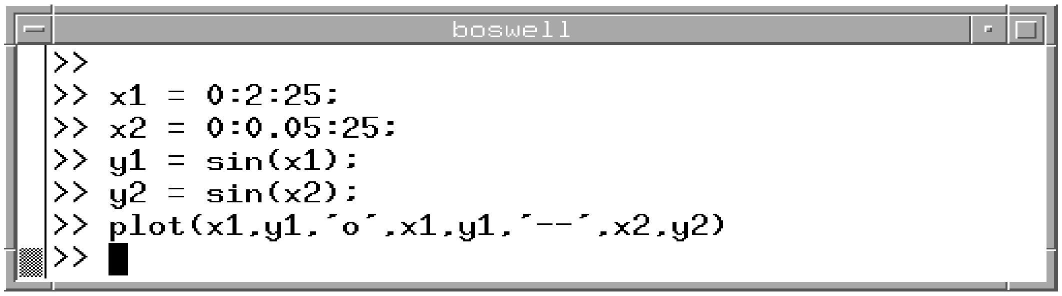
MATLAB TRAINING SESSION IV

SIMULATING DYNAMIC SYSTEMS

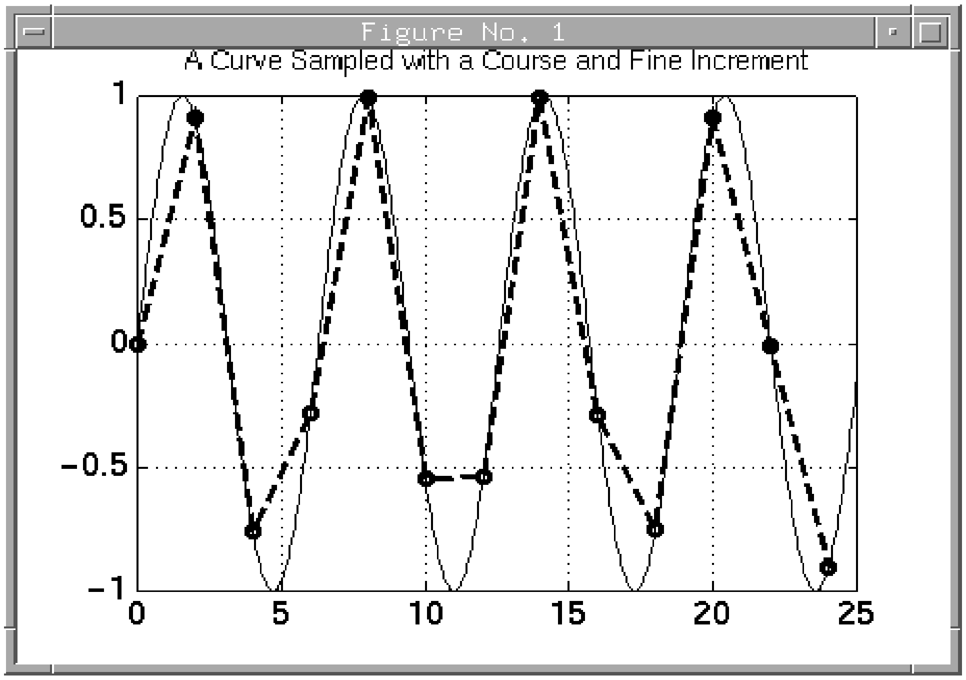
MATLAB provides a variety of techniques for solving dynamics problems numerically. On-line help for each of the commands shown below is available by typing `help 'command_name'` or `doc 'command_name'` at the MATLAB prompt.

Sampling the Solution Equation:

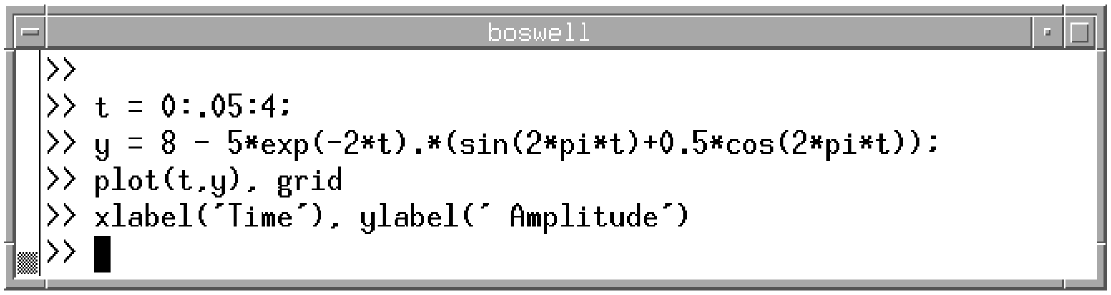
Suppose you already have the solution to a dynamics problem and you want to obtain a plot of the response. The task would reduce to implementing the equation in MATLAB and sampling the function over some domain of interest. For a high fidelity reproduction of the function, the function must be sampled with a spacial or time increment that captures many samples for each frequency that comprises the Fourier series for the function. In simpler terms, we can tell if we have used a small enough increment if the resulting plot appears smooth. The example shows how the same function can look jagged or smooth in the plot depending on the sampling increment.



