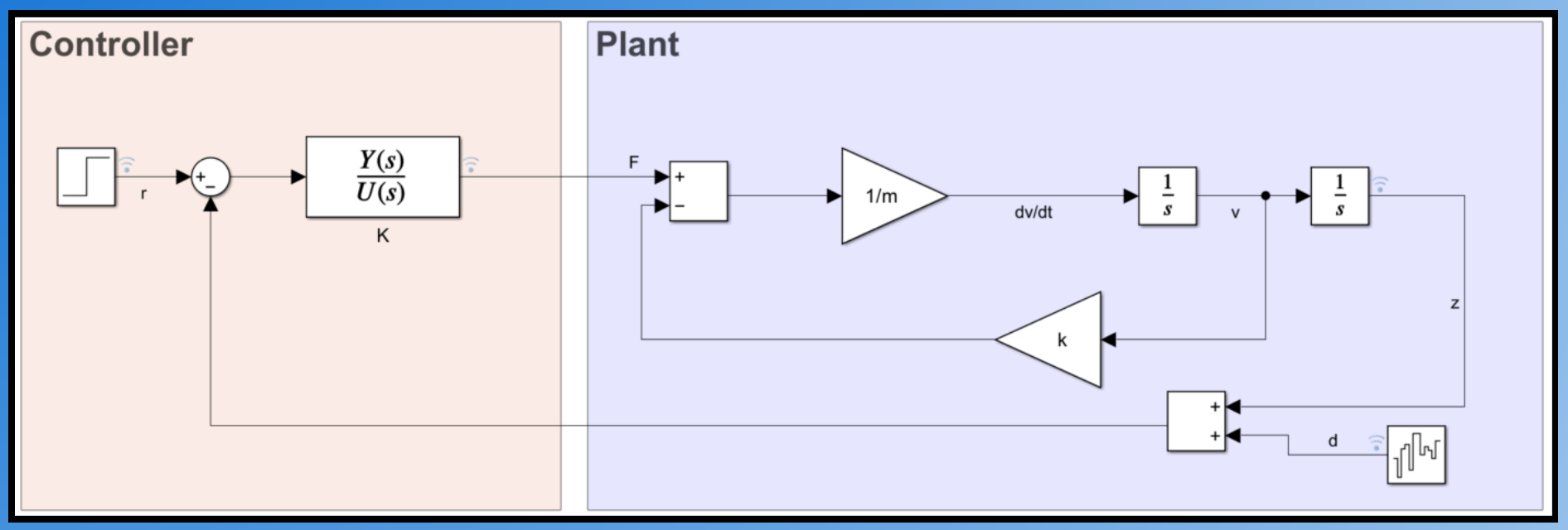
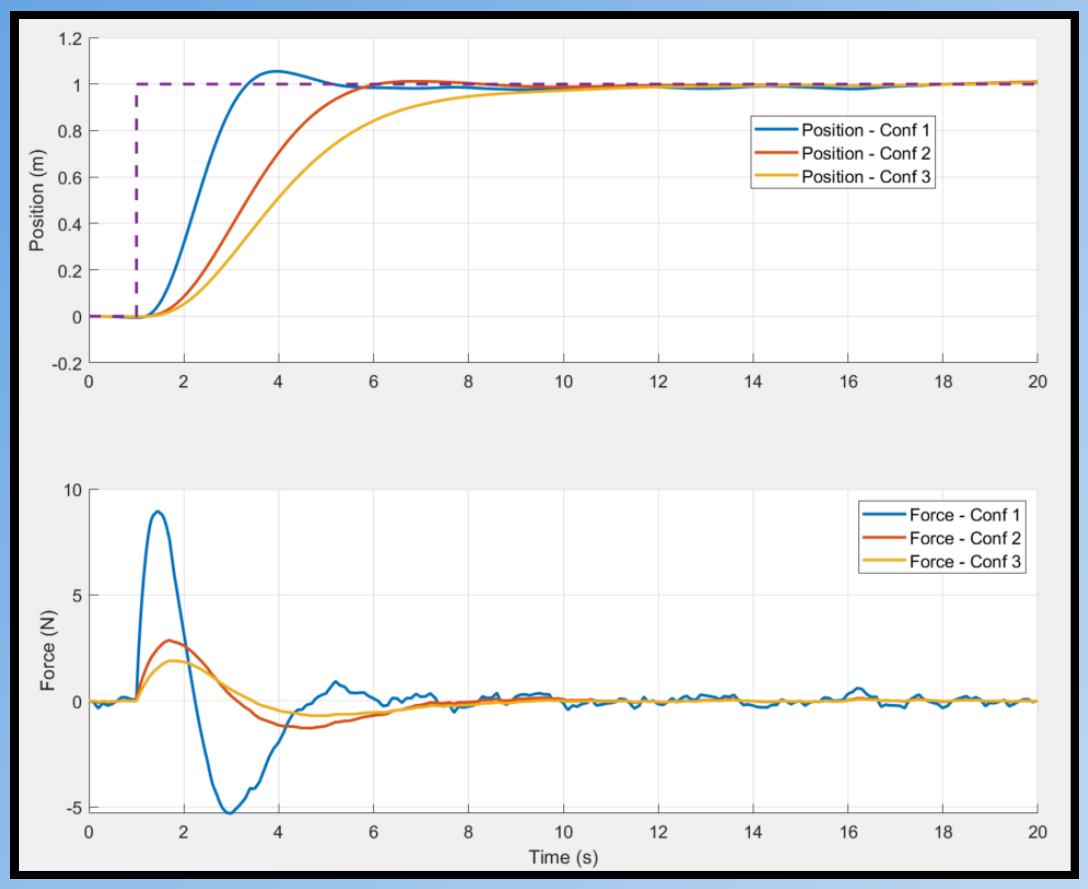
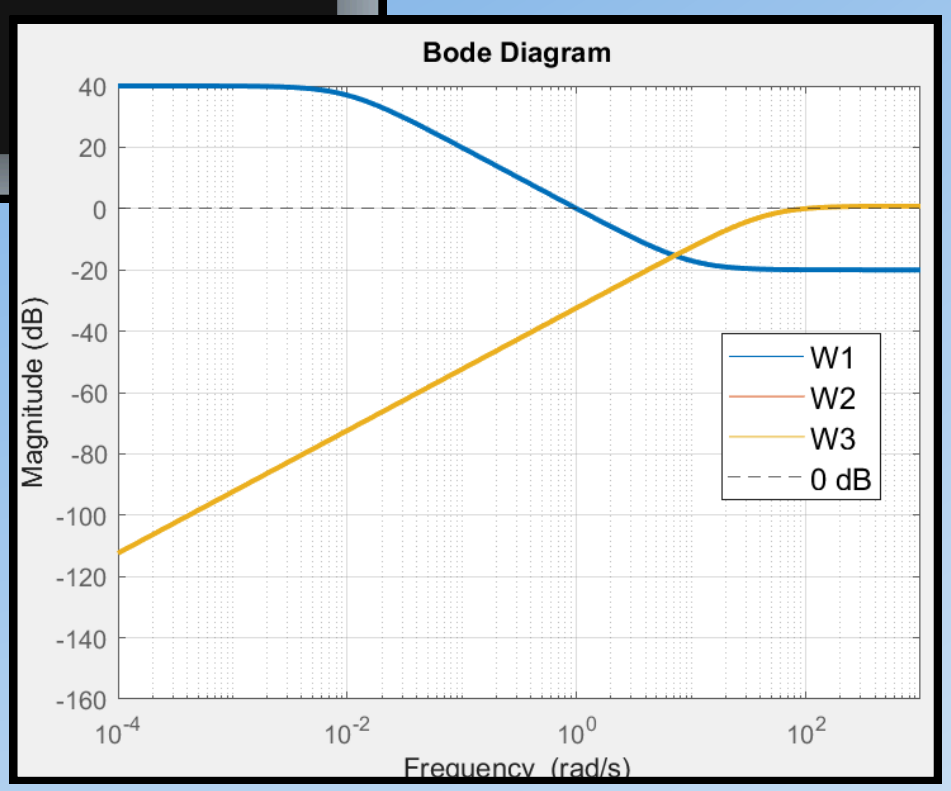
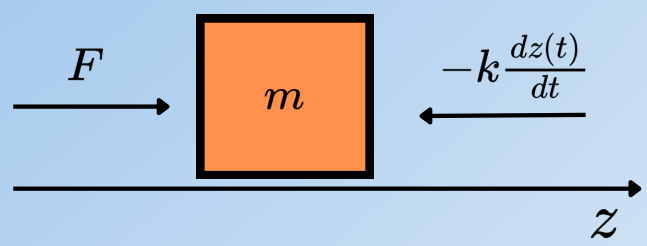
H ∞ Position Control




```m
%% Plant
% Create 's'
s = tf('s');

% System parameters
m = 10;
k = 0.5;

% System transfer function
G = 1/(s*(m*s+k));

%% Controller synthesis - multiple configurations

% Error weight function:
% High low frequency gain to achieve good rejection of constant
% dusturbance
% Define configurations
configs = {
    makeweight(100, 1, 0.1); % Configuration 1
    makeweight(100, 0.1, 0.1)   % Configuration 2
    makeweight(100, 0.05, 0.1)   % Configuration 2
};

% Control effort and controlled variable weights:
% Big high frequency gain to limit control effort at high frequency and
% achieve good robustness
W2 = makeweight(0, 100, 1.1);
W3 = makeweight(0, 100, 1.1);

% Display transfer functions:
W2_tf = tf(W2)
W3_tf = tf(W3)

% Preallocate arrays for storage
K = cell(size(configs));
CL = cell(size(configs));
gamma = cell(size(configs));
results = struct([]);
```





