University of Cyprus
Department of Electrical and
Computer Engineering

ECE 327: Lecture 2

Second Order System's Features

Second-order systems are commonly encountered in practice, and are the simplest type of dynamic systems to exhibit oscillations and overshoot. In fact, many real higher order systems are modelled as second-order systems to facilitate analysis. Typical examples are the mass-spring-damper systems and RLC circuits.

The **canonical (or standard) form** of a second-order system is given by:

$$G(s) = \frac{w_n^2}{s^2 + 2\zeta w_n s + w_n^2}.$$

where ζ denotes the damping ratio, and w_n denotes the natural frequency. Depending on the value of the damping ratio the system exhibits different behavior.

Underdamped system:

If $\zeta < 1$, then the system is **underdamped**. In this case, both poles are complex-valued with negative real parts; therefore, the system is stable but oscillates while approaching the steady-state value.

Overdamped system:

If $\zeta > 1$, then the system is **overdamped**. Both poles are real and negative; therefore, the system is stable and does not oscillate.

Critically-damped system:

If $\zeta = 1$, then the system is **critically damped**. Both poles are real and have the same magnitude. For a canonical second-order system, the quickest settling time is achieved when the system is critically damped.

Undamped System:

If $\zeta = 0$, then the system is **undamped**. In this case, the poles are purely imaginary; therefore, the system is marginally stable and the step response oscillates indefinitely.

