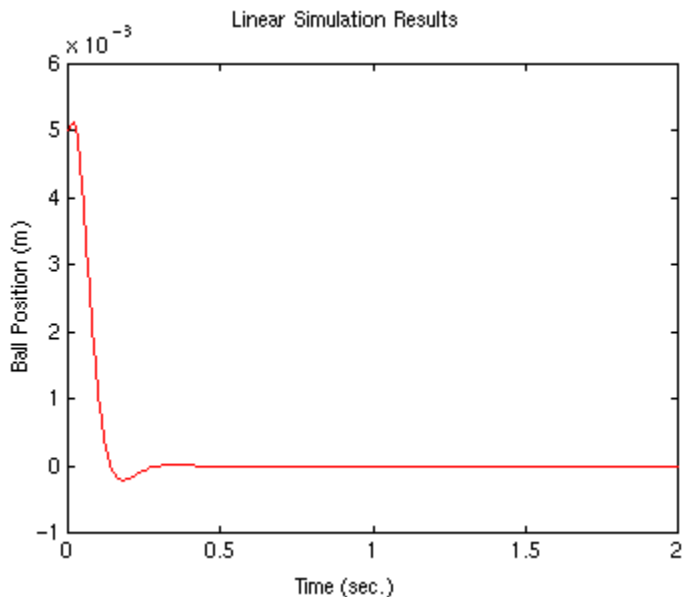
```
p1 = -10 + 10i;
p2 = -10 - 10i;
p3 = -50;

K = place(A,B,[p1 p2 p3]);
sys_cl=ss(A-B*K,B,C,0);
lsim(sys_cl,u,t,x0);
```



The overshoot is too large (there are also zeros in the transfer function which can increase the overshoot; you do not see the zeros in the state-space formulation). Try placing the poles further to the left to see if the transient response improves (this should also make the response faster).

```
p1 = -20 + 20i;
p2 = -20 - 20i;
p3 = -100;
K = place(A,B,[p1 p2 p3]);
sys_cl = ss(A-B*K,B,C,0);
lsim(sys_cl,u,t,x0);
```



This time the overshoot is smaller. Consult your textbook for further suggestions on choosing the desired closed-loop poles.

Compare the control effort required ( $K$ ) in both cases. In general, the farther you move the poles, the more control effort it takes.

---

Note: If you want to place two or more poles at the same position, `place` will not work. You can use a function called `acker` which works similarly to `place`:

```
K = acker(A,B,[p1 p2 p3])
```

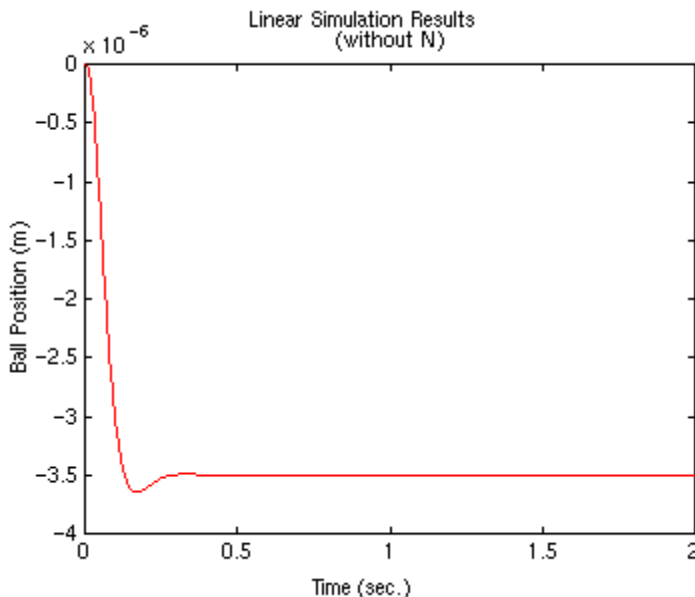
---

## Introducing the reference input

Now, we will take the control system as defined above and apply a step input (we choose a small value for the step, so we remain in the region where our linearization is valid).