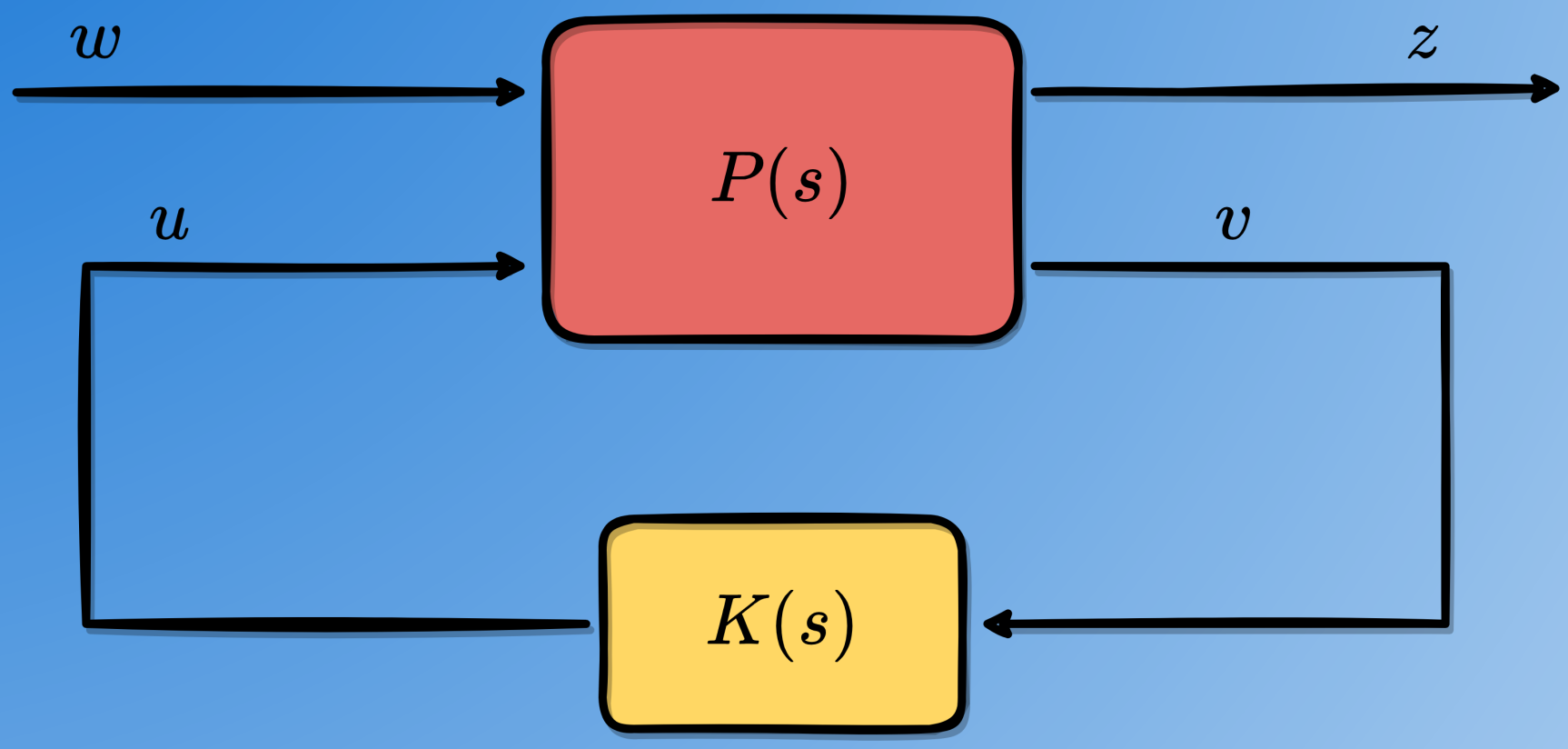


Model

https://github.com/simorxb/H-Infinity-Position-Control-Matlab

SIMONE BERTONI
CONTROL LAB

Control Problem Formulation

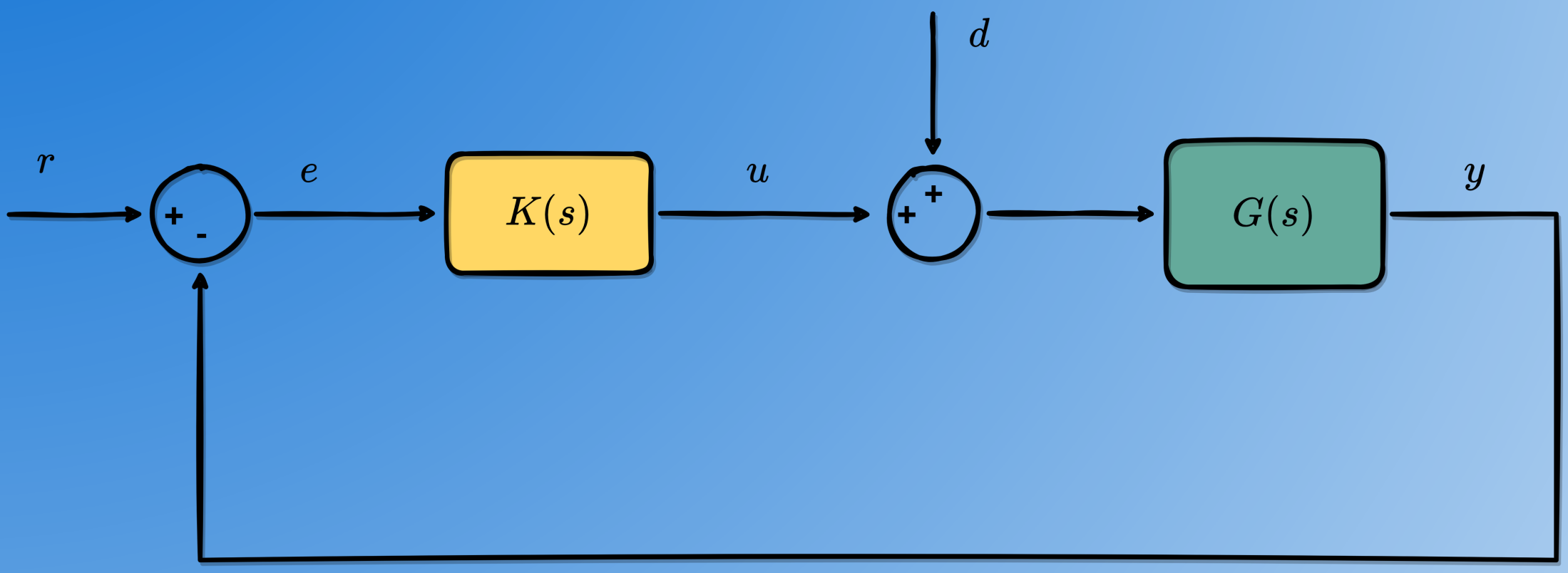


$$\begin{bmatrix} z \\ v \end{bmatrix} = P(s) \begin{bmatrix} w \\ u \end{bmatrix} = \begin{bmatrix} P_{11}(s) & P_{12}(s) \\ P_{21}(s) & P_{22}(s) \end{bmatrix} \begin{bmatrix} w \\ u \end{bmatrix}$$

w: Exogenous input (references signal and disturbances)
u: Control input
z: Error signals (that should be minimised)
v: Measured variables (used to control the system)

