5ELEN018W - Robotic Principles
Lecture 5: Control - Part 3

Dr Dimitris C. Dracopoulos

Analysis of Control Systems - Performance of Controllers

- steady-state error
- transient response
- overshooting
- settling time
- stability

Stability

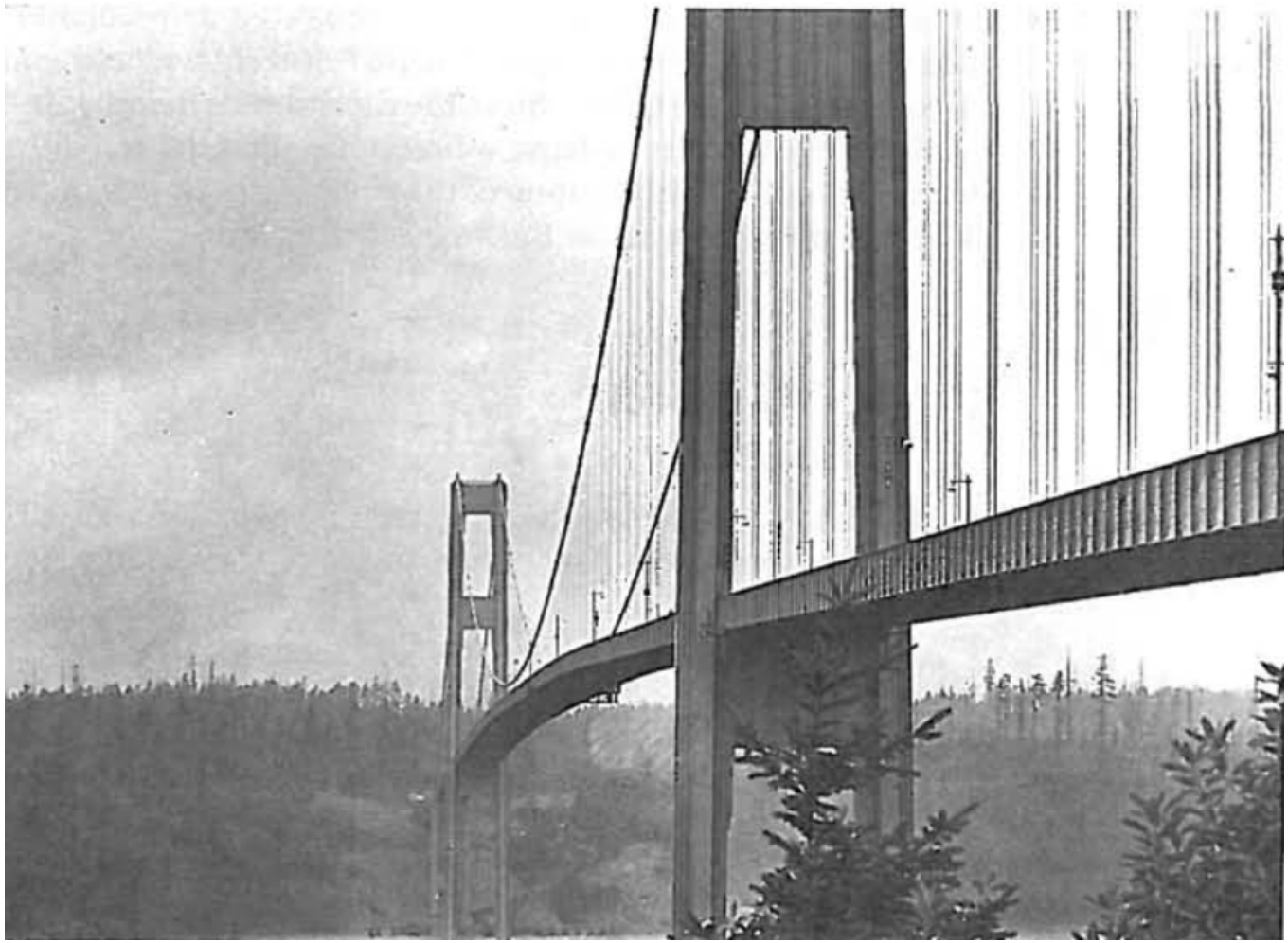
Stability is very important.

- A stable system should have a bounded output if the corresponding input is bounded.

Stability in Systems - Example

The Tacoma Narrows bridge in Washington was opened to traffic on 1st of July 1940.

- The bridge was oscillating every time the wind blew.



Stability in Systems - Example (cont'd)

- After 4 months (November 7th 1940) the bridge collapsed.



https://www.youtube.com/watch?v=y0xohjV7Avo

Time Response Analysis

Plots of the plant's output vs the desired behaviour can be used for the analysis of controllers.