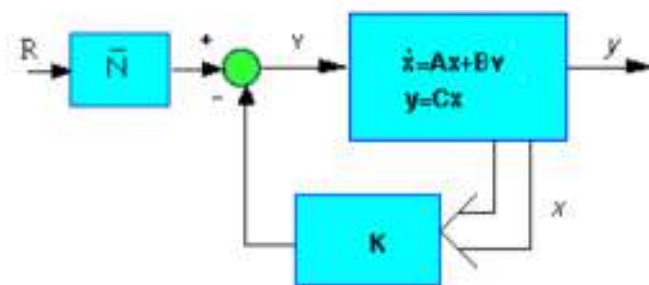
Replace `t`, `u` and `lsim` in your m-file with the following,

```
t = 0:0.01:2;
u = 0.001*ones(size(t));
sys_cl = ss(A-B*K,B,C,0);
lsim(sys_cl,u,t)
```



The system does not track the step well at all; not only is the magnitude not one, but it is negative instead of positive!

Recall the schematic above, we don't compare the output to the reference; instead we measure all the states, multiply by the gain vector  $K$ , and then subtract this result from the reference. There is no reason to expect that  $K*x$  will be equal to the desired output. To eliminate this problem, we can scale the reference input to make it equal to  $K*x_{\text{steadystate}}$ . This scale factor is often called  $Nbar$ ; it is introduced as shown in the following schematic:

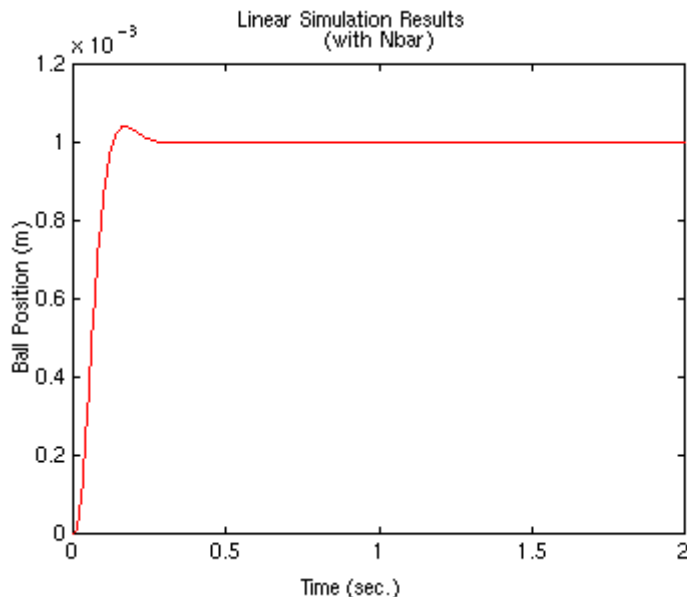


We can get  $Nbar$  from MATLAB by using the function `rscale` (place the following line of code after `K = ...`).

```
Nbar=rscale(sys,K)
```

Note that this function is not standard in MATLAB. You will need to copy it to a new m-file to use it. Click [here](#) for more information on using functions in MATLAB. Now, if we want to find the response of the system under state feedback with this introduction of the reference, we simply note the fact that the input is multiplied by this new factor, **Nbar**:

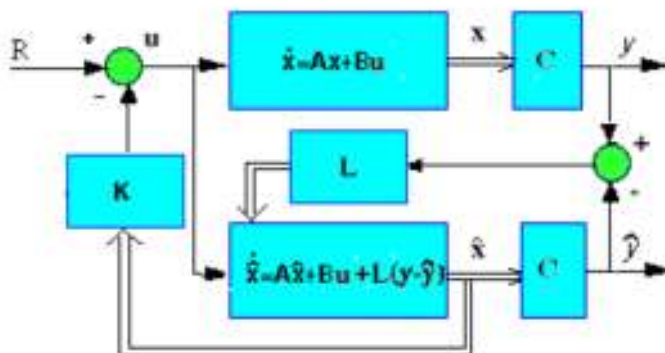
```
lsim(sys_cl,Nbar*u,t)
```



and now a step can be tracked reasonably well.