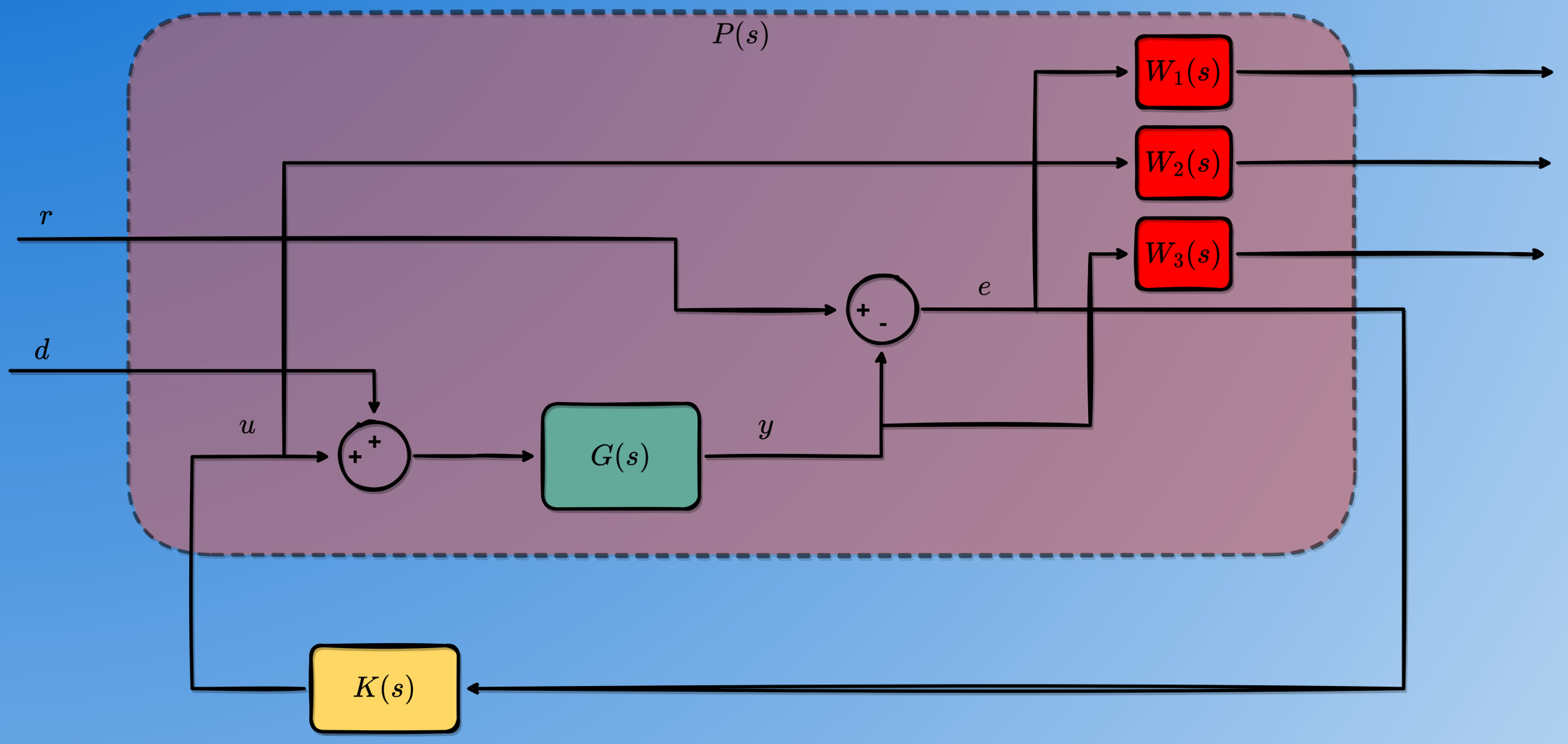
The objective of H_∞ control design is to find a controller $K(s)$ that minimises the H_∞ norm of the close loop system between w and z.

Standard Feedback Control Architecture



This is a typical way to represent a feedback control system. How can we express it in a framework that allows us to find $K(s)$ using the H_∞ optimisation?

H ∞ Optimisation Framework



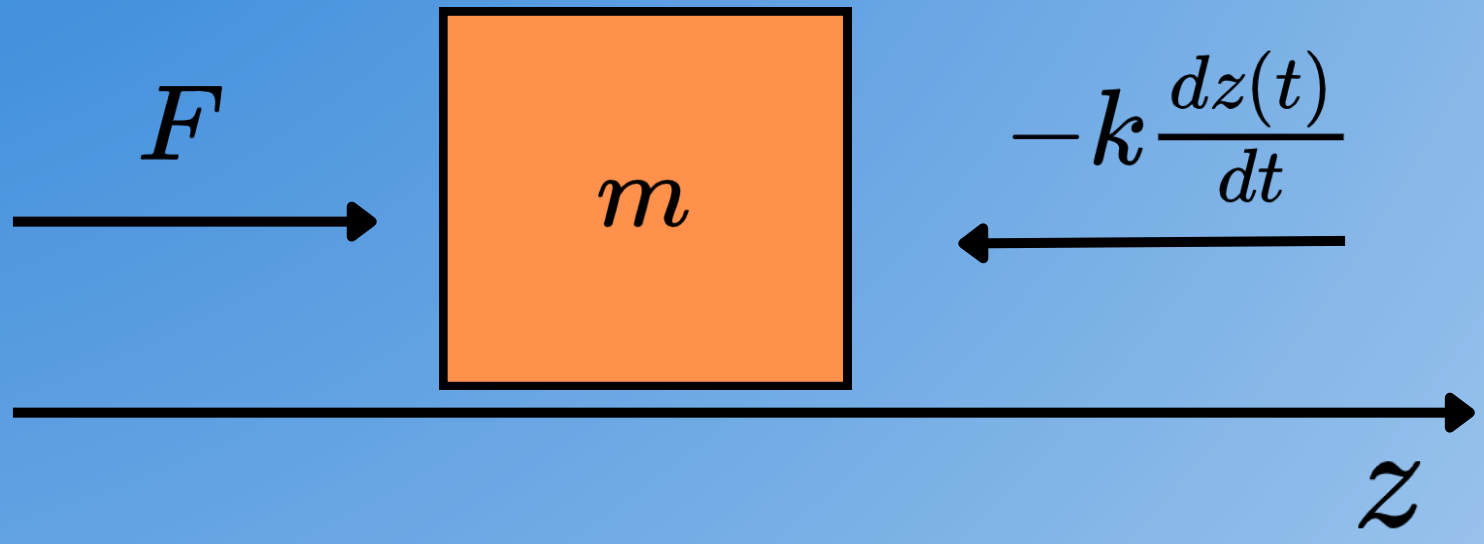
$$w = \begin{bmatrix} r \\ d \end{bmatrix} \qquad z = \begin{bmatrix} W_1(s)e \\ W_2(s)u \\ W_3(s)y \end{bmatrix}$$

$W_1(s)$, $W_2(s)$ and $W_3(s)$ are the optimisation weight functions.

$$v = e$$
$$e = r - y$$

$$P(s) = \begin{bmatrix} W_1(s) & -W_1(s)G(s) & -W_1(s)G(s) \\ 0 & 0 & W_2(s) \\ 0 & W_3(s)G(s) & W_3(s)G(s) \\ 1 & -G(s) & -G(s) \end{bmatrix}$$

Plant



$$m\frac{d^2z(t)}{dt^2} = F - k\frac{dz(t)}{dt}$$

$$m = 10kg$$
$$k = 0.5\frac{Ns}{m}$$

Optimisation – 1

The optimisation is done using the mixsyn function in Matlab. The weight functions chosen are defined using the function makeweight where you can define low frequency gain, crossover frequency, and high frequency gain, in this order.

The idea is to have $W_1(s)$ dominant in the low frequency band (at steady state we want to minimise the control error) and have $W_2(s)$ and $W_3(s)$ dominant at high frequency, where we want to minimise chattering and have measurement disturbance rejection.

