Exercise 12.1 scallywag

Python Solution

Load Python packages

```
import numpy as np
import matplotlib.pyplot as plt
import sympy as sp
import dysys as ds
import control
```

Define the System

Define the transfer function by using the dysys package to create a symbolic transfer function and converting it to a control package transfer function object (this is easier because $H(s)$ is given in neither expanded form nor zero-pole-gain form) as follows:

```
s = sp.symbols("s", complex=True)
H = ds.tfs(10*(s + 3)/((s + 2)*(s**2 + 8*s + 41)))
H_ = H.to_control()
print(H_)
```

```
       10 s + 30
------------------------
s^3 + 10 s^2 + 57 s + 82
```

Poles and Zeros

The poles and zeros can be computed as follows:

```
poles = H_.poles()
zeros = H_.zeros()
print(f"Poles: {poles}", f"\nZeros: {zeros}")
```

```
Poles: [-4.+5.j -4.-5.j -2.+0.j]
Zeros: [-3.+0.j]
```

The real parts of the poles are all negative; therefore, the system is asymptotically stable. # Pole-Zero Plot {-} A pole-zero plot can be created automatically with `control.pzmap()`, but it can be created manually (and more configurably) as follows:

```
fig, ax = plt.subplots()
ax.scatter(
    np.real(poles), np.imag(poles), s=50, marker="x", facecolors="k"
)
ax.scatter(
    np.real(zeros), np.imag(zeros),
    s=50, marker="o", facecolors="none", edgecolors="k"
)
ax.plot([1],[0])  # Force origin in view
ax.spines[['left', 'bottom']].set_position('zero')
ax.spines[['top', 'right']].set_visible(False)
ax.set_xlabel("$\\Re(s)$", loc="right")
ax.set_ylabel("$\\Im(s)$", loc="top")
plt.show()
```

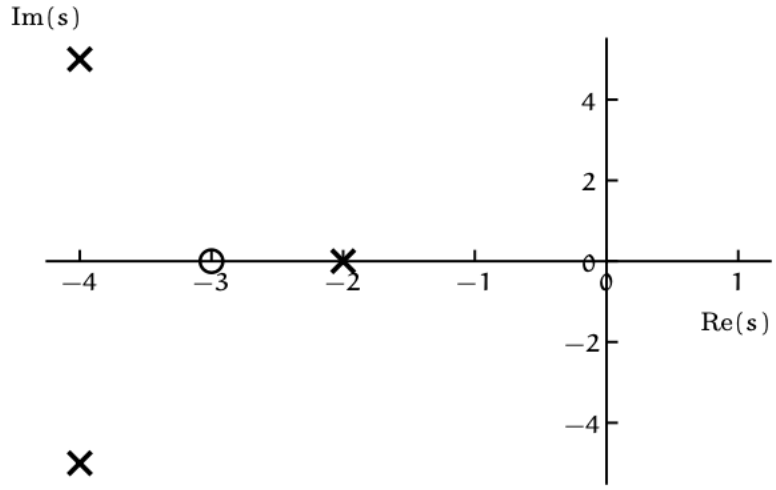


Figure tf.1: The pole-zero plot for $H(s)$.

Step Response

The step response can be simulated with `control.step_response()` as follows:

```
t = np.linspace(0, 3, 101)
y = control.step_response(H_, T=t, squeeze=True).outputs
```

Plot the response as follows:

```
fig, ax = plt.subplots()
ax.plot(t, y)
ax.set_xlabel("Time (s)")
ax.set_ylabel("Unit step response")
plt.show()
```

