Disk-Torsional Spring-Chain with Houdaille Damper Absorbers

by

Sina Atalay, Cem Geçgel, Mustafa Çağatay Sipahioğlu

Department of Mechanical Engineering
Bogazici University
Turkey

Spring 2022

Disk-Torsional Spring-Chain with Houdaille Damper Absorbers

by

Sina Atalay, Cem Geçgel, Mustafa Çağatay Sipahioğlu

Abstract

This project aims to analyze a disk-torsional spring-chain. The mode shapes of the system are found using modal analysis. The transmissibility from the base excitation to the last disk is found by using the corresponding element of the receptance matrix. The transmissibility is reduced by adding Houdaille dampers, which were optimzied using the built-in fminimax function of MATLAB. The damped frequencies were found by searching a small range before the the natural frequencies using built-in fminbnd function of MATLAB. The results showed that the minimum transmissibility is achived by putting a single absorber at the last disk and the limit on the total viscosity of the absorbers is the important parameter, while the number of disks has negligible effects.

Table of Contents

1 Introduction

In Figure 1, an (n+2)-degrees-of-freedom system that is going to be designed is shown, where $I_i = 100/n$, $k_i = 25n$. Four problems will be solved to design the system:

1. What are the natural frequencies and mode shapes of the n-degrees-of-freedom disk-torsional spring chain without the viscous torsional dampers for a given n?
2. Let the system without the viscous torsional dampers to be base excited. Find transmissibility for the last disk (Θ_n/Φ), assuming $\overrightarrow{\theta}(t) = \overrightarrow{\Theta}e^{i\omega t}$, $\overrightarrow{\varphi}(t) = \Phi e^{i\omega t}$, and plot the absolute value of the transmissibility in log-log axes for $0 \leq \omega \leq 1.5\omega_{n,\max}$.
3. For the (n+2)-degrees-of-freedom system shown in Figure 1 (viscous dampers are attached to the first and last disks) find the optimum c_{a1} and c_{a2} values that minimize the peak transmissibility for the last disk. I_{a1} and I_{a2} are given and between 0.1 and 0.3.
4. For a given $I_{a1} + I_{a2} = \mu$ value, find the optimum I_{a1}, I_{a2}, c_{a1}, c_{a2} values and location of the torsional dampers that minimize the peak transmissiblity for the last disk.

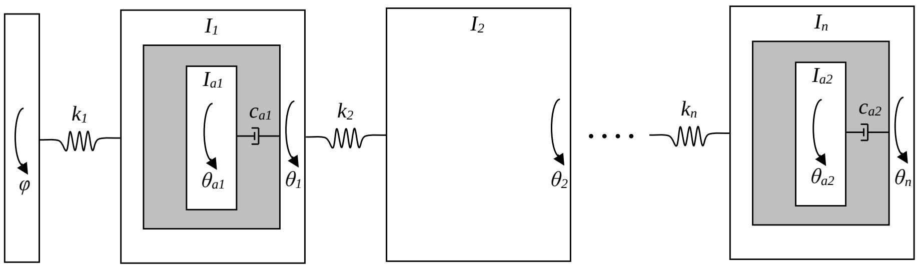

Figure 1: The system to be designed.

2 Theory

Equation of motion is given below in the matrix form.

$$\boldsymbol{M}\ddot{\vec{\theta}} + \boldsymbol{C}\dot{\vec{\theta}} + \boldsymbol{K}\vec{\theta} = \vec{F}$$

The force vector's first component can be written in terms of the excitation angular displacelemnt by multiplying the latter by the stiffness between the base and the first inertia. Rest of its components are zero. The excitation and the resulting angular displacements are expressed as fallows.

$$F_1 = k\Phi$$

$$\phi = \Phi\exp(i\omega t)$$

$$\vec{\theta} = \vec{\Theta} \exp(i\omega t)$$

The equation of motion can be reduced using the preciding expressions as the exponential terms cannot be zero. There will be no vibration in that case.

$$\boldsymbol{M}\ddot{\vec{\Theta}} + \boldsymbol{C}\dot{\vec{\Theta}} + \boldsymbol{K}\vec{\Theta} = \vec{F}$$

The damping matrix and the force vector are zero in the first part. Modal analysis was used to get the mode shapes and natural frequencies.

$$\boldsymbol{M}\ddot{\vec{\Theta}} + \boldsymbol{K}\vec{\Theta} = \vec{0}$$

$$\boldsymbol{M}\boldsymbol{M}^{-1/2}\ddot{\vec{Q}} + \boldsymbol{K}\boldsymbol{M}^{-1/2}\vec{Q} = \vec{0}$$

$$\boldsymbol{M}^{-1/2}\boldsymbol{M}\boldsymbol{M}^{-1/2}\ddot{\vec{Q}} + \boldsymbol{M}^{-1/2}\boldsymbol{K}\boldsymbol{M}^{-1/2}\vec{Q} = \vec{0}$$

$$\boldsymbol{I}\ddot{\vec{Q}} + \tilde{\boldsymbol{K}}\vec{Q} = \vec{0}$$

As you can see, with the above transformation finding the natural frequencies become an eigenvalue problem. The eigenvectors of the mass normalized stiffness matrix are the mode shapes and square root of its eigenvalues are the natural frequencies.