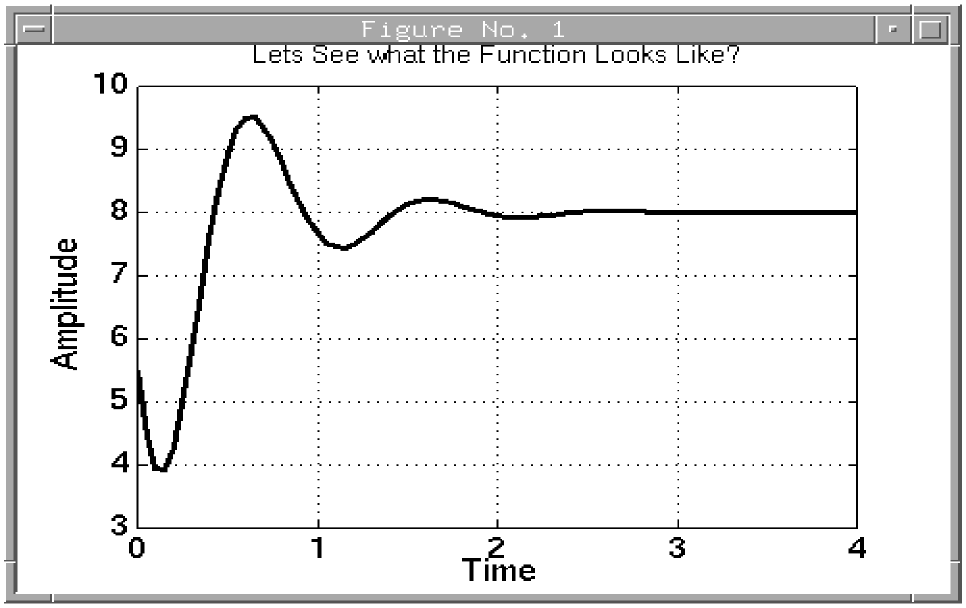
```
>>
>> x1 = 0:2:25;
>> x2 = 0:0.05:25;
>> y1 = sin(x1);
>> y2 = sin(x2);
>> plot(x1,y1,'o',x1,y1,'--',x2,y2)
>>
```



Suppose we find: $y(t) = 8 - 5e^{-2t}(\sin(2\pi t) + 0.5\cos(2\pi t))$



```
>>
>> t = 0:.05:4;
>> y = 8 - 5*exp(-2*t).*(sin(2*pi*t)+0.5*cos(2*pi*t));
>> plot(t,y), grid
>> xlabel('Time'), ylabel(' Amplitude')
>>
```



Linear Systems:

There are three general purpose methods for solving dynamics problems; transfer function method, state space integration method, and the MATLAB function `lsim`.

Transfer Function Method:

The transfer function method uses partial fraction expansion of the transfer function multiplied by the Laplace transform of the forcing function to obtain the solution. The solution of the differential equation is expressed in terms of a linear combonation of complex exponentials that are determined by the poles and the corresponding residues in the partial fraction expansion. Thus the form of the solution is:

$$y(t) = \sum_{i=1}^{n} r(i)\exp(p(i) \cdot t)$$

Suppose we have $\ddot{y} + 2\dot{y} + 5y = 5u(t) + \dot{u}(t)$ *and* $u(t) = input$