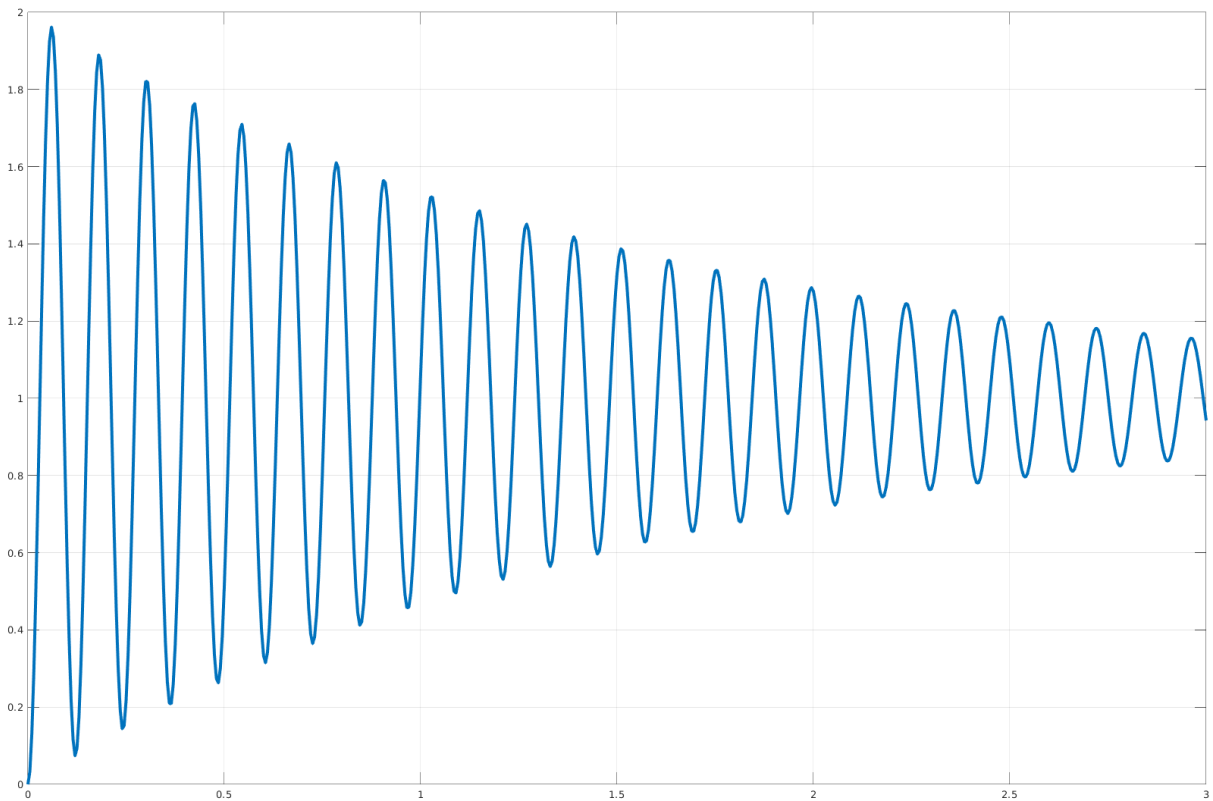
- **Step response:** the control system performance specifications are often given in terms of the *unit step response*.



Example of a Unit Step Response

System having the following transfer function:

$$\frac{5400}{2s^2 + 2.5s + 5402} \tag{1}$$



Example of a Unit Step Response (cont'd)

Matlab code:

```
num = [5400];
den = [2 2.5 5402];
sys = tf(num, den)
t = [0:0.005:3];
[y,t] = step(sys,t);
plot(t,y,'LineWidth',3),grid
xlabel('time(s)')
ylabel('System output')
```

Frequency Response

The frequency response of a system is defined as the steady-state response of the system to a sinusoidal input signal. The output for a linear system is also sinusoidal with the same frequency.

