System Modeling and State-Space

There are several ways to express a dynamic system. In ECE 343, you used transfer functions, such as

$$Y = \left(\frac{10}{s^2+2s+10}\right) X$$

In this class, we'll be using a formulation called State Space. State-space is an energy-based system to describe the dynamics of a system. In essence, the states, X, define the energy in the system. The change in the energy describes how the system behaves as

$$sX = AX + BU$$

What you measure is also a funciton of the energy in the system.

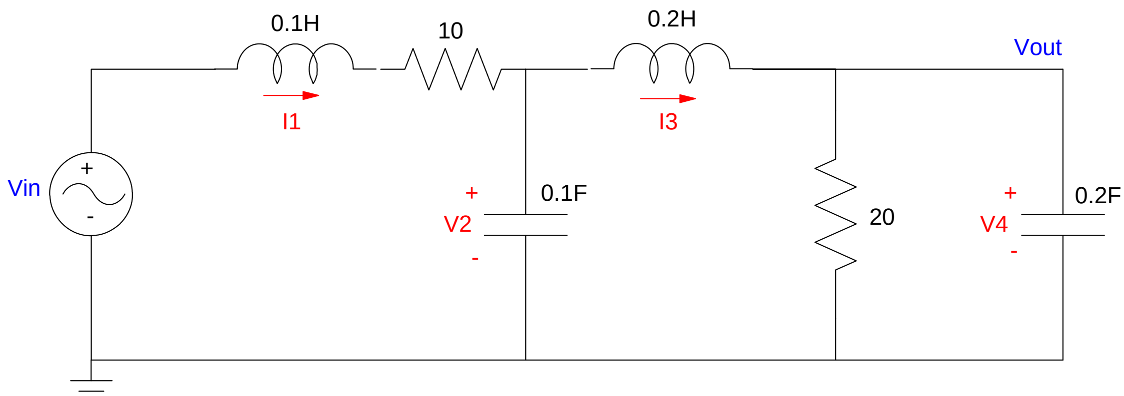
$$Y = CX + DU$$

Together, this gives you a state-space representation for a system:

Example 1: RLC Circuit

For the following RLC circuit,

- Find the transfer function for the following circuit from Vin to Vout
- Find the dominant pole of this system
- Determine a 1st or 2nd-order approximation for this system:



Example 1: RLC Circuit

Let the states, X, define the energy in the system (the current through the inductors and the voltage across the capacitors):

$$X = \begin{bmatrix} I_1 \\ V_2 \\ I_3 \\ V_4 \end{bmatrix}$$

The change in the states can then be found using the basic equations for an inductor and capacitor:

$$V = L\frac{dI}{dt}$$

$$I = C\frac{dV}{dt}$$

The voltage across the first inductor is then

$$V_{L1} = 0.1sI_1 = V_{in} - (V_2 + 10I_1)$$

The current to capacitor 2 is

$$I_{C2} = 0.1sV_2 = I_1 - I_3$$

The voltage across inductor 3 is

$$V_{L3} = 0.2sI_3 = V_2 - V_4$$

The current to capacitor 4 is

$$I_{C4} = 0.2sV_4 = I_3 - \frac{V_4}{20}$$

Put these together and you get

$$sI_1 = 10V_{in} - 10V_2 - 100I_1$$

$$sV_2 = 10I_1 - 10I_3$$

$$sI_3 = 5V_2 - 5V_4$$

$$sV_4 = 5I_3 - 0.25V_4$$

In matrix form (a.k.a. state-space form):

$$\begin{bmatrix} sI_1 \\ sV_2 \\ sI_3 \\ sV_4 \end{bmatrix} = \begin{bmatrix} -100 & -10 & 0 & 0 \\ 10 & 0 & -10 & 0 \\ 0 & 5 & 0 & -5 \\ 0 & 0 & 5 & -0.25 \end{bmatrix} \begin{bmatrix} I_1 \\ V_2 \\ I_3 \\ V_4 \end{bmatrix} + \begin{bmatrix} 10 \\ 0 \\ 0 \\ 0 \end{bmatrix} V_{in}$$

The output is equal to V4, so

$$V_{out} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_1 \\ V_2 \\ I_3 \\ V_4 \end{bmatrix} + [0]V_{in}$$