Then we can write

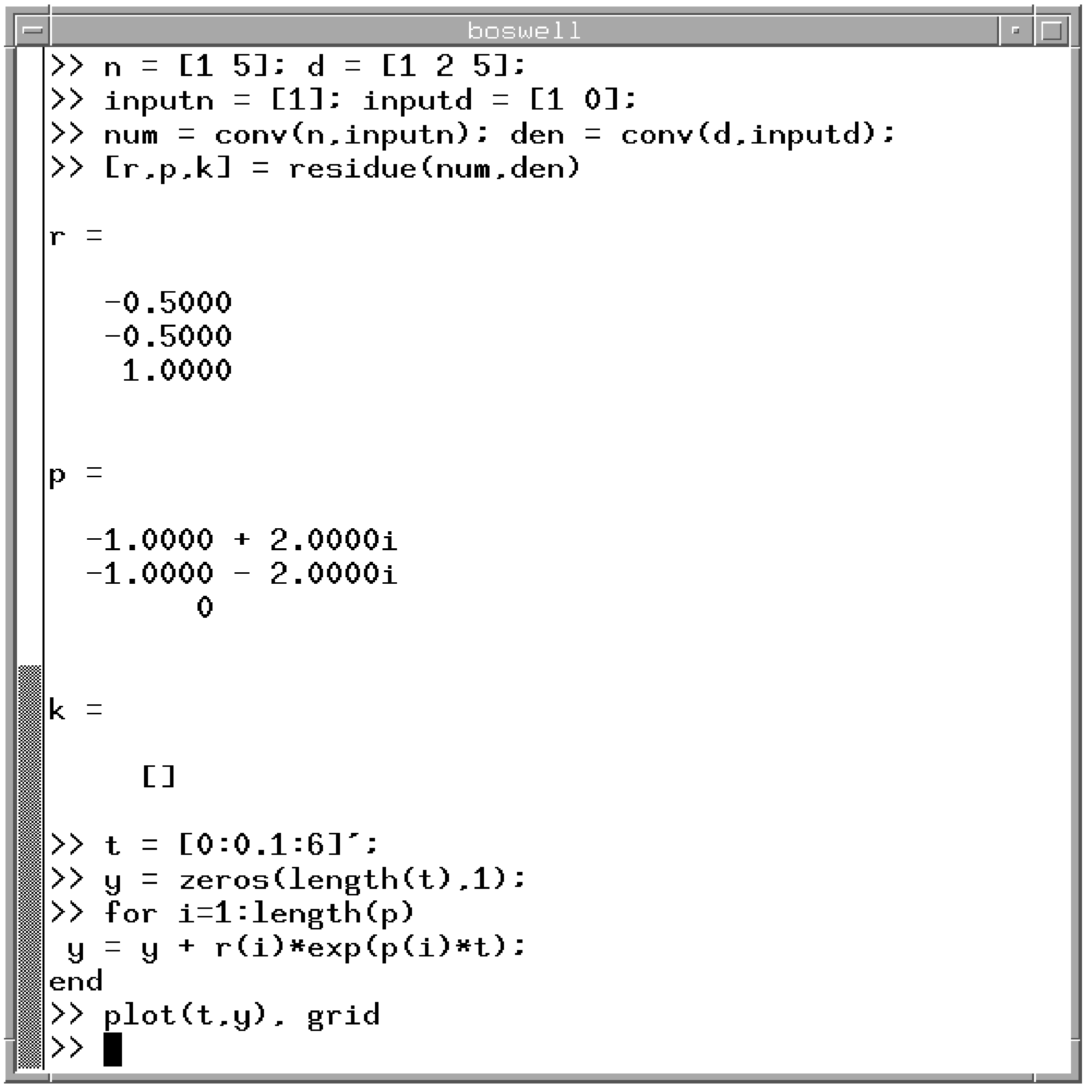
$$s^2Y(s) - sy(0) - \dot{y}(0) + 2sY(s) - 2y(0) + 5Y(s) = 5U(s) + sU(s) + u(0)$$

$$Let \qquad y(0) = 0 \qquad \dot{y}(0) = 0 \qquad u(0) = 0$$

$$Then \qquad Y(s) = \frac{s+5}{s^2+2s+5}U(s)$$

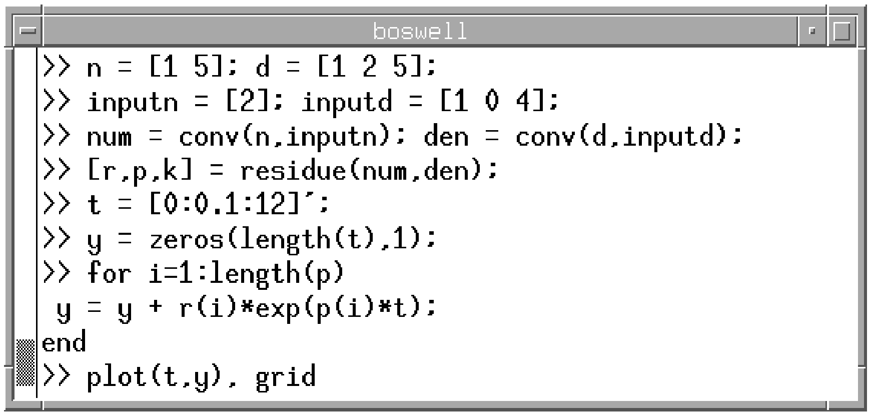
Let the input `u(t)` be the step function and to solve for `y(t)` do

With $U(s) = \frac{1}{s}$ then $Y(s) = \frac{s+5}{s(s^2+2s+5)}$

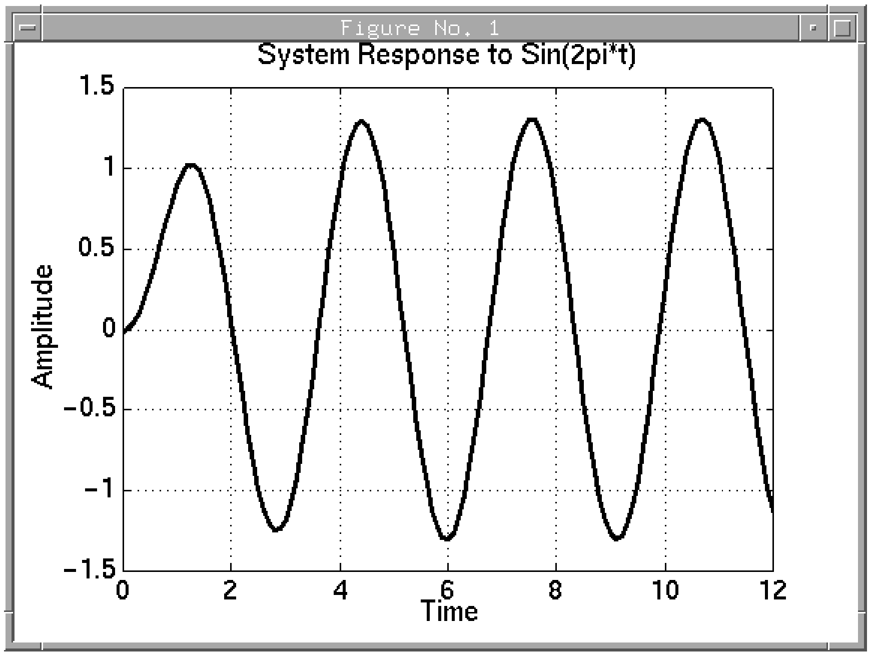


```
>> n = [1 5]; d = [1 2 5];
>> inputn = [1]; inputd = [1 0];
>> num = conv(n,inputn); den = conv(d,inputd);
>> [r,p,k] = residue(num,den)

r =

   -0.5000
   -0.5000
    1.0000


p =

  -1.0000 + 2.0000i
  -1.0000 - 2.0000i
        0


k =

     []

>> t = [0:0.1:6]';
>> y = zeros(length(t),1);
>> for i=1:length(p)
 y = y + r(i)*exp(p(i)*t);
end
>> plot(t,y), grid
>> 
```