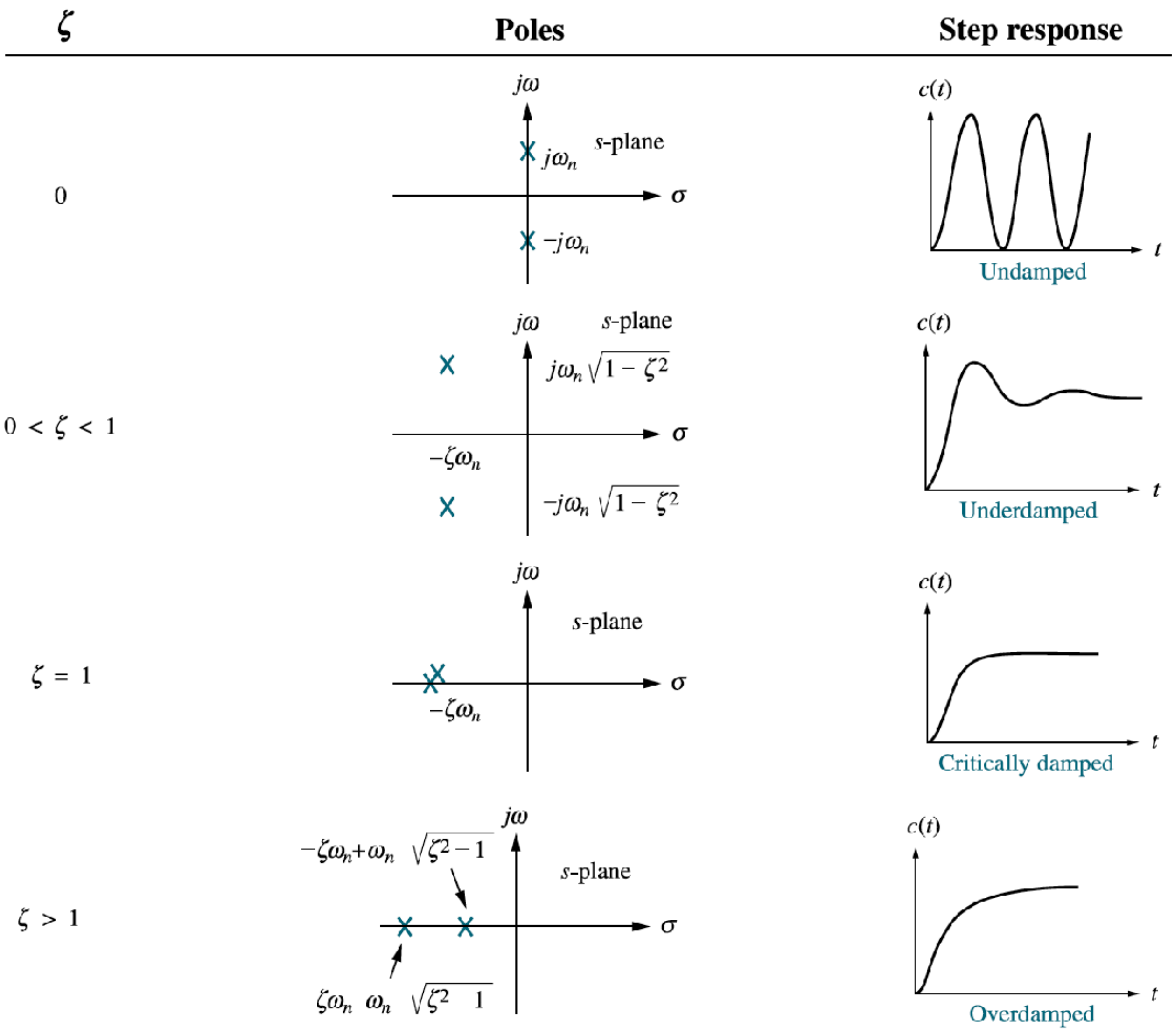


FIGURE. Second-order response as a function of damping ratio

Effects of additional poles and zeros on the second-order response:

Addition of a pole: The poles of a transfer function affect the transient response of the system. The rise time and the peak time are increased with a reduction in the overshoot. As the pole moves away from the dominant pole, it has less effect. Since this additional exponential term decays after five time-constants and if the pole is five times farther to the left than the dominant poles, the system can be represented by a second-order model.

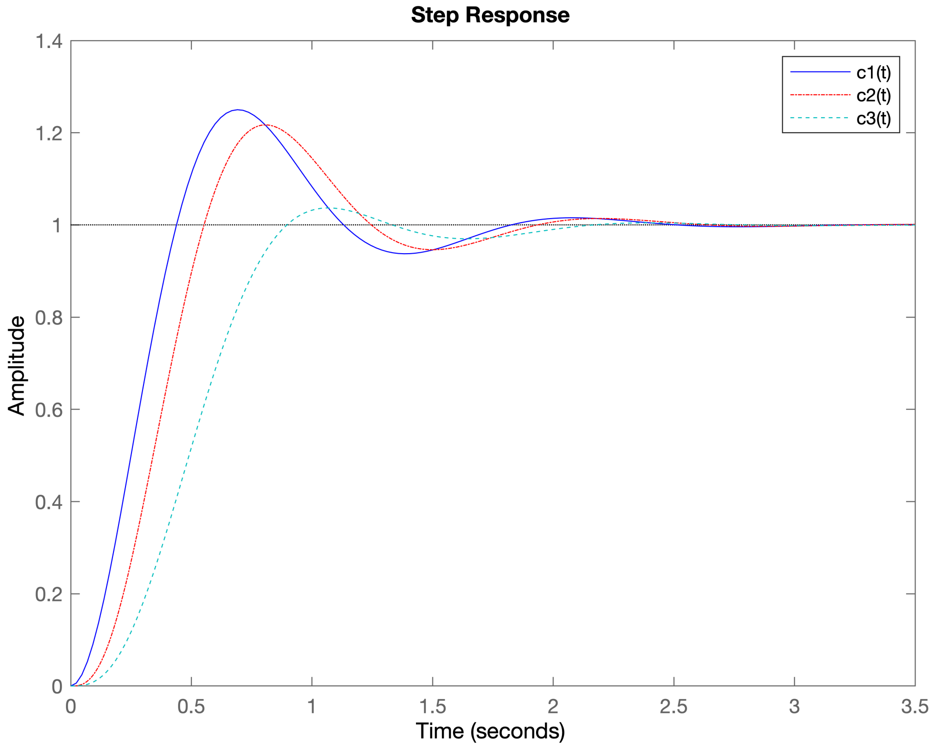
Example. Find the step response of each of the following transfer functions and compare them.

$$T_1(s) = \frac{24.542}{s^2 + 4s + 24.542}$$

$$T_2(s) = \frac{245.42}{(s + 10)(s^2 + 4s + 24.542)}$$

$$T_3(s) = \frac{73.626}{(s+3)(s^2+4s+24.542)}$$

```
sys1 = tf(24.542,[1 4 24.542]);
sys2 = tf(245.542,conv([1 10],[1 4 24.542]));
sys3 = tf(73.626,conv([1 3],[1 4 24.542]));
step(sys1,'b',sys2,'-.r',sys3,'--c')
legend('c1(t)','c2(t)','c3(t)');
```



```
pole(sys1)
```

```
ans = 2×1 complex
  -2.0000 + 4.5323i
  -2.0000 - 4.5323i
```

The three responses are plotted as shown above. Notice that $c_2(t)$ with its third pole at 10 and farthest from the dominant poles, is the better approximation of $c_1(t)$, the pure second-order system response. $c_3(t)$ with a third pole close to the dominant poles, yields the most error.

