Introduction: MATLAB was *created* for use in state space, and its enormous capabilities go far beyond the scope of a single introductory course like MAE 340. Here, we highlight some of the most fundamental usage.

Getting started in MATLAB: Our standard-form state space equations are given by:

$$\dot{\vec{z}}(t) = A\vec{z} + B\vec{u}$$
$$\vec{y} = C\vec{z} + D\vec{u}$$

In MATLAB, the standard model form is understood. We thus define any system by simply inputting the A, B, C, and D matrices.

MATLAB function ss

used for convenience, to assign a name to a state-space system. The format of the command is

$$>> \text{ systemname} = \text{ss (A,B,C,D)}$$

which creates an *object* in MATLAB, named *systemname* , that defines the state space model

Example: In the state-space notes, we converted

$$\dddot{x}(t) + 4\ddot{x}(t) - 3\dot{x}(t) - 12x(t) = 5sin(8t) + 12 + 3e^{-t}$$

where the output is $7\ddot{x} + 3\dot{x} + 2x$, into the state-space model:

$$\vec{z}(t) = \begin{pmatrix} z_1(t) \\ z_2(t) \\ z_3(t) \end{pmatrix} \equiv \begin{pmatrix} x(t) \\ \dot{x}(t) \\ \ddot{x}(t) \end{pmatrix} \quad , \quad \vec{u}(t) = \begin{pmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \end{pmatrix} \equiv \begin{pmatrix} sin(8t) \\ 1 \\ e^{-t} \end{pmatrix}$$

$$\dot{\vec{z}}(t) = \begin{pmatrix} \dot{z}_1(t) \\ \dot{z}_2(t) \\ \dot{z}_3(t) \end{pmatrix} \equiv \begin{pmatrix} z_2(t) \\ z_3(t) \\ 12z_1(t) + 3z_2(t) - 4z_3(t) + 5u_1(t) + 12u_2(t) + 3u_3(t) \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 12 & 3 & -4 \end{pmatrix} \vec{z}(t) + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 5 & 12 & 3 \end{pmatrix} \vec{u}(t) = A\vec{z} + B\vec{u}$$

and the output equation:

$$\vec{y}(t) = \left(7\ddot{x} + 3\dot{x} + 2x\right) = \left(2z_1 + 3z_2 + 7z_3\right)$$

$$= \begin{pmatrix} 2 & 3 & 7 \end{pmatrix} \vec{z}(t) + \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} \vec{u}(t)$$

$$= C\vec{z}(t) + D\vec{u}(t)$$

To create this model in MATLAB and name it Model1, we may use the following MATLAB commands:

```
>> A = [ 0 1 0 ; 0 0 1 ; 12 3 -4 ];
>> B = [ 0 0 0 ; 0 0 0 ; 5 12 3 ];
>> C = [ 2 3 7 ] ;
>> D = [ 0 0 0 ];
>> Model1 = ss (A,B,C,D);
```

We may also use the following single command:

```
>> Model1 = ss ([ 0 1 0 ; 0 0 1 ; 12 3 -4 ] , [ 0 0 0 ; 0 0 0 ; 5 12 3 ] , [ 2 3 7 ] , [ 0 0 0 ]);
```

Regardless of how it is created, the object **Model1** may now be used as the input to many other commands.

MATLAB functions eig, poly, det, and solve

The *eigenvalues* of any matrix $A_{n\times n}$ are the n roots of the *determinant* polynomial defined by:

$$|A-\lambda I_{n\times n}| \equiv \det \underbrace{\begin{pmatrix} A_{11}-\lambda & A_{12} & ... & A_{1n} \\ A_{21} & A_{22}-\lambda & ... & A_{2n} \\ ... & & & \\ A_{n1} & A_{n2} & ... & A_{nn}-\lambda \end{pmatrix}}_{n\times n \text{ matrix}} = n^{th}\text{-order polynomial in } \lambda \qquad (1)$$

Eq. (1) defines an n^{th}-order polynomial in λ; the n roots of this polynomial are A's eigenvalues.

