Nama : ABELLEO DWI PUTRA

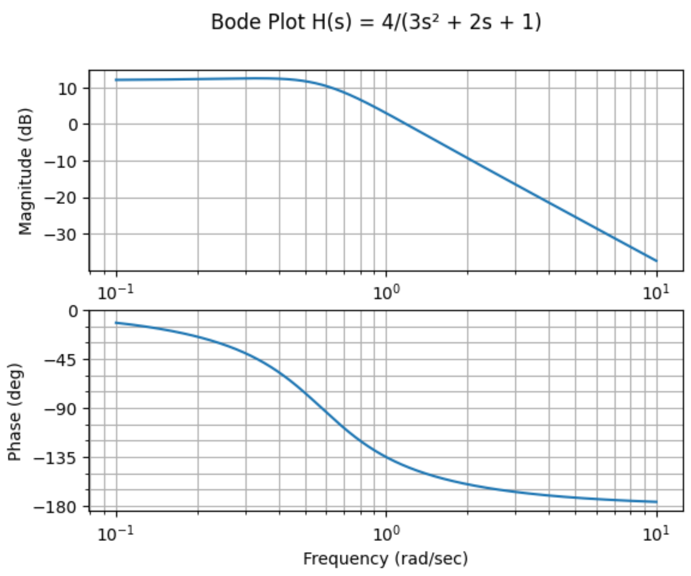
NIM : 2411010018

Jawaban soal

No. 1.

```
[4]: #No: 1. Transfer fungsi
import numpy as np
import matplotlib.pyplot as plt
import control as ctrl
num = [4]
den = [3, 2, 1]
H = ctrl.tf(num, den)
print("Transfer Function H(s):")
print(H)

# Plot respon frekuensi
plt.figure()
ctrl.bode(H, dB=True)
plt.suptitle("Bode Plot H(s) = 4/(3s² + 2s + 1)")
plt.show()
# Cek kestabilan
poles = ctrl.pole(H)
print("Poles:", poles)
if np.all(np.real(poles) < 0):
    print("Sistem stabil karena semua pole berada di sisi kiri bidang s.")
else:
    print("Sistem tidak stabil karena terdapat pole di sisi kanan bidang s.")
```

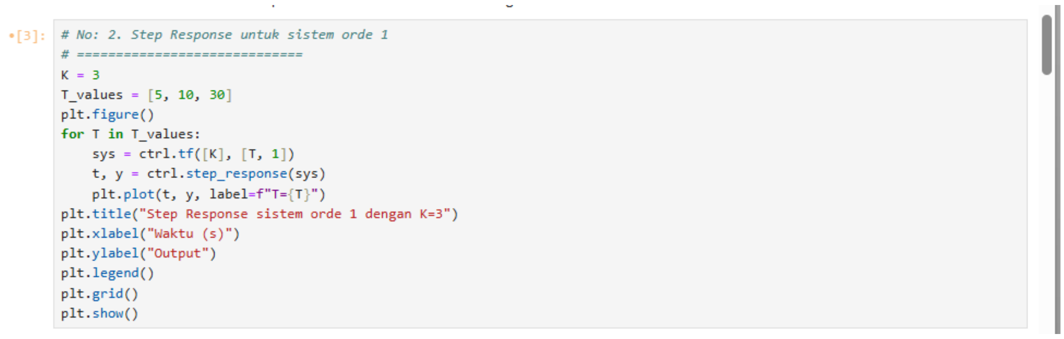
```
Transfer Function H(s):

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



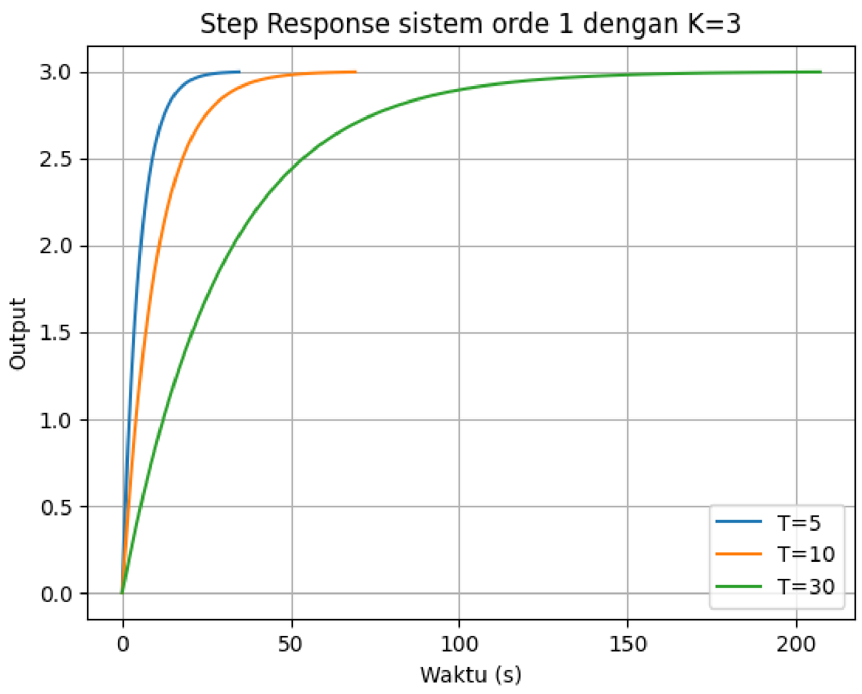
```
Poles: [-0.33333333+0.47140452j -0.33333333-0.47140452j]
Sistem stabil karena semua pole berada di sisi kiri bidang s.
```

Jawaban soal

No. 2

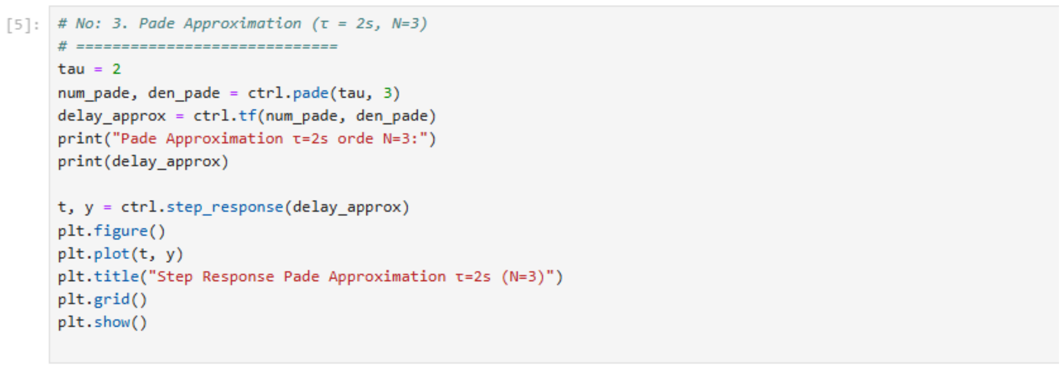


