```
clear; close all; clc;
```

Assignment 1 - Piezoelectric pressure sensor

Table of Contents

Numerical data:

```
pmax = 1e3;         % [Pa] rated max pressure
l = 20e-3;          % [m] length
w = 20e-3;          % [m] width
h = 0.15e-3;        % [m] thickness of each layer
rho = 7600;         % [kg/m^3] beam density
Rf = 200e6;         % [Ohm] OpAmp param
Cf = 4e-9;          % [F] OpAmp param
kt = 1;             %[Nm]
sE11 = 16.4e-12;    % [m^2/N]
% sE33 = 18.8e-12; % [m^2/N]
d13 = -171e-12;     % [C/N]
csi = 0.2;          % [-]
```

Useful quantities:

```
Yp = 1/sE11;              % Young modulus
Mp = w*Yp*d13*(2*h)^2/4; % Moment Electric Field unit
J = 1/12*w*(2*h)^3;       % Moment of inertia
```

Static Analysis

1.1) Expression for deformed shape of the beam

The model can assumed to be a doubly pinned beam subjected to a distributed load due to the external pressure and two lumped moment due to the torsional springs.

$M_y(x)$ is the internal moment along the beam due to:

- torsional springs (lumped loads)
- external applied pressure (distributed load)

$$M_y(x) = -M_0 + \frac{L\,W\,p}{2}\,x - \frac{W\,p}{2}\,x^2$$

(from now on assume the torsional springs moment M_0 known)

Use the generic solution of the dispacement of the beam under the effects of the internal moment

$$u_z(x) = -\frac{1}{\mathrm{YJ}}\int_0^x \int_0^\zeta M_y(\xi)\,d\xi\,d\zeta - \frac{1}{2}\frac{M_p}{\mathrm{YJ}}\,x^2 E_3 + C_1 x + C_2$$

```
syms x p W YJ M_p L E_3 C1 C2 V_0 h_p M_0 k_t
assume(L>0)
assume(W>0)

My = -(p*W)/2 * x^2 + p*W*L/2 * x - M_0
```

My =

$$-\frac{W\,p\,x^2}{2} + \frac{L\,W\,p\,x}{2} - M_0$$

```
uz_x_C = -int(int(My,x),x)/YJ -1/2*M_p/YJ*E_3*x^2 +C1*x + C2
```

uz_x_C =

$$C_2 + C_1\,x + \frac{\frac{W\,p\,x^4}{24} - \frac{L\,W\,p\,x^3}{12} + \frac{M_0\,x^2}{2}}{\mathrm{YJ}} - \frac{E_3\,M_p\,x^2}{2\,\mathrm{YJ}}$$

Find the constants C_1 and C_2 with the boundary conditions:

- $u_z(x=0) = 0$
- $u_z(x=L) = 0$

```
C = solve([subs(uz_x_C,x,0); subs(uz_x_C,x,L)],[C1 C2]);
uz_x = simplify(subs(uz_x_C,{C1 C2},{C.C1 C.C2}))
```

uz_x =

$$\frac{x\,(L-x)\,\left(W\,p\,L^2 + W\,p\,L\,x - W\,p\,x^2 - 12\,M_0 + 12\,E_3\,M_p\right)}{24\,\mathrm{YJ}}$$

Now it is possible to find the value of torsional springs moemnt M_0, assuming short circuit conditions ($E_3 = 0$)

$M_0 = k_t * \frac{\mathrm{d}}{\mathrm{d}x} u_z(x)$ --> evaluated at spring position.

```
eqa = subs(k_t * diff(uz_x, x) - M_0, [x, M_0, E_3],[0, M_0, 0])
```

eqa =

$$-M_0 - \frac{L\,k_t\,\left(12\,M_0 - L^2\,W\,p\right)}{24\,\mathrm{YJ}}$$

```
eqb = solve(eqa, M_0)
```

eqb =

$$\frac{L^3\,W\,k_t\,p}{24\,\mathrm{YJ} + 12\,L\,k_t}$$

```
M0_numeric = eval(subs(eqb,{p W YJ L k_t},{pmax w Yp*J l kt}))
```

```
Warning: The function sym/eval is deprecated and will be removed in a future release. Depending on the
usage, use subs or double instead.
M0_numeric =
5.2313e-04
```

Lets write the deform expression by explicit linear relation in respect the pressure 'p' and the electric field 'E_3':

We expect the vertical deformation $u_z(x)$ to be linearly dependent from ***Pressure p*** and ***Voltage V***

$u_z(x) = f_1(x)\,p + f_2(x)\,V$

```
uz_f = collect(simplify(subs(uz_x, [M_0],[eqb])),[p, E_3])
```

uz_f =

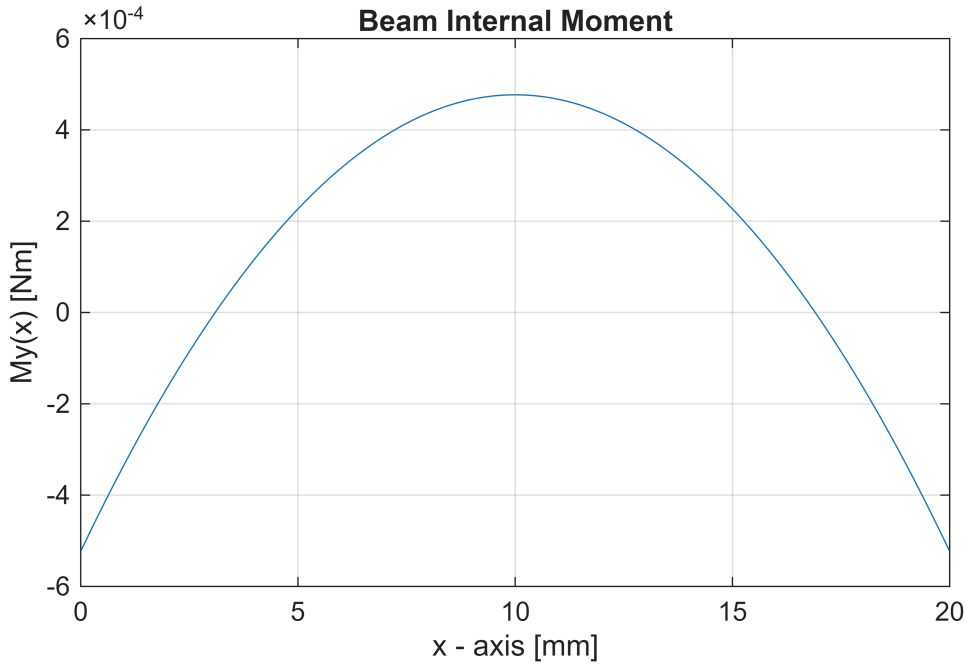
$$\frac{x\,(L-x)\left(L^2\,W - W\,x^2 + L\,W\,x - \frac{L^3\,W\,k_t}{2\,\mathrm{YJ} + L\,k_t}\right)}{24\,\mathrm{YJ}}\,p + \frac{M_p\,x\,(L-x)}{2\,\mathrm{YJ}}\,E_3$$

Plot the internal moment $M_y(x)$ evaluated at the maxinum pressure:

```
% numeric array for plot:
x1 = (0:(l/1e3):l)';
My_num = eval(subs(My, {p W L M_0 x},{pmax w l M0_numeric x1}));
```

```
Warning: The function sym/eval is deprecated and will be removed in a future release. Depending on the
usage, use subs or double instead.
```

```
figure();
plot(x1 *1e3, My_num)
title('Beam Internal Moment')
xlabel('x - axis [mm]')
ylabel('My(x) [Nm]')
grid on
```