In Matlab, you can find the transfer function:

```
>> A = [-100,-10,0,0;10,0,-10,0;0,5,0,-5;0,0,5,-0.25]

 -100.0000  -10.0000         0         0
   10.0000         0  -10.0000         0
         0    5.0000         0   -5.0000
         0         0    5.0000   -0.2500

>> B = [10;0;0;0]

    10
     0
     0
     0

>> C = [0,0,0,1]

     0     0     0     1

>> D = 0

     0

>> G4 = ss(A,B,C,D);
```

The transfer function from Vin to Vout is:

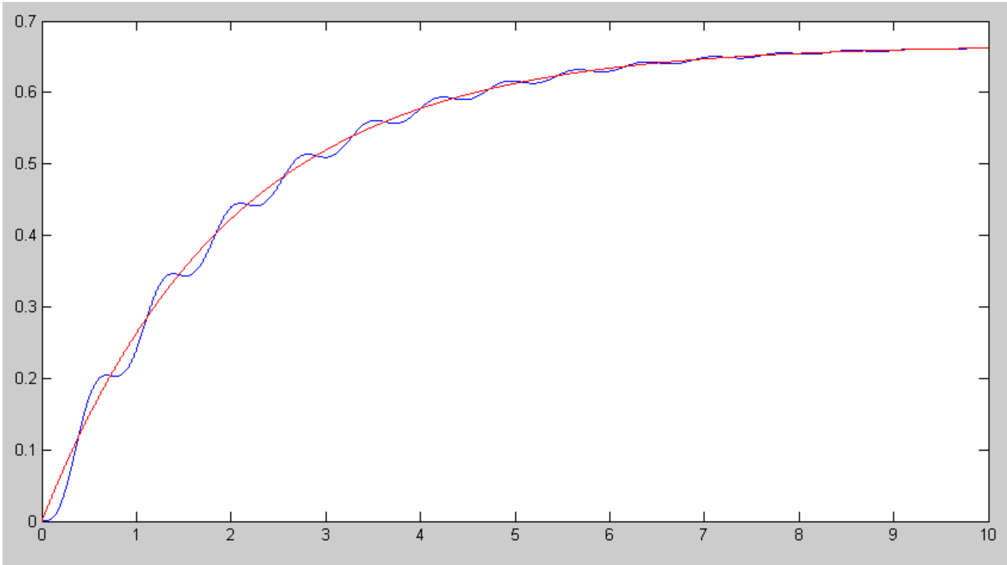
```
>> zpk(G4)

                    2500
--------------------------------------------
(s+98.99) (s+0.5025) (s^2 + 0.7525s + 75.38)
```

Its first-order approximation is:

- A 1st-order system
- With a DC gain of 0.6667, and
- A 2% settling time of 8 seconds (4 / 0.5025).

```
>> DC = evalfr(G4,0)

    0.6667

>> G1 = tf(0.6667*0.5025,[1,0.5025])


  0.335
----------
s + 0.5025

>> y4 = step(G4,t);
>> y1 = step(G1,t);
>> plot(t,y4,'b',t,y1,'r');
```

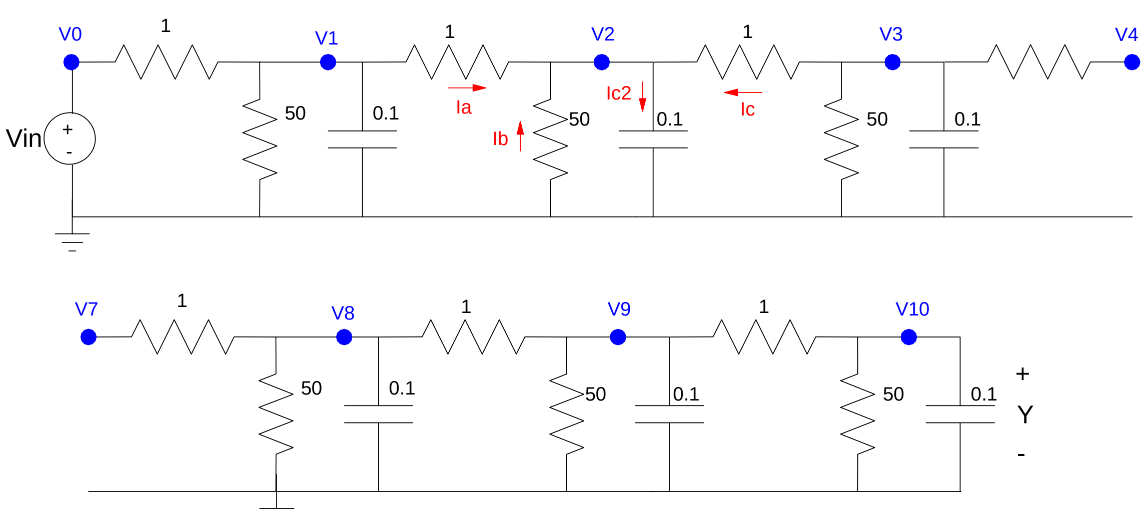
The actual step response from Matlab is as follows:



Step Response of the 4th-order RLC circuit (blue) and its 1st-order approximation (red)

Example 2: RC Filter (Heat Equation)

Consider next the following 10-stage RC filter



Example 2: 10-stage RC filter

While this looks daunting, in state-space it isn't that bad. Since each stage is identical, the equations will be the same. Take node 2 for example. The current to the capacitor (IC2) is the sum of the currents coming in:

$$Ic2 = Ia + Ib + Ic$$

$$I_{c2} = CsV_2 = \left(\frac{V_1 - V_2}{1}\right) + \left(\frac{0 - V_2}{50}\right) + \left(\frac{V_3 - V_2}{1}\right)$$

or

$$sV_2 = 10V_1 - 20.2V_2 + 10V_3$$

This repeats for nodes 1..9. Node #10 is slightly different since it only connects to one other node::

$$I_{c10} = CsV_{10} = \left(\frac{V_9 - V_{10}}{1}\right) + \left(\frac{0 - V_{20}}{50}\right)$$

$$sV_{10} = 10V_9 - 10.2V_{10}$$

In Matlab:

```
>> A = zeros(10,10);
>> for i=1:9
   A(i,i) = -20.2;
   A(i,i+1) = 10;
   A(i+1,i) = 10;
   end
>> A(10,10) = -10.2;

  -20.2000   10.0000         0         0         0         0         0         0         0         0
   10.0000  -20.2000   10.0000         0         0         0         0         0         0         0
         0   10.0000  -20.2000   10.0000         0         0         0         0         0         0
         0         0   10.0000  -20.2000   10.0000         0         0         0         0         0
         0         0         0   10.0000  -20.2000   10.0000         0         0         0         0
         0         0         0         0   10.0000  -20.2000   10.0000         0         0         0
         0         0         0         0         0   10.0000  -20.2000   10.0000         0         0
         0         0         0         0         0         0   10.0000  -20.2000   10.0000         0
         0         0         0         0         0         0         0   10.0000  -20.2000   10.0000
         0         0         0         0         0         0         0         0   10.0000  -10.2000

>> B = [10;0;0;0;0;0;0;0;0;0]

    10
     0
     0
     0
     0
     0
     0
     0
     0
     0

>> C = [0,0,0,0,0,0,0,0,0,1];
>> D = [0];
>> G = ss(A,B,C,D);
>> evalfr(G,0)

    0.4325

>> zpk(G)

                                               10000000000
----------------------------------------------------------------------------------------------------
(s+39.31) (s+36.72) (s+32.67) (s+27.51) (s+21.69) (s+15.75) (s+10.2) (s+5.539) (s+2.181) (s+0.4234)
```