Optimisation – 2

To show how the result changes depending on the weights, three different configurations have been used, where $W_2(s)$ and $W_3(s)$ remain the same, and $W_1(s)$ varies:

Configuration 1

$$W_1(s) = \frac{0.1s+0.995}{s+0.00995}$$

$$W_2(s) = \frac{1.1s}{s+45.83}$$

$$W_3(s) = \frac{1.1s}{s+45.83}$$

Configuration 2

$$W_1(s) = \frac{0.1s+0.0995}{s+0.000995}$$

$$W_2(s) = \frac{1.1s}{s+45.83}$$

$$W_3(s) = \frac{1.1s}{s+45.83}$$

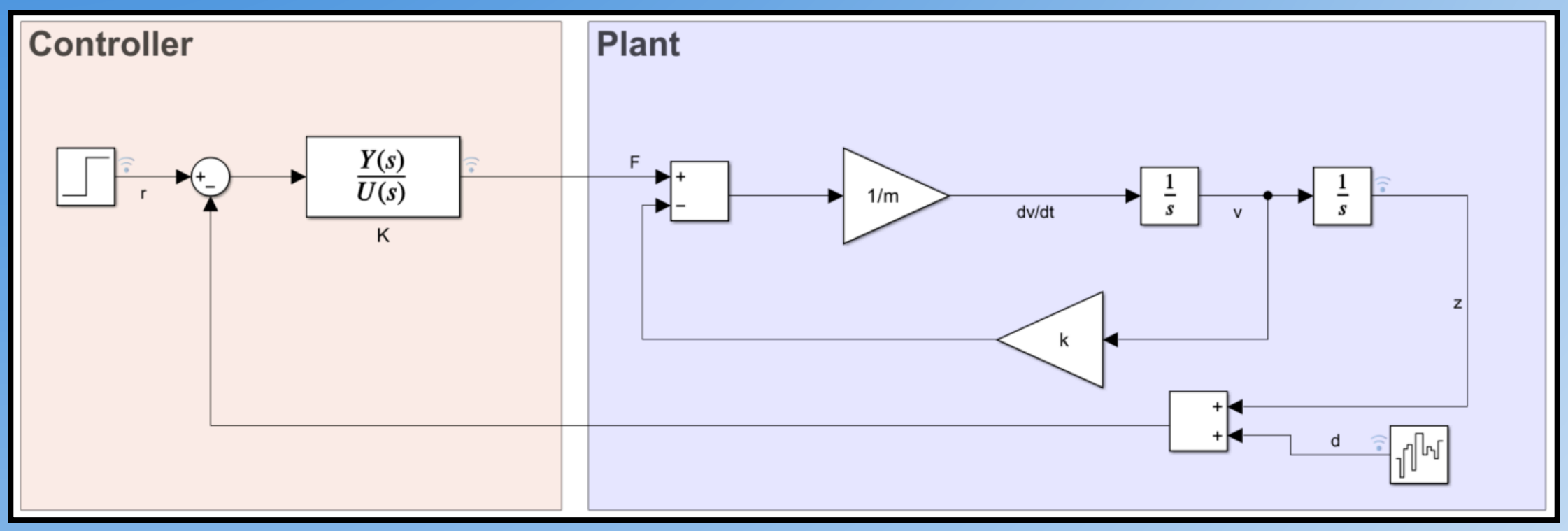
Configuration 3