The eigenvalues of a system's state matrix A are equal to the roots of its characteristic equation

Example:
Consider the LTI ODE model

$$\ddot{x}+\dot{x}+9x = 2t-3e^{-2t}$$

We may define $x_h(t)=Ce^{\lambda t}$ and substitute to find the characteristic equation

$$\lambda^2+\lambda+9=0$$

Now convert the system into state space, and find the eigenvalues of the state matrix A:

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} x \\ \dot{x} \end{pmatrix} ; \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} t \\ e^{-2t} \end{pmatrix} ; \begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \end{pmatrix} = \begin{pmatrix} z_2 \\ -9z_1 - z_2 + 2u_1 - 3u_2\} \end{pmatrix}$$

$$\Rightarrow A = \begin{pmatrix} 0 & 1 \\ -9 & -1 \end{pmatrix} ; B = \begin{pmatrix} 0 & 0 \\ 2 & -3 \end{pmatrix}$$

The eigenvalues of the A matrix are found from

$$|A-\lambda I| = \det \begin{pmatrix} 0-\lambda & 1 \\ -9 & -1-\lambda \end{pmatrix} = \lambda^2+\lambda+9=0$$

which is exactly equal to the characteristic equation. Thus, the eigenvalues of A are the roots of the CE.

1. The MATLAB function **eig** may be used to get eigenvalues of an A matrix (i.e., roots of the CE):

   >> lambda = eig(A)

   The actual characteristic equation itself can be constructed from its roots using **poly**:

   >> CE = poly (eig(A));

   The vector CE produced by this command contains the coefficients of the n^{th}-order polynomial defined by Eq. (1), in order from highest-power to lowest, scaled so that the highest power coefficient is always 1.

   Function **eig** may also be applied to *symbolic* matrices, but for our purposes, the combination of functions **det** and **solve** are usually more useful.

2. **MATLAB functions det , solve**

   det finds the determinant of a matrix, and may be applied to symbolic matrices; thus, we may use **det** along with Eq. (1) to find the characteristic equation for systems with algebraic coefficients in their LTI ODE's.

   Then, if we wish to obtain the algebraic expressions for the actual eigenvalues, we may use **solve** on the symbolic output of **det**

Example:
Consider the LTI ODE model

$$\ddot{x} + c\dot{x} + 9x = 2t - 3e^{-2t}$$

We may define $x_h(t) = Ce^{\lambda t}$ and substitute to find the characteristic equation

$$\lambda^2 + c\lambda + 9 = 0$$

Now convert the system into state space, and find the eigenvalues of the state matrix A:

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} x \\ \dot{x} \end{pmatrix} \; ; \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} t \\ e^{-2t} \end{pmatrix} \; ; \; \begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \end{pmatrix} = \begin{pmatrix} z_2 \\ -9z_1 - cz_2 + 2u_1 - 3u_2\} \end{pmatrix}$$

$$\Rightarrow A = \begin{pmatrix} 0 & 1 \\ -9 & -c \end{pmatrix} \; ; \; B = \begin{pmatrix} 0 & 0 \\ 2 & -3 \end{pmatrix}$$

The eigenvalues of the A matrix are found from

$$|A - \lambda I| = \det \begin{pmatrix} 0-\lambda & 1 \\ -9 & -c-\lambda \end{pmatrix} = \lambda^2 + c\lambda + 9 = 0$$

which is exactly equal to the characteristic equation.