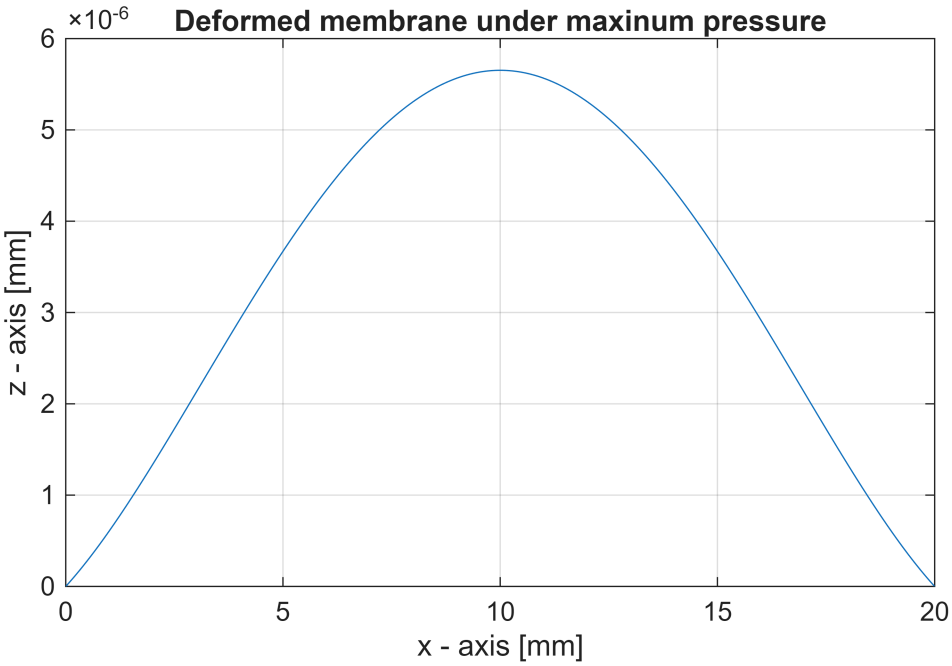


Plot the shape of the deformed membrane $u_z(x)$ at the maxinum pressure:

```
uz_f_n = eval(subs(uz_f, {p W YJ M_p L E_3 k_t x},{pmax w Yp*J Mp l 0 kt x1}));
```

```
Warning: The function sym/eval is deprecated and will be removed in a future release. Depending on the
usage, use subs or double instead.
```

```
figure(1)
plot(x1 *1e3, uz_f_n)
title('Deformed membrane under maxinum pressure')
xlabel('x - axis [mm]')
ylabel('z - axis [mm]')
grid on
```



1.2) Volume subtended by the membrane and displaced charge

$$\Omega = w \int_0^L u_z(x)\,\mathrm{dx}$$

```
Omega_Ep = simplify(w * int(uz_f,x,0,l))
```

```
Omega_Ep =
```

$$\frac{6\,W\,\mathrm{YJ}\;p - 300000\,E_3\,M_p\,\mathrm{YJ} + 11250000\,E_3\,L^2\,M_p\,k_t + 1875000\,L^3\,W\,\mathrm{YJ}\;p - 375\,L^2\,W\,k_t\,p + 12500\,L^3}{5625000000000\,\mathrm{YJ}\;(2\,\mathrm{YJ} +}$$

```
Omega_num_max = eval(subs(Omega_Ep,{p W YJ M_p L E_3 k_t},{pmax w Yp*J Mp l 0
kt})); % [m^3]
```

```
Warning: The function sym/eval is deprecated and will be removed in a future release. Depending on the
usage, use subs or double instead.
```

```
disp(['Maxinum subtended volume: ',num2str(Omega_num_max*1e9),' mm^3'])
```

```
Maxinum subtended volume: 1.3454 mm^3
```

Transducer equation

Linear relation with *pressure* and *voltage*

- K_Ω volumetric stiffness
- α electric coupling constant

MECHANICAL EQUATION (volume in static conditions)

$$\Omega(p, V) = \frac{1}{K_\Omega}\, p + \alpha\, V$$

ELECTRICAL EQUATION (amount of charge displaced by the piezo)

$$q(p, V) = \alpha\, p + C\, V$$

Let's focus on the Mechanical Equation to extract the constant coefficient

```
Omega_eq = collect(Omega_Ep,[p, E_3]) % with collected terms for visualization
```

```
Omega_eq =
```

$$\frac{6\, W\, \mathrm{YJ} - 750\, L\, W\, \mathrm{YJ} + 3\, L\, W\, k_t + 1875000\, L^3\, W\, \mathrm{YJ} - 375\, L^2\, W\, k_t + 12500\, L^3\, W\, k_t}{\sigma_1}\, p + \left(-\frac{300000\, M_L}{}\right.$$

where

$$\sigma_1 = 5625000000000\, \mathrm{YJ}\ (2\, \mathrm{YJ} + L\, k_t)$$

Extrapolate the target coefficients

```
terms = children(Omega_eq);
```

Volumetric stiffness K_Ω

```
K_omega = (coeffs(terms(1), p))^(-1);
K_omega_num = eval(subs(K_omega, {W YJ L k_t},{w Yp*J l kt})) % [N/m^5]
```

```
Warning: The function sym/eval is deprecated and will be removed in a future release. Depending on the
usage, use subs or double instead.
K_omega_num =
7.4327e+11
```

Electric coupling constant α

```
alpha = coeffs(terms(2), E_3)
```

```
alpha =
```

$$-\frac{300000\, M_p\, \mathrm{YJ} - 22500000\, L\, M_p\, \mathrm{YJ} + 150000\, L\, M_p\, k_t - 11250000\, L^2\, M_p\, k_t}{5625000000000\, \mathrm{YJ}\ (2\, \mathrm{YJ} + L\, k_t)}$$

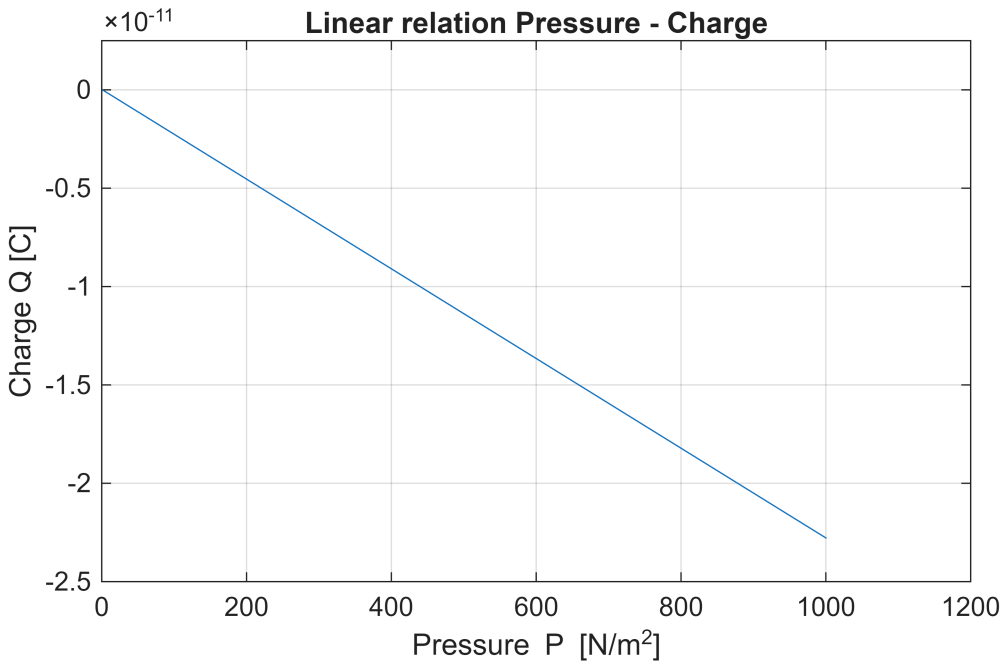
```
alpha_num = eval(subs(alpha, {W YJ M_p L k_t},{w Yp*J Mp l kt})) % [m^4/V]
```

```
Warning: The function sym/eval is deprecated and will be removed in a future release. Depending on the
usage, use subs or double instead.
alpha_num =
-2.2800e-14
```

Since the transducer is used as a sensor, the voltage in the circuit is so low such that the elecrical part is cutted out from the equation. A short-circuit condition is assumed.