Let `u(t)` be `sin(2t)` and to solve for `y(t)` do $U(s) = \frac{2}{s^2+4}$



```
>> n = [1 5]; d = [1 2 5];
>> inputn = [2]; inputd = [1 0 4];
>> num = conv(n,inputn); den = conv(d,inputd);
>> [r,p,k] = residue(num,den);
>> t = [0:0.1:12]';
>> y = zeros(length(t),1);
>> for i=1:length(p)
 y = y + r(i)*exp(p(i)*t);
end
>> plot(t,y), grid
```



Integration Method:

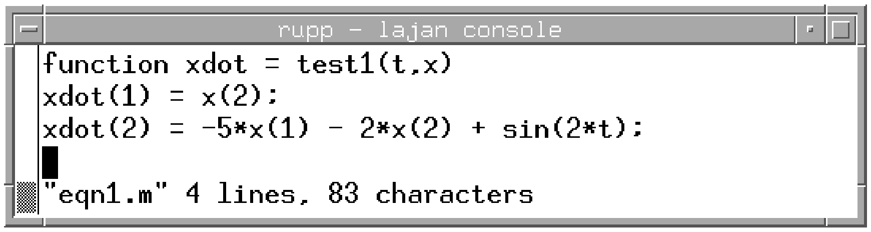
The integration method utilizes the state space representation of the system and the MATLAB `ode23` or the `ode45` integrator. One advantage of this method is that it is much easier to incorporate the initial conditions in the solution without having to find the new transfer function. Lets look at the last example given above.

The state space representation is

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ -5 & -2 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \qquad \dot{x} = Ax + Bu$$

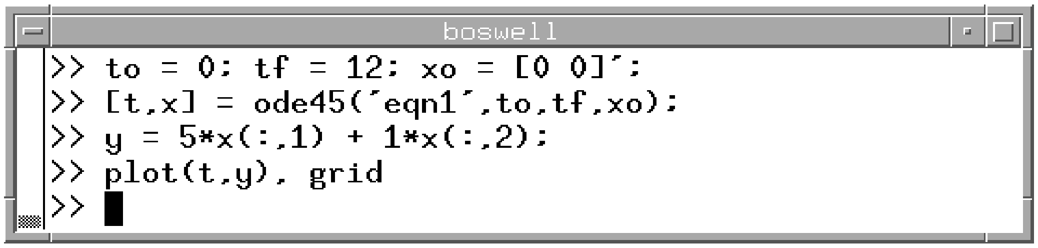
$$y = \begin{bmatrix} 5 & 1 \end{bmatrix} x(t) \qquad y = Cx + Du$$

To implement the model create the M-file eqn1.m as shown

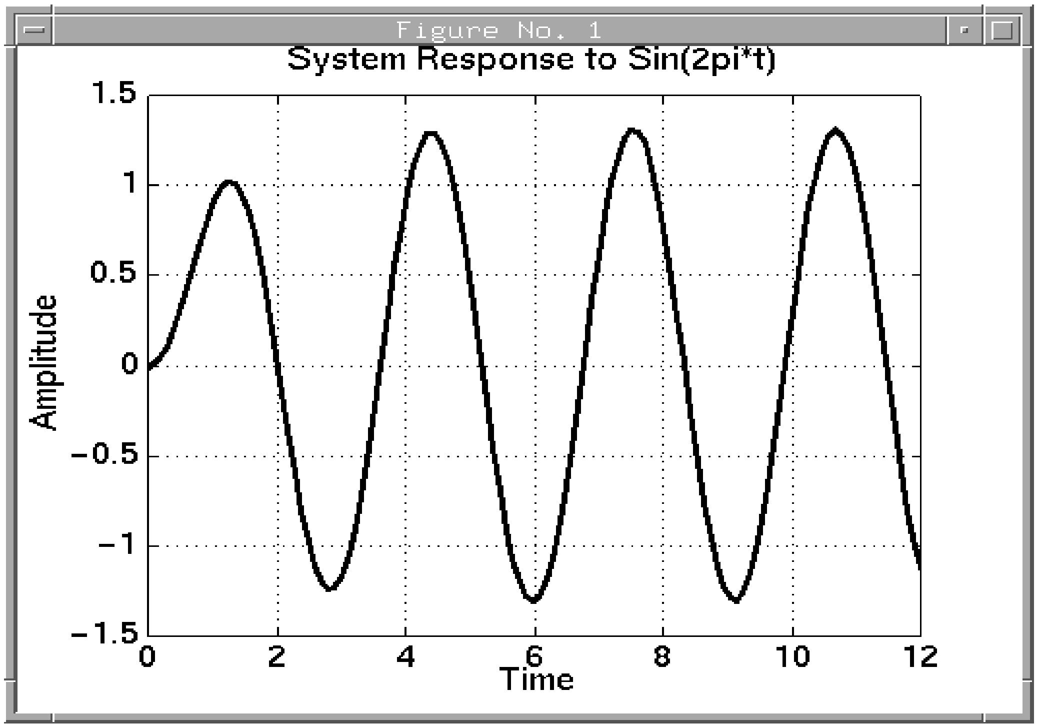


```
function xdot = test1(t,x)
xdot(1) = x(2);
xdot(2) = -5*x(1) - 2*x(2) + sin(2*t);

"eqn1.m" 4 lines, 83 characters
```