We now illustrate the use of **det** to obtain this characteristic equation:

```
>> syms A c lambda
>> A = [ -lambda 1 ; -9 -c-lambda] ;
>> det(A)
    ans  = lambda^2 + c*lambda + 9
```

There are other combinations of commands that can do this. The MATLAB function **eye(n)**, for example, represents the $n \times n$ identity matrix, so we could let MATLAB construct $|A - \lambda I|$ for us:

```
>> A=[ 0 1 ; -9 -1 ]
>> syms lambda
>> det(A-lambda*eye(2))
    ans  =  lambda^2 + lambda + 9
```

If you want the solution for lambda, you can use MATLAB function **solve**:

```
>>  solve(det(A-lambda*eye(2))), lambda)
                                        ans =
                                            (35^(1/2) * i)/2 - 1/2
                                            - (35^(1/2) * i)/2 - 1/2
```

solve can also be used on characteristic equations with unknown constants:

```
>> syms A c lambda
>> Alam = [ 0 1 ; -9 -c-] - lambda*eye(2);
>> solve(det(Alam), lambda )
    ans  =
((c - 6) * (c + 6))^(1/2)/2 - c/2
- c/2 - ((c - 6) * (c + 6))^(1/2)/2
```

Note that the output of a symbolic **solve** may be less useful than drawing the root locus.

Example:
MATLAB function **eig** may be used to determine the roots of the characteristic equation for the system

$$5\dddot{x} + 3\ddot{x} + 12\dot{x} + x = 2t - 3e^{-2t}$$

Defining the state and input vectors in the usual way, we obtain:

$$\begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix} ; \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} t \\ e^{-2t} \end{pmatrix} ; \begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \end{pmatrix} = \begin{pmatrix} z_2 \\ z_3 \\ 1/5\{-z_1 - 12z_2 - 3z_3 + 2u_1 - 3u_2\} \end{pmatrix}$$

$$\Rightarrow A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -0.2 & -2.4 & -0.6 \end{pmatrix} ; B = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0.4 & -0.6 \end{pmatrix}$$

The MATLAB commands

```
>> A=[0 1 0 ; 0 0 1 ; -0.2 -2.4 -0.6] ; lambda = eig(A)
```

or simply

```
>>  lambda = eig([0 1 0 ; 0 0 1 ; -0.2 -2.4 -0.6])
```

produce the roots of the characteristic equation:

```
lambda  =
        - 0.0849
        - 0.2576 + 1.5133i
        - 0.2576 + 1.5133i
```

The coefficients of the characteristic equation itself can be found from

```
>>  CE = poly ( eig (A) ))

CE = 1.0000 0.6000 2.4000 0.2000
```

Note that **poly** automatically scales the coefficient of the highest-order term to 1.

If we want to find the settling time, we could use (for example)

```
>>  settling = -4/max(real(eig(A)))
```

MATLAB function initial

The MATLAB function **initial** can be used to calculate the *homogeneous* solution of an LTI ODE:

$$>> \text{ initial (model , z0)}$$

where *model* is the name of the system created by **ss** , and z0 is a vector of the initial conditions corresponding to the states, i.e., it is $\vec{z}(0)$

Use MATLAB function **initial** to find the homogeneous solution of

$$\dddot{x} + 3\ddot{x} + 12\dot{x} + 6x = 2t - 3e^{-2t}$$

to the initial conditions $x(0) = 10$, $\dot{x}(0) = -2$, $\ddot{x}(0) = 0$

The states and inputs are defined as

$$\begin{aligned} z_1 &\equiv x(t) & u_1(t) &= t \\ z_2 &\equiv \dot{x}(t) & u_2(t) &= e^{-2t} \\ z_3 &\equiv \ddot{x}(t) \end{aligned}$$

Using these definitions, the state equations are

$$\dot{\vec{z}}(t) = \begin{pmatrix} \dot{z}_1(t) \\ \dot{z}_2(t) \\ \dot{z}_3(t) \end{pmatrix} \equiv \begin{pmatrix} z_2(t) \\ \dot{z}_3(t) \\ -6z_1(t) - 12z_2(t) - 3z_3(t) + 2u_1(t) - 3u_2(t) \end{pmatrix}$$