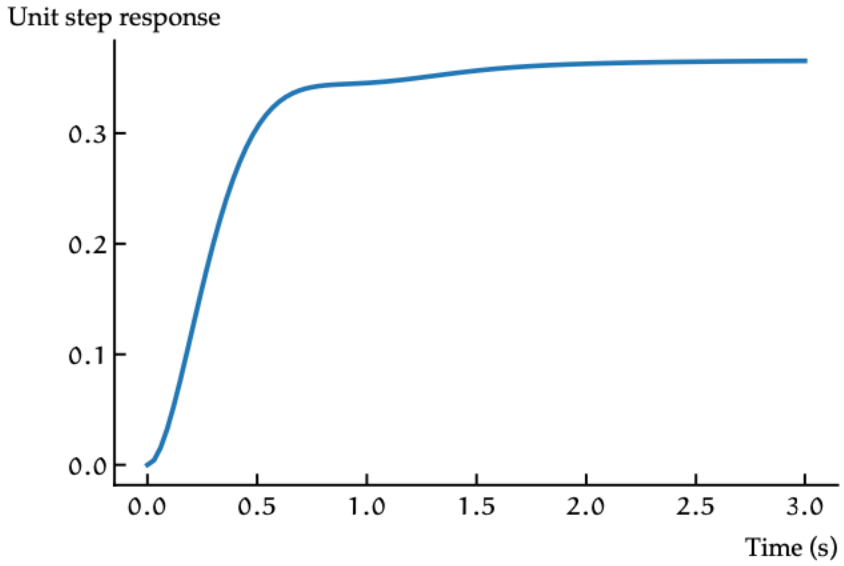


Figure tf.2: The unit step response for $H(s)$.

MATLAB Solution

Solution for exercise tf.scallywag

Define the transfer function model

The numerator and denominators need expanded for entering to `tf`. The numerator is trivial, but let's use Matlab's symbolics for the denominator.

```
syms s
disp(...
  expand(... % expands the expression
    (s+2)*(s^2 + 8*s + 41) ...
  )...
)
```

```
s^3 + 10*s^2 + 57*s + 82
```

Now we can use `tf`.

```
H = tf(...
  [10,30],... % numerator
  [1,10,57,82] ... % denominator
)
```

```
H =

          10 s + 30
  -----------------------
  s^3 + 10 s^2 + 57 s + 82

Continuous-time transfer function.
```

a. Poles and zeros

The zeros can be computed with the function `zero`.

```
zeros_H = zero(H);
disp(zeros_H)
```

```
    -3
```

The poles can be computed with the function `pole`.

```
poles_H = pole(H);
disp(poles_H)
```

```
  -4.0000 + 5.0000i
  -4.0000 - 5.0000i
  -2.0000 + 0.0000i
```