## Observer design

When we can't measure all the states  $x$  (as is commonly the case), we can build an observer to estimate them, while measuring only the output  $y = Cx$ . For the magnetic ball example, we will add three new, estimated states to the system. The schematic is as follows:



The observer is basically a copy of the plant; it has the same input and almost the same differential equation. An extra term compares the actual measured output  $y$  to the estimated output  $\hat{y}$ ; this will cause the estimated states  $\hat{x}$  to approach the values of the actual states  $x$ . The error dynamics of the observer are given by the poles of  $(A - L \cdot C)$ .

First we need to choose the observer gain  $L$ . Since we want the dynamics of the observer to be much faster than the system itself, we need to place the poles at least five times farther to the left than the dominant poles of the system. If we want to use `place`, we need to put the three observer poles at different locations.

```

op1 = -100;
op2 = -101;
op3 = -102;

```

Because of the duality between controllability and observability, we can use the same technique used to find the control matrix, but replacing the matrix B by the matrix C and taking the transposes of each matrix (consult your text book for the derivation):

```
L = place(A',C',[op1 op2 op3])';
```

The equations in the block diagram above are given for  $\hat{\mathbf{x}}$ . It is conventional to write the combined equations for the system plus observer using the original state  $\mathbf{x}$  plus the error state:  $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}$ . We use as state feedback  $\mathbf{u} = -\mathbf{K}\hat{\mathbf{x}}$ . After a little bit of algebra (consult your textbook for more details), we arrive at the combined state and error equations with the full-state feedback and an observer:

```

At = [A - B*K      B*K
      zeros(size(A))  A - L*C];
Bt = [  B*Nbar
      zeros(size(B))];
Ct = [  C      zeros(size(C))];

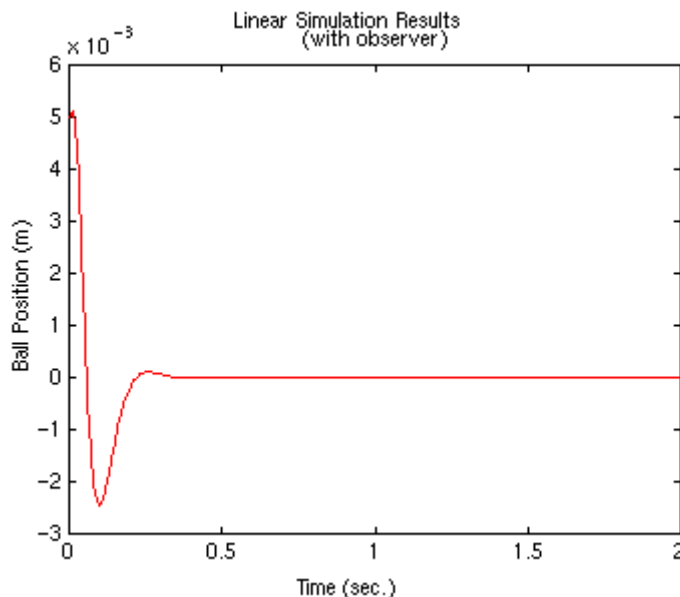
```

To see how the response looks to a nonzero initial condition with no reference input, add the following lines into your m-file. We typically assume that the observer begins with zero initial condition,  $\hat{\mathbf{x}} = 0$ . This gives us that the initial condition for the error is equal to the initial condition of the state.