Linear relation Charge - Pressure

```
p_array = 0:1:pmax;
q_plot = alpha_num * p_array;

figure;
plot(q_plot);
xlabel('Pressure P [N/m^2]');
ylabel('Charge Q [C]');
ylim([-2.5e-11,0.25e-11])
title('Linear relation Pressure - Charge');
grid on;
```



Linear relation Volume - Pressure

```
Omega_p = double(1/K_omega_num) * p
```

```
Omega_p =
```

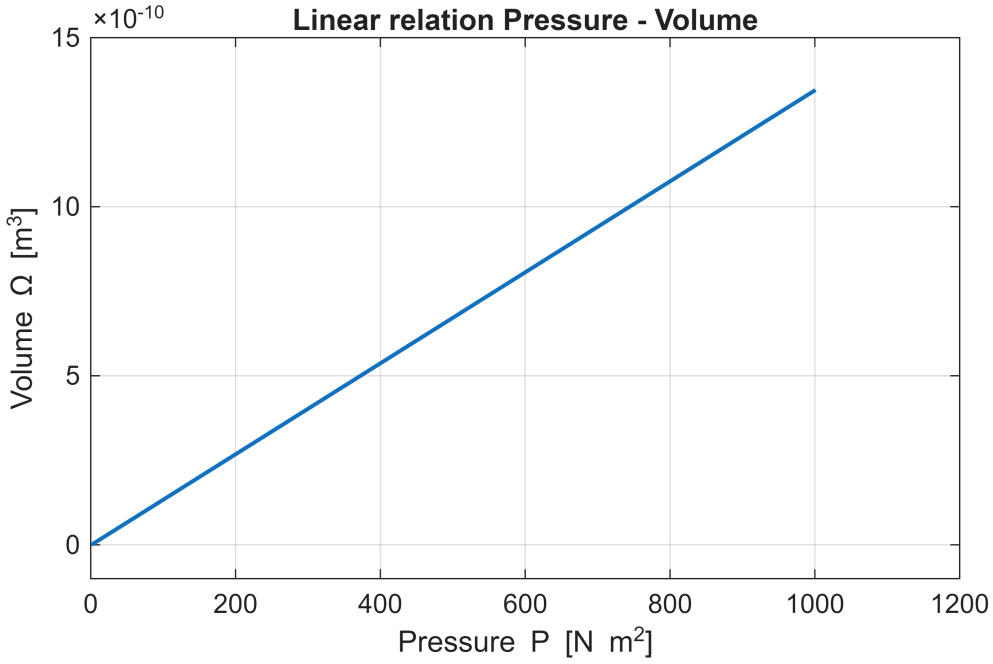
$$\frac{3331055404994553\,p}{24758800785707605497982484 48}$$

```
Omega_plot = eval(subs(Omega_p,p_array));
```

```
Warning: The function sym/eval is deprecated and will be removed in a future release. Depending on the
usage, use subs or double instead.
```

```
figure;
plot(Omega_plot, 'LineWidth', 1.5);
xlabel('Pressure P [N m^2]');
ylabel('Volume \Omega [m^3]');
ylim([-0.1e-9,1.5e-9])
title('Linear relation Pressure - Volume');
```

```
grid on;
```



Dynamic Analysis (A)

Assume the system dynamics are linear and the oscillations are damped.

Find the frequency reponse, the useful working range, the amplitude of the voltage ouput

2.1) Frequency Response

Step 1: pressure variations lead to **volume variations** through the beam deformation

Step 2: volume variations lead to **current variations** through the piezo deformation

Step 3: current variations lead to **voltage variations** through the Amplifier

By using three different Transfer Functions in cascade, it is possible to establish a clear relation between **pressure variations** and **voltage variations**

2.1.1) Equation of Motion and its constants

The considered Pressure (p) - Volume (Ω) function in static was in this form:

$$p = K_\Omega \Omega$$

We update the formulation introduciong the dynamic terms:

$$p = K_\Omega \Omega + B_\Omega \dot{\Omega} + M_\Omega \ddot{\Omega}$$

Coefficient K_Ω found previousy from the Mechanical Equation

It is possible to find coefficients B_Ω and M_Ω individually

Lets start from inertial coefficient M_Ω

We make a step back to the relation:

$u_z(x) = f_1(x)\, p + f_2(x)\, V$

Short circuit assumption is still valid: V = 0

$$u_z = f_1(x)p$$

Consider the static relation $p = K_\Omega * \Omega$ is valid also for dynamic conditions, under the assumptions:

- higher resonant frequencies are not reached by the system
- p is now an equivalent pressure that considers also the inertia

$$u_z = f_1(x)K_\Omega\Omega$$

rename: $g_1(x) = f_1(x)\, K_\Omega$ --> $u_z(x) = g_1(x)\, K_\Omega$

NOTE: $g_1(x)$ is the *zero order* modshape

(the modshape is constant, for higher value the zero modshape is scaled by a factor)

Compute the vertical velocity:

$$v_z = \frac{\delta u_z}{\delta t} = g_1(x)\dot{\Omega} + \cancel{\dot{g_1}(x)\Omega}$$

Let's compute the kinetic energy in respect to the Volume:

$$E_k = \frac{1}{2}mv^2 = \frac{1}{2}M_\Omega\dot{\Omega}^2 = \frac{1}{2}\int_U v_x^2 du = \frac{1}{2}\rho w h_p \int_0^l g_1(x)^2 dx \dot{\Omega}^2$$

```
terms_b = children(uz_f);
f1 = coeffs(terms_b(1), p);
f1_num_x = eval(subs(f1, {W YJ M_p L k_t},{w Yp*J Mp l kt})); % [m^3/N] f1
```

```
Warning: The function sym/eval is deprecated and will be removed in a future release. Depending on the
usage, use subs or double instead.
```

```
g1 = f1_num_x*K_omega_num;
```

Now we have the element to extract the Inertial Coefficient M_Ω:

$$M_\Omega = \rho\, w\, h_p \int_0^L g_1(x)^2\, \mathrm{dx}$$

```
disp('Inertia coeff: ')
```

```
Inertia coeff:
```

```
M_omega_num = double(rho*w*2*h * int(g1^2,x,0,l)) % [N s^2 / m^5]
```

```
M_omega_num =
7.3861e+03
```

and to extract the Damping Coefficient given the damping ratio of $\xi = 0.2$:

$$B_\Omega = 2\,\xi\,\sqrt{K_\Omega\, M_\Omega}$$

```
disp('Damping coeff: ')
```

```
Damping coeff:
```

```
B_omega_num = 2*csi*sqrt(K_omega_num*M_omega_num)%*M_omega_num % [N s / m^5]
```

```
B_omega_num =
2.9637e+07
```

and the Natural Frequency

```
disp('natural freq: ')
```

```
natural freq:
```

```
w_n = sqrt(K_omega_num/M_omega_num) % [rad/s]
```

```
w_n =
1.0032e+04
```