Then at the MATLAB prompt, enter:



```
>> to = 0; tf = 12; xo = [0 0]';
>> [t,x] = ode45('eqn1',to,tf,xo);
>> y = 5*x(:,1) + 1*x(:,2);
>> plot(t,y), grid
>> 
```

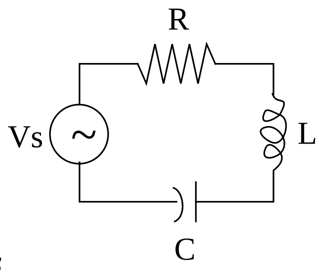


Special Purpose MATLAB Commands:
The MATLAB control systems toolbox provides several built in functions for solving for the solution to linear dynamic equations. All functions except either the transfer function or the state space representations. When the functions are invoked with left hand arguments, the variable are filled and a no plot is made. Each function has a variable number of possible valid argument configurations. Most function support both transfer function (num/den) and state space (A,B,C,D) forms. More information about each function can be found by typing `help 'function_name'`. A list is provided below.

• `initial(A,B,C,D,x0)`	Initial condition response of cont-time LTI systems
• `impulse(num,den,t)`	Impulse response of continuous time LTI systems
`impulse(A,B,C,D,IU,t)`	IU = input channel number; t = sample time vector
• `step(num,den,t)`	Step response of continuous time LTI systems
`step(A,B,C,D,IU,t)`	
• `lsim(num,den,U,t)`	Simulation of c-time LTI systems to arbitrary inputs
`lsim(A,B,C,D,U,t,x0)`	U = input time history (# of columns = # of inputs)

For example lets work with the series resonant circuit:

$$Ri + L\frac{di}{dt} + \frac{1}{C}\int_0^t i\,d\tau = V_s$$

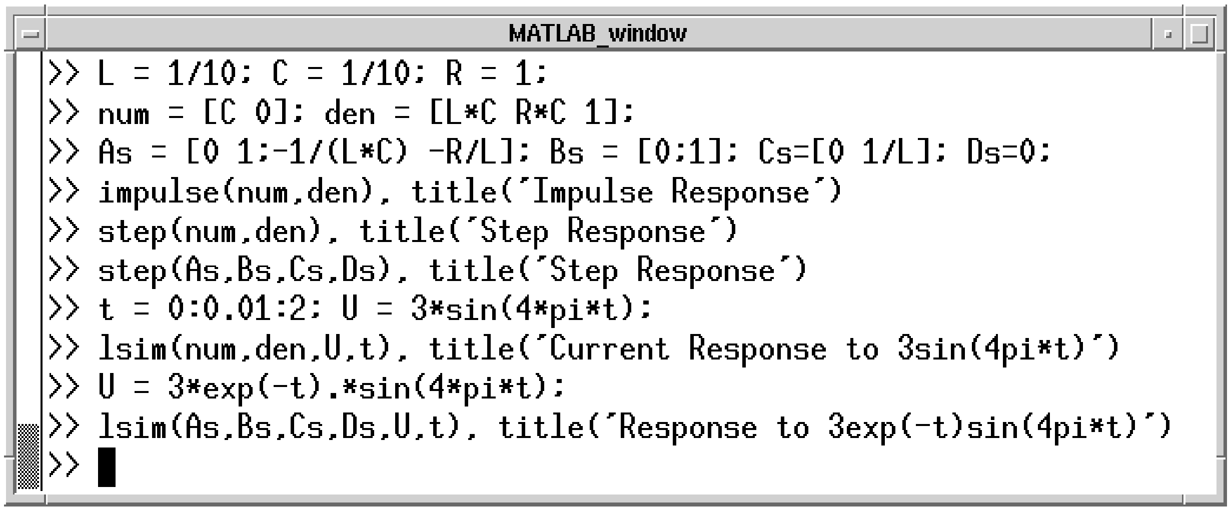


$$THEN \qquad LC\frac{d^2 i}{dt} + RC\frac{di}{dt} + i = C\frac{dV}{dt}$$

$$(RCs^2 + LCs + 1)I(s) = CsV(s) \qquad \frac{I(s)}{V(s)} = \frac{Cs}{(LCs^2 + RCs + 1)}$$

$$OR \qquad x_1 = i \qquad x_2 = \dot{x}_1 \qquad \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{1}{LC} & -\frac{R}{L} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}u \qquad y = \begin{bmatrix} 0 & \frac{1}{L} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