```
[3]: # No: 2. Step Response untuk sistem orde 1
     # ============================
     K = 3
     T_values = [5, 10, 30]
     plt.figure()
     for T in T_values:
         sys = ctrl.tf([K], [T, 1])
         t, y = ctrl.step_response(sys)
         plt.plot(t, y, label=f"T={T}")
     plt.title("Step Response sistem orde 1 dengan K=3")
     plt.xlabel("Waktu (s)")
     plt.ylabel("Output")
     plt.legend()
     plt.grid()
     plt.show()
```



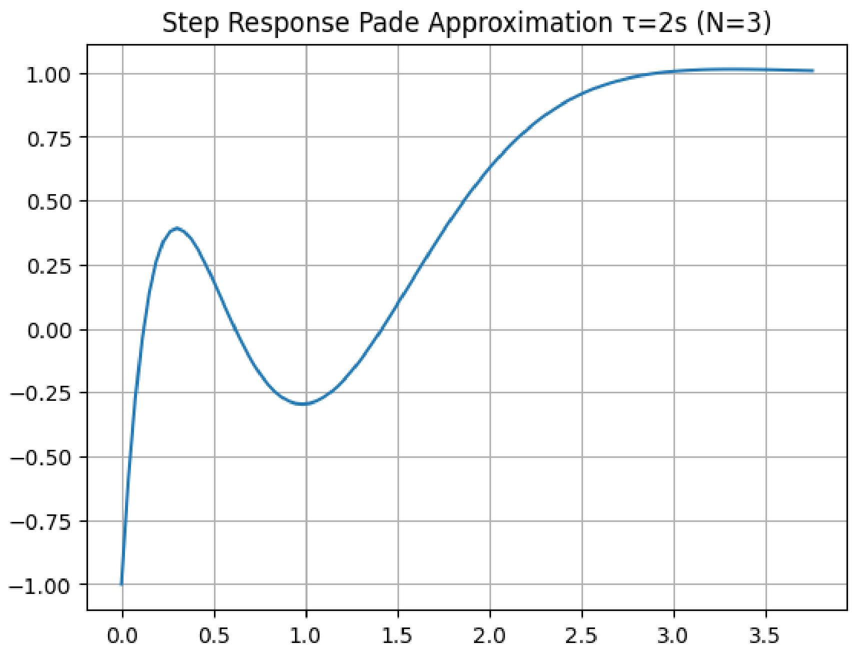
Jawaban soal

no.3



```
[5]: # No: 3. Pade Approximation (τ = 2s, N=3)
     # ============================
     tau = 2
     num_pade, den_pade = ctrl.pade(tau, 3)
     delay_approx = ctrl.tf(num_pade, den_pade)
     print("Pade Approximation τ=2s orde N=3:")
     print(delay_approx)

     t, y = ctrl.step_response(delay_approx)
     plt.figure()
     plt.plot(t, y)
     plt.title("Step Response Pade Approximation τ=2s (N=3)")
     plt.grid()
     plt.show()
```

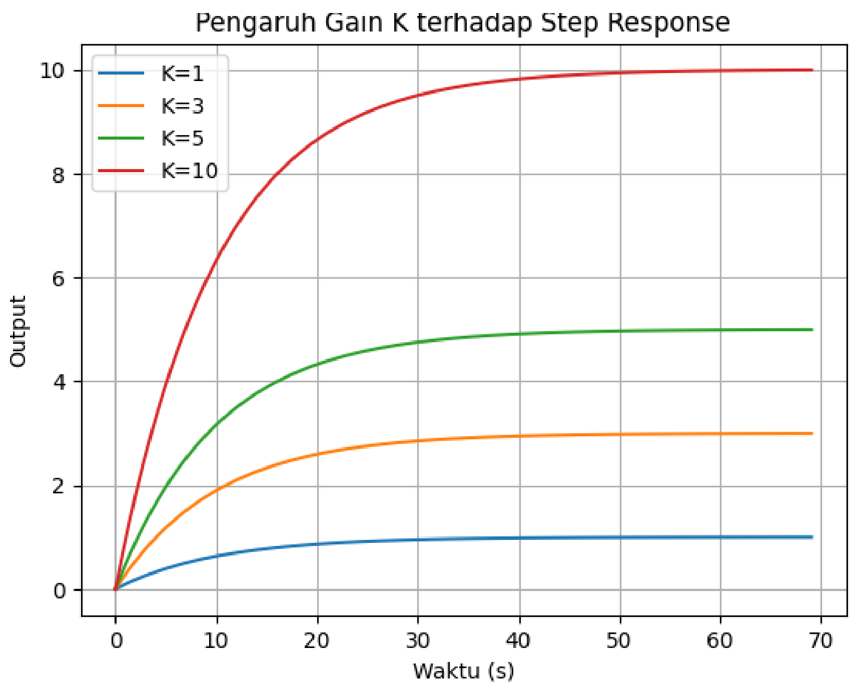
```
Pade Approximation τ=2s orde N=3:

-s^3 + 6 s^2 - 15 s + 15
------------------------
s^3 + 6 s^2 + 15 s + 15
```



Jawaban soal

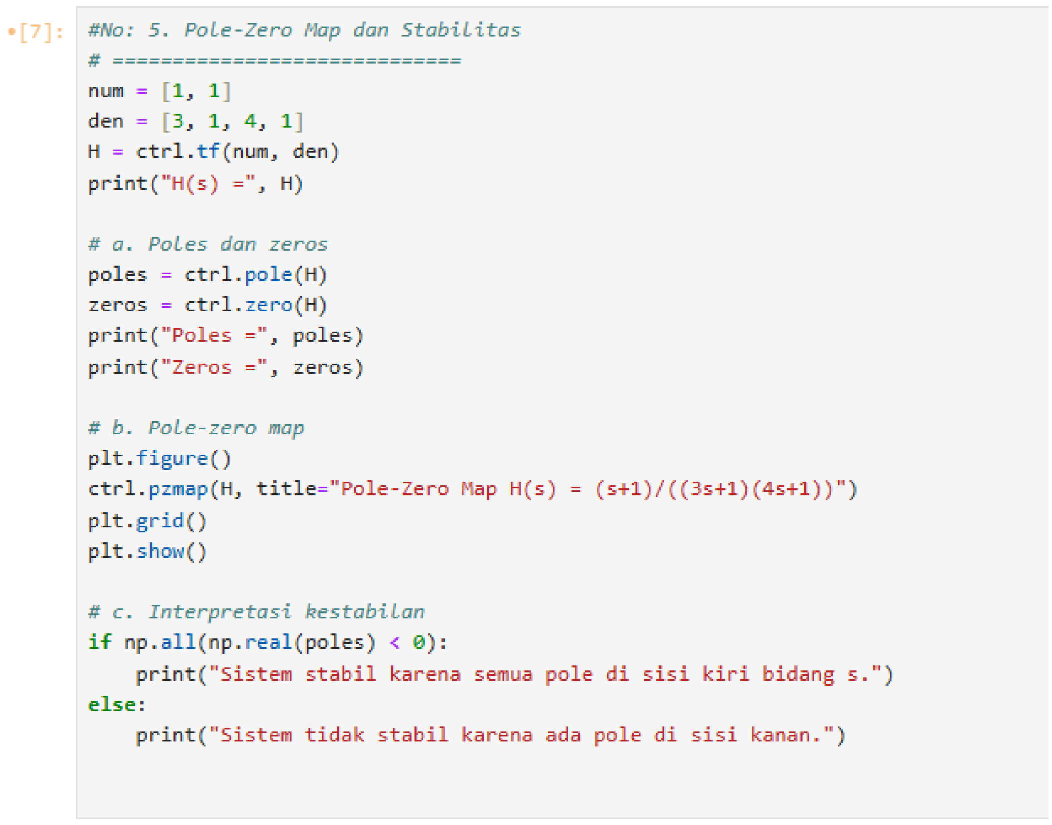
no. 4

```
[6]: #No: 4. Pengaruh K (Gain) terhadap Step Response
     # =============================
     K_values = [1, 3, 5, 10]
     plt.figure()
     for K in K_values:
         sys = ctrl.tf([K], [10, 1])
         t, y = ctrl.step_response(sys)
         plt.plot(t, y, label=f"K={K}")
     plt.title("Pengaruh Gain K terhadap Step Response")
     plt.xlabel("Waktu (s)")
     plt.ylabel("Output")
     plt.legend()
     plt.grid()
     plt.show()
```



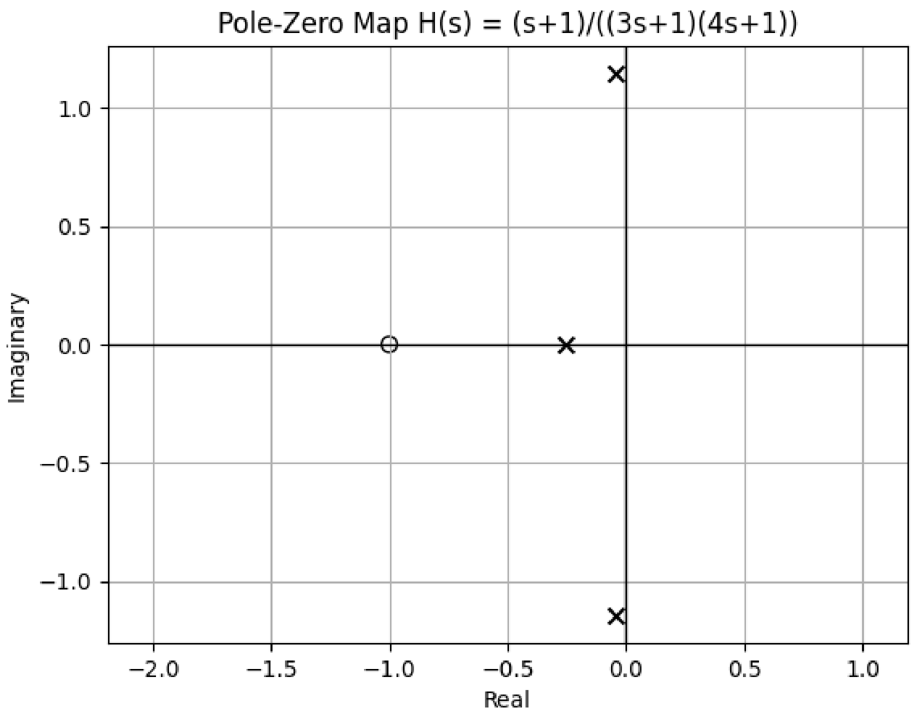
Jawaban Soal

no. 5