$$W_1(s) = \frac{0.1s+0.04975}{s+0.0004975}$$

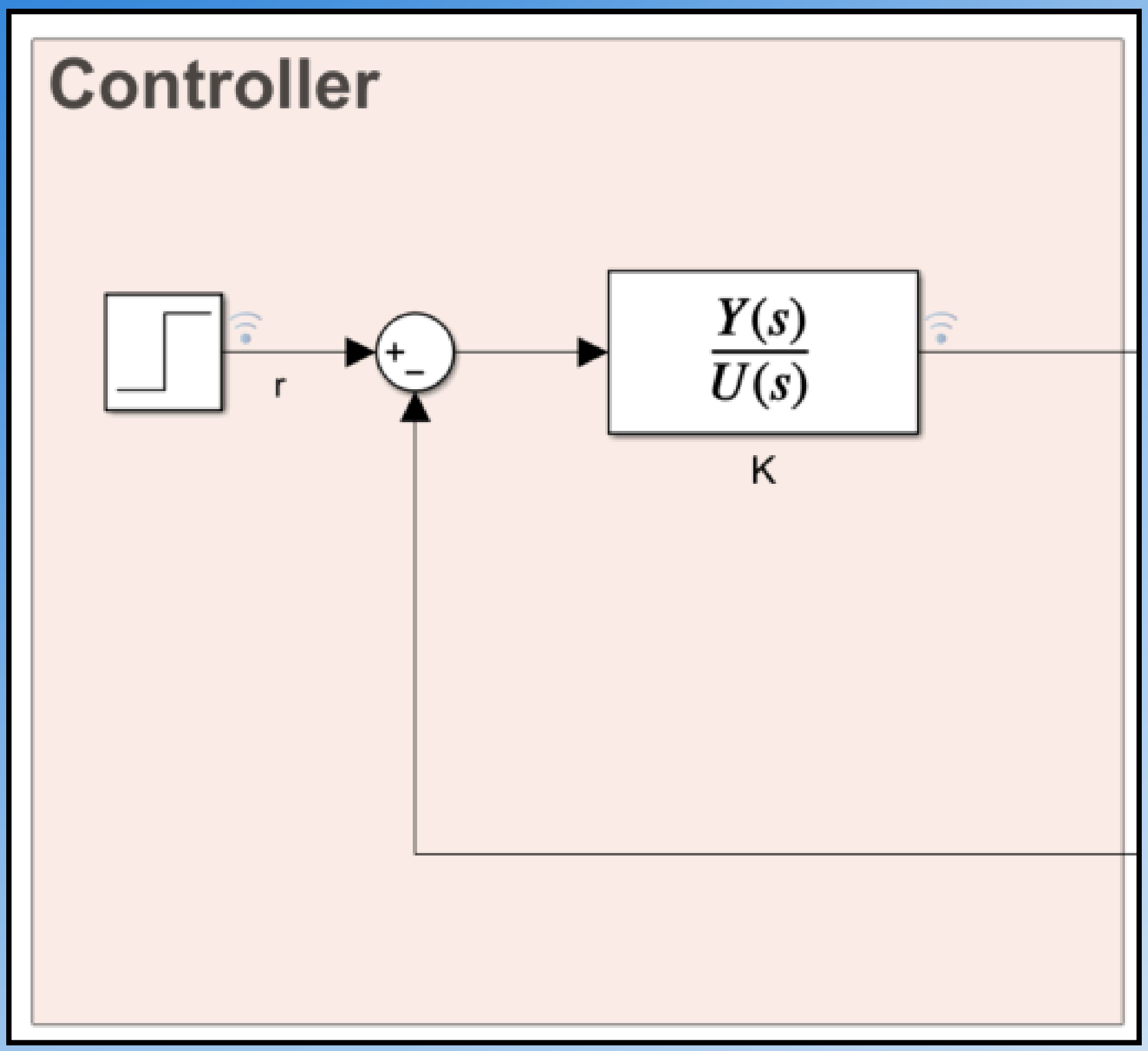
$$W_2(s) = \frac{1.1s}{s+45.83}$$

$$W_3(s) = \frac{1.1s}{s+45.83}$$

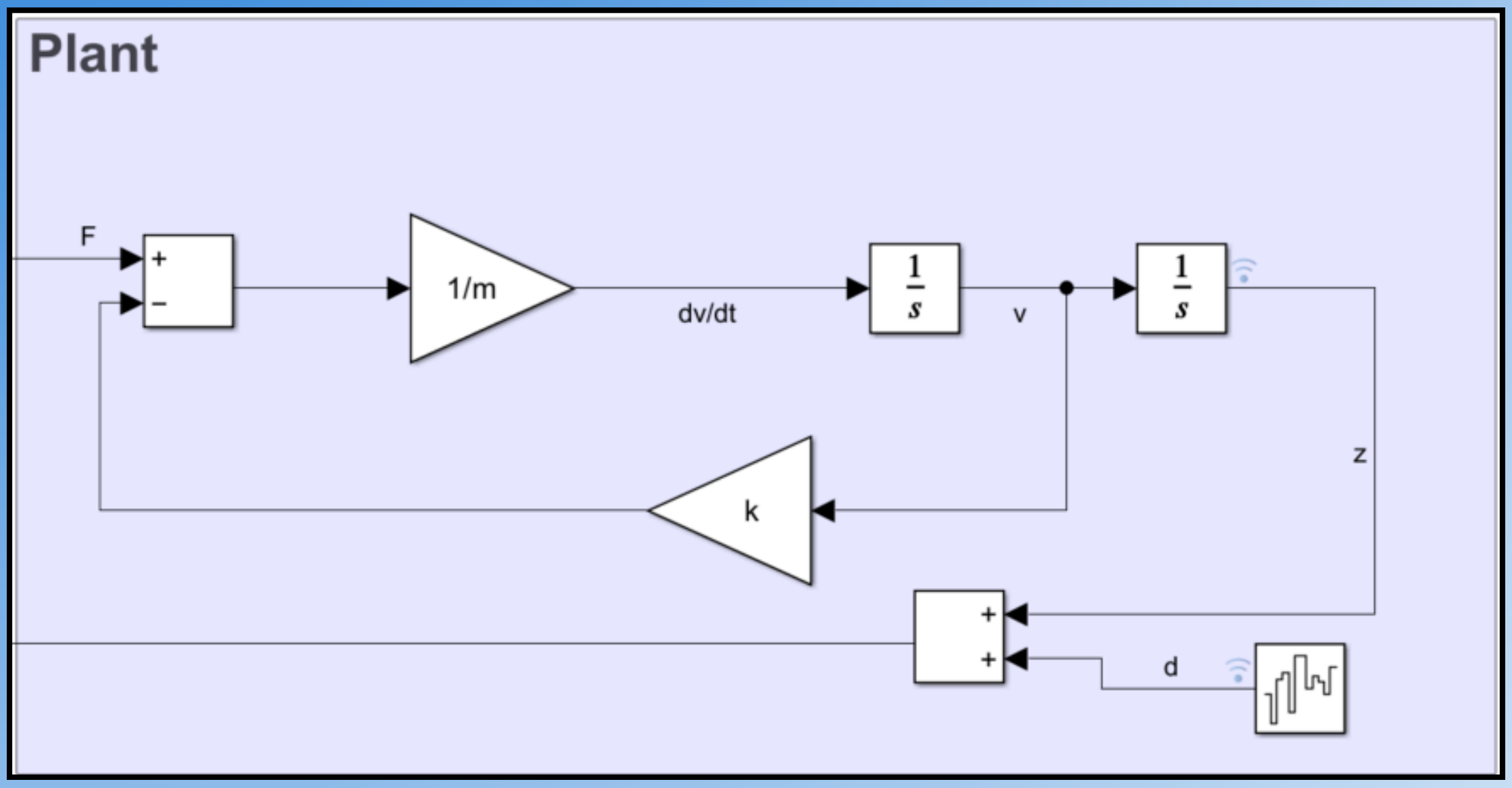
Simulink Model



Simulink Model – Controller



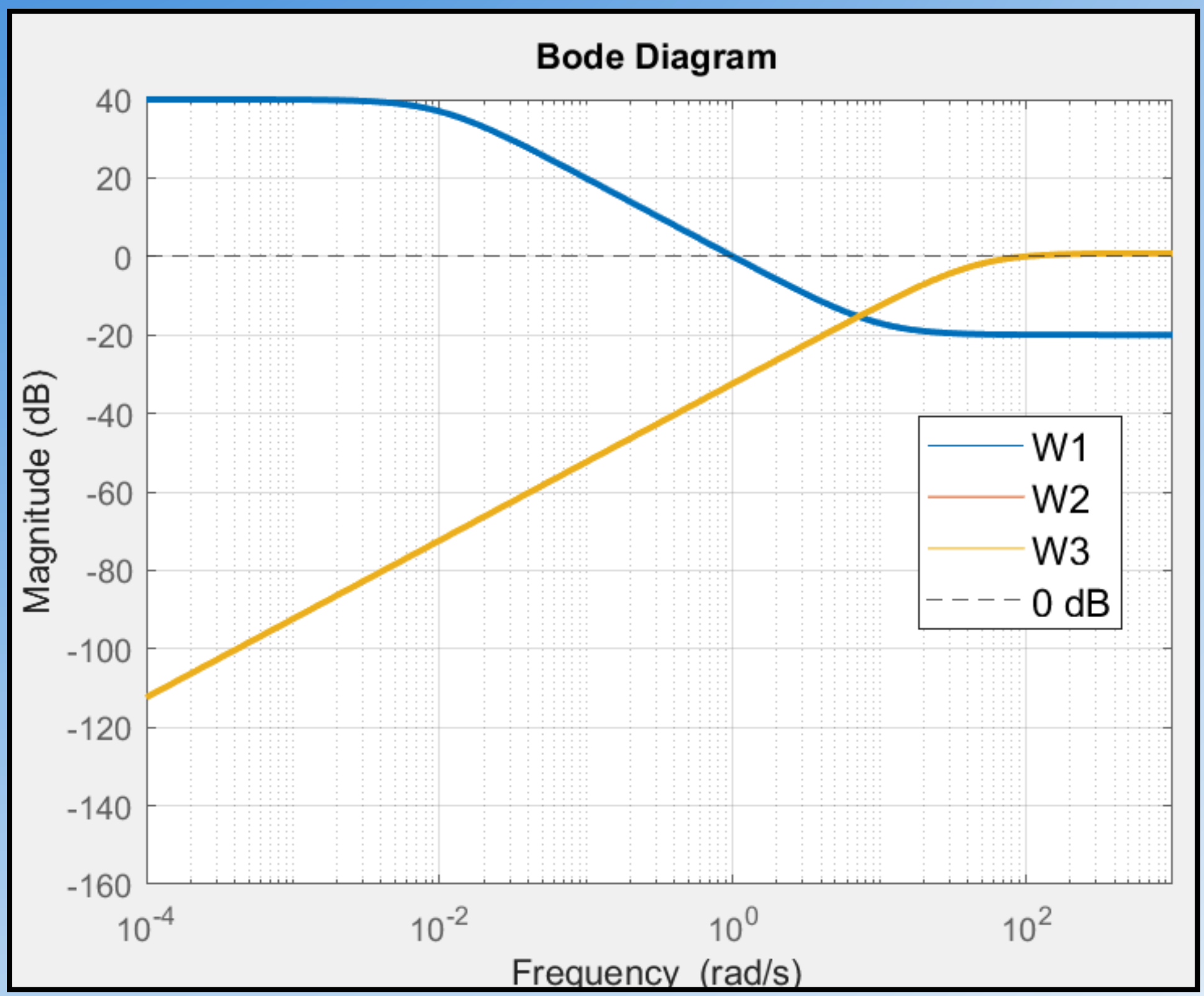
Simulink Model – Plant



Configuration 1 – Analysis and Results

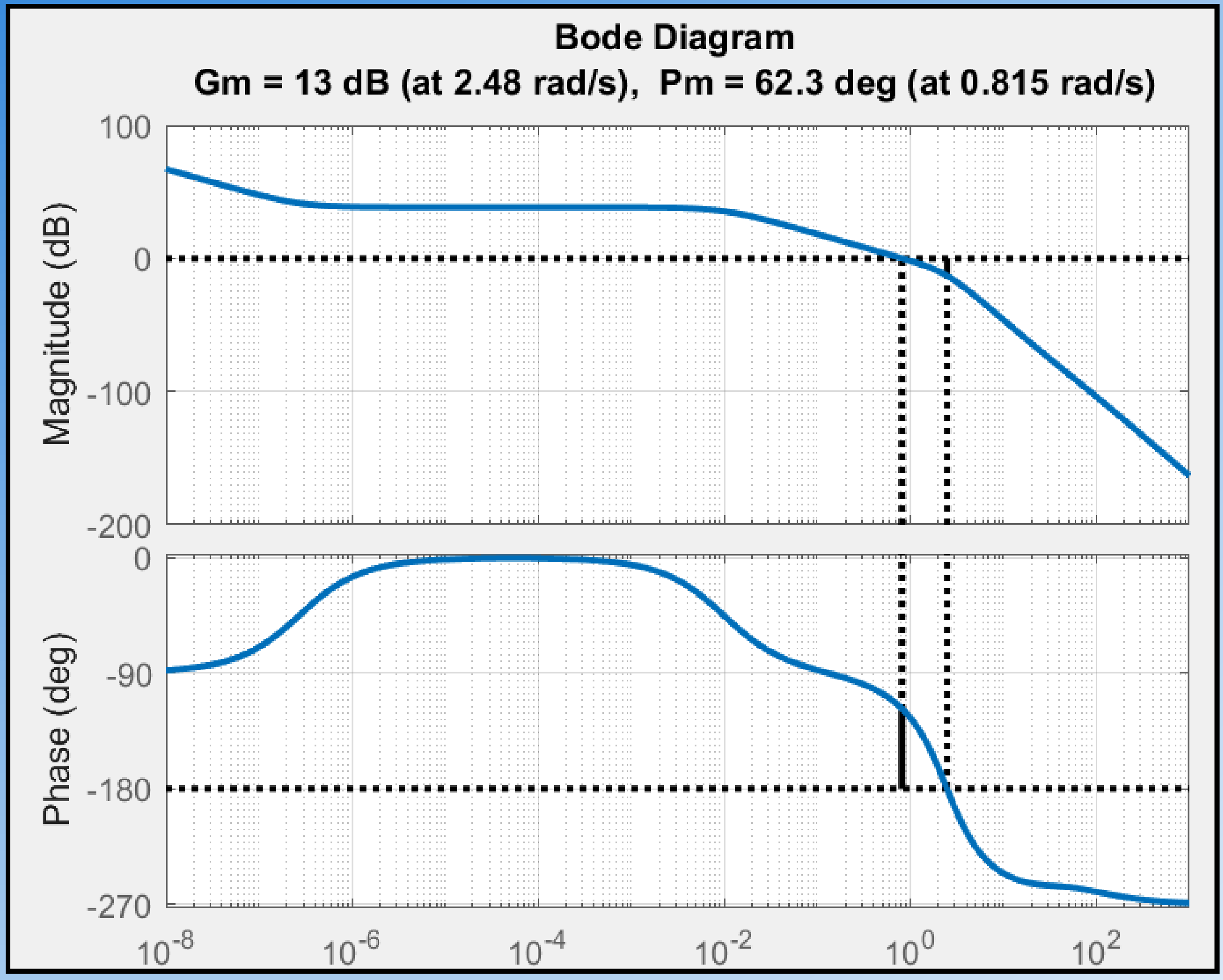
$$K(s) = \frac{68.74s^4+6303s^3+1.447e05s^2+7217s+0.001979}{s^5+113.1s^4+3319s^3+1.111e04s^2+1.759e04s+173.9}$$