2.1.2) Transfer Function Volume / Pressure

- Transfer Function of the system $\frac{\Omega(s)}{p(s)}$

$$p = K_\Omega \Omega + B_\Omega \dot{\Omega} + M_\Omega \ddot{\Omega}$$

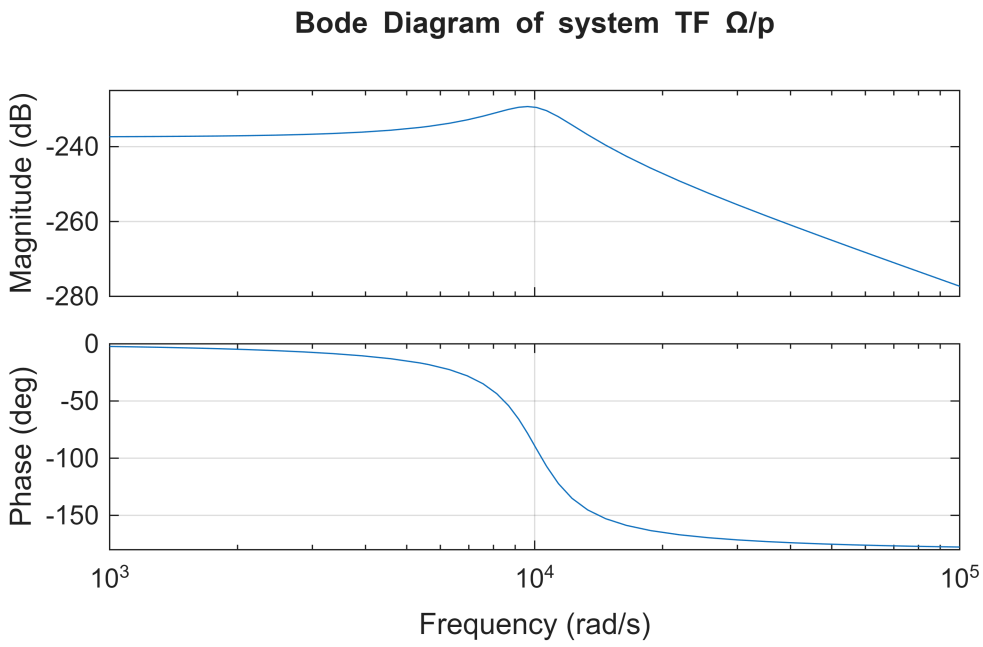
$$\frac{\Omega(s)}{p(s)} = \frac{1}{s^2 M_\Omega + s\, B_\Omega + K_\Omega}$$

```
numerator_sys = [0 0 1];
denominator_sys = [M_omega_num, B_omega_num, K_omega_num];
% H_sys = omega/p
H_sys = tf(numerator_sys, denominator_sys)
```

```
H_sys =

                1
```

```
  --------------------------------
  7386 s^2 + 2.964e07 s + 7.433e11

Continuous-time transfer function.
Model Properties
```

```
bode(H_sys)
grid on
title('Bode Diagram of system TF \Omega/p')
```



2.1.3) Transfer Function Current / Volume

- Transfer Function of the system $\frac{I(s)}{\Omega(s)}$

Remember the Electrical Equation:

$q(p, V) = \alpha\, p + C\, V$

Short Circuit assumption is still valid

$q = \alpha\, p$

Remember fundamental relation:

$p = K_\Omega\, \Omega$

Compute the derivative:

$i = \alpha\, K_\Omega\, \dot{\Omega}$

Switch to Laplace domain, assuming inital volume equal to zero.

$I(s) = s\, \alpha\, K_\Omega\, \Omega(s)$

Rearrange:

$$\frac{I(s)}{\Omega(s)} = s\,\alpha\,K_\Omega$$

$$\Omega(s) = H_1(s) * p(s)$$

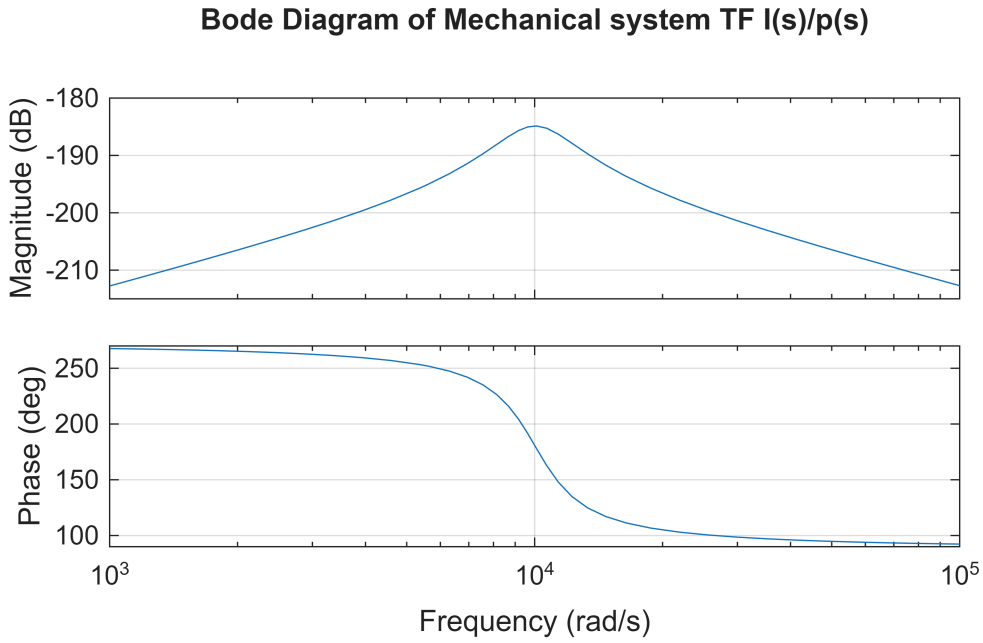
$$\frac{I(s)}{p(s)} = s\,\alpha\,K_\Omega * H_1(s)$$

$$\frac{I(s)}{p(s)} = \frac{s\,\alpha\,K_\Omega}{s^2 M_\Omega + s\,B_\Omega + K_\Omega}$$

```
numerator_sys2 = [0 alpha_num*K_omega_num 0];
denominator_sys2 = [M_omega_num, B_omega_num, K_omega_num];
H_sys2 = tf(numerator_sys2, denominator_sys2)
```

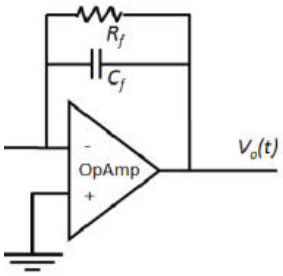
```
H_sys2 =

             -0.01695 s
  --------------------------------
  7386 s^2 + 2.964e07 s + 7.433e11

Continuous-time transfer function.
Model Properties
```

```
bode(H_sys2)
title('Bode Diagram of Mechanical system TF I(s)/p(s)')
grid on
```