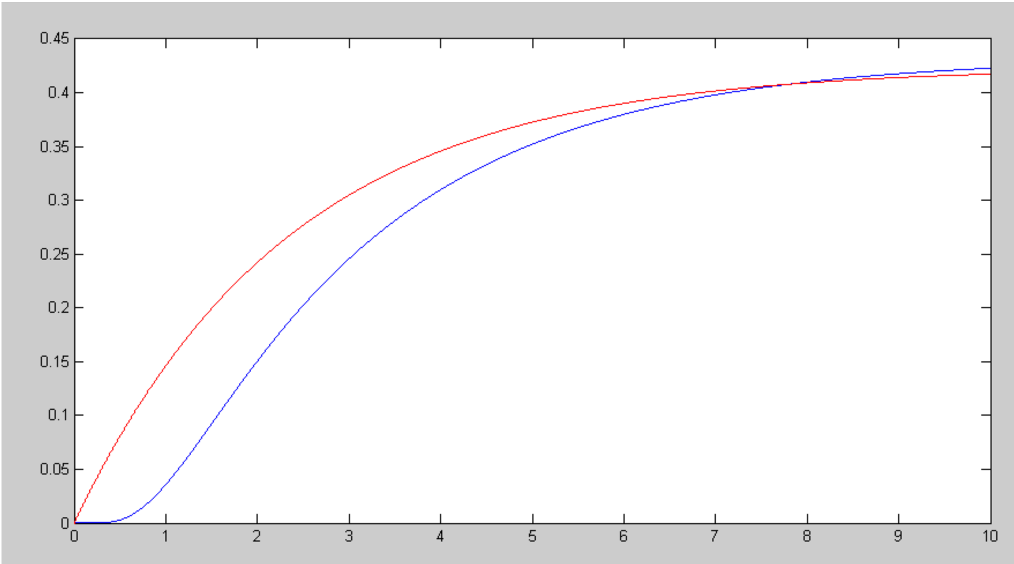
This is the transfer function for this RC filter. Its response should

- Have a DC gain of 0.4325, and
- A 2% settling time of 9.44 seconds (4 / 0.4234)

$$G(s) \approx \left(\frac{0.1793}{s + 0.4234}\right)$$

The actual step-response from Matlab is

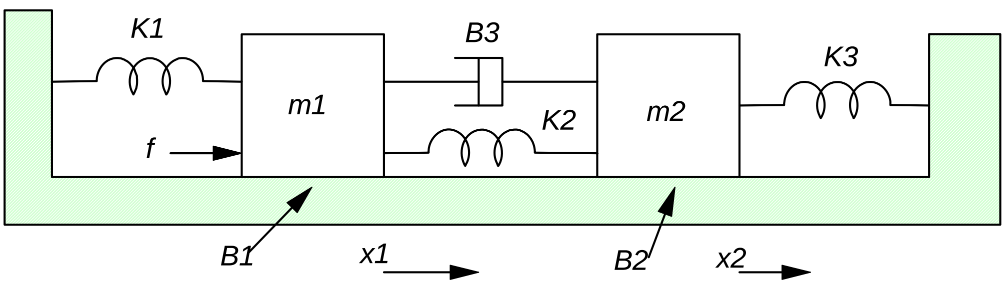
```
>> G1 = zpk([],[-0.4234],0.1793)

Zero/pole/gain:
  0.1793
----------
(s+0.4234)

>> t = [0:0.001:10]';
>> y1 = step(G1,t);
>> y10 = step(G10,t);
??? Undefined function or variable 'G10'.

>>
>> y10 = step(G,t);
>> plot(t,y10,'b',t,y1,'r');
```



Step Response of the 10th Order RC Filter (blue) and its 1st-Order Approximation (red)

Example 3: Mass-Spring System

Finally, consider a mass-spring system:



Example 3: Mass Spring System

To put this in state-space form, redraw this as a circuit equivalent using the following dual:

$$I = \frac{1}{R} \cdot V$$

$$F = Ms^2 \cdot X$$

If

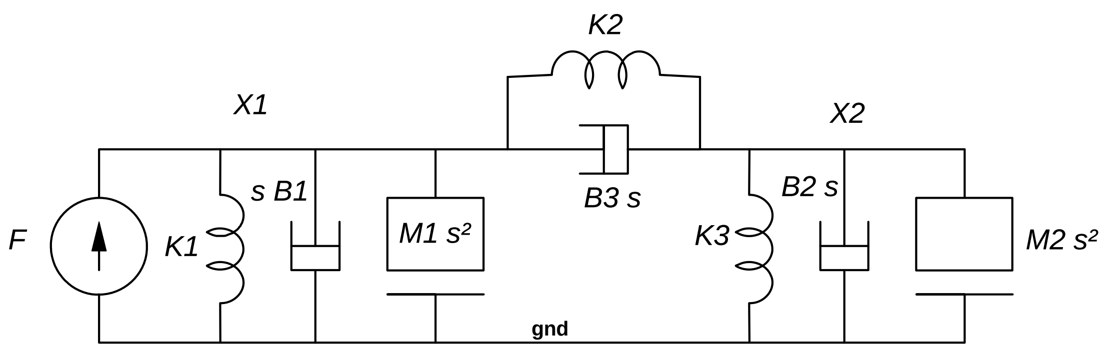
- Current is the analog of force, and
- Voltage is the analog of position, then
- The admittance is the analog of
  - Ms2 (for a mass)
  - Bs (for friction)
  - K (for a spring)

This mass-spring system has three displacements. The circuit equivalent has three node voltages (X1, X2, X3).