With Matlab:

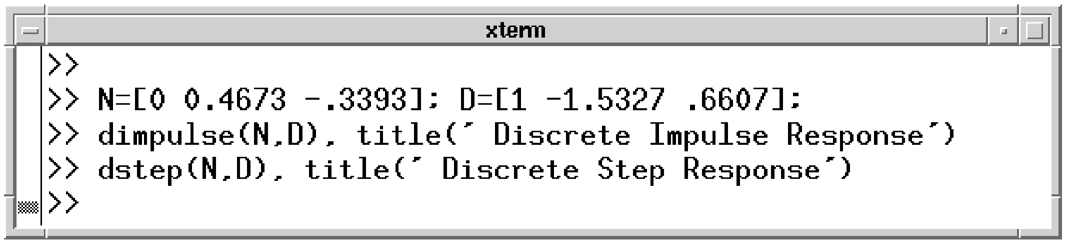


```
>> L = 1/10; C = 1/10; R = 1;
>> num = [C 0]; den = [L*C R*C 1];
>> As = [0 1;-1/(L*C) -R/L]; Bs = [0;1]; Cs=[0 1/L]; Ds=0;
>> impulse(num,den), title('Impulse Response')
>> step(num,den), title('Step Response')
>> step(As,Bs,Cs,Ds), title('Step Response')
>> t = 0:0.01:2; U = 3*sin(4*pi*t);
>> lsim(num,den,U,t), title('Current Response to 3sin(4pi*t)')
>> U = 3*exp(-t).*sin(4*pi*t);
>> lsim(As,Bs,Cs,Ds,U,t), title('Response to 3exp(-t)sin(4pi*t)')
>> 
```

There is a corresponding function for discrete time dynamics for the functions mentioned above . The function would start with a ‘d’ like `dstep` for example.

Suppose we find the z-transform to be $G(z) = \frac{0.4673z - 0.3393}{z^2 - 1.5327z + 0.6607}$

Then



```
>>
>> N=[0 0.4673 -.3393]; D=[1 -1.5327 .6607];
>> dimpulse(N,D), title(' Discrete Impulse Response')
>> dstep(N,D), title(' Discrete Step Response')
>>
```

Nonlinear Systems:
Nonlinear dynamics are solved by the integration method. The system equations are expressed in state form described in a M-file function and the `ode` integrators are used to solve for the state trajectories.

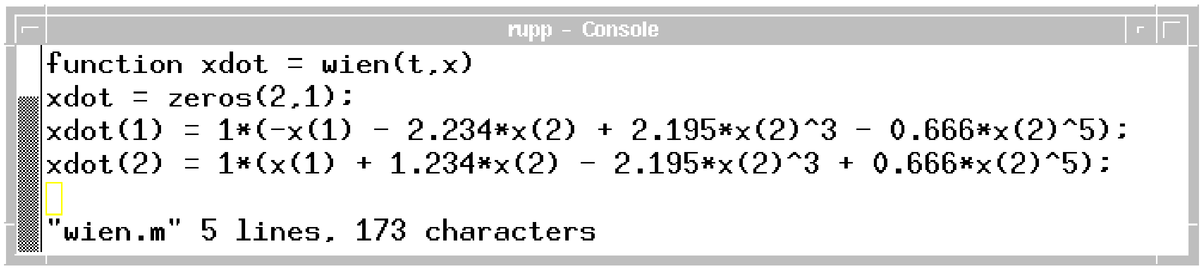
Consider the Wien-Bridge Oscillator:

$$\dot{x}_1 = \frac{1}{C_1R_1}[-x_1 + x_2 - g(x_2)]$$

$$\dot{x}_2 = \frac{1}{C_2R_1}[-x_1 + x_2 - g(x_2)] - \frac{1}{C_2R_2}x_2$$

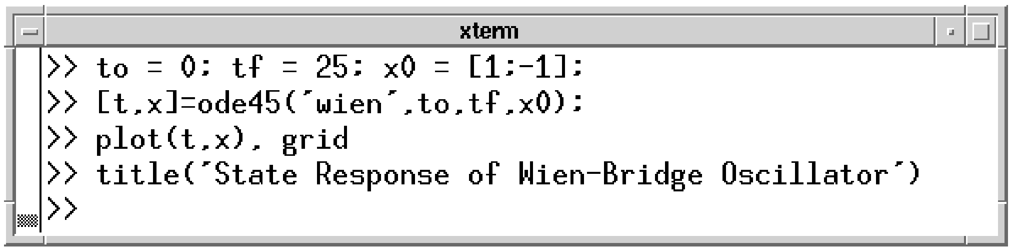
$$g(v) = 3.234v - 2.195v^3 + 0.666v^5$$

Create the M-file wien.m as shown (Let R1 = R2 = C1 = C2 = 1.0)