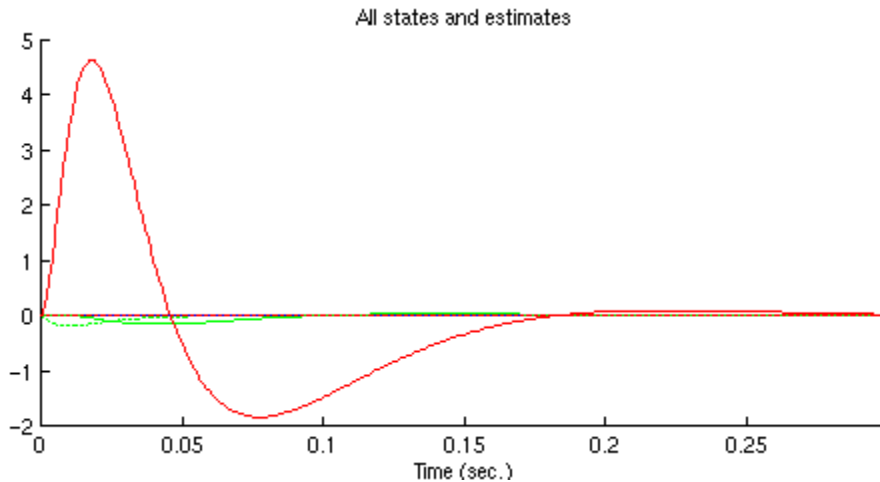
```

sys=ss (At,Bt,Ct,0);
lsim(sys,zeros(size(t)),t,[x0 x0])

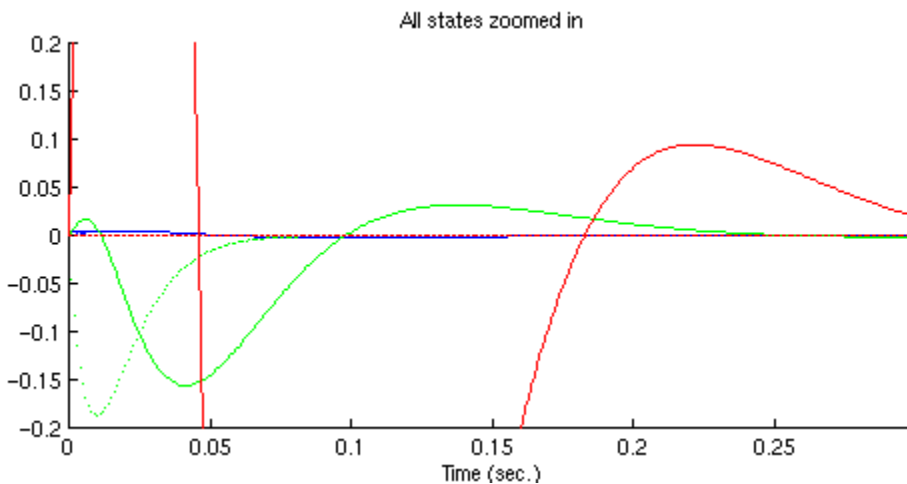
```



Responses of all the states are plotted below. Recall that lsim gives us  $\mathbf{x}$  and  $\mathbf{e}$ ; to get  $\hat{\mathbf{x}}$  we need to compute  $\mathbf{x} - \mathbf{e}$ .



Zoom in to see some detail:



The blue solid line is the response of the ball position  $\Delta h$ , the blue dotted line is the estimated state  $\hat{\Delta h}$ ;

The green solid line is the response of the ball speed  $\dot{\Delta h}$ , the green dotted line is the estimated state  $\hat{\dot{\Delta h}}$ ;

The red solid line is the response of the current  $\Delta i$ , the red dotted line is the estimated state  $\hat{\Delta i}$ .

We can see that the observer estimates the states quickly and tracks the states reasonably well in the steady-state.

The plot above can be obtained by using the `plot` command as follows:

```
[y,t,x] = lsim(sys,zeros(size(t)),t,[x0 x0]);
plot(t,x)
axis([0,.3,-.2,.2])
```

## State Space Examples

[Cruise Control](#) | [Motor Speed](#) | [Motor Position](#) | [Bus Suspension](#) | [Inverted Pendulum](#)

[Pitch Controller](#) | [Ball & Beam](#)

## Tutorials

[MATLAB Basics](#) | [MATLAB Modeling](#) | [PID Control](#) | [Root Locus](#) | [Frequency Response](#) | [State Space](#) | [Digital Control](#) | [Simulink Basics](#) | [Simulink Modeling](#) | [Examples](#)

[Home Page](#)

[MATLAB®  
Commands](#)

[simulink®  
Blocks](#)

[Tutorial  
Index](#)