Addition of a zero: The zeros of a transfer function affect the amplitude of a response component but do not affect the nature of the response - exponential, damped sinusoid, and so on. The closer the zero is to the dominant poles, the more effect it has on the transient response. The rise time and the peak time are decreased, while the oveshoot is increased. As the zero moves away from the dominant poles, the response approaches that of the second-order system.

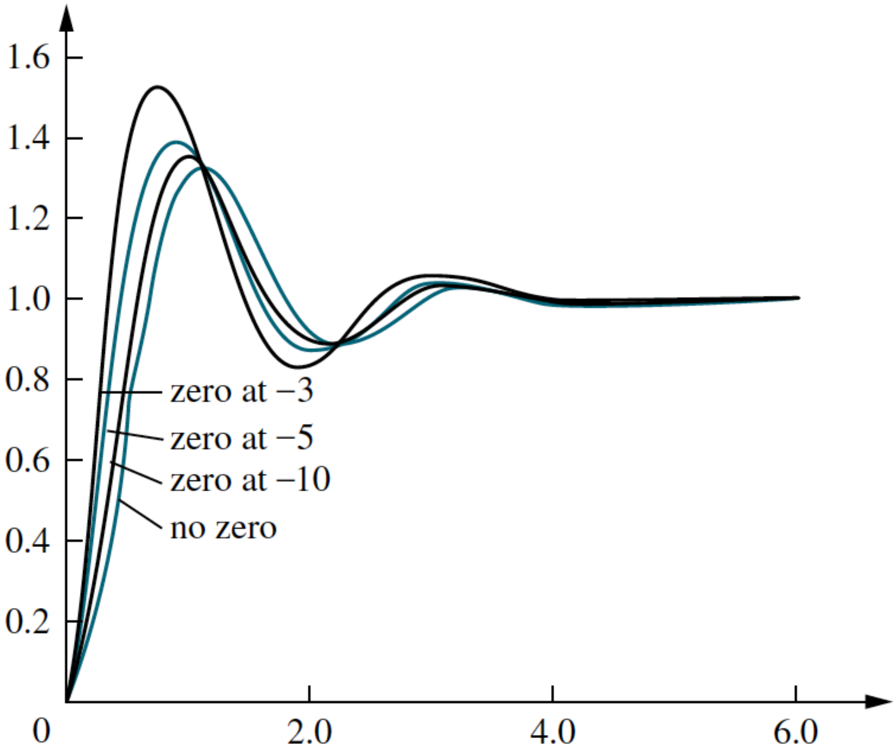
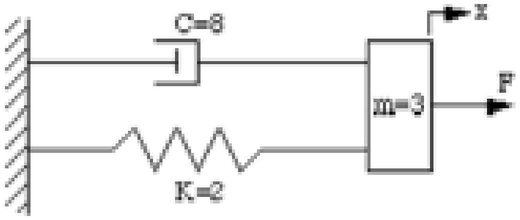


FIGURE. Effect of adding a zero to a two-pole system

Spring-mass-damper system:



The spring force is proportional to the displacement of the mass, x, and the viscous damping force is proportional to the velocity of the mass, $\dot{x}$. Both forces oppose the motion of the mass and are, therefore, shown in the negative direction. Applying Newton's law of motion, the force equation of the above system is

$$m\frac{d^2x(t)}{dt^2} + C\frac{dx(t)}{dt} + Kx(t) = F(t)$$

where m is the mass, K is the spring constant, C is the damping constant, and F is the input force.

Laplace transforms:

$$\mathscr{L}\{x(t)\} = X(s)$$
$$\mathscr{L}\{\dot{x}(t)\} = sX(s) - x(0)$$
$$\mathscr{L}\{\ddot{x}(t)\} = s^2X(s) - sx(0) - \dot{x}(0)$$

Numerical Solution of Differential Equations of Physical Systems:

Analytical solutions of linear time-invariant equations are obtained through the Laplace transform and its inversion. There are other techniques which use the state transition matrix $\Phi(t)$ to provide a solution. These analytical methods are normally restricted to linear differential equations with constant coefficients. On the other hand, **numerical techniques** solve differential equations directly in the time domain; they apply not only to linear time invariant but also to nonlinear and time varying differential equations. The value of the function obtained at any step is an approximation of the value which would have been obtained analytically. However, an analytical solution may be difficult, time consuming or even impossible to find. MATLAB provides the **ODE function** for numerical solution of differential equations employing the Runge-Kutta method.

Ordinary Differential Equations.