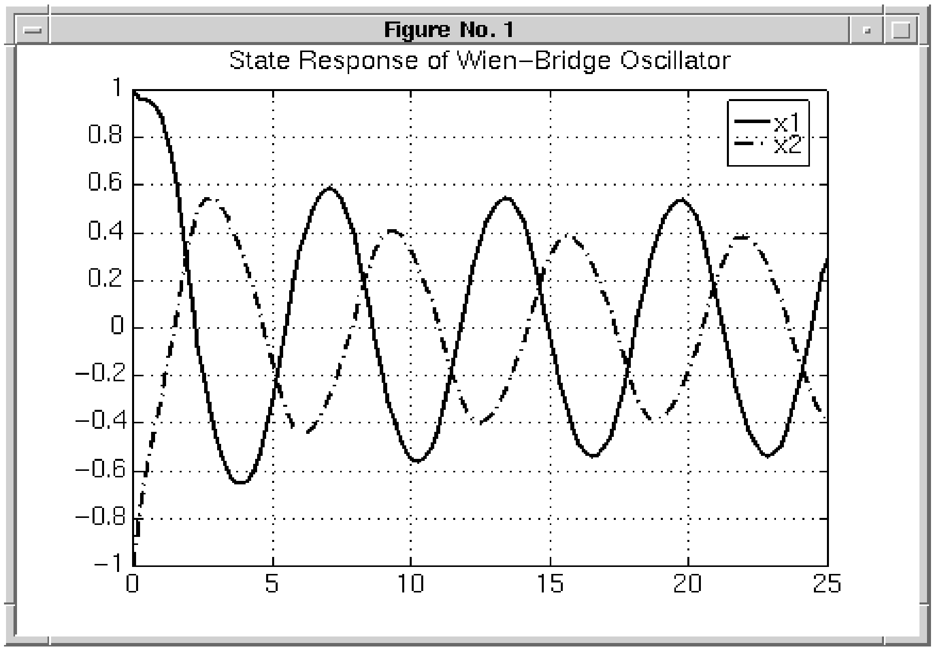


```
function xdot = wien(t,x)
xdot = zeros(2,1);
xdot(1) = 1*(-x(1) - 2.234*x(2) + 2.195*x(2)^3 - 0.666*x(2)^5);
xdot(2) = 1*(x(1) + 1.234*x(2) - 2.195*x(2)^3 + 0.666*x(2)^5);

"wien.m" 5 lines, 173 characters
```

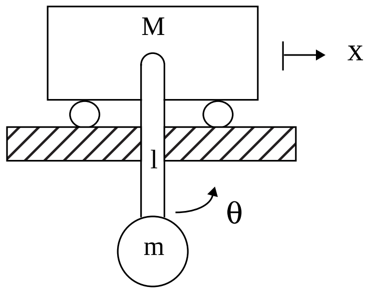
Then with MATLAB



```
>> to = 0; tf = 25; x0 = [1;-1];
>> [t,x]=ode45('wien',to,tf,x0);
>> plot(t,x), grid
>> title('State Response of Wien-Bridge Oscillator')
>>
```



Consider the Cart and Pendulum Problem



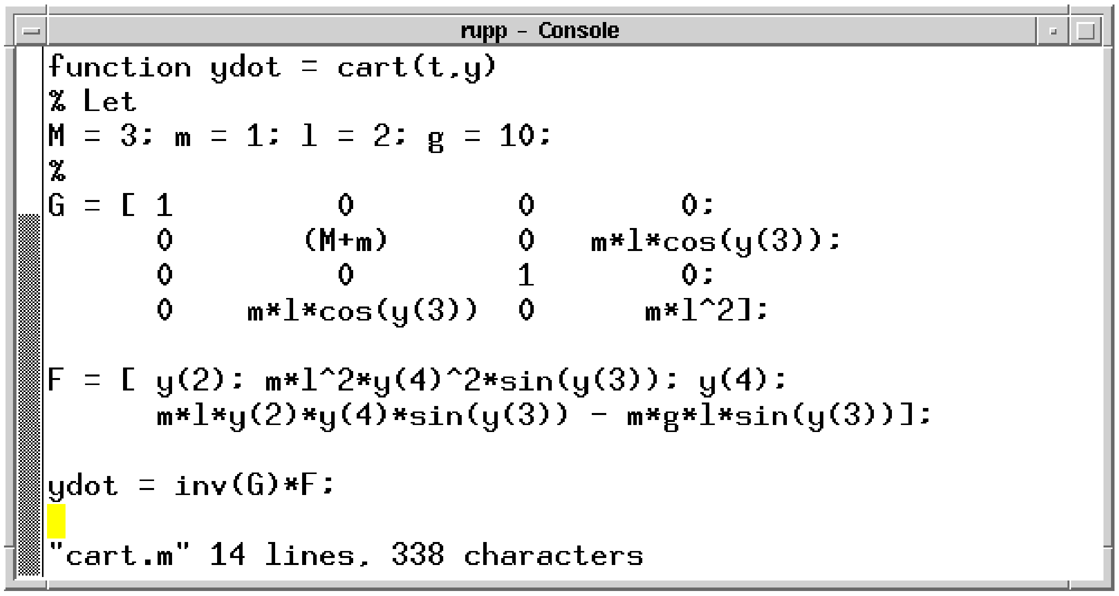
$$(M+m)\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = 0$$

$$ml^2\ddot{\theta} + ml\ddot{x}\cos\theta - ml\dot{x}\dot{\theta}\sin\theta + mgl\sin\theta = 0$$