Each node has mass relative to ground (i.e. Einstein's theory of relativity)

The other nodes are the springs and friction

Redrawing this mass-spring circuit:



Writing the node equations then results in

$$(K_1 + B_1 s + M_1 s^2 + K_2 + B_3 s)X_1 - (K_2 + B_3 s)X_2 = F$$

$$(M_2 s^2 + B_2 s + K_3 + K_2 + B_3 s)X_2 - (K_2 + B_3 s)X_1 = 0$$

Solving for the highest derivative:

$$M_1 s^2 X_1 = -(K_1 + K_2 + B_1 s + B_3 s)X_1 + (K_2 + B_3 s)X_2 + F$$

$$M_2 s^2 X_2 = -(B_2 s + K_3 + K_2 + B_3 s)X_2 + (K_2 + B_3 s)X_1$$

The states (that which determines the energy in the system) are

- position, and
- velocity.

Defining the states this way results in the matrix formulation of dynamics (i.e. the state-space model) being:

$$s\begin{bmatrix} X_1 \\ X_2 \\ \cdots \\ sX_1 \\ sX_2 \end{bmatrix} = \left[\begin{array}{cc:cc} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \cdots & \cdots & \cdots & \cdots \\ \left(\frac{-(K_1+K_2)}{M_1}\right) & \left(\frac{K_2}{M_1}\right) & \left(\frac{-(B_1+B_3)}{M_1}\right) & \left(\frac{B_3}{M_1}\right) \\ \left(\frac{K_2}{M_2}\right) & \left(\frac{-(K_2+K_3)}{M_2}\right) & \left(\frac{B_3}{M_2}\right) & \left(\frac{-(B_2+B_3)}{M_2}\right) \end{array}\right] \begin{bmatrix} X_1 \\ X_2 \\ \cdots \\ sX_1 \\ sX_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \cdots \\ \left(\frac{1}{M_1}\right) \\ 0 \end{bmatrix} F$$

$$Y = X_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ sX_1 \\ sX_2 \end{bmatrix} + [0]F$$

Note that

- You have 2N states, where N is the number of masses. Each mass has two energy states (kinetic and potential energy) giving your 2N state variables.
- The first N rows are [0 : I] where I is the identity matrix. This tells MATLAB that the states are position and velocity.
- The last N rows are where the dynamics come into play.

Also also, you can have real or complex poles for mass-spring systems - unlike the heat equation which always has real poles.

Finding the Transfer Function:

To find the transfer function, use MATLAB or SciLab. Assume for example that

- M = 1kg
- B = 2 Ns/m
- K =10 N/m

Then the state-space model is:

$$s\begin{bmatrix} X_1 \\ X_2 \\ sX_1 \\ sX_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -20 & 10 & -4 & 2 \\ 10 & -20 & 2 & -4 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ sX_1 \\ sX_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} F$$

MATLAB Code: Input the A B C D matrices:

```
-->a11 = zeros(2,2);
-->a12 = eye(2,2);
-->a21 = [-20,10;10,-20];
-->a22 = [-4,2;2,-4];
-->A = [a11,a12;a21,a22]
```

```
       0.      0.      1.      0.
       0.      0.      0.      1.
     - 20.     10.   - 4.      2.
       10.   - 20.     2.    - 4.

-->B = [0;0;1;0];
-->C = [0,1,0,0];
-->D = 0;
```

Use these to define G(s):

```
-->G = ss(A,B,C,D)
```

Once G(s) is in MATLAB, find the transfer function

```
-->[z,p,k] = zpk(G)
 k  =
    2.

 p  =
  - 3. + 4.5825757i
  - 3. - 4.5825757i
  - 1. + 3.i
  - 1. - 3.i

 z  =
  - 5.
```

meaning:

$$X_2 = \left(\frac{2(s+5)}{(s+3\pm j4.58)(s+1\pm j3)}\right)F$$

If you want to approximate this with a 2nd-order model, keep the slowest pole and match the DC gain

```
-->DC = evalfr(G,0)
    0.0333333
```

So

$$X_2 \approx \left(\frac{0.3333}{(s+1\pm j3)}\right)F$$

To check the response in MATLAB, take the two step responses.