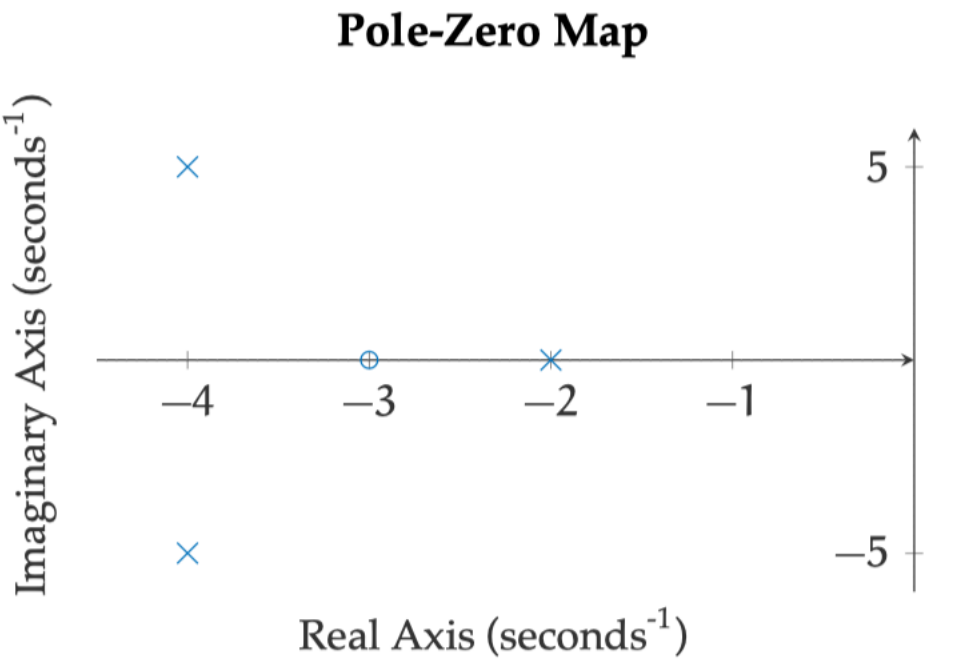
b. Stability

The poles all have negative real parts; therefore, the system is stable. There is a complex pair of poles, so system responses will tend to oscillate.

c. Pole-zero plot

The `pzmap` function gives the pole-zero plot.

```
pzmap(H)
```



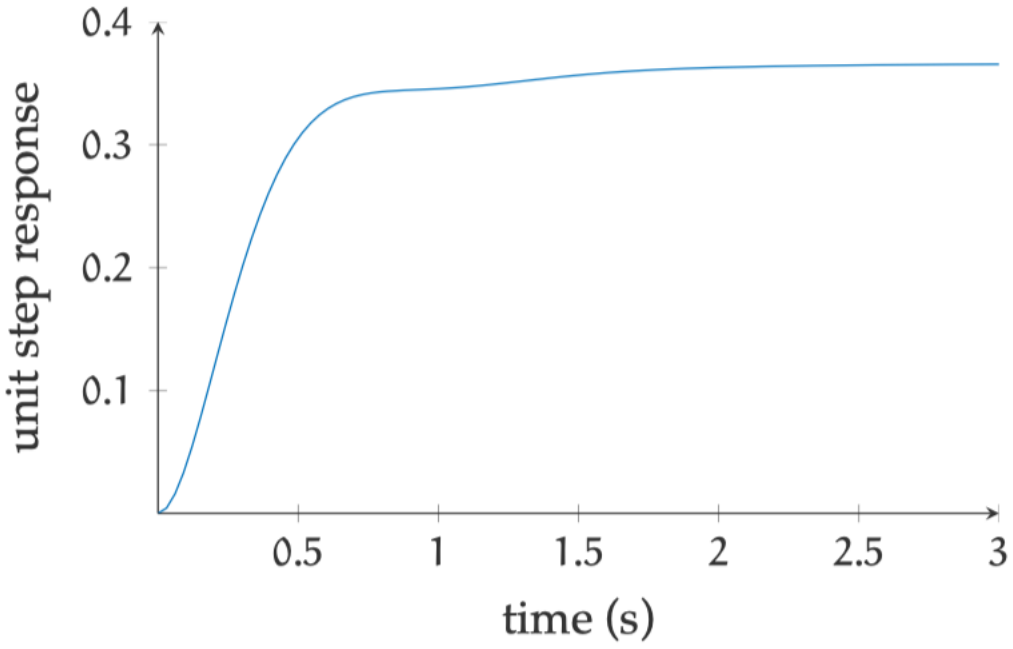
d. Step response

The unit step response is easily computed using the `step` command.

```
t = linspace(0,3,100); % 3 seconds
y = step(H,t); % step response
```

This can be plotted in the usual way.

```
figure
plot(t,y,...
  'linewidth',1 ...
)
xlabel('time (s)')
ylabel('step response')
```



Exercise 12.2 swashbuckling

Python Solution

Load Python packages

```
import numpy as np
import numpy.linalg
import matplotlib.pyplot as plt
import sympy as sp
import dysys as ds
import control
```

Define the System

Define the state-space model as follows:

```
A = np.array([[-1, 4], [0, -3]])
B = np.array([[1], [-1]])
C = np.array([[1, 0]])
D = np.array([[0]])
sys = control.ss(A, B, C, D)
```