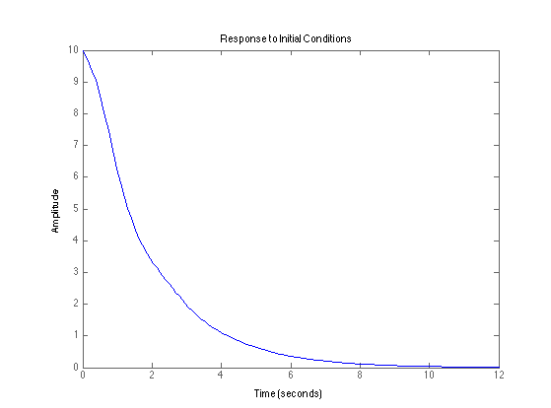
$$= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -6 & -12 & -3 \end{pmatrix} \vec{z}(t) + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 2 & -3 \end{pmatrix} \vec{u}(t) = A\vec{z} + B\vec{u}$$

Let's assume that our only desired output is $x(t)$:

$$\vec{y}(t) = (x) = (z_1) = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \vec{z}(t) + \begin{pmatrix} 0 & 0 \end{pmatrix} \vec{u}(t) = C\vec{z}(t) + D\vec{u}(t)$$

The following MATLAB commands produce the homogeneous solution for the specified initial conditions:

```
>> model = ss([ 0 1 0 ; 0 0 1 ; -6 -12 -3 ], [ 0 0 ; 0 0 ; 2 -3 ],[ 1 0 0 ] , [ 0 0 ])
>> initial( model, [ 10 -2 0 ] )
```



The output is simply the plot.

If desired, the numerical solution may be stored as vectors:
>> [Y, T] = initial (model, z0)
stores the vector output Y at specific times T

The states can also be stored:
>> [Y, T, X] = initial (model, z0)
stores the n states at each T

Example: HW3 reworked using MATLAB functions **eig** and **initial**

Original HW3: For each of the following homogeneous LTI ODE's, with accompanying initial conditions, use MATLAB to do the following:

1. Find the roots of the characteristic equation (function ROOTS)
2. Apply the initial conditions to find the complete homogeneous solution (function INV)
3. Plot the solution (functions PLOT, TITLE, XLABEL, YLABEL, LEGEND, GRID)

Solution using state-space functions:

(a)

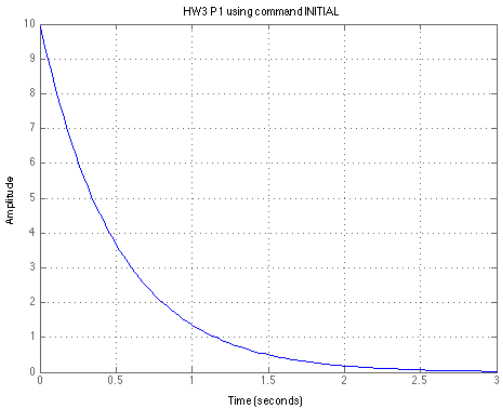
$$\dot{x}(t) + 2x(t) = 0 \qquad , \qquad x(0) = 10$$

Solution

$$(z_1) = (x) \;\; ; \;\; (\dot{z}_1) = (-2z_1) \;\; ; \;\; (y) = (z_1)$$
$$\Rightarrow A = (-2) \;\; ; \;\; B = (0) \;\; ; \;\; C = (1) \;\; ; \;\; D = (0)$$

In MATLAB:

```
>> P1 = ss( [-2] , [0] , [1] , [0]
>> initial(P1 , [10])
>> lambda=eig([-2])
```



(b)

$$3\ddot{x}(t) = 0 \quad , \quad x(0) = 1 \ , \ \dot{x}(0) = 3$$

Solution

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} x \\ \dot{x} \end{pmatrix} \ ; \ \begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \end{pmatrix} = \begin{pmatrix} z_2 \\ 0 \end{pmatrix} \ ; \ (y) = (z_1)$$

$$\Rightarrow A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \ ; \ B = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \ ; \ C = \begin{pmatrix} 1 & 0 \end{pmatrix} \ ; \ D = \begin{pmatrix} 0 \end{pmatrix}$$