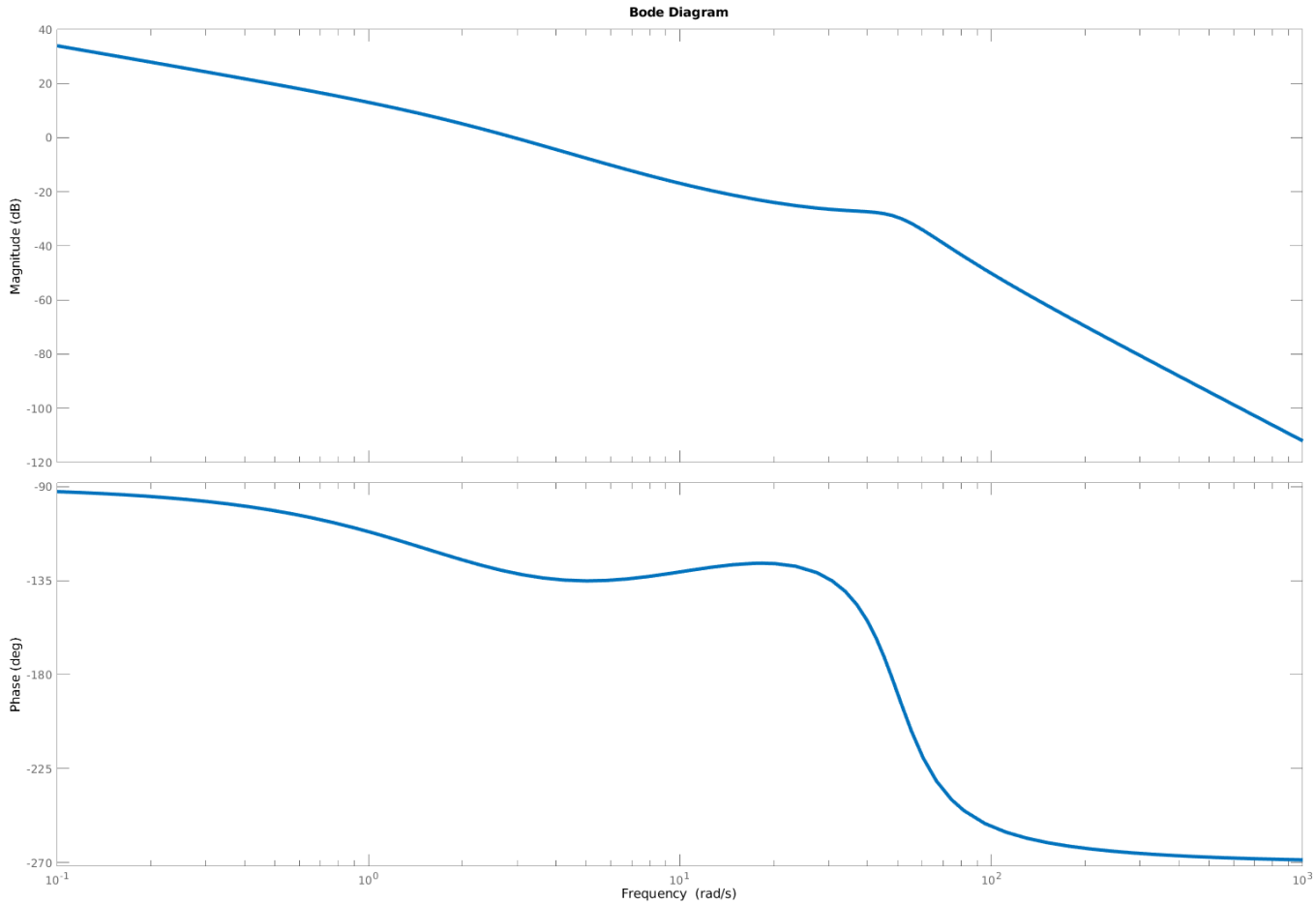
- Although the output response has the same frequency as the input, they have different amplitude and phase angle.

Bode Plots

Hendrik Wade Bode an engineer working in Bell Labs in the 1930s came up with Bode plots, which can be used to determine the stability of systems.
The plot consists of 2 diagrams:

- The magnitude plot (magnitude in decibels), frequency in rad/s. (frequency values are logarithmic, while the magnitude $|H|$ values are in decibels, i.e. $20 \log_{10} |H|$.
- The phase plot (phase in degrees), frequency in rad/s. Frequency values are in the logarithmic scale but the phase values are in the linear scale.

Example of a Bode Plot




```
sys=tf(num, den);   % previous system example
bode(sys)
```

Gain Margin and Phase Margin in Bode Plots

The gain margin and the phase margin can be used to determine the stability of the system.
To calculate the **gain margin**:

1. Find the frequency corresponding to a phase of -180^o from the phase plot.
2. For that frequency find the corresponding magnitude M_1 from the magnitude plot.
3. The **gain margin** is the difference between the magnitude of $0dB$ and M_1: $0 - M_1$.