The transfer function can be found from the sys model as follows:

```
H = control.tf(sys)
print(H)
```

```
    s - 1
-------------
s^2 + 4 s + 3
```

Poles and Zeros

The poles and zeros can be computed as follows:

```
poles = H.poles()
zeros = H.zeros()
print(f"Poles: {poles}", f"\nZeros: {zeros}")
```

```
Poles: [-3.+0.j -1.+0.j]
Zeros: [1.+0.j]
```

Comparing Poles and Eigenvalues

The poles were computed above. The eigenvalues of A can be computed as follows:

```
evals, evecs = np.linalg.eig(A)
print(f"Eigenvalues: {evals}")
```

```
Eigenvalues: [-1. -3.]
```

So, as we expected, the eigenvalues of A are equal to the poles of $H(s)$. # Pole-Zero Plot {-} A pole-zero plot can be created automatically with `control.pzmap()`, but it can be created manually (and more configurably) as follows:

```
fig, ax = plt.subplots()
ax.scatter(
    np.real(poles), np.imag(poles), s=50, marker="x", facecolors="k"
)
ax.scatter(
    np.real(zeros), np.imag(zeros),
    s=50, marker="o", facecolors="none", edgecolors="k"
)
ax.plot([1],[0])  # Force origin in view
ax.spines[['left', 'bottom']].set_position('zero')
ax.spines[['top', 'right']].set_visible(False)
ax.set_xlabel("$\\Re(s)$", loc="right")
ax.set_ylabel("$\\Im(s)$", loc="top")
plt.show()
```

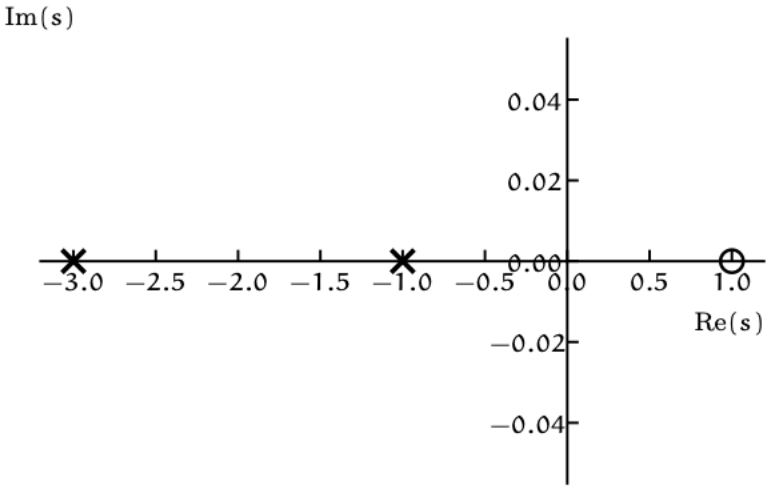


Figure tf.3: The pole-zero plot for $H(s)$.

Stability and Free Response Characteristics

The real parts of the poles are all negative; therefore, the system is asymptotically stable. The poles are real, so we expect the free response to decay to zero from the initial conditions without oscillation at two different

decay rates, one for each pole. The right half-plane zero means the system is non-minimum phase; this means the free response may initially move in the direction opposite zero. # Step Response {-}
Define a symbolic transfer function as follows:

```
s = sp.symbols("s", complex=True)
H_ = (s - 1)/(s**2 + 4*s + 3)
```

The Laplace transform of the input is, by inspection,

```
U_ = 9/s
```

So the output in the Laplace domain is

```
Y_ = H_ * U_
```

Take the inverse Laplace transform as follows:

```
t = sp.symbols("t", real=True)  # Time variable
y = sp.inverse_laplace_transform(Y_, s, t).collect(sp.Heaviside(t))
print(y)
```

```
(-3 + 9*exp(-t) - 6*exp(-3*t))*Heaviside(t)
```

