Solution Review: Transfer Functions

This lesson discusses solutions to the tasks in the exercise about transfer functions.

WE'LL COVER THE FOLLOWING

Solution 1

```
from sympy import *

s = Symbol('s')

G1 = 1 / s
G2 = 1 / (s + 1)
G3 = s / (s + 2)
G4 = 4
G5 = 1 / (s**2 + 1)
G6 = 3 * s

tf = together(G1 * ((G2 * G3) + (G4 * G5)) * G6)

print(tf)
```

Explanation

- In lines 5 - 10, we have defined the transfer function of the system components in terms of `s`.
- In line 12, we used the rules of transfer functions to reduce the system to the following equation:

$$(G_1 \times ((G_2 \times G_3) + (G_4 \times G_5)) \times G_6)$$

Solution 2

```
from sympy import *
import numpy as np

s = Symbol('s')
tf = 3 * (s * (s**2 + 1) + 4 * (s + 1) * (s + 2)) / ((s + 1) * (s + 2) * (s**2 + 1))

n_solve = solve(numer(tf))
d_solve = solve(denom(tf))

zeros = np.array([complex(item) for item in n_solve])
poles = np.array([complex(item) for item in d_solve])

print(zeros)
print(poles)
```

Explanation

- In line 5, we are directly using the transfer function `tf` that we obtained in solution 1.
- Line 7 is a two-step command. First, we extract the numerator from the transfer function `tf` using `numer`. Then, we find its roots using the `solve` function.
- Similar steps follow in line 8, except that we extract the denominator using the `denom` function to extract the poles of the system.
- Lines 10 and 11 are interesting. We use the `complex()` function to convert the SymPy imaginary numbers into regular Python imaginary numbers. See here.

Solution 3

```
from sympy import *
import numpy as np
import matplotlib.pyplot as plt

s = Symbol('s')
tf = 3 * (s * (s**2 + 1) + 4 * (s + 1) * (s + 2))/((s + 1) * (s + 2) * (s**2 + 1))

n_solve = solve(numer(tf))
d_solve = solve(denom(tf))

zeros = np.array([complex(item) for item in n_solve])
poles = np.array([complex(item) for item in d_solve])

plt.axhline(y=0, color='k')
plt.axvline(x=0, color='k')
plt.title('Pole-Zero Plot')
plt.xlabel('real')
plt.ylabel('imaginary')
plt.scatter(zeros.real, zeros.imag, marker='o', label='zero')
plt.scatter(poles.real, poles.imag, marker='x', label='pole')
plt.legend()
plt.savefig('output/plot.png')
```

Explanation

- In lines 14 - 15, we have plotted lines on the x-axis and the y-axis.
- In lines 16 - 18, we have added labels for the axes and a title for the plot.
- In line 19, we have plotted the real part of `zeros` against the imaginary part of `zeros`.
- We have done the same for `poles` in line 20.

Solution 4

```
from sympy import *
s = Symbol('s')

in1 = 1 + s
in2 = s
G1 = 1 / s
G2 = 1 / (s + 1)
G3 = 1 / (s**2 + 2)
G4 = 4

in11 = ((in1 + in2) * in2) * G1 * G2
in22 = in2 * G3 * G4
```

```
output = together(in11 - in22)
x = output.subs({s: 1})
print(x)
```

Explanation

- In lines 4 - 5, we have defined the inputs `in1` and `in2`.
- In lines 6 - 9, we have defined the systems G_1 to G_4.
- `in11` is the upper input of the last subtraction operation. It is given by the equation

$$(((1+s)+s)\times s))\times G_1\times G_2$$

- `in22` is the lower input to the last subtraction operation. It is given by:

$$s\times G_3\times G_4$$

- In line 14, we find the output and use the `together` function to get the output as a single fraction.
- In line 15, we substitute the value of `s` with `1` to find the output as $s=1$.

The next lesson gives you a preview of a scientific tool: the harmonograph.