Gain Margin and Phase Margin in Bode Plots (cont'd)

To calculate the **phase margin**:

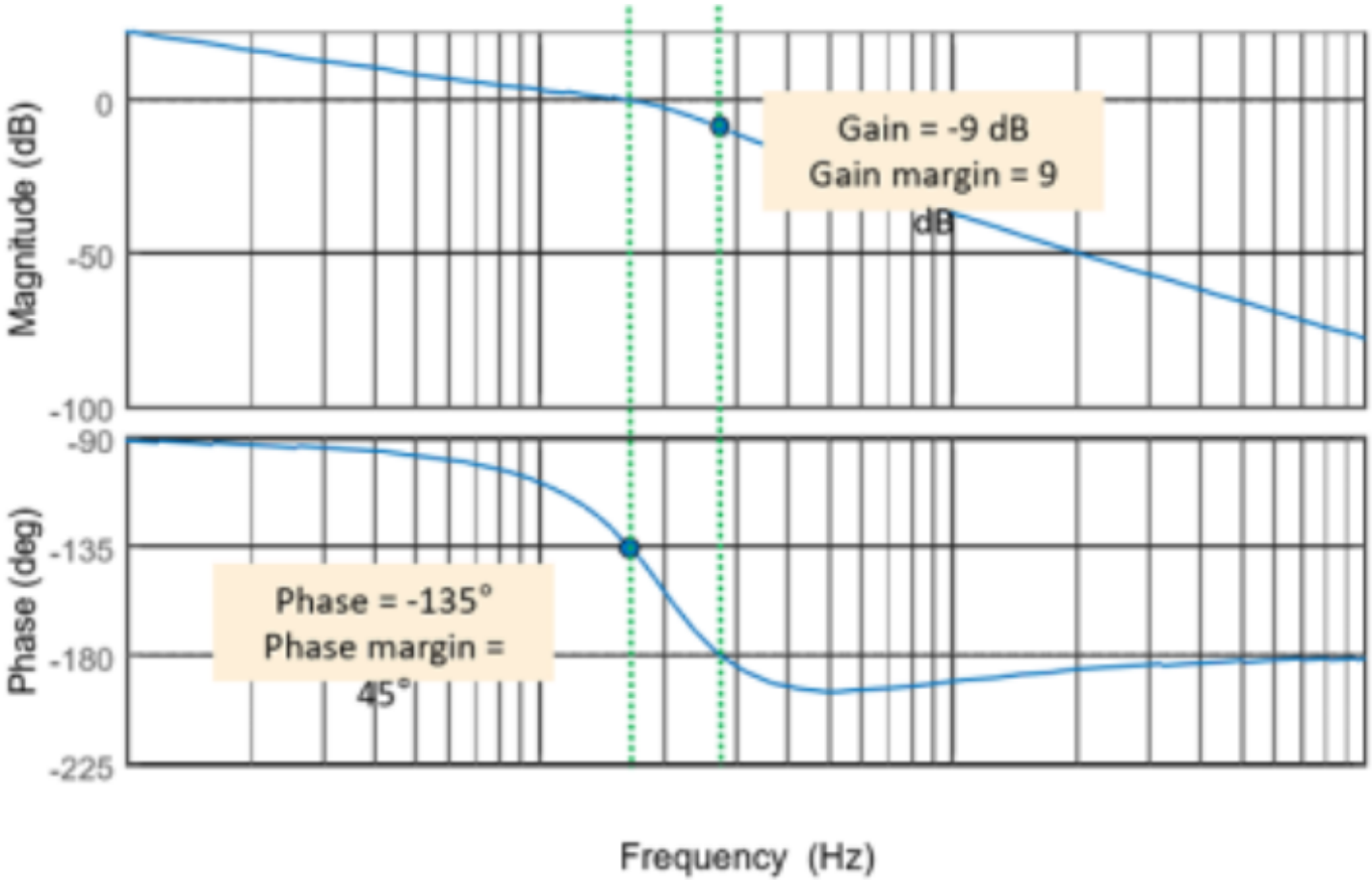
1. Find the frequency corresponding to a magnitude of $0dB$ from the magnitude plot.
2. For that frequency find the corresponding phase P_1 from the phase plot.
3. The **phase margin** is the difference between P_1 and -180^o: $P_1 - (-180) = P_1 + 180$.