Bode diagram for weight functions



Configuration 1 – Analysis and Results

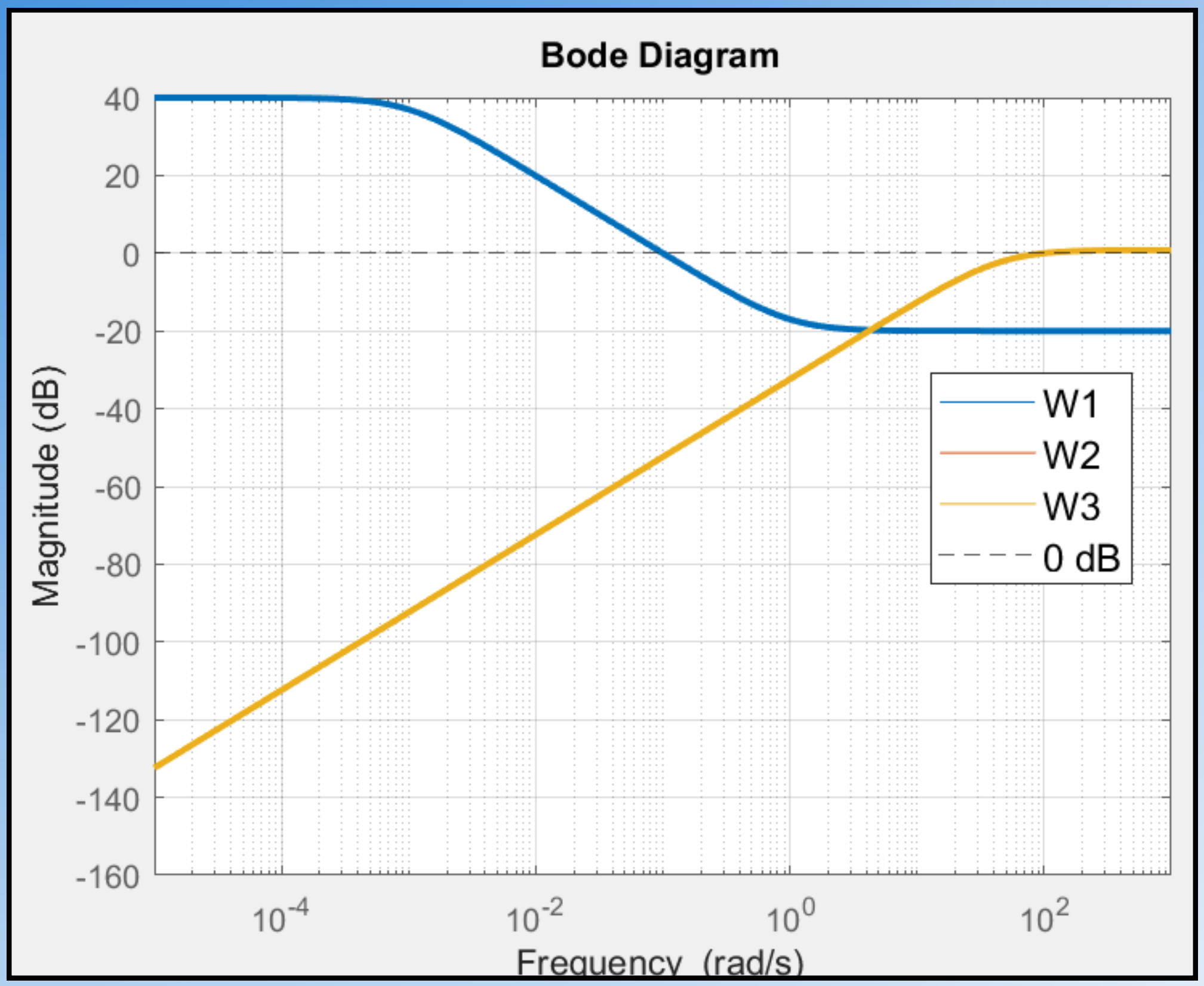
Bode diagram for K(s)G(s)



Configuration 2 – Analysis and Results

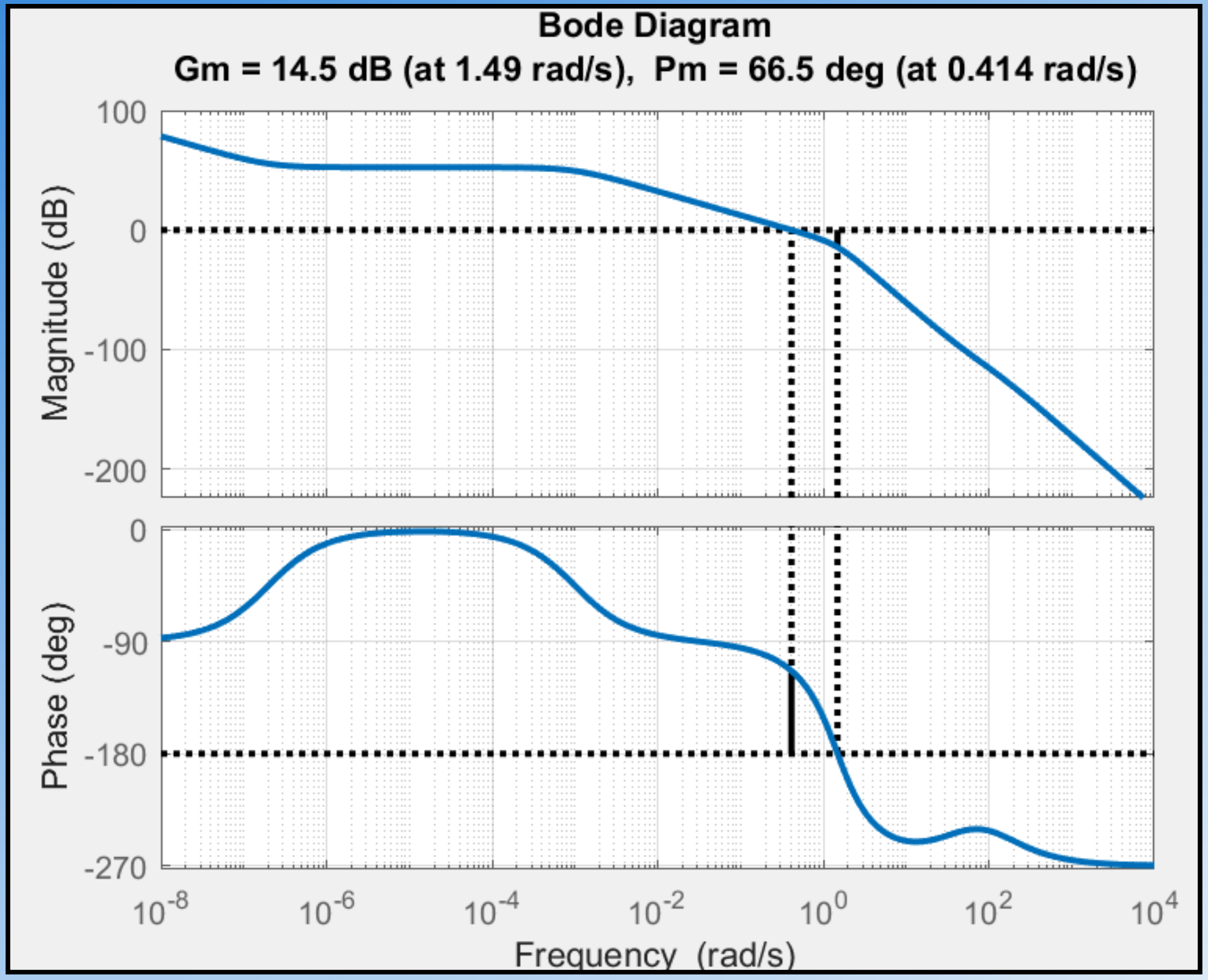
$$K(s) = \frac{24.88s^4+2281s^3+5.236e04s^2+2612s+0.0005227}{s^5+174.5s^4+6171s^3+1.278e04s^2+1.255e04s+12.47}$$