Problem 12.1

(a) $$H(s) = \frac{s+2}{s+3}$$

Pole: $s = -3$
Zero: $s = -2$

(b) $$H(s) = \frac{2s+1}{s^2+7s+12} = 2\frac{s+0.5}{(s+3)(s+4)}$$

Poles: $s = -3 \quad s = -4$
Zero: $s = -0.5$

(c) $$H(s) = \frac{s}{s^2+5s+7} = \frac{s}{(s+(5/2+j\sqrt{3}/2))(s+(5/2-j\sqrt{3}/2))}$$

Poles: $s = -5/2 + j\sqrt{3}/2 \quad s = -5/2 - j\sqrt{3}/2$
Zero: $s = 0$

Problem 12.3

In Fig. 12.12a the two networks have transfer functions

$$H_1(s) = \frac{1}{0.01s+1} \qquad H_2(s) = \frac{1}{0.02s+1}$$

The amplifiers effectively isolates the two networks (no loading effect of $H_2(s)$ on $H_1(s)$) so that the complete transfer function is:

$$H_c(s) = H_1(s)H_2(s) = \frac{1}{(0.01s+1)(0.02s+1)} = \frac{1}{0.0002s^2+0.03s+1} = \frac{5000}{s^2+150s+5000}$$

In Fig 12.12b the two systems interact, and the overall system must be modeled as a unit. Using the linear graph method the transfer function is

$$H(s) = \frac{5000}{s^2+250s+5000}$$

The two systems vary in their damping. In general it is not valid to cascade tthe transfer function of two systems unless it is known that they do not load each other.

Problem 12.5

(a) Define the output of the summer to be the error $e(t)$. Then for exponential inputs

$$\begin{aligned} E(s) &= U(s) - Y(s)H_2(s) \\ Y(s) &= E(s)H_1(s) \end{aligned}$$

which are comined to give

$$H(s) = \frac{H_1(s)}{1 + H_1(s)H_2(s)}$$

(b) Let the two transfer functions be written as

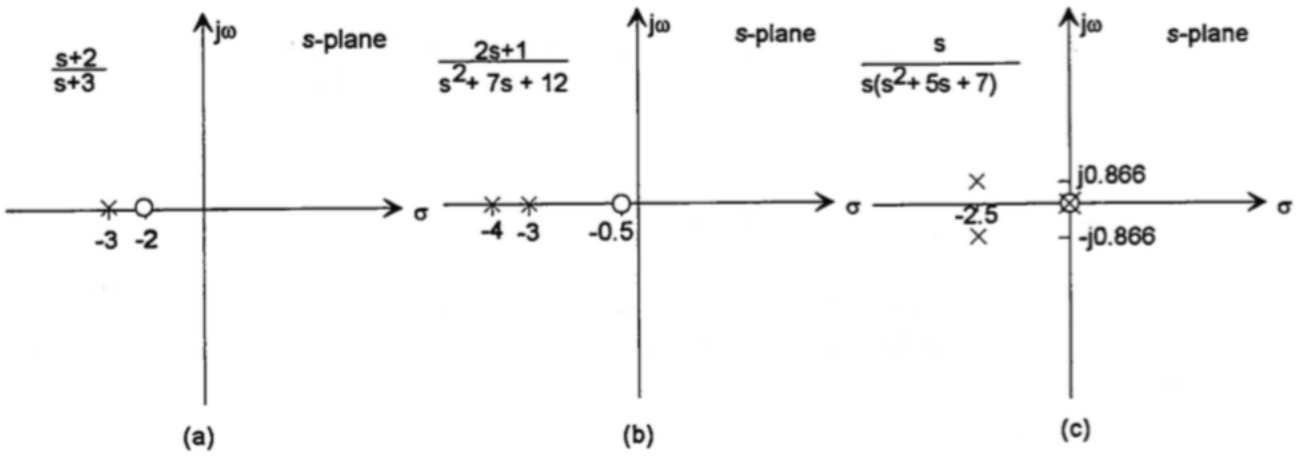
$$H_1(s) = \frac{N_1(s)}{D_1(s)} = \frac{K_1 \prod_{i=1}^{n_1}(s - z_{1,i})}{\prod_{i=1}^{m_1}(s - p_{1,i})} \qquad H_2(s) = \frac{N_2(s)}{D_2(s)} = \frac{K_2 \prod_{i=1}^{n_2}(s - z_{2,i})}{\prod_{i=1}^{m_2}(s - p_{2,i})}$$