2.1.4) Transfer Function Voltage / Current

- Transfer function of the OpAmp $\frac{V(s)}{I(s)}$



$$i = i_R + i_C$$

$$R_f\, i_R = \frac{\int i_C}{C_f}$$

$$Z_R = R \quad Z_C = \frac{1}{j\,w\,C} \quad Z_L = j\,w\,L$$

$$Z_{\text{parallel}} = \sum_i \frac{1}{Z_i}$$

$$Z_p = \frac{R}{1 + j\,w\,R\,C}$$

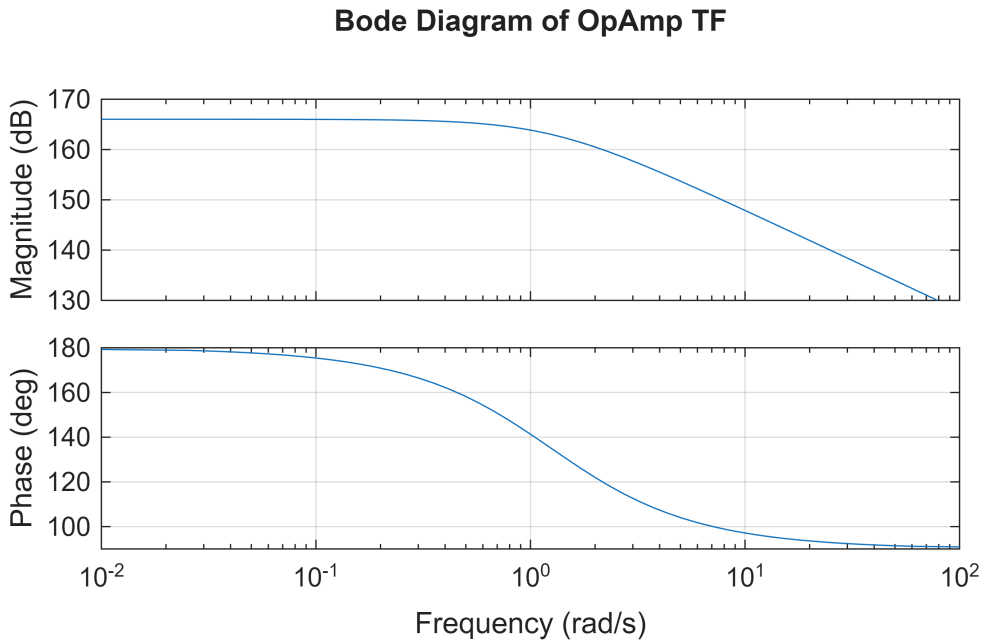
$$\frac{V(s)}{I(s)} = -Z_p$$

$$\frac{V(s)}{I(s)} = -\frac{R}{1 + j\,w\,R\,C}$$

```
numerator_OpAmp = [0 0 -Rf];
denominator_OpAmp = [0, Cf*Rf, 1];
H_OpAmp = tf(numerator_OpAmp, denominator_OpAmp)
```

```
H_OpAmp =

    -2e08
  ---------
  0.8 s + 1

Continuous-time transfer function.
Model Properties
```

```
bode(H_OpAmp)
title('Bode Diagram of OpAmp TF')
grid on
```



2.1.5) Global Transfer Function

We are interested in this relation: $\frac{V_0(s)}{p(s)}$

Given the pressure ***p***, we directly have the output voltage ***V***

$$H(s) = \frac{I(s)}{p(s)} \frac{V_0(s)}{I(s)}$$

$$H(s) = \frac{V_0(s)}{p(s)}$$

```
% H = I/p * V_0/I --> V_0/p

% INPUT: pressure; OUTPUT: voltage

H = H_sys2*H_OpAmp
```

```
H =

                    3.389e06 s
  -----------------------------------------------
  5909 s^3 + 2.372e07 s^2 + 5.946e11 s + 7.433e11

Continuous-time transfer function.
Model Properties
```

```
figure;
[mag, phase] = bode(H);
bode(H)
```

```
H_param = findall(gcf, 'Type', 'axes'); % Get all axes in the figure
title('Bode Diagram of Global TF')
grid on;
```

2.2) Useful Working Range

Let's find the interesting frequencies:

- Cut-in freq.

$$f_c = \frac{1}{2\,\pi\,C_f\,R_f}$$

```
fc = 1/(2*pi) * 1/(Cf*Rf)  % [Hz]
```

```
fc =
0.1989
```

```
fc_lim = 5 * (fc);       % [Hz]
wc = fc * 2*pi;            % [rad/s]
wc_lim = 5 * wc;         % [rad/s]
```

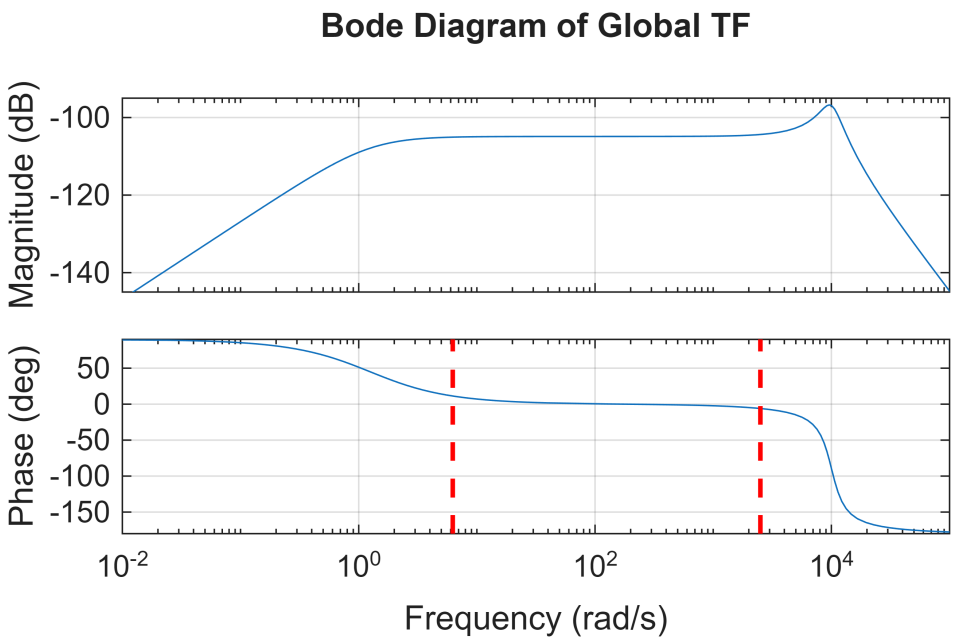
- Undamped resonance freq.

$$f_n = \frac{1}{2\,\pi}\,w_n$$

```
fn = 1/(2*pi) * w_n % [Hz]
```

```
fn =
1.5966e+03
```

```
fn_lim = 1/4 * fn;  % [Hz]
wn = w_n;                % [rad/s] previously computed
wn_lim = 1/4 * wn;  % [rad/s]

axes(H_param); % h(2) is usually the magnitude plot
line([wc_lim wc_lim], ylim, 'Color', 'r', 'LineStyle', '--', 'LineWidth', 1.5);
line([wn_lim wn_lim], ylim, 'Color', 'r', 'LineStyle', '--', 'LineWidth', 1.5);
```



The effectice linear range is not given by the cut-in and the natural frequencies;

The linear range is close to those frequency:

```
disp(['Cut-in freq:                          ',num2str(2*pi*fc),' rad/s'])
```

```
Cut-in freq:                       1.25 rad/s
```

```
disp(['Effective Cut-in freq:                ',num2str(wc_lim),' rad/s'])
```

```
Effective Cut-in freq:             6.25 rad/s
```

```
disp(['Undamped resonance freq:              ',num2str(2*pi*fn),' rad/s'])
```

```
Undamped resonance freq:           10031.5128 rad/s
```

```
disp(['Effective Undamped resonance freq: ',num2str(wn_lim),' rad/s'])
```

```
Effective Undamped resonance freq: 2507.8782 rad/s
```

```
disp(['-'])
```

```
-
```

```
disp(['The useful operating range is: [',num2str(fc_lim),', ',num2str(fn_lim),']
[Hz]'])
```

```
The useful operating range is: [0.99472, 399.1412] [Hz]
```