How to determine Stability from the Bode Plots

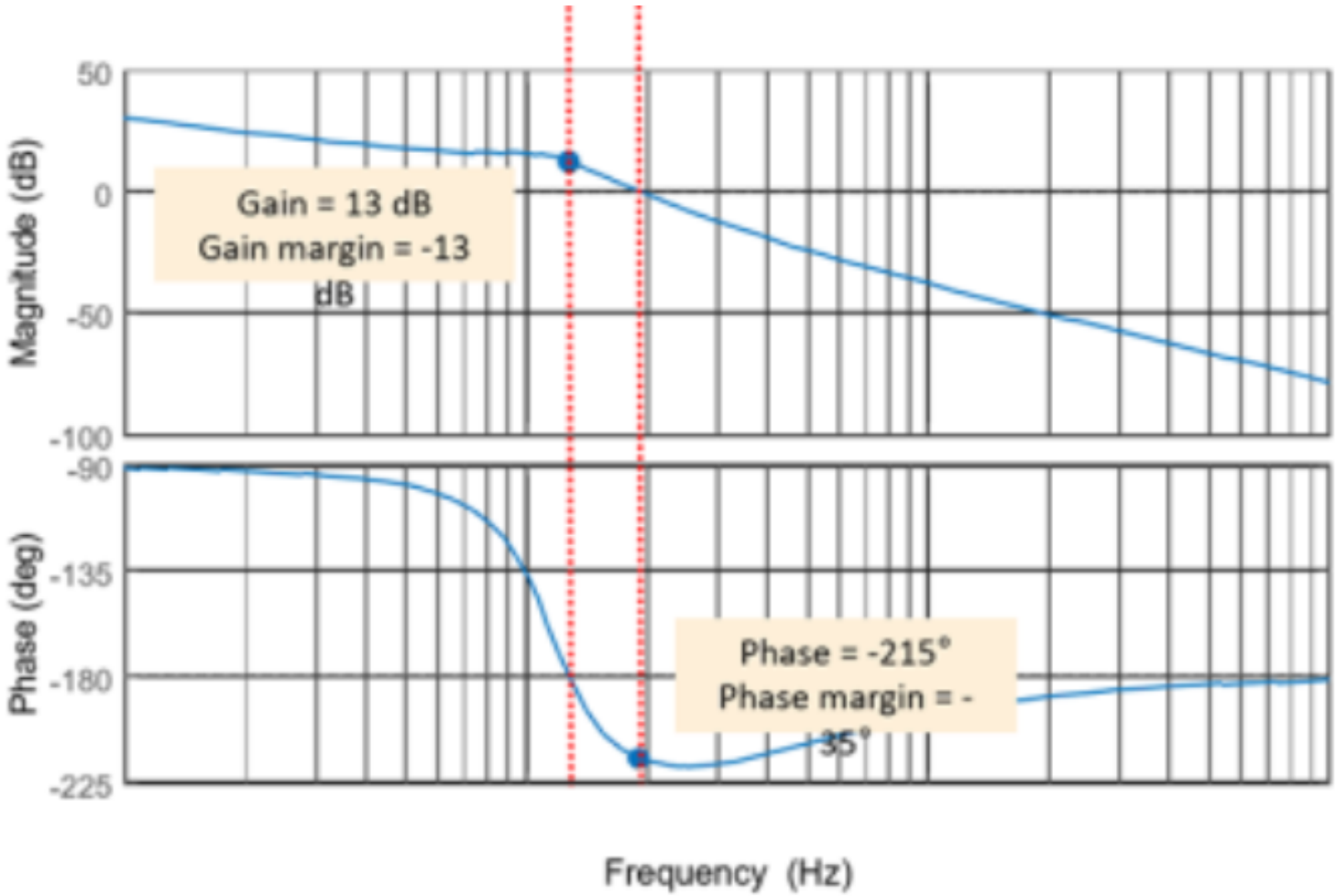
Stability can be determined by calculating either the gain margin or the phase margin:

- A positive gain margin indicates a stable system, while a negative gain margin indicates an unstable system. The larger the positive gain margin, the system has more "margin" before going to instability.
- A positive phase margin indicates a stable system, while a negative phase margin indicates an unstable system. The larger the positive margin the more stable is the system and a value of at least 45 degrees is desirable.