In MATLAB:

```
>> P2 = ss( [0 1 ; 0 0 ] , [0 ; 0] , [1 0] , [0]
>> lambda=eig([0 1 ; 0 0])
>> initial(P2 , [1 3])
```

(c)

$$\ddot{x}(t) + \dot{x}(t) + 9x(t) = 0 \quad , \quad x(0) = 5 \ , \ \dot{x}(0) = 1$$

Solution

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} x \\ \dot{x} \end{pmatrix} \ ; \ \begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \end{pmatrix} = \begin{pmatrix} z_2 \\ -9z_1 - z_2 \end{pmatrix} \ ; \ (y) = (z_1)$$

$$\Rightarrow A = \begin{pmatrix} 0 & 1 \\ -9 & -1 \end{pmatrix} \ ; \ B = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \ ; \ C = \begin{pmatrix} 1 & 0 \end{pmatrix} \ ; \ D = \begin{pmatrix} 0 \end{pmatrix}$$

In MATLAB:

```
>> P3 = ss( [0 1 ; -9 -1 ] , [0 ; 0] , [1 0] , [0]
>> lambda=eig([0 1 ; -9 -1 ] )
>> initial(P3 , [5 1])
```

(d)

$$\ddot{x}(t) + 9\dot{x}(t) + 9x(t) = 0 \quad , \quad x(0) = 5 \ , \ \dot{x}(0) = 1$$

Solution

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} x \\ \dot{x} \end{pmatrix} \ ; \ \begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \end{pmatrix} = \begin{pmatrix} z_2 \\ -9z_1 - 9z_2 \end{pmatrix} \ ; \ (y) = (z_1)$$

$$\Rightarrow A = \begin{pmatrix} 0 & 1 \\ -9 & -9 \end{pmatrix} \ ; \ B = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \ ; \ C = \begin{pmatrix} 1 & 0 \end{pmatrix} \ ; \ D = \begin{pmatrix} 0 \end{pmatrix}$$

In MATLAB:

```
>> P4 = ss( [0 1 ; -9 -9 ] , [0 ; 0] , [1 0] , [0]
>> lambda=eig(A)
>> initial(P4 , [5 1])
```

(e)

$$\dddot{x}(t) + 6\ddot{x}(t) + 11\dot{x}(t) + 6x(t) = 0 \quad , \quad x(0) = 5 \ , \ \dot{x}(0) = 1 \ , \ \ddot{x}(0) = 1$$

Solution

$$\begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix} \ ; \ \begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \end{pmatrix} = \begin{pmatrix} z_2 \\ z_3 \\ -6z_1 - 11z_2 - 6z_3 \end{pmatrix} \ ; \ (y) = (z_1)$$

$$\Rightarrow A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -6 & -11 & -6 \end{pmatrix} \ ; \ B = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \ ; \ C = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \ ; \ D = \begin{pmatrix} 0 \end{pmatrix}$$

In MATLAB:

```
>> P5 = ss( [0 1 0 ; 0 0 1 ; -6 -11 -6] , [0 ; 0 ; 0] , [1 0 0] , [0]
>> lambda=eig(A)
>> initial(P5 , [5 1 1])
```

(f)

$$\dddot{x}(t) + 2\ddot{x}(t) + 10\dot{x}(t) + 9x(t) = 0 \quad , \quad x(0) = 5 \ , \ \dot{x}(0) = 1 \ , \ \ddot{x}(0) = 0$$

Solution

$$\begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix} \ ; \ \begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \end{pmatrix} = \begin{pmatrix} z_2 \\ z_3 \\ -9z_1 - 10z_2 - 2z_3 \end{pmatrix} \ ; \ (y) = (z_1)$$

$$\Rightarrow A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -9 & -10 & -2 \end{pmatrix} \ ; \ B = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \ ; \ C = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \ ; \ D = \begin{pmatrix} 0 \end{pmatrix}$$

In MATLAB:

```
>> P6 = ss( [0 1 0 ; 0 0 1 ; -9 -10 -2] , [0 ; 0 ; 0] , [1 0 0] , [0]
>> lambda=eig(A)
>> initial(P5 , [5 1 0])
```

MATLAB function step

Calculates the step response(s) of a state-space system.