2.3) Amplitude of voltage output

Compute the max range of the amplified voltage, assume p > 0 and p <= pmax

```
disp('---------------------------------')
```

```
---------------------------------
```

```
disp('Used voltage range: [V]')
```

```
Used voltage range: [V]
```

```
V_0_min = abs(evalfr(H,1i * wc_lim)) * pmax
```

```
V_0_min =
0.0056
```

```
V_0_max = abs(evalfr(H,1i * wn_lim)) * pmax
```

```
V_0_max =
0.0060
```

Dynamic Analysis (B)

Build a time-domain dynamic model of the system

Test the system response in the presence of multichromatic pressure signals

Discretise the frequency range, with frequencies and a step Δ.

```
Fs = 10000;  % Sampling frequency (Hz) fs > 2*fmax
T = 1;      % Duration (s)
N = Fs * T; % Number of points
t = linspace(0, T, N);  % Time vector

% Frequency range
f_min = 1;
f_max = Fs/2;
df = 1; % Frequency step
f = f_min:df:f_max;
N_freq = length(f);


f0 = fn; % [Hz] mean of the gaussian PSD
sigma = 1e8; % variance of the gaussian PSD
```

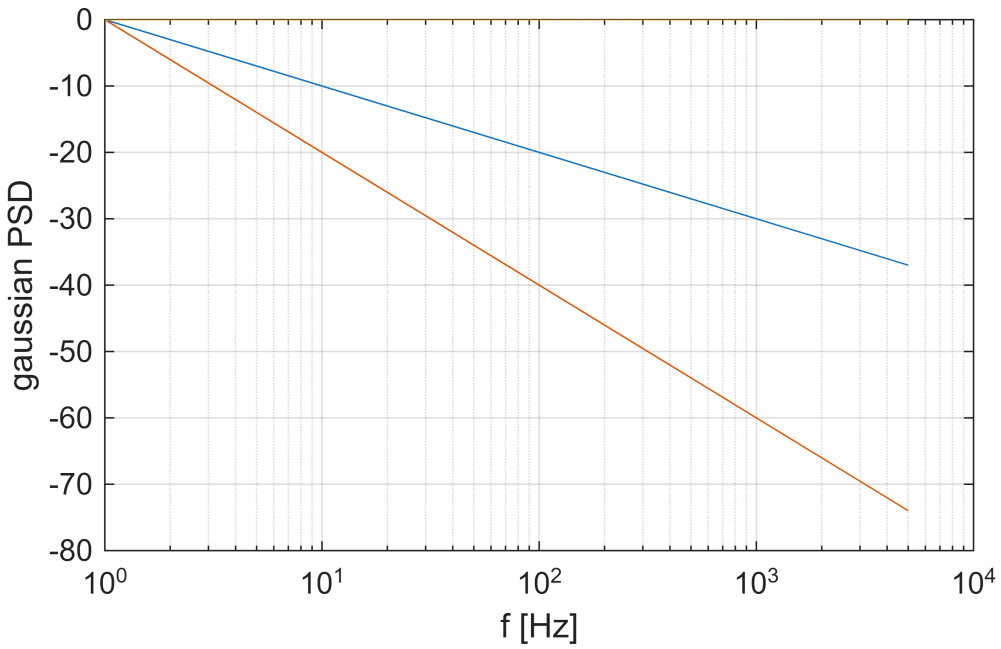
Buid the noise signal:

```
% Desired total power of the signal prsm = 3*sigma
prms = pmax / 3; % Compute RMS pressure
```

Factor 3 is reasonable assumption for maxinum pressure in Gaussian-distributed signals (like pink noise) because about 99.7% of the energy of the signal lies between $\pm 3\,\sigma$

```
P_total = prms^2; % Compute total power;
% Prms^2 reflects the pressure wave's energy density
```

```
% Compute S0
% S0 = P_total / sum(f_min ./ f);
S0 = 1;

phi = 2 * pi * rand(1, N_freq); % random phases [0,2*pi]

% NOISE PSDs
S_pi = S0./f; % pink noise
S_br = S0./(f.^2); % brownian noise
S_ga = S0*exp(-((f-f0)/sigma).^2); % gaussian-shaped

figure();
semilogx(f, 10*log10(S_pi))
hold on
xlabel('f [Hz]')
ylabel('Pink PSD')
grid on
semilogx(f, 10*log10(S_br))
xlabel('f [Hz]')
ylabel('Brownian PSD')
grid on
semilogx(f, 10*log10(S_ga))
xlabel('f [Hz]')
ylabel('gaussian PSD')
grid on
hold off
```

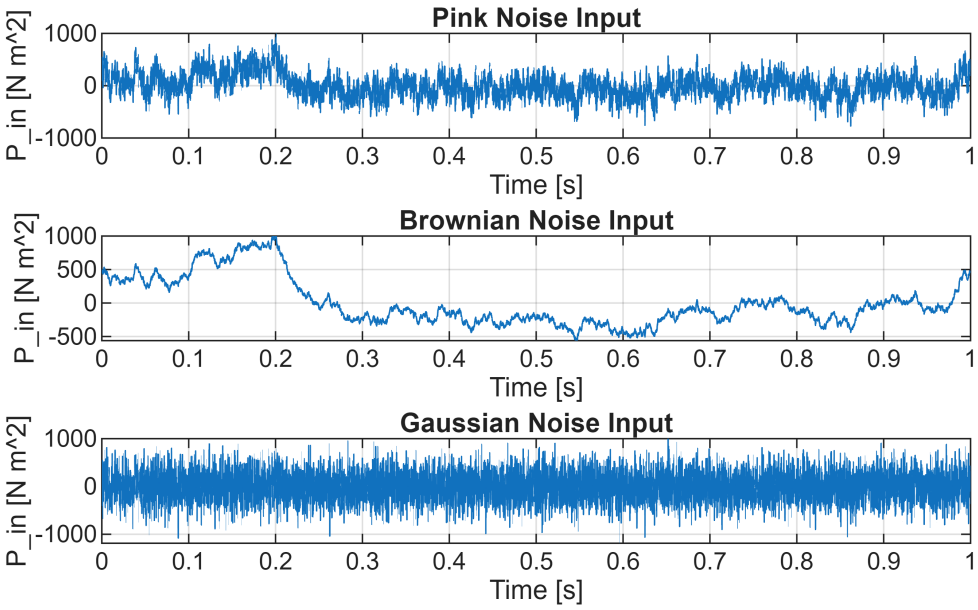


These curves are plausible for theoretical values

- A white noise signal has equal power across all frequencies. Therefore, the PSD plot should de flat.