Example 2 of a Bode Plot



Example 3 of a Bode Plot



Bode Plots in Simulink

Problem: Watertank

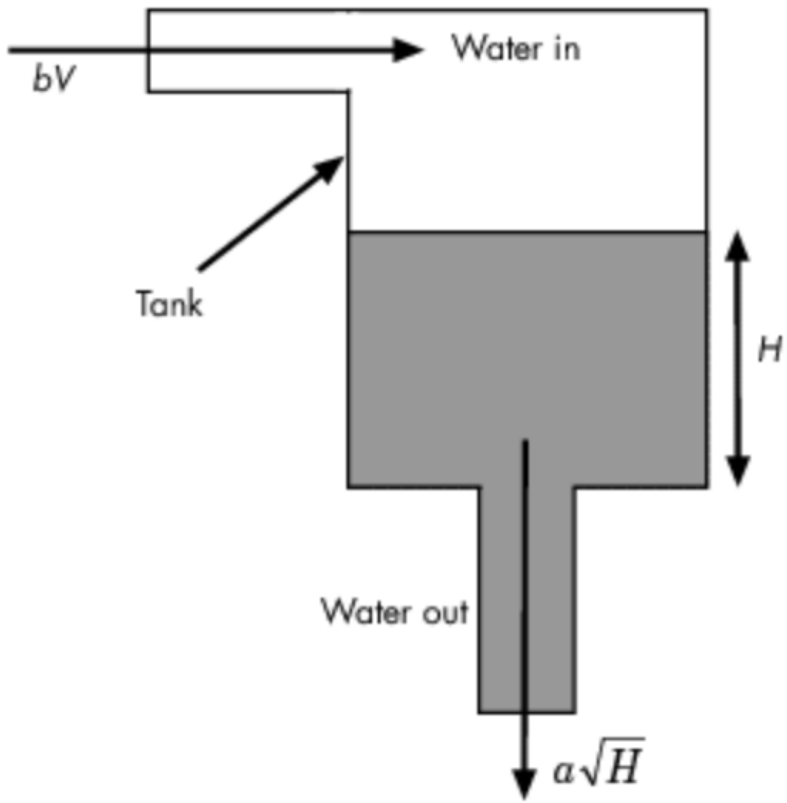


Figure 1: The watertank dynamic system.

Bode Plots in Simulink (cont'd)

$$\frac{d}{dt}Vol = A\frac{dH}{dt} = bV - \alpha\sqrt{H}$$

- H is the height of the water in the tank
- Vol is the volume of the water in the tank
- V is the control input (water to put into the tank).
- $b = 5$ is a constant related to the flow rate into the tank
- $A = 20$ relates to the area of the tank
- $\alpha = 2$ is a constant related to the flow rate out of the tank
- Initial condition for H is $H(0) = 1$

Bode Plots in Simulink (cont'd)

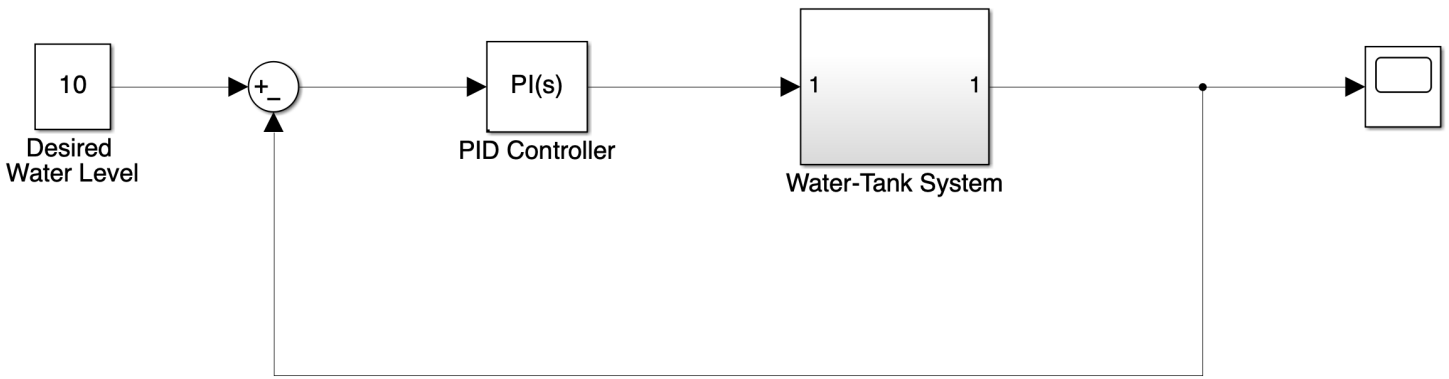


Figure 2: The watertank dynamic system in Matlab (built-in example up to version 2022).