Input the 2nd-order approximation:

```
-->G2 = zpk([],[-1+j*3,-1-j*3],0.3333)
```

Take the step response of the two systems:

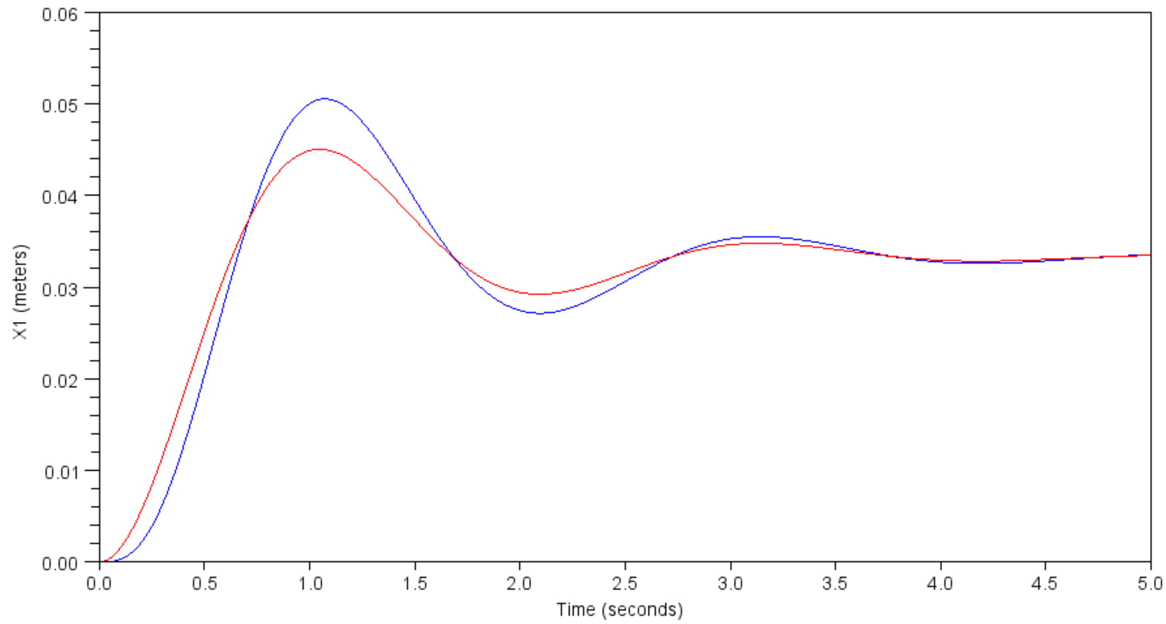
```
-->t = [0:0.1:100]';
-->x2 = step(G,t);
-->x2a = step(G2,t);
```

and plot

```
-->plot(t,x2,t,x2a)
```

```
-->xlabel('Time (seconds)');
-->ylabel('X2 (meters)');
```

Note that the 2nd-order model isn't that good of an approximation: the 'fast' pole is only 3x faster.



Step Response of the 4th-Order Mass-Spring System (blue) and its 2nd-Order Approximation (red)

Matlab Code:

```
a11 = zeros(2,2);
a12 = eye(2,2);
a21 = [-20,10;10,-20];
a22 = [-4,2;2,-4];
A = [a11,a12;a21,a22]

     0     0     1     0
     0     0     0     1
   -20    10    -4     2
    10   -20     2    -4


B = [0;0;1;0];
C = [0,1,0,0];
D = 0;
G = ss(A,B,C,D);

zpk(G)

            2 (s+5)
-------------------------------
(s^2 + 2s + 10) (s^2 + 6s + 30)


 tf(G)


             2 s + 10
----------------------------------
s^4 + 8 s^3 + 52 s^2 + 120 s + 300
```

$$X_2 = \left(\frac{2(s+5)}{(s+3\pm j4.58)(s+1\pm j3)} \right) F$$