Define the Laplace variable 's'

```
s = tf('s');
```

Define the system components

```
GC1 = 2.5*(1 + 1/(5*s))
```

```
GC1 =

  12.5 s + 2.5
  ------------
      5 s

Continuous-time transfer function.
Model Properties
```

```
GC2 = 0.1
```

```
GC2 = 0.1000
```

```
GP1 =1.0 * (1/(10*s + 1))
```

```
GP1 =

     1
  --------
  10 s + 1

Continuous-time transfer function.
Model Properties
```

```
GP2 = 5/(2*s + 1)
```

```
GP2 =

     5
  -------
  2 s + 1

Continuous-time transfer function.
Model Properties
```

Define the time delays

```
delay1 = 1 - 4*s
```

```
delay1 =

  -4 s + 1

Continuous-time transfer function.
Model Properties
```

```
delay2 = 1 -s
```

```
delay2 =

  -s + 1

Continuous-time transfer function.
Model Properties
```

```
% using Pade approximation
% delay1 = pade(exp(-4*s), 1) % First-order Pade approximation for e^-4s
% delay2 = pade(exp(-s), 1)   % First-order Pade approximation for e^-s
```

Inner loop analysis

Get the characteristic polynomial from the denominator of the closed-loop transfer function

```
char_eq = 1 + GC2 * GP2 * delay2
```

```
char_eq =

  1.5 s + 1.5
  -----------
    2 s + 1

Continuous-time transfer function.
Model Properties
```

We are interested in the numerator of the characteristic equation.

```
[num,~] = tfdata(char_eq, 'v')
```

```
num = 1×2
    1.5000    1.5000
```

Find the roots of the characteristic equation (poles of the closed-loop system). `'v'` specifies that the coefficients should be returned as vectors.

```
roots_char_eq = roots(num)
```

```
roots_char_eq = -1
```

```
real_parts = real(roots_char_eq)
```

```
real_parts = -1
```

```
% Stability check
if all(real(real_parts) < 0)
    disp('The system is stable (All poles are in the left-half of the s-plane).');
else
    disp('The system is unstable (There are poles in the right-half of the s-plane).');
end
```

```
The system is stable (All poles are in the left-half of the s-plane).
```

Outer loop analysis

Inner loop transfer function

```
HR2 = GC2 * GP2 * delay2/ (1 + GC2*GP2*delay2)
```

```
HR2 =

  -s^2 + 0.5 s + 0.5
  -------------------
  3 s^2 + 4.5 s + 1.5

Continuous-time transfer function.
Model Properties
```

Combine the components to get the open-loop transfer function

```
L = GC1 * HR2 * GP1 * delay1
```

```
L =

  50 s^4 - 27.5 s^3 - 26.25 s^2 + 2.5 s + 1.25
  --------------------------------------------
      150 s^4 + 240 s^3 + 97.5 s^2 + 7.5 s

Continuous-time transfer function.
Model Properties
```

Define the characteristic equation 1 + L = 0

Get the characteristic polynomial from the denominator of the closed-loop transfer function

```
char_eq = 1 + L
```

```
char_eq =

  200 s^4 + 212.5 s^3 + 71.25 s^2 + 10 s + 1.25
  ---------------------------------------------
      150 s^4 + 240 s^3 + 97.5 s^2 + 7.5 s

Continuous-time transfer function.
Model Properties
```

```
[num,~] = tfdata(char_eq, 'v')
```

```
num = 1×5
  200.0000  212.5000   71.2500   10.0000    1.2500
```

Find the roots of the characteristic equation (poles of the closed-loop system)

```
roots_char_eq = roots(num)
```

```
roots_char_eq = 4×1 complex
  -0.5000 + 0.0000i
  -0.4584 + 0.0000i
  -0.0521 + 0.1567i
  -0.0521 - 0.1567i
```

```
real_parts = real(roots_char_eq)
```

```
real_parts = 4×1
   -0.5000
   -0.4584
   -0.0521
   -0.0521
```

```
% Stability check
if all(real(real_parts) < 0)
    disp('The system is stable (All poles are in the left-half of the s-
plane).');
else
    disp('The system is unstable (There are poles in the right-half of the s-
plane).');
end
```

The system is stable (All poles are in the left-half of the s-plane).