```
•[7]: #No: 5. Pole-Zero Map dan Stabilitas
      # =============================
      num = [1, 1]
      den = [3, 1, 4, 1]
      H = ctrl.tf(num, den)
      print("H(s) =", H)

      # a. Poles dan zeros
      poles = ctrl.pole(H)
      zeros = ctrl.zero(H)
      print("Poles =", poles)
      print("Zeros =", zeros)

      # b. Pole-zero map
      plt.figure()
      ctrl.pzmap(H, title="Pole-Zero Map H(s) = (s+1)/((3s+1)(4s+1))")
      plt.grid()
      plt.show()

      # c. Interpretasi kestabilan
      if np.all(np.real(poles) < 0):
          print("Sistem stabil karena semua pole di sisi kiri bidang s.")
      else:
          print("Sistem tidak stabil karena ada pole di sisi kanan.")
```

```
H(s) =
        s + 1
---------------------
3 s^3 + s^2 + 4 s + 1

Poles = [-0.03974588+1.14524027j -0.03974588-1.14524027j -0.25384157+0.j        ]
Zeros = [-1.+0.j]
```



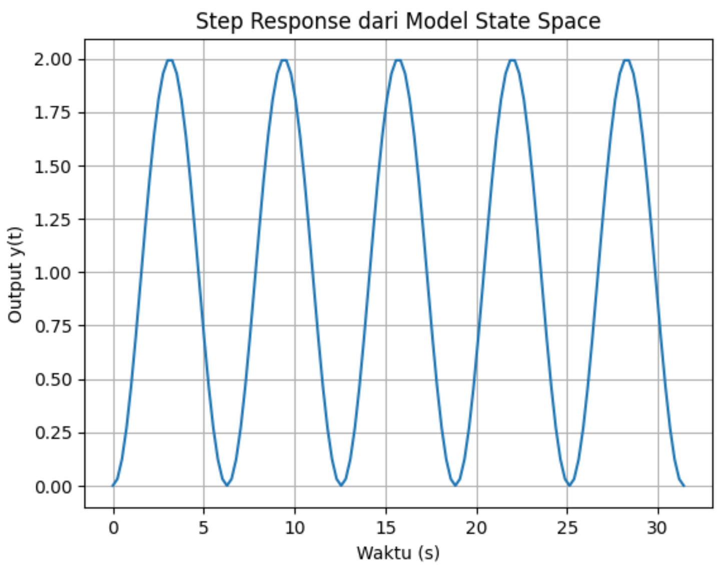
```
Sistem stabil karena semua pole di sisi kiri bidang s.
```

Jawaban Soal

no.6

```
•[8]:
# No: 6. State Space Model
# ============================
A = np.array([[0, 1],
              [-1, 0]])
B = np.array([[0],
              [1]])
C = np.array([[1, 0]])
D = np.array([[0]])

sys_ss = ctrl.ss(A, B, C, D)
print("Model State-Space (A,B,C,D):\n", sys_ss)

# Step response
t, y = ctrl.step_response(sys_ss)
plt.figure()
plt.plot(t, y)
plt.title("Step Response dari Model State Space")
plt.xlabel("Waktu (s)")
plt.ylabel("Output y(t)")
plt.grid()
plt.show()

# Interpretasi stabilitas
eig_values = np.linalg.eigvals(A)
print("Eigenvalue =", eig_values)
if np.all(np.real(eig_values) < 0):
    print("Sistem stabil.")
else:
    print("Sistem tidak stabil (osilasi / marginally stable).")
```

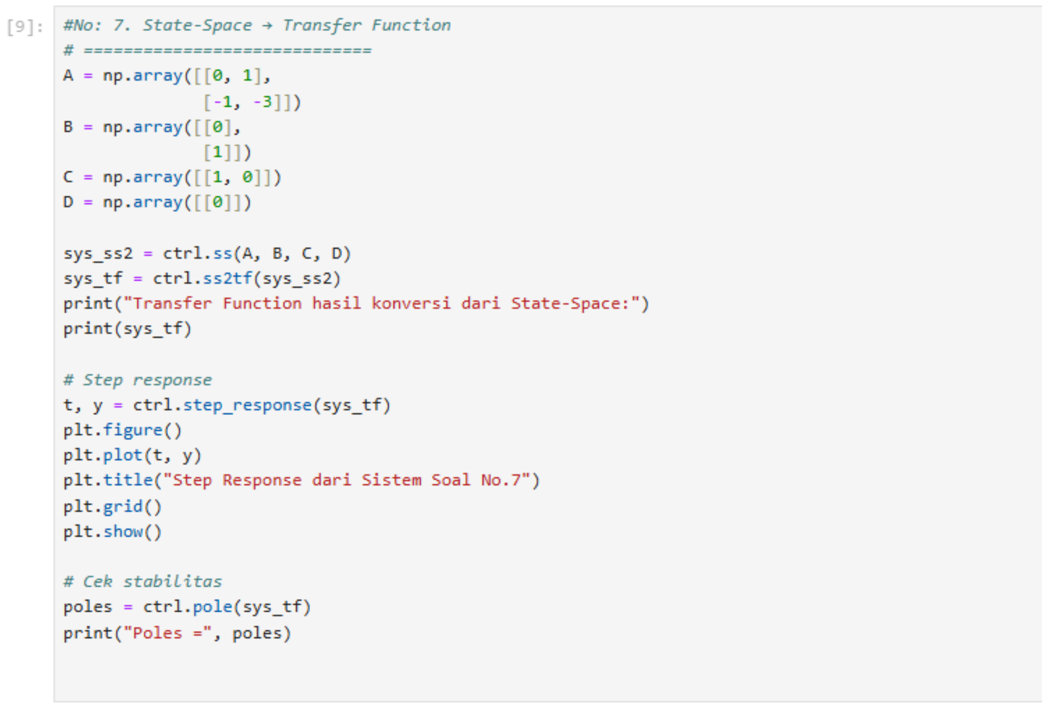
```
Model State-Space (A,B,C,D):
 <LinearIOSystem>: sys[31]
Inputs (1): ['u[0]']
Outputs (1): ['y[0]']
States (2): ['x[0]', 'x[1]']