then

$$\begin{aligned} H(s) &= \frac{N_1(s)/D_1(s)}{1 + N_1(s)N_2(s)/D_1(s)D_2(s)} \\ &= \frac{N_1(s)D_2(s)}{D_1(s)D_2(s) + N_1(s)N_2(s)} \\ &= \frac{K_1 \prod_{i=1}^{n_1}(s - z_{1,i}) \prod_{i=1}^{m_2}(s - p_{2,i})}{\prod_{i=1}^{m_1}(s - p_{1,i}) \prod_{i=1}^{m_2}(s - p_{2,i}) + K_1 K_2 \prod_{i=1}^{n_1}(s - z_{1,i}) \prod_{i=1}^{n_2}(s - z_{2,i})} \end{aligned}$$

(c) From (b) the closed-loop zeros consist of the set of unmodified zeros of $H_1(s)$ and the poles of the feedback system $H_2(s)$. The closed-loop poles cannot be simply stated, except to note that as $K_1K_2 \to 0$ the poles tend to the open-lop poles, that is the roots of $D_1(s)D_2(s) = 0$, while if $K_1K_2 \to \infty$ the poles approach the open-loop zeros, that is the roots of $N_1(s)N_2(s) = 0$

Problem 12.7



Problem 12.15

The pole-zero plots are not given here.

(a) Pole as $s = -4$. zero at $s = 0$. The system is stable. The initial condition response will be of the form

$$y(t) = Ce^{-4t}$$

(b) Poles at $s = -1$ and $s = -4$. The system is stable. The initial condition response is of the form

$$y(t) = C_1 e^{-t} + C_2 e^{-4t}$$

(c) Poles at $s = -2$ and $s = +2$. The system is unstable. The initial condition response is of the form

$$y(t) = C_1 e^{-2t} + C_2 e^{2t}$$

(d) Poles at $s = -1 + j\sqrt{(3)}$ and $s = -1 - j\sqrt{(3)}$, and a zero at $s = +2$. This is an underdamped non-minimum phase system – it is stable. The initial condition response is of the form

$$y(t) = Ce^{-t}\sin(3t + \phi$$

where C and ϕ are found from the initial conditions.

Problem 12.23

For each of the following systems

System 1: **(a)** The system eigenvalues are: $\lambda_1, \lambda_2 = -4, -1$. The system characterstic equation is $s^2 + 5s + 4 = 0$.

(b)

$$\frac{Y(s)}{U(s)} = \mathbf{C}\left[s\mathbf{I} - \mathbf{A}\right]^{-1}\mathbf{B} = \frac{4}{s^2 + 5s + 4}$$

Poles at $s_1, s_2 = -4, 1$.

(c) The transfer function poles are identical to the eigenvalues of the **A** matrix.

System 2: **(a)** The system eigenvalues are: $\lambda_1, \lambda_2 = -3.225, -0.775$. The system characterstic equation is $2s^2 + 8s + 5 = 0$.

(b)

$$\frac{Y(s)}{U(s)} = \mathbf{C}\left[s\mathbf{I} - \mathbf{A}\right]^{-1}\mathbf{B} = \frac{2}{2s^2 + 8s + 54}$$

Poles at $s_1, s_2 = - - 3.225, -0.775$.

(c) The transfer function poles are identical to the eigenvalues of the **A** matrix.