- The pink noise signal has a dependency about $1/f$. This will create a PSD curve with an inclination of $-10\,\mathrm{dB/dec}$ because $10\log_{10}(S(f)) = 10\log_{10}\left(\frac{S_0}{f}\right) = 10\log_{10}(S_0) - 10\log_{10}(f)$
- The pink noise signal has a dependency about $1/f^2$. This will create a PSD curve with an inclination of $-20\,\mathrm{dB/dec}$ because $10\log_{10}(S(f)) = 10\log_{10}\left(\frac{S_0}{f^2}\right) = 10\log_{10}(S_0) - 10\log_{10}(f)$

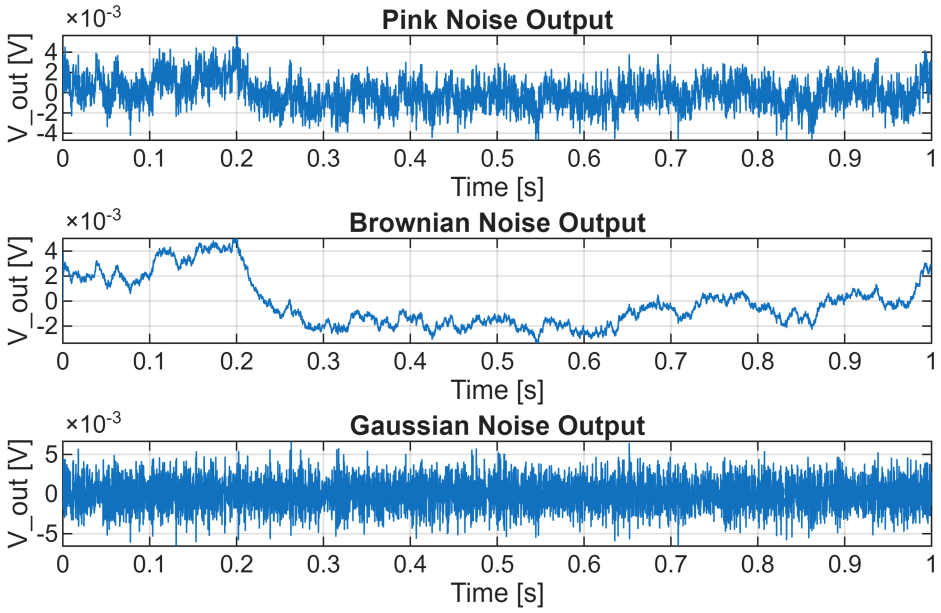
```
p_pi = sum(sqrt(2 * df * S_pi(:)) .* sin(2 * pi * f(:) * t + phi(:)), 1);
p_br = sum(sqrt(2 * df * S_br(:)) .* sin(2 * pi * f(:) * t + phi(:)), 1);
p_ga = sum(sqrt(2 * df * S_ga(:)) .* sin(2 * pi * f(:) * t + phi(:)), 1);
p_pi = p_pi/max(p_pi) * pmax;
p_br = p_br/max(p_br) * pmax;
p_ga = p_ga/max(p_ga) * pmax;


noise_ts_1 = timeseries((p_pi)', (t)');
noise_ts_2 = timeseries((p_br)', (t)');
noise_ts_3 = timeseries((p_ga)', (t)');

figure();
subplot(3,1,1)
plot(noise_ts_1)
title('Pink Noise Input')
xlabel('Time [s]')
ylabel('P_in [N m^2]')
grid on
subplot(3,1,2)
plot(noise_ts_2)
title('Brownian Noise Input')
xlabel('Time [s]')
ylabel('P_in [N m^2]')
grid on
subplot(3,1,3)
plot(noise_ts_3)
title('Gaussian Noise Input')
xlabel('Time [s]')
ylabel('P_in [N m^2]')
grid on
```



Start simulink and bring V0 signal to workspace

```
inputData = {noise_ts_1, noise_ts_2, noise_ts_3};
size_inputData = length(inputData);
T; % simulation time [s]

% Initialize a cell array to store output data
outVoltage = cell(1,size_inputData);

modelName = 'Dynamic_model';

for i = 1:size_inputData
    % Assign the current time series to the Simulink input
    assignin('base', 'inputTimeSeries', inputData{i});

    % Run the simulation
    simOut = sim(modelName, 'ReturnWorkspaceOutputs', 'on');

    % Extract output data from the simulation
    outVoltage{i} = simOut.get('V0sim');

    % Save output with a unique name in the workspace
    assignin('base', ['outVoltage_' num2str(i)], outVoltage{i});
end

figure();
subplot(3,1,1)
plot(outVoltage_1)
```

```
title('Pink Noise Output')
xlabel('Time [s]')
ylabel('V_out [V]')
grid on
subplot(3,1,2)
plot(outVoltage_2)
title('Brownian Noise Output')
xlabel('Time [s]')
ylabel('V_out [V]')
grid on
subplot(3,1,3)
plot(outVoltage_3)
title('Gaussian Noise Output')
xlabel('Time [s]')
ylabel(['V_out [V]'])
grid on
```



Compute the FFT and visualize magnitude spectrum

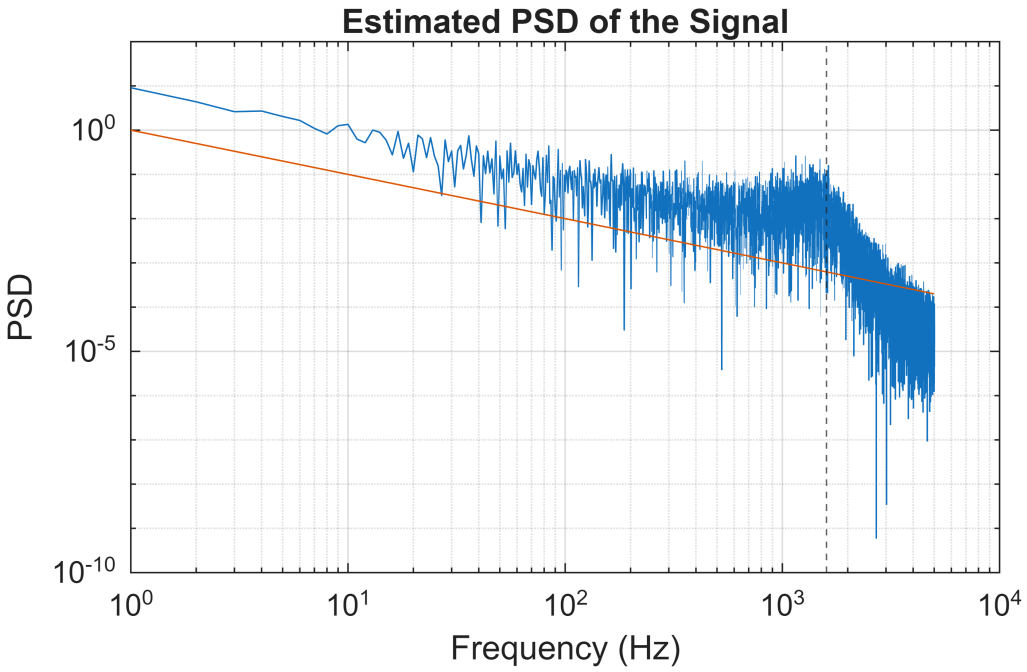
```
% from Pink noise
fs_V_1 = 1 / mean(diff(outVoltage_1.Time)); %  % Compute sampling frequency from
time steps
N_1 = length(outVoltage_1.Data);       % Number of samples
V_1 = fft(outVoltage_1.Data);          % Compute FFT
f_1 = (0:N_1-1) * (fs_V_1 / N_1);      % Frequency vector
PSD_1 = (abs(V_1).^2);% / (fs_V_1 * N_1);  % Compute PSD (normalization)

% from Brownian noise
```

```
fs_V_2 = 1 / mean(diff(outVoltage_2.Time)); %  % Compute sampling frequency from
time steps
N_2 = length(outVoltage_2.Data);       % Number of samples
V_2 = fft(outVoltage_2.Data);          % Compute FFT
f_2 = (0:N_2-1) * (fs_V_2 / N_2);      % Frequency vector
PSD_2 = (abs(V_2).^2);% / (fs_V_2 * N_2);  % Compute PSD (normalization)

% from Gaussian noise
fs_V_3 = 1 / mean(diff(outVoltage_3.Time)); %  % Compute sampling frequency from
time steps
N_3 = length(outVoltage_3.Data);       % Number of samples
V_3 = fft(outVoltage_3.Data);          % Compute FFT
f_3 = (0:N_3-1) * (fs_V_3 / N_3);      % Frequency vector
PSD_3 = (abs(V_3).^2);% / (fs_V_3 * N_3);  % Compute PSD (normalization)

% Voltage plot from Pink noise
% figure();
% loglog(f_1(1:N_1 / 2), abs(V_1(1:N_1 / 2)));  % Plot positive frequencies
% hold on
% xlabel('Frequency (Hz)');
% ylabel('Magnitude [V]');
% title('Signal Spectrum Pink');
% xline(fn, '--')
% grid on
% hold off
% % Voltage plot from Brownian noise
% figure();
% loglog(f_2(1:N_2 / 2), abs(V_2(1:N_2 / 2)));  % Plot positive frequencies
% hold on
% xlabel('Frequency (Hz)');
% ylabel('Magnitude [V]');
% title('Signal Spectrum Brownian');
% xline(fn, '--')
% grid on
% hold off
% % Voltage plot from Gaussian noise
% figure();
% loglog(f_3(1:N_3 / 2), abs(V_3(1:N_3 / 2)));  % Plot positive frequencies
% hold on
% xlabel('Frequency (Hz)');
% ylabel('Magnitude [V]');
% title('Signal Spectrum Gaussian');
% xline(fn, '--')
% grid on
% hold off
```