A = [[ 0.  1.]
     [-1.  0.]]

B = [[0.]
     [1.]]

C = [[1. 0.]]

D = [[0.]]
```



```
Eigenvalue = [0.+1.j 0.-1.j]
Sistem tidak stabil (osilasi / marginally stable).
```

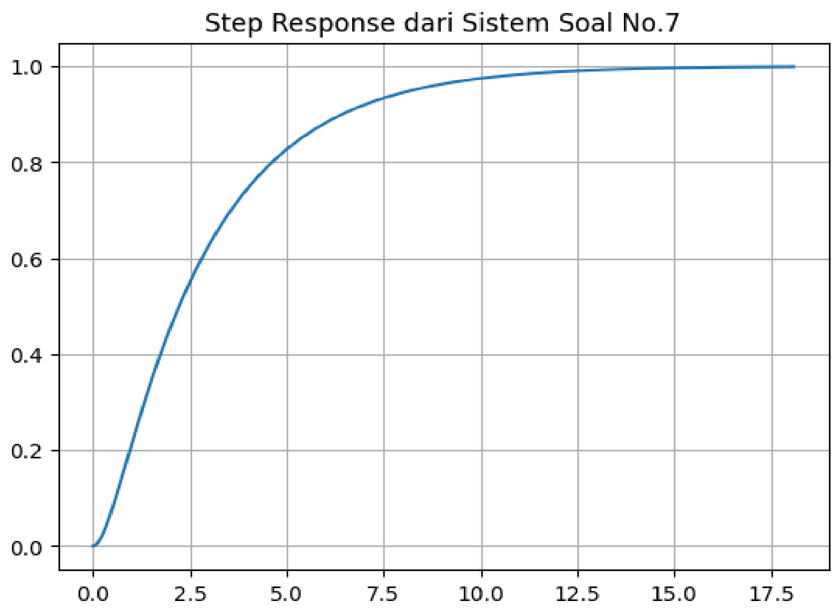
Jawaban Soal

no.7



```
[9]: #No: 7. State-Space → Transfer Function
     # ============================
     A = np.array([[0, 1],
                   [-1, -3]])
     B = np.array([[0],
                   [1]])
     C = np.array([[1, 0]])
     D = np.array([[0]])

     sys_ss2 = ctrl.ss(A, B, C, D)
     sys_tf = ctrl.ss2tf(sys_ss2)
     print("Transfer Function hasil konversi dari State-Space:")
     print(sys_tf)

     # Step response
     t, y = ctrl.step_response(sys_tf)
     plt.figure()
     plt.plot(t, y)
     plt.title("Step Response dari Sistem Soal No.7")
     plt.grid()
     plt.show()

     # Cek stabilitas
     poles = ctrl.pole(sys_tf)
     print("Poles =", poles)
```

```
Transfer Function hasil konversi dari State-Space:

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



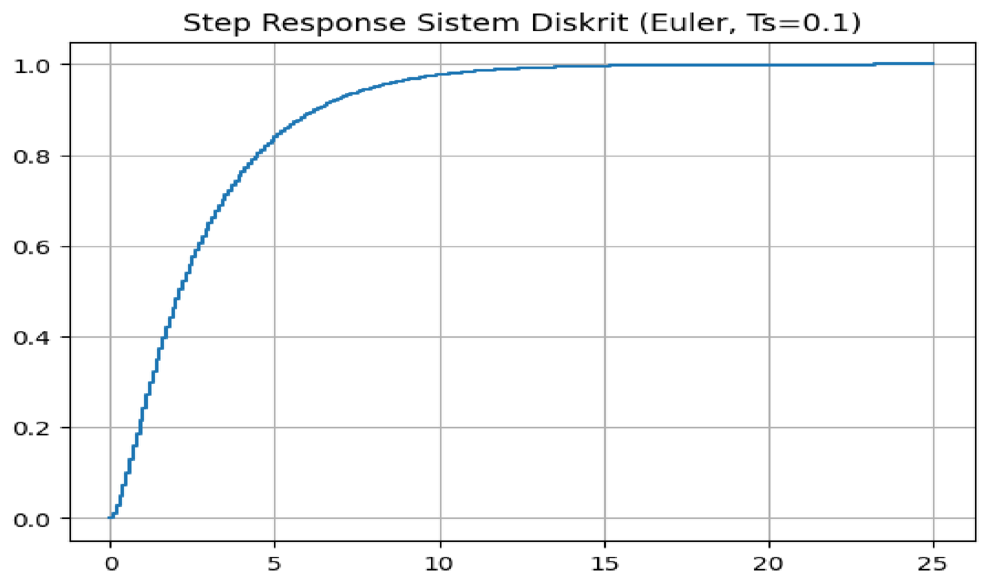
```
Poles = [-2.61803399+0.j -0.38196601+0.j]
```