Bode Plots in Simulink (cont'd)

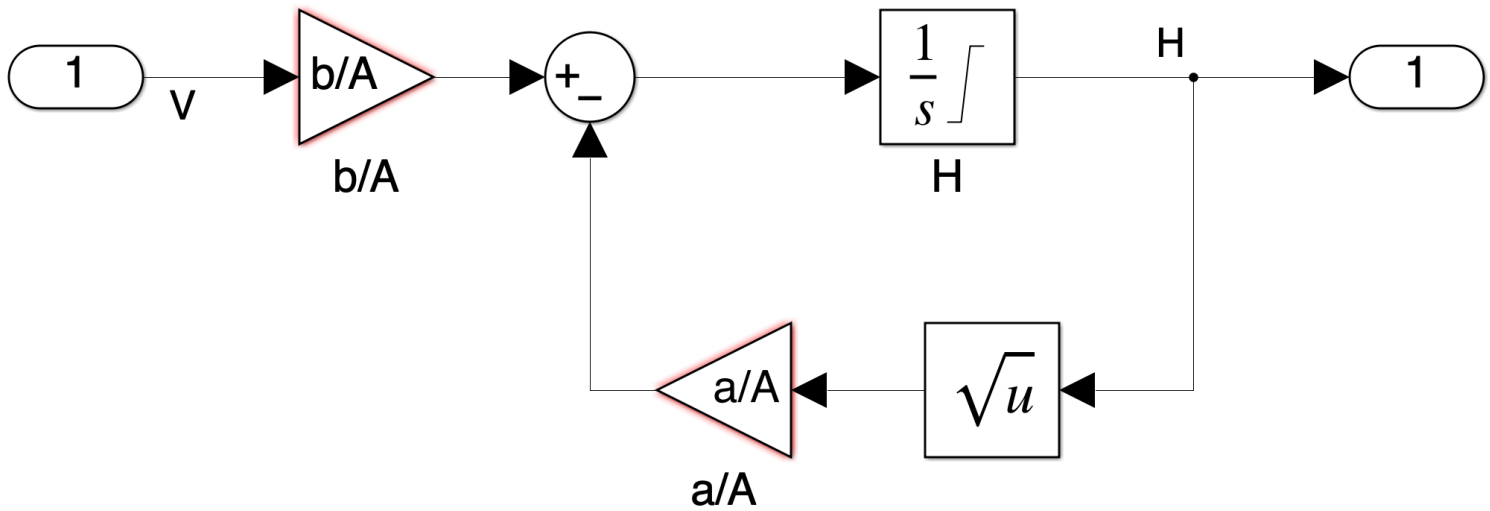


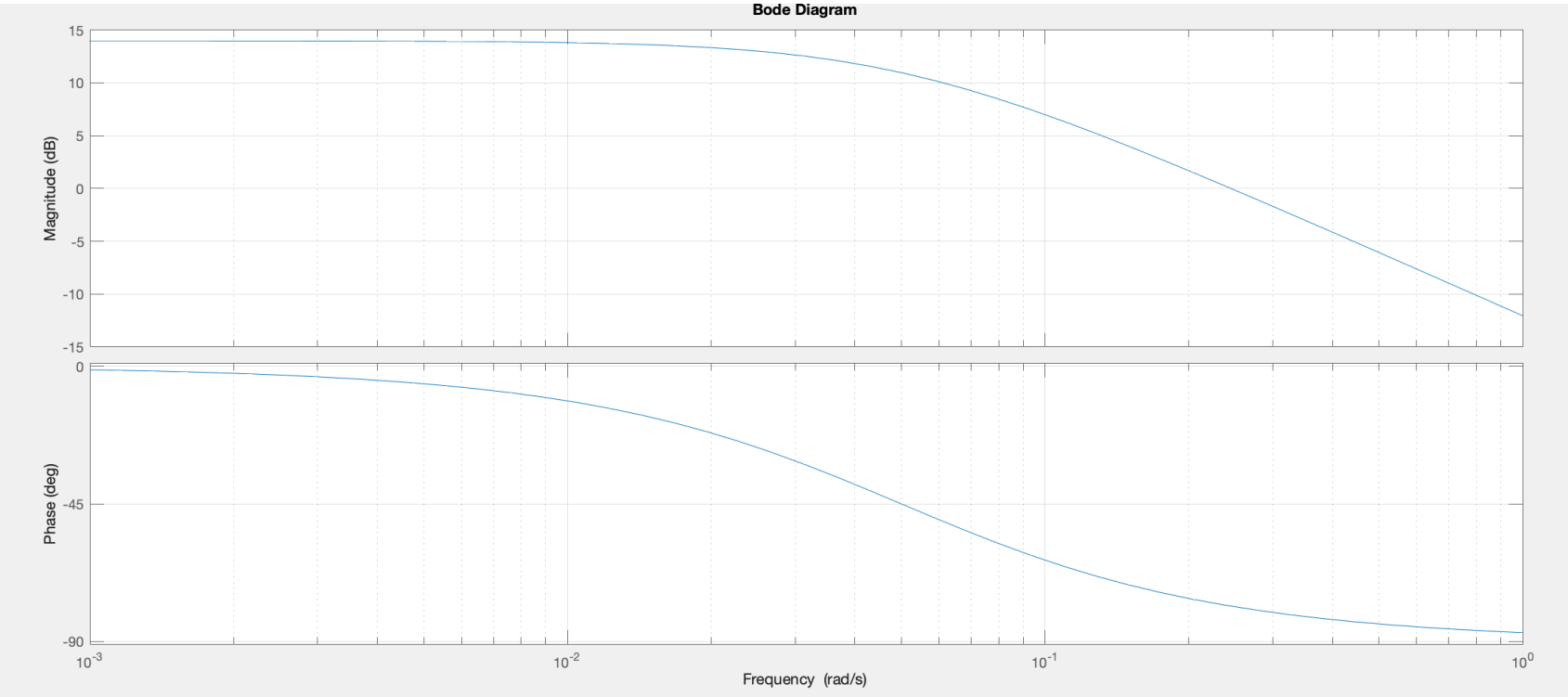
Figure 3: The watertank dynamic system in Matlab (built-in example up to version 2022).