Bode diagram for weight functions



Configuration 2 – Analysis and Results

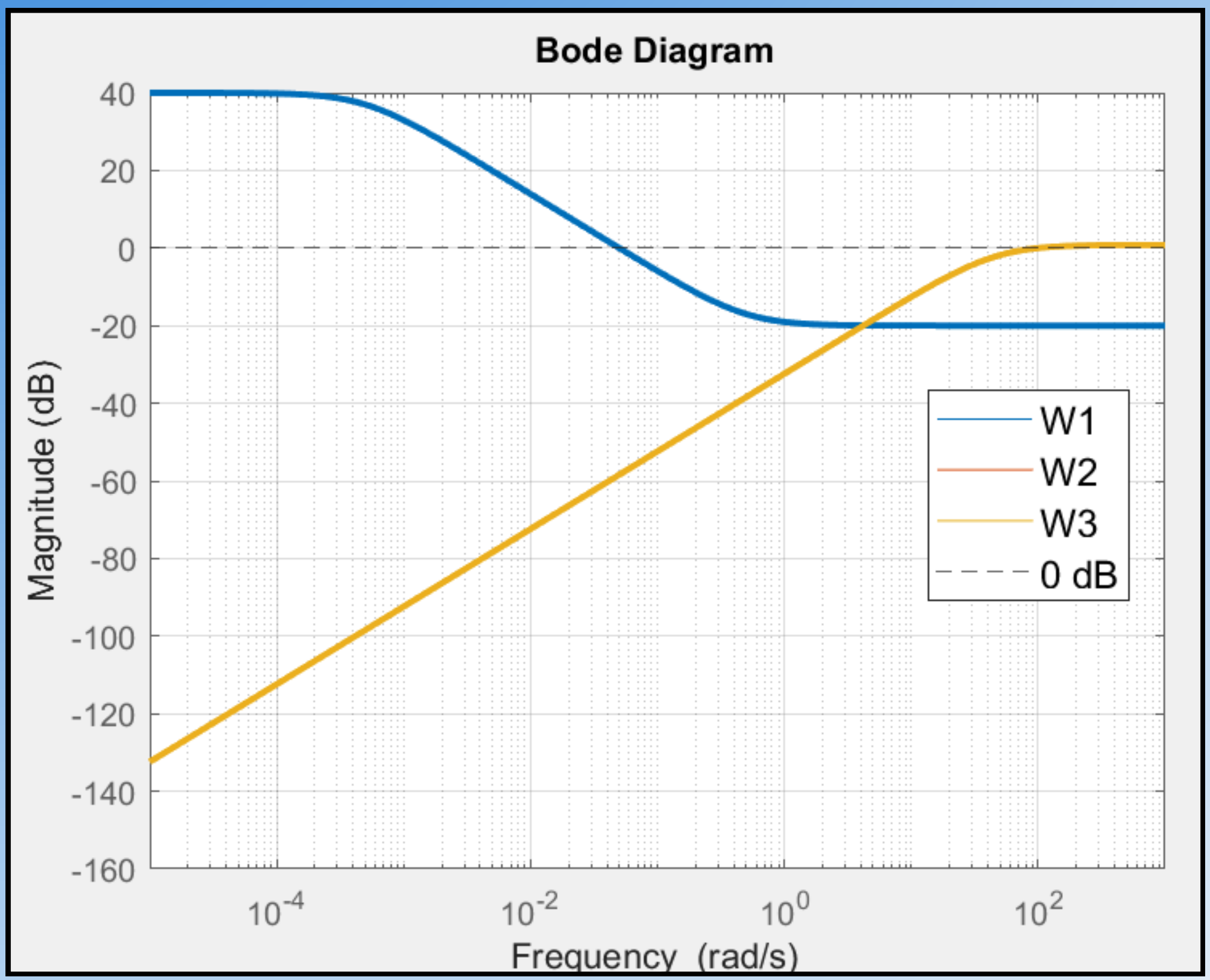
Bode diagram for K(s)G(s)



Configuration 3 – Analysis and Results

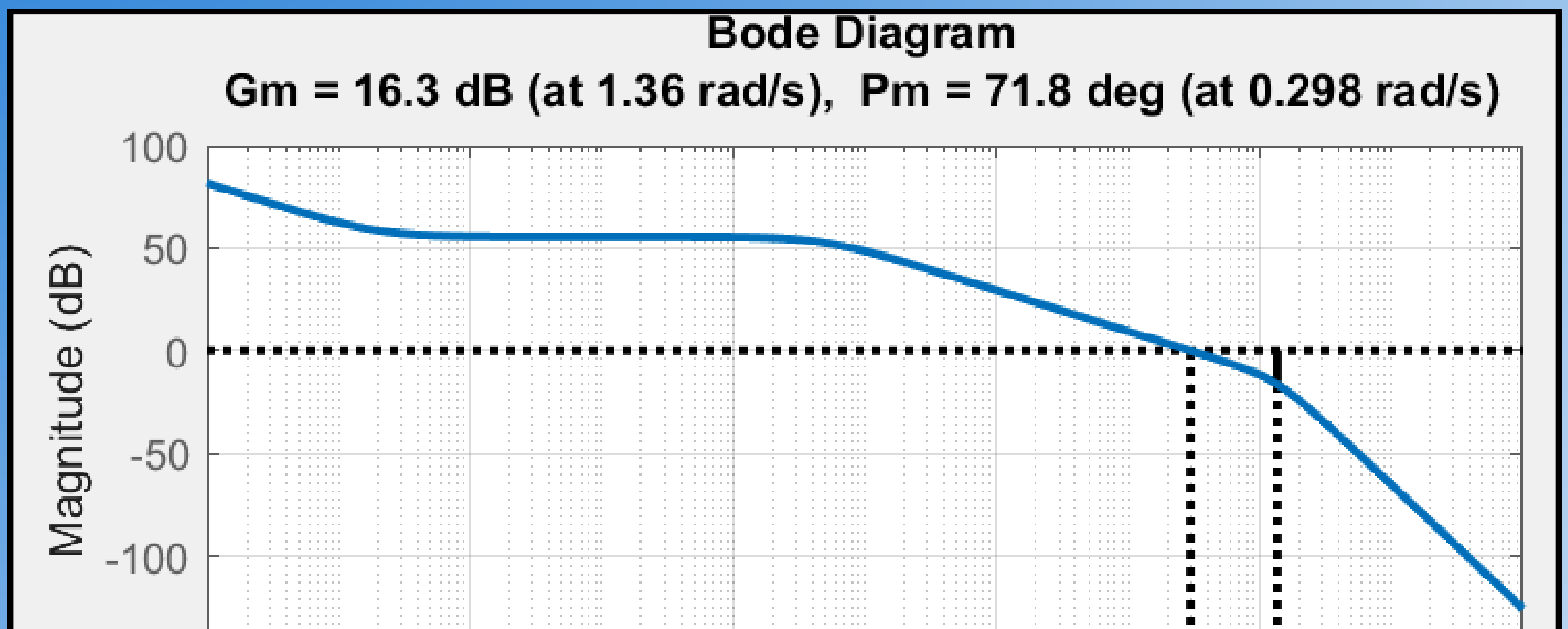
$$K(s) = \frac{5.16s^4+473.2s^3+1.086e04s^2+541.8s+0.0001083}{5.16s^4+473.2s^3+1.086e04s^2+541.8s+0.0001083}$$

Bode diagram for weight functions



Configuration 3 – Analysis and Results

Bode diagram for K(s)G(s)