Compute and plot the PSD

```
figure();
loglog(f_1(1:N_1/2), PSD_1(1:N_1/2));  % Log-log plot for better visibility
```

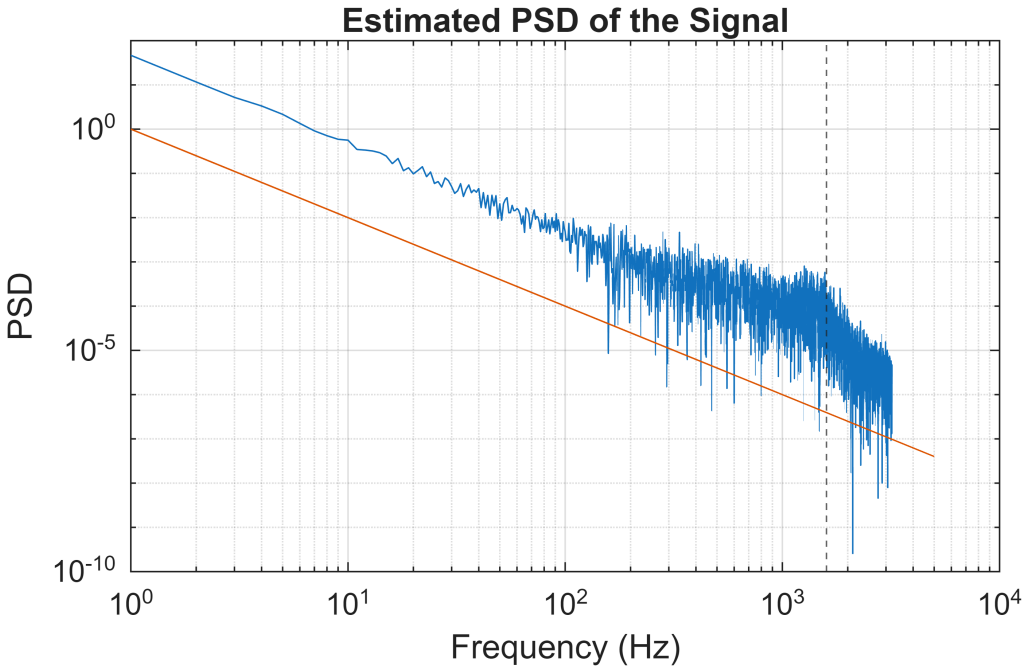
```
hold on
loglog(f,S_pi)
xlabel('Frequency (Hz)');
ylabel('PSD');
xline(fn, '--')
title('Estimated PSD of the Signal');
grid on
hold off
```




```
figure();
loglog(f_2(1:N_2/2), PSD_2(1:N_2/2));  % Log-log plot for better visibility
```

```
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
```

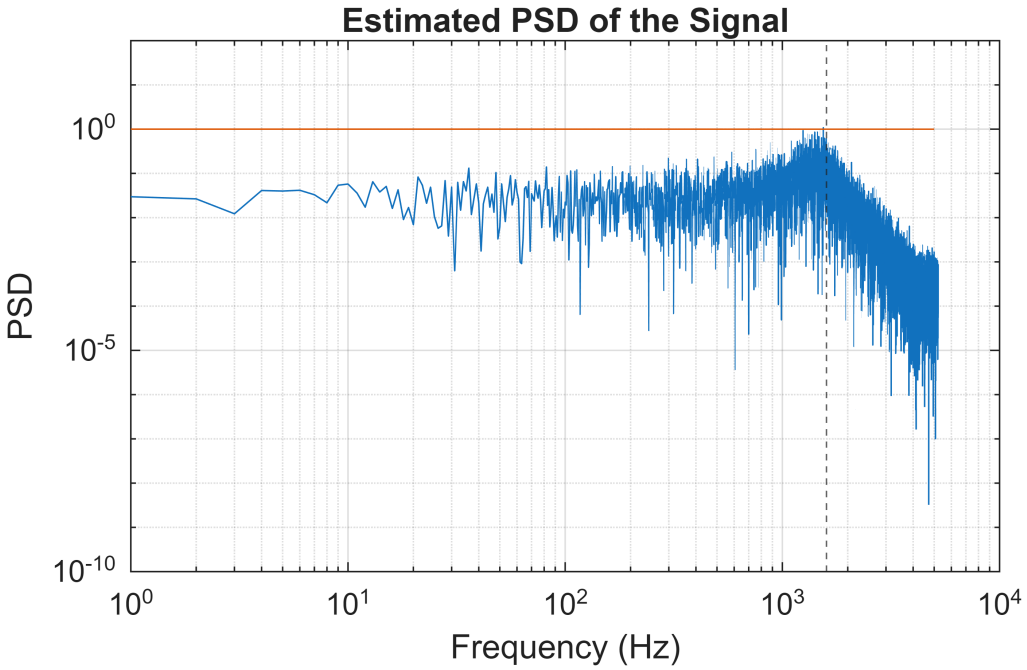
```
hold on
loglog(f,S_br)
xlabel('Frequency (Hz)');
ylabel('PSD');
xline(fn, '--')
title('Estimated PSD of the Signal');
grid on
hold off
```




```
figure();
loglog(f_3(1:N_3/2), PSD_3(1:N_3/2));  % Log-log plot for better visibility
```

```
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
```

```
hold on
% semilogx(f_out_brownian_sim,10*log10(P_f_out_brownian_sim),'LineWidth',0.5)
loglog(f,S_ga)
xlabel('Frequency (Hz)');
ylabel('PSD');
xline(fn, '--')
title('Estimated PSD of the Signal');
grid on
hold off
```



Conclusion:

As we can observe from the time domain input ando outputs the conclusion is not abvious

It may be more readable in the frequency domain

By comuting the FFT and make it the power of 2 the plot of the corresponding PSD can be observed.

The output PSD has the same trend of the input PSD, with 2 differences:

- the output is attenuated

- we have maxinum values where the input PSD has concentration of power

e.g. at low frequencies for Pink and Brownian noise and at the Gaussian center

- we have a slight peak at the natural frequency