Bode Plots in Simulink (cont'd)

From the Simulink Library Browser:

1. Choose Simulink Control Design
2. Choose Linear Analysis Plots
3. Choose and add a Bode Plot block
4. Open the Bode Plot block and in Linearisation add 2 signals:
   - → The output of the PID controller as input-perturbation
   - → The output of the plant as open-loop output
5. Run to display the Bode plot.

Bode Plots in Simulink (cont'd)



Implementing a Dynamic System and a Controller Programmatically (not Simulink)

Car cruise control can be found in most of the modern cars. The cruise control system keeps the car at a constant speed despite any external disturbances, such as the road surface and incline.

- The car's mass m is controlled by a force u. To simplify the situation, we assume that the force u that our controller applies is not affected by other parameters such as tires, etc.

The equation of the system is described by the following:

$$m\dot{v} + bv = u \tag{2}$$

where v is the speed of the car, u is the control action and b is the damping coefficient due to friction.

The Car Cruise Control System

$$m\dot{v} + bv = u \tag{3}$$

- $m = 1000, b = 50$.
- The desired (reference) speed is $v_{ref} = 10$.
- $t = 10.0$ is the total simulation time
- The parameters of the PID controller are:
  $K_p = 800, K_i = 0, K_d = 40$ (i.e. this is a PD controller).

Implement in Python a PID controller to bring the system to the desired speed.

The Car Cruise Control System (cont'd)

```
# Implements the dynamic system (plant) - system inputs are
# action_u and the current speed v.
# The function returns the output of the plant.
def plant(action_u, v):
    # equations of motion:
    # m \dot{v} + b v  = u
    # m=1000, b=50, u = 500
    m = 1000.0
    b = 50

    v_dot = (action_u - b*v)/m
    new_speed = v + v_dot*dt

    return new_speed
```

The Car Cruise Control System (cont'd)

```
v = 0  # current speed initialised to 0
previous_v = 0  # the speed at the previous time step
dt = 0.01  # time step for the simulation

start_time = 0.0
end_time = 10.0

current_time = start_time

v_ref = 10  # the desired speed

K_p = 800  # proportional gain
K_i = 41    # integral gain
K_d = 40   # derivative gain

previous_error = 0.0
integral = 0

# log values to file
f = open('myfile.txt', "w")
# write initial values to the file
f.write(f'{v}  {current_time}\n')
```

The Car Cruise Control System (cont'd)

```
    ''' Simulate the system operation from the beginning till
        the end of the simulation '''
    while current_time <= end_time:
        error = v_ref - v

        # I(integral) component of the PID controller
        integral = integral + error*dt

        # (D)erivative component of the PID controller
        deriv = (error - previous_error)/dt

        # the output (action) of the PID controller
        action = K_p*error + K_i*integral + K_d*deriv

        # remember the last error when the previous
        # action was applied to the plant
        previous_error = error

        # apply the new action to the plant to calculate the new (current) speed
        v = plant(action, v)

        print(f'Time: {current_time} Action: {action}, Speed={v}')
        f.write(f'{v}  {current_time}\n')

        # advance the time
        current_time += dt

        f.close()
```

Plotting the Response Program

File `cruisecontrol_plot.py`

```
import matplotlib.pyplot as plt

f = open('myfile.txt')
v = []
time =[]
for i in f:
    [y, t] = i.split()
    #print(y, t)
    v.append(float(y))
    time.append(float(t))

print(v)
plt.plot(time, v)
plt.axis([-1, 10, 0, 12])
plt.xlabel('Time t')
plt.ylabel('Speed v')
plt.show()
```

Running the Plot Program