For systems with multiple inputs and/or outputs, **step** sets *one input at a time* equal to the unit step input, and all other inputs equal to zero. It then calculates and plots each individual output for that step input. It then repeats this process, one-at-a-time, for each input. Thus, the result of a single call to **step** is a set of $m * p$ plots, which show the individual unit step responses of each output to each input, where the inputs are used one-at-a-time.

Example:

Find the unit step response of the system

$$\dddot{x}(t) + 2\ddot{x}(t) + 10\dot{x}(t) + 9x(t) = f(t)$$

Solution

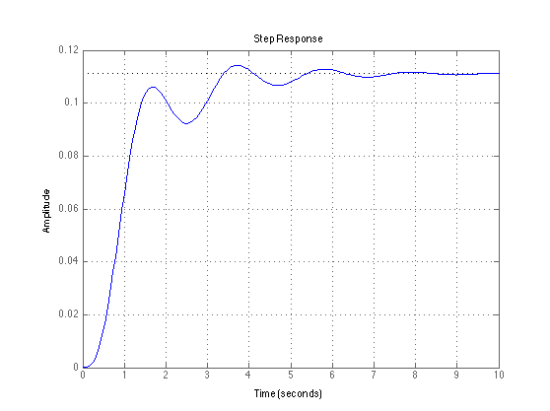
$$\begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix} \; ; \; \begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \end{pmatrix} = \begin{pmatrix} z_2 \\ z_3 \\ -9z_1 - 10z_2 - 2z_3 + f(t) \end{pmatrix} \; ; \quad (y) = (z_1)$$

$$\Rightarrow A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -9 & -10 & -2 \end{pmatrix} \; ; \; B = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \; ; \; C = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \; ; \; D = \begin{pmatrix} 0 \end{pmatrix}$$

In MATLAB:

```
>> Example = ss( [0 1 0 ; 0 0 1 ; -9 -10 -2] , [0 ; 0 ; 1] , [1 0 0] , [0]
>> step (Example)
```

The output of this command is a plot of the step response:



Like most MATLAB commands, the calculated response can be stored as numbers:

```
>>  [ Y,T ] = step(Example)
```

stores the calculated response Y at the discrete times in T. If for some reason all of the states are also desired,

```
>>  [ Y,T,Z ] = step(Example)
```

may be used.

The user may also specify the final time of the solution:

```
>>  step(Example , tfinal)
```

or, if desired, the entire time vector to be used

```
>>  step(Example , T )
```

Using the property of linearity, we can use **step** to find the response of system

$$\dddot{x}(t) + 2\ddot{x}(t) + 10\dot{x}(t) + 9x(t) = f(t) = \begin{pmatrix} 0 & t < 0 \\ 14.7 & t \geq 0 \end{pmatrix}$$

The response can be found as

```
>>  14.7*step(Example)
```

MATLAB function lsim

The most *generic* MATLAB function for calculating LTI ODE solutions is **lsim** , which calculates the complete solution for a forced system with any initial conditions.

lsim can do everything that **step** and **initial** can do, plus much more.