Use a change of variables to y then we can write

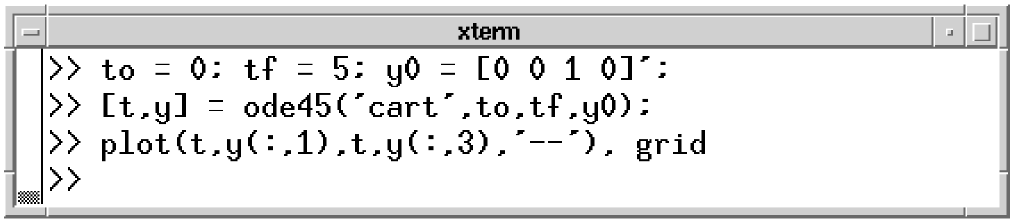
$$\begin{bmatrix} x(t) \\ \dot{x}(t) \\ \theta(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & (M+m) & 0 & ml\cos y_3 \\ 0 & 0 & 1 & 0 \\ 0 & ml\cos y_3 & 0 & ml^2 \end{bmatrix} \cdot \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \\ \dot{y}_4 \end{bmatrix} = \begin{bmatrix} y_2 \\ mly_4^2\sin y_3 \\ y_4 \\ ml(y_2y_4 - g)\sin y_3 \end{bmatrix}$$

Write the M-file cart.m



```
function ydot = cart(t,y)
% Let
M = 3; m = 1; l = 2; g = 10;
%
G = [ 1         0         0        0;
      0       (M+m)       0   m*l*cos(y(3));
      0         0         1        0;
      0    m*l*cos(y(3))  0      m*l^2];

F = [ y(2); m*l^2*y(4)^2*sin(y(3)); y(4);
      m*l*y(2)*y(4)*sin(y(3)) - m*g*l*sin(y(3))];

ydot = inv(G)*F;

"cart.m" 14 lines, 338 characters
```

Then with MATLAB



```
>> to = 0; tf = 5; y0 = [0 0 1 0]';
>> [t,y] = ode45('cart',to,tf,y0);
>> plot(t,y(:,1),t,y(:,3),'--'), grid
>>
```

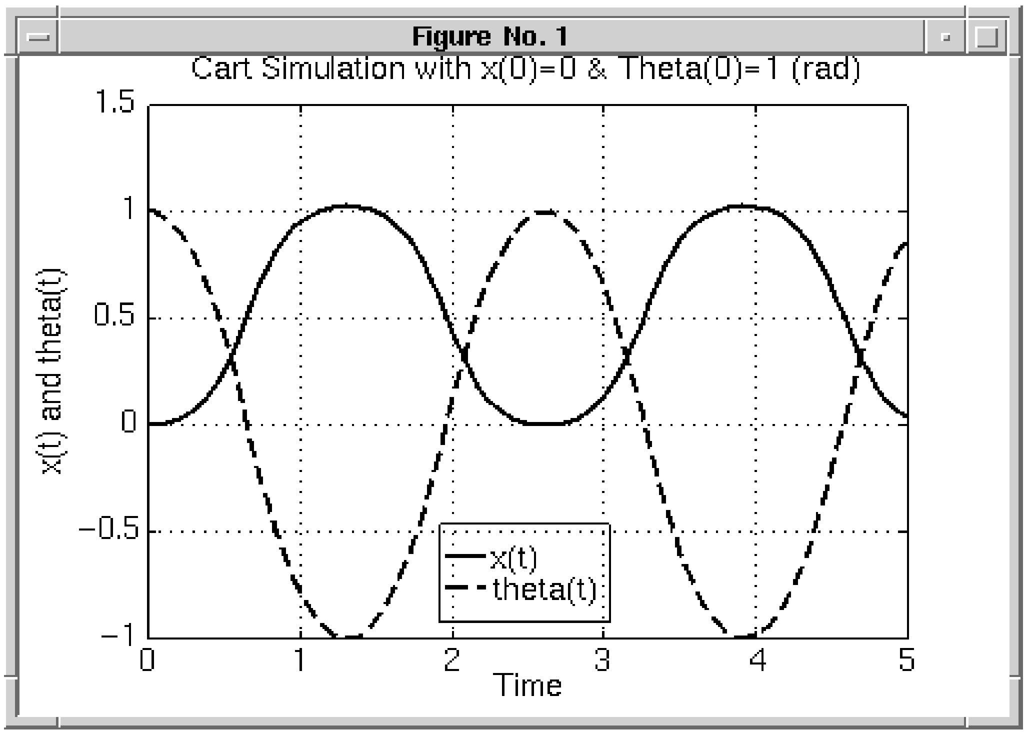


Figure No. 1
Cart Simulation with x(0)=0 & Theta(0)=1 (rad)
1.5
1
0.5
0
−0.5
−1
0
1
2
3
4
5
Time
x(t) and theta(t)
x(t)
theta(t)