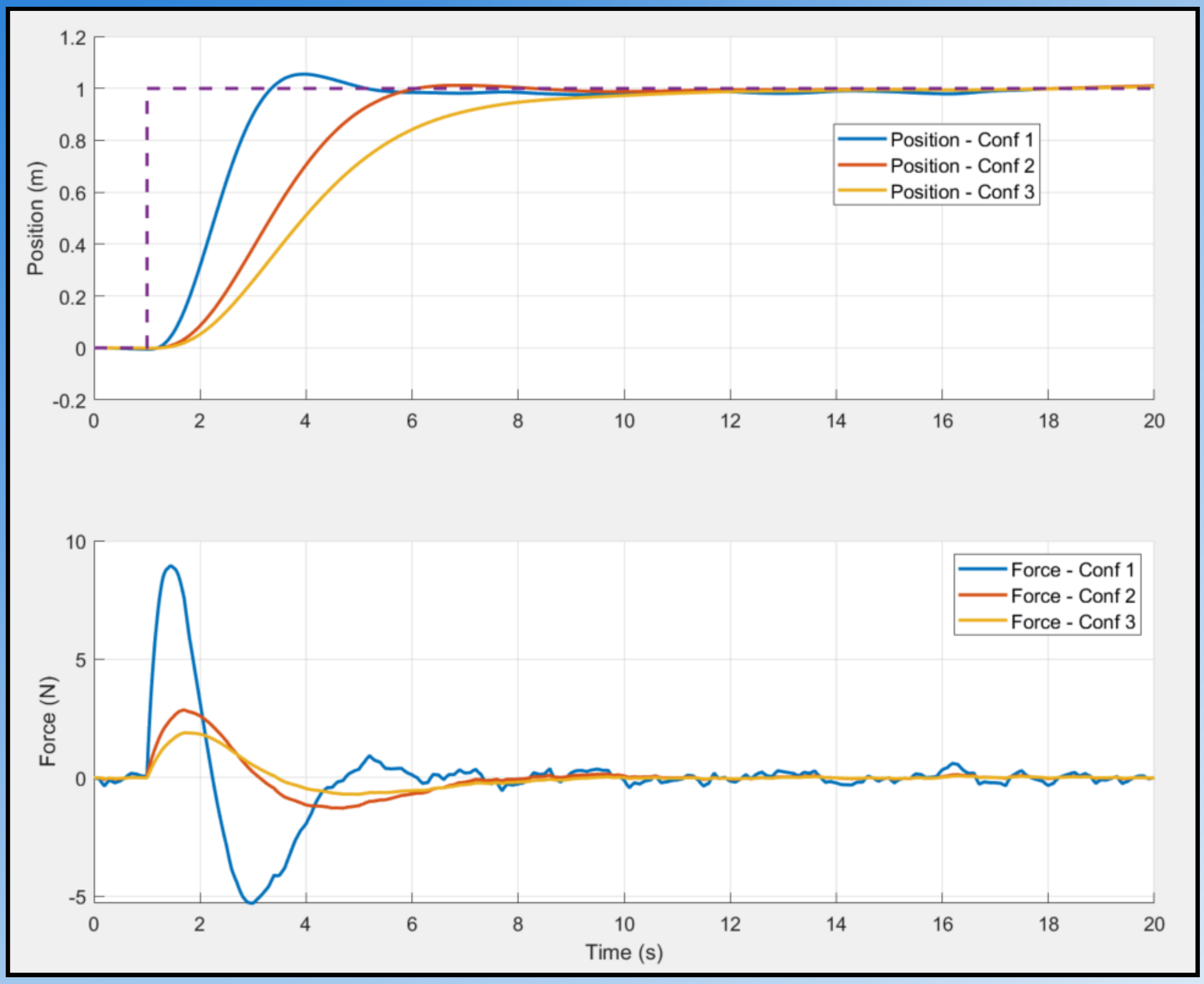
Matlab Code – 1

```matlab
%% Plant
% Create 's'
s = tf('s');

% System parameters
m = 10;
k = 0.5;

% System transfer function
G = 1/(s*(m*s+k));

%% Controller synthesis - multiple configurations

% Error weight function:
% High low frequency gain to achieve good rejection of constant
% dusturbance
% Define configurations
configs = {
    makeweight(100, 1, 0.1); % Configuration 1
    makeweight(100, 0.1, 0.1)   % Configuration 2
    makeweight(100, 0.05, 0.1)   % Configuration 2
};

% Control effort and controlled variable weights:
% Big high frequency gain to limit control effort at high frequency and
% achieve good robustness
W2 = makeweight(0, 100, 1.1);
W3 = makeweight(0, 100, 1.1);

% Display transfer functions:
W2_tf = tf(W2)
W3_tf = tf(W3)

% Preallocate arrays for storage
K = cell(size(configs));
CL = cell(size(configs));
gamma = cell(size(configs));
results = struct([]);
```

Matlab Code – 2

```matlab
% Loop through configurations
for i = 1:length(configs)
    W1 = configs{i}; % Use the ith configuration
    W1_tf = tf(W1)

    % Plot Bode diagram for weights
    figure;
    bodemag(W1, W2, W3);
    set(findall(gcf,'type','line'),'LineWidth',2);
    yline(0,'--');
    legend({'W1', 'W2', 'W3', '0 dB'}, 'FontSize', 12);
    grid on;

    % Compute K using mixed-sensitivity loop shaping
    [K{i}, CL{i}, gamma{i}] = mixsyn(G, W1, W2, W3);

    % Find K's transfer function
    K_tf = tf(K{i})

    % Open loop function's Bode diagram to check stability margins
    figure;
    margin(K_tf*G); % Plot Bode diagram with gain and phase margins
    set(findall(gcf,'type','line'),'LineWidth',2);
    grid on;

    % Run the model with the ith configuration
    simOut = sim("H_inf_pos_control");

    % Access the signals from out.logsout
    results(i).r = simOut.logsout.get('r').Values;
    results(i).F = simOut.logsout.get('F').Values;
    results(i).z = simOut.logsout.get('z').Values;

end

%% Plotting Results
figure;
% Subplot for response
subplot(2, 1, 1);
hold on;
for i = 1:length(configs)
    plot(results(i).z.Time, results(i).z.Data, 'LineWidth', 2);
end
plot(results(1).r.Time, results(1).r.Data, '--', 'LineWidth', 2);
hold off;
ylabel('Position (m)');
legend(arrayfun(@(x) sprintf('Position - Conf %d', x), 1:length(configs), 'UniformOutput', false), 'FontSize', 12);
set(gca, 'FontSize', 12);
grid on;

% Subplot for control effort
subplot(2, 1, 2);
hold on;
for i = 1:length(configs)
    plot(results(i).F.Time, results(i).F.Data, 'LineWidth', 2);
end
hold off;
ylabel('Force (N)');
legend(arrayfun(@(x) sprintf('Force - Conf %d', x), 1:length(configs), 'UniformOutput', false), 'FontSize', 12);
set(gca, 'FontSize', 12);
grid on;
```

Result



PID Controller Course

https://simonebertonilab.com

Understand the control theory

★★★★★ *April 28, 2024*

I think the most important thing is to understand the meaning behind the mathematical formula. I guess this is the mission of Simone in this course and from my point of view he fully achivied this target. I hope to see in the future other courses (e.g advanced controls) structered in the same way with the same passion and examples.

Thank you Simone. Show less

Emidio ✔ Verified

★★★★★

Very helpful and practical

Yoav Golan

I enjoyed this course very much. I learned a lot of practical knowledge in a short time. Simone is very clear and teaches well, thank you! In the future, I would be very interested if Simone added a course with more subjects, such as cascading controllers, rate limiting, and how the controllers look in actual code. Thanks again!

★★★★★

Intuitive and Practical

Ranya Badawi

Simone's explanation of PID control was very intuitive. This is a great starter course to gain a fundamental understanding and some practical knowledge of PID controllers. I highly recommend it. For future topics, I'd be interested in frequency response, transfer functions, Bode plots (including phase/gain margin), Nyquist plots, and stability.

★★★★★

Very good sharing of experience

Romy Domingo Bompart Ballache

I have background in control system for power electronics, I see every lesson very useful.

Great course

★★★★★ *April 15, 2024*

Right to the point, easy to follow and very practical. I missed the zero/pole placement and phase margin analysis. It would also be interesting if you could provide other plants examples. Anyway, a great course to help designing and tuning a PID controller.

Leonardo Starling ✔ Verified

A different way to learn PID !

★★★★★ *May 31, 2023*

The teacher explains PID in a clear way adding his experience there where formulas alone cannot do much. Furthermore, each topic covered is included in a practical example to better fix ideas.

Michele De Palma