In its most basic form, we use the command

```
>> lsim(sys , u , t, z0)
```

to plot the outputs of the system named sys (defined using **ss**) subject to the initial conditions z0 and the forcing u(t). Note the following:

1. The *vectors* u and t must be the same length; and, the elements of t must be evenly spaced

   (a) The following creates an input of $sin(4t)$ from $0 \leq t \leq 10$ seconds in increments of 0.01 sec:

   ```
   >> t = [0 : .01 : 10] ; u = sin(4*t);
   ```

   (b) The following creates a constant input of $u = 22$ from $0 \leq t \leq 5.5$ seconds in increments of 0.05 sec:

   ```
   >> t = [0 : .05 : 5.5] ; u = 22*ones(length(t));
   ```

   (c) The following creates multiple inputs $u_1 = 5$, $u_2 = 3e^{-2t}$, and $u_3 = 12cos(2t)$, for $0 \leq t \leq 8$:

   ```
   >> t = [o : .01 : 8] ; u = [5*ones(length(t) ; 3*exp(-2*t) ; 12*cos(2*t)]' ;
   ```

   Note the *transpose* symbol ' at the end of the definition of u. For multiple inputs in a state-space model, each individual input should be its own column in u for **lsim**. The individual columns must all be the same length, and they must all correspond to the same t vector.

2. The vector z0 may be omitted if the initial conditions are zero. Otherwise, it should be a *column vector* of initial conditions corresponding to the states.

3. The command

   ```
   >> lsim(sys,u,t,z0)
   ```

   draws a plot of the system outputs for the input vector u. If you want to store outputs for some other use,

   ```
   >> [Y,T]=lsim(sys, u, t, z0)
   ```

   may be used. Note that, just like **step** and **initial**, if this form is used, no plot is generated.

   If the full states are desired, they may also be stored, as in **step** and **initial**:

   ```
   >> [Y,T,Z]=lsim(sys,u,t,z0)
   ```

4. For example, to simulate the command **initial** for initial conditions z0, for the time period $0 \leq t \leq tfinal$,

   ```
   >> t=[0: .01: tfinal] ; u=0*ones(1,length(t)) ; lsim(sys, u, t, z0)
   ```

5. For example, to simulate the command **step**, for a system with one input, for the time period $0 \leq t \leq tfinal$,

   ```
   >> t=[0: .01: tfinal] ; u=ones(1,length(t)) ; lsim(sys, u, t )
   ```

Example: HW12

Find and plot the complete solution until it settles, for the following system:

$$\ddot{x}(t) + 0.6\dot{x}(t) + 9x(t) = f(t) \quad , \quad f(t) = \begin{pmatrix} 0 & t < 0 \\ 2t & 0 \le t \le 2 \\ 4 & 2 \le t \le 12 \\ 28 - 2t & 12 \le t \le 14 \\ 0 & t > 14 \end{pmatrix}$$

The initial conditions are $x(0) = 0$ and $\dot{x}(0) = 0$

Solution: We obtain the standard state-space form of this system as:

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} x \\ \dot{x} \end{pmatrix} \; ; \; \begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \end{pmatrix} = \begin{pmatrix} z_2 \\ -9z_1 - 0.6z_2 + u(t) \end{pmatrix} \; ; \; (y) = (z_1)$$

$$\Rightarrow A = \begin{pmatrix} 0 & 1 \\ -9 & -0.6 \end{pmatrix} \; ; \; B = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \; ; \; C = \begin{pmatrix} 1 & 0 \end{pmatrix} \; ; \; D = \begin{pmatrix} 0 \end{pmatrix}$$

In MATLAB:

Define the state-space system	>> HW12 = ss([0 1 ; -9 -0.6] , [0 ; 1] , [1 0] , [0])
Calculate the settling time	>> settle=-4/max(real(eig(A)))
Define the time vector	>> t=[0 : .01 : 14+settle];
Define the input vector	>> u= 0*ones(1,length(t);
	>> u(1:201)=2*t(1:201);
	>> u(202:1201)=4*ones(1,1000)
	>> u(1202:1401)=28*ones(1,200)-2*t(1202,1401);
Calculate and plot	>> lsim(HW12, u, t)