Jawaban Soal

no.8

```
[10]: #No: 8. Discrete System (Euler method)
      # ============================
      Ts = 0.1
      sys_d = ctrl.c2d(sys_ss2, Ts, method='euler')
      print("Model Diskrit (Euler, Ts=0.1):\n", sys_d)

      t, y = ctrl.step_response(sys_d)
      plt.figure()
      plt.step(t, y)
      plt.title("Step Response Sistem Diskrit (Euler, Ts=0.1)")
      plt.grid()
      plt.show()
```

```
Model Diskrit (Euler, Ts=0.1):
 <LinearIOSystem>: sys[32]$sampled
Inputs (1): ['u[0]']
Outputs (1): ['y[0]']
States (2): ['x[0]', 'x[1]']

A = [[ 1.    0.1]
     [-0.1  0.7]]

B = [[0. ]
     [0.1]]

C = [[1. 0.]]

D = [[0.]]

dt = 0.1
```

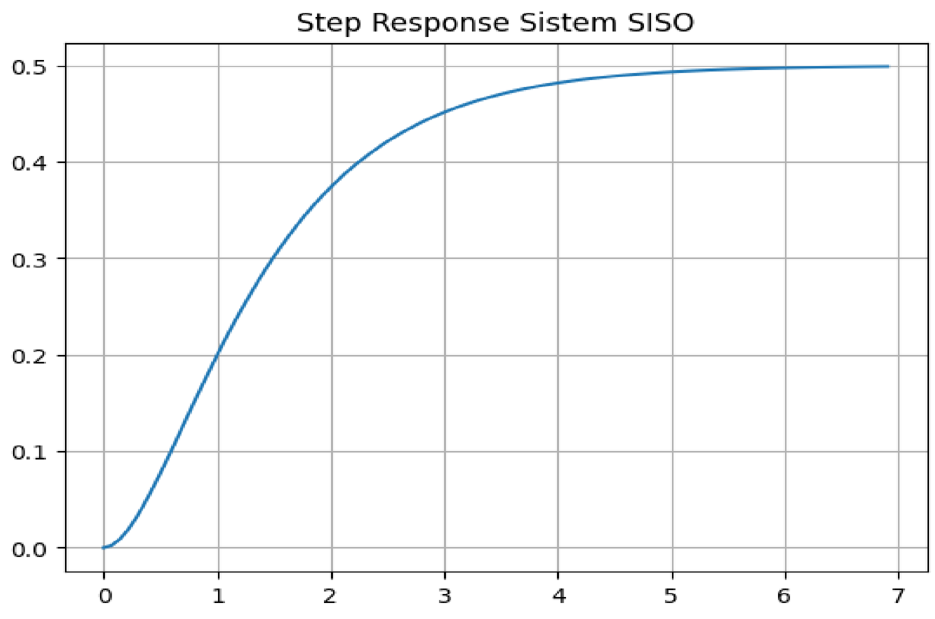


Jawaban Soal

no.9

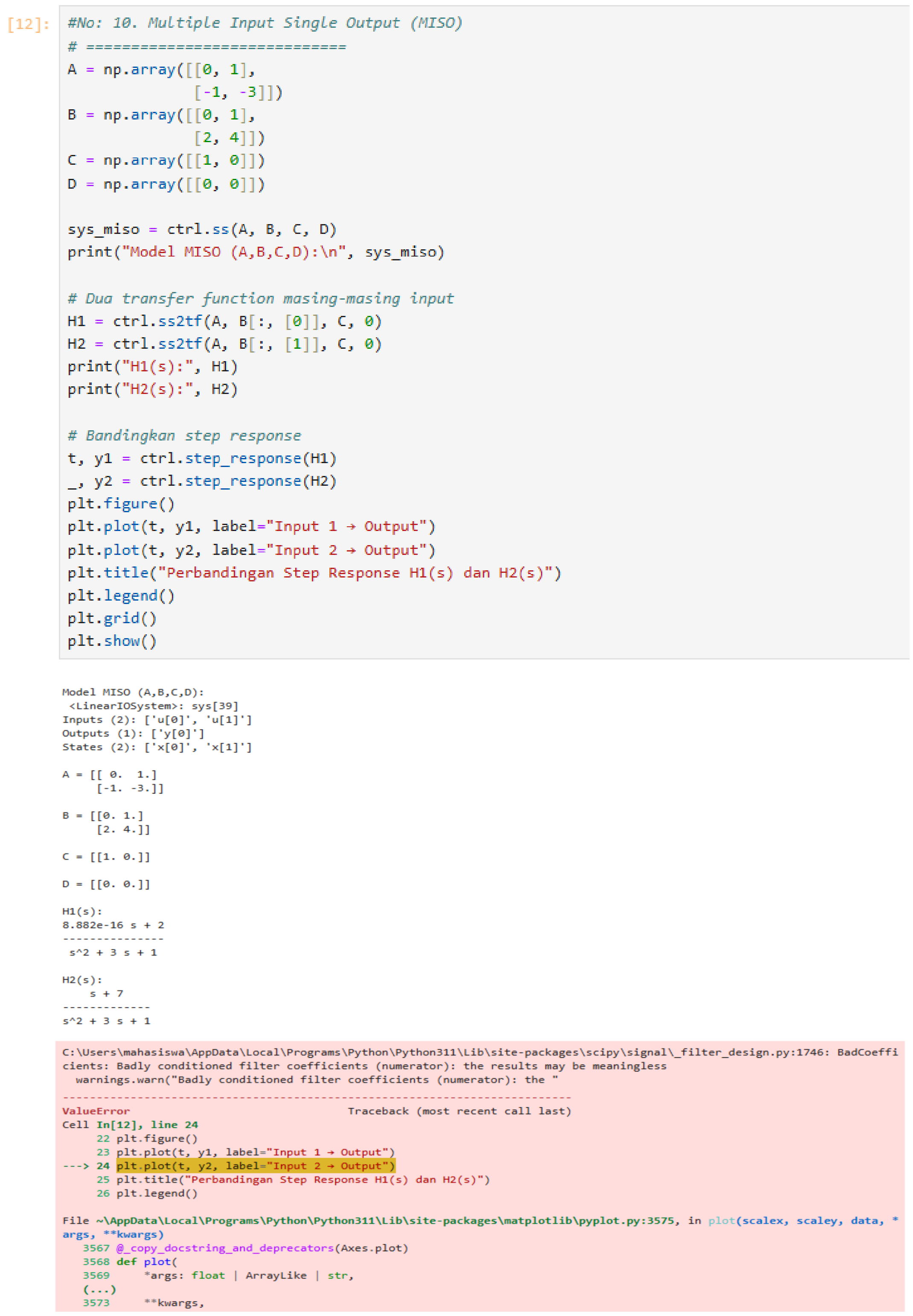
```
[11]: #No: 9. Single Input Single Output (SISO)
      # =============================
      A = np.array([[0, 1],
                    [-2, -3]])
      B = np.array([[0],
                    [1]])
      C = np.array([[1, 0]])
      D = 0

      sys_siso = ctrl.ss(A, B, C, D)
      H_siso = ctrl.ss2tf(sys_siso)
      print("Transfer Function H(s) untuk SISO:")
      print(H_siso)

      t, y = ctrl.step_response(sys_siso)
      plt.figure()
      plt.plot(t, y)
      plt.title("Step Response Sistem SISO")
      plt.grid()
      plt.show()
```

```
Transfer Function H(s) untuk SISO:

8.882e-16 s + 1
---------------
 s^2 + 3 s + 2
```