An *ordinary differential equation* (ODE) contains one or more derivatives of a dependent variable, y, with respect to a single independent variable, t, usually referred to as time. In an *initial value problem*, the ODE is solved by starting from an initial state. Using the initial condition, y_0, as well as a period of time over which the answer is to be obtained, (t_0, t_f), the solution is obtained iteratively. At each step the solver applies a particular algorithm to the results of previous steps. The final result is that the ODE solver returns a vector of time steps $t = [t_0, t_1, \ldots, t_f]$ as well as the corresponding solution at each step $y = [y_0, y_1, \ldots, y_f]$.

Note: **The MATLAB ODE solvers only solve first-order equations.** You must rewrite higher-order ODEs as an equivalent system of first-order equations. For example, for an nth order linear differential equation

$$\frac{d^n y(t)}{dt^n} + \alpha_{n-1}\frac{d^{n-1}y(t)}{dt^{n-1}} + \ldots + \alpha_0 y(t) = b_0 u(t)$$

choose the state variables

$$\begin{aligned} x_1(t) &= y(t) \\ x_2(t) &= \frac{dy(t)}{dt} \\ x_3(t) &= \frac{d^2y(t)}{dt^2} \\ &\vdots \\ x_n(t) &= \frac{d^{n-1}y(t)}{dt^{n-1}} \end{aligned}$$

The result of these substitutions is a system of n first-order equations

$$\begin{aligned} &\dot{x}_1(t) = \dot{y}(t) = x_2(t) \\ &\dot{x}_2(t) = x_3(t) \\ &\vdots \\ &\dot{x}_{n-1}(t) = x_n(t) \\ &\dot{x}_n(t) = -\alpha_0 x_1(t) - \alpha_1 x_2(t) - \ldots - \alpha_{n-1}(t)x_n(t) + b_0 u(t) \end{aligned}$$

Example. Consider the second order differential equation

$$\ddot{y} + 2\dot{y} + y = u$$

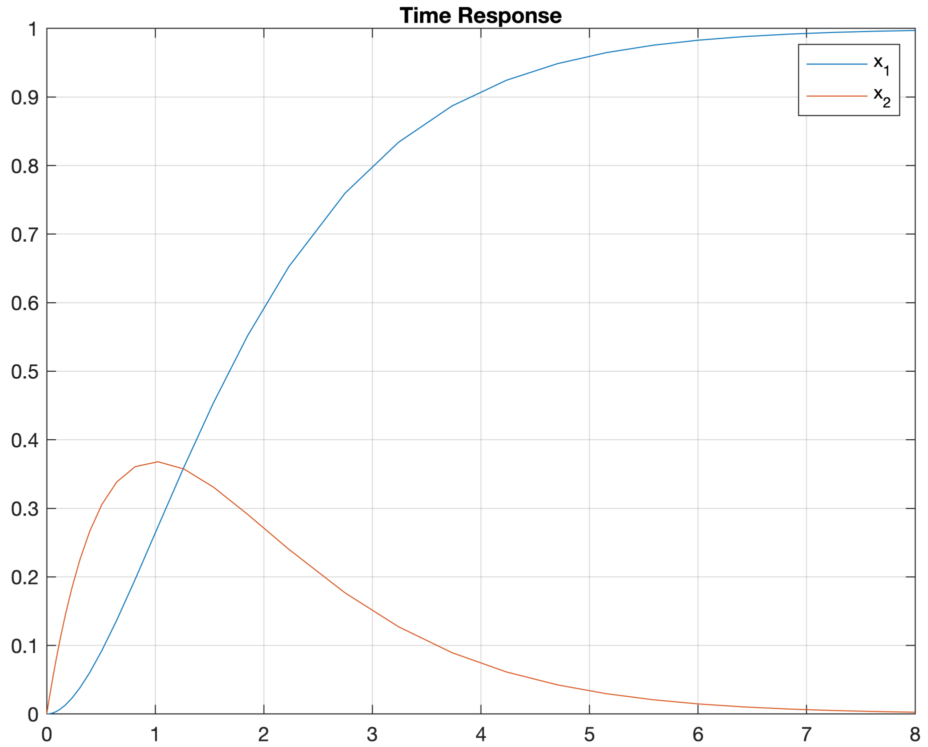
where $y(0) = \dot{y}(0) = 0$ and $u(t)$ is a unit step. Determine the solution $y(t)$ analytically.

Solution. Let $x_1 = y$ and $x_2 = \dot{y}$, then

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = u - 2x_2 - x_1$$

```
tspan=[0 8];
x0=[0;0];
[t,x] = ode23(@just_an_example,tspan,x0);
plot(t,x(:,1),t,x(:,2));grid;
title('Time Response');
legend('x_1','x_2')
```



```
function xdot = just_an_example(t,x)
u=1;
xdot = [x(2); u-2*x(2)-x(1)];
end
```