Jawaban Soal

no.10

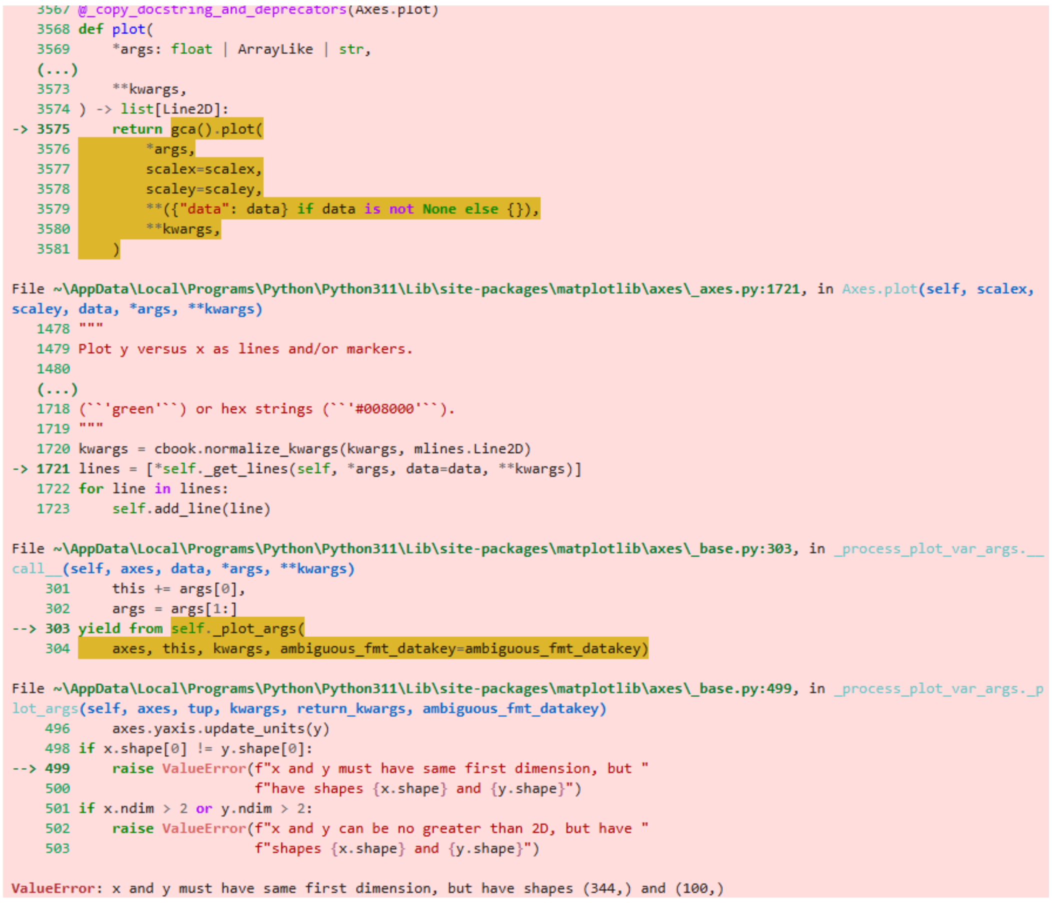


[12]:

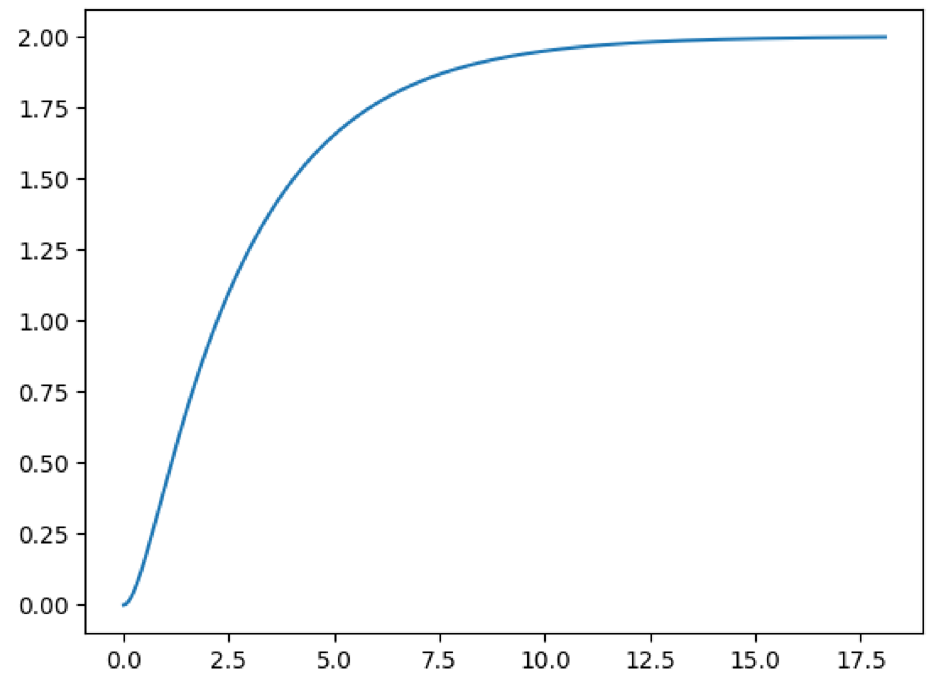
```
#No: 10. Multiple Input Single Output (MISO)
# ==============================
A = np.array([[0, 1],
              [-1, -3]])
B = np.array([[0, 1],
              [2, 4]])
C = np.array([[1, 0]])
D = np.array([[0, 0]])

sys_miso = ctrl.ss(A, B, C, D)
print("Model MISO (A,B,C,D):\n", sys_miso)

# Dua transfer function masing-masing input
H1 = ctrl.ss2tf(A, B[:, [0]], C, 0)
H2 = ctrl.ss2tf(A, B[:, [1]], C, 0)
print("H1(s):", H1)
print("H2(s):", H2)

# Bandingkan step response
t, y1 = ctrl.step_response(H1)
_, y2 = ctrl.step_response(H2)
plt.figure()
plt.plot(t, y1, label="Input 1 → Output")
plt.plot(t, y2, label="Input 2 → Output")
plt.title("Perbandingan Step Response H1(s) dan H2(s)")
plt.legend()
plt.grid()
plt.show()
```

```
Model MISO (A,B,C,D):
 <LinearIOSystem>: sys[39]
Inputs (2): ['u[0]', 'u[1]']
Outputs (1): ['y[0]']
States (2): ['x[0]', 'x[1]']

A = [[ 0.  1.]
     [-1. -3.]]

B = [[0. 1.]
     [2. 4.]]

C = [[1. 0.]]

D = [[0. 0.]]

H1(s):
8.882e-16 s + 2
---------------
 s^2 + 3 s + 1

H2(s):
    s + 7
-------------
s^2 + 3 s + 1
```

```
C:\Users\mahasiswa\AppData\Local\Programs\Python\Python311\Lib\site-packages\scipy\signal\_filter_design.py:1746: BadCoeffi
cients: Badly conditioned filter coefficients (numerator): the results may be meaningless
  warnings.warn("Badly conditioned filter coefficients (numerator): the "

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[12], line 24
     22 plt.figure()
     23 plt.plot(t, y1, label="Input 1 → Output")
---> 24 plt.plot(t, y2, label="Input 2 → Output")
     25 plt.title("Perbandingan Step Response H1(s) dan H2(s)")
     26 plt.legend()

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\matplotlib\pyplot.py:3575, in plot(scalex, scaley, data, *
args, **kwargs)
   3567 @_copy_docstring_and_deprecators(Axes.plot)
   3568 def plot(
   3569     *args: float | ArrayLike | str,
   (...)
   3573     **kwargs,
```



```
   3567 @_copy_docstring_and_deprecators(Axes.plot)
   3568 def plot(
   3569     *args: float | ArrayLike | str,
   (...)
   3573     **kwargs,
   3574 ) -> list[Line2D]:
-> 3575     return gca().plot(
   3576         *args,
   3577         scalex=scalex,
   3578         scaley=scaley,
   3579         **({"data": data} if data is not None else {}),
   3580         **kwargs,
   3581     )

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\matplotlib\axes\_axes.py:1721, in Axes.plot(self, scalex,
scaley, data, *args, **kwargs)
   1478 """
   1479 Plot y versus x as lines and/or markers.
   1480
   (...)
   1718 (``'green'``) or hex strings (``'#008000'``).
   1719 """
   1720 kwargs = cbook.normalize_kwargs(kwargs, mlines.Line2D)
-> 1721 lines = [*self._get_lines(self, *args, data=data, **kwargs)]
   1722 for line in lines:
   1723     self.add_line(line)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\matplotlib\axes\_base.py:303, in _process_plot_var_args.__
call__(self, axes, data, *args, **kwargs)
    301     this += args[0],
    302     args = args[1:]
--> 303 yield from self._plot_args(
    304     axes, this, kwargs, ambiguous_fmt_datakey=ambiguous_fmt_datakey)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\matplotlib\axes\_base.py:499, in _process_plot_var_args._p
lot_args(self, axes, tup, kwargs, return_kwargs, ambiguous_fmt_datakey)
    496     axes.yaxis.update_units(y)
    498 if x.shape[0] != y.shape[0]:
--> 499     raise ValueError(f"x and y must have same first dimension, but "
    500                      f"have shapes {x.shape} and {y.shape}")
    501 if x.ndim > 2 or y.ndim > 2:
    502     raise ValueError(f"x and y can be no greater than 2D, but have "
    503                      f"shapes {x.shape} and {y.shape}")

ValueError: x and y must have same first dimension, but have shapes (344,) and (100,)
```


2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5