Going back to the initial equation of motion, the receptance matrix can be deriven to find the transmissibility that is wanted.

$$\boldsymbol{M}\ddot{\vec{\Theta}} + \boldsymbol{C}\dot{\vec{\Theta}} + \boldsymbol{K}\vec{\Theta} = \vec{F}$$

$$-\omega^2\boldsymbol{M}\vec{\Theta} + i\omega\boldsymbol{C}\vec{\Theta} + \boldsymbol{K}\vec{\Theta} = \vec{F}$$

$$\left(-\omega^2\boldsymbol{M} + i\omega\boldsymbol{C} + \boldsymbol{K}\right)\vec{\Theta} = \vec{F}$$

$$\vec{\Theta} = \left(-\omega^2\boldsymbol{M} + i\omega\boldsymbol{C} + \boldsymbol{K}\right)^{-1}\vec{F}$$

$$\boldsymbol{\alpha} = \left(-\omega^2\boldsymbol{M} + i\omega\boldsymbol{C} + \boldsymbol{K}\right)^{-1}$$

The transmissibility is defined from the excitation on the first degree of freedom to the output of the last degree of freedom. Then, only a single enrty of the receptance matrix can be used to improve speed.

$$\Theta_n = \boldsymbol{\alpha}_{1,n}F_1$$

$$\Theta_n = \boldsymbol{\alpha}_{1,n}k\Phi$$

$$\frac{\Theta_n}{\Phi} = \boldsymbol{\alpha}_{1,n}k$$

$$\left|\frac{\Theta_n}{\Phi}\right| = |\boldsymbol{\alpha}_{1,n}k|$$

This transmissibility is plotted for a range of excitation frequencies and also optimized using MATLAB. For finding the peaks of the transmissibility expression, the natural frequencies found in the first part were used. After the damping is added, the damped frequencies are expected to be slightly smaller than the natural frequencies. Thus the maximum of the transmissibility expression is searched n times in a range from 95% of the natural frequencies to themselves.

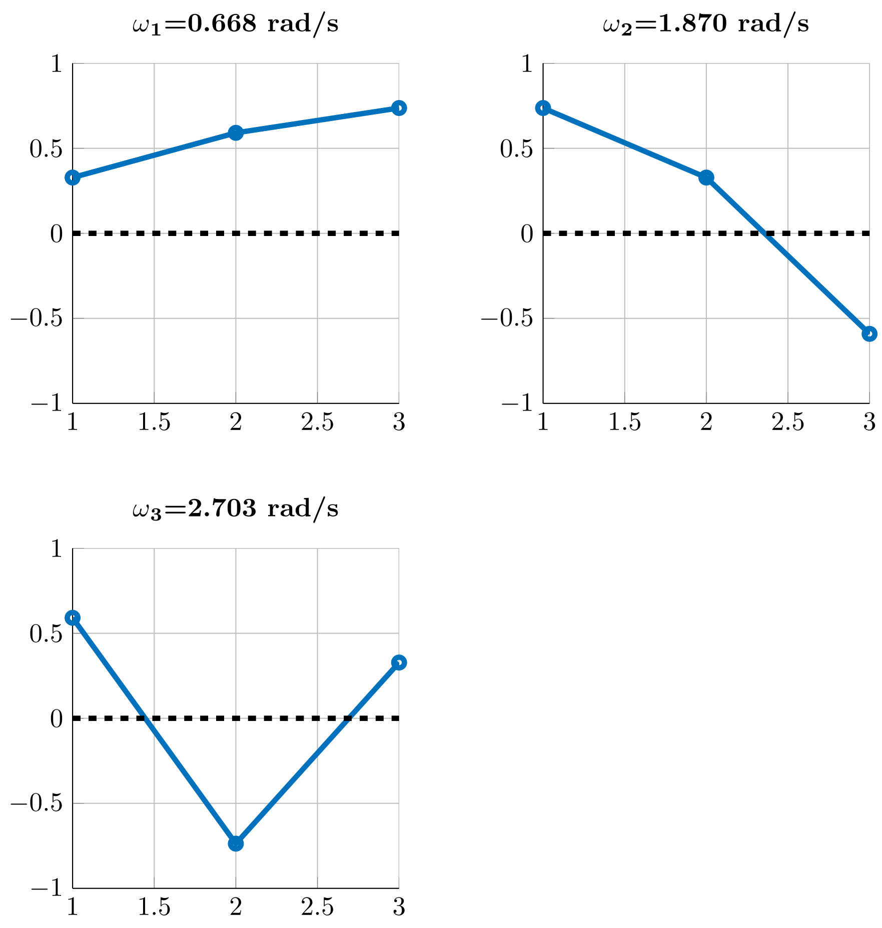

Figure 2: Mode Shapes n=3

3 Results

For the first two parts, where there are no absorbers, the mode shapes for the case with 3 and 5 degrees of freedom can be seen in Figure 2 and Figure 3, respectively. Here it is visible that, the first three mode shapes are similar. This is the expected result as the underlying physics of the system are the same.

The transmissibilities for these two cases for the undamped system are given below in Figure 4 and Figure 5. The resonance can be observed for the last disk for both cases where the transmissibility goes to infinity theoretically. Additionally, there are equal amount of peaks and degrees of freedom.

As the absorbers are placed in the first and last disk, the peaks in Figure 4 and Figure 5 shift to the left slightly. This fact was used to find the maximum transmissibility of a design efficiently as discussed in the previous section. The transmissibility of the solutions for the third part are visible in Figure 6 and Figure 7. The resonance is gone; the transmissibility is greatly reduced in this step.

In Figure 6 and Figure 7, all the peaks are in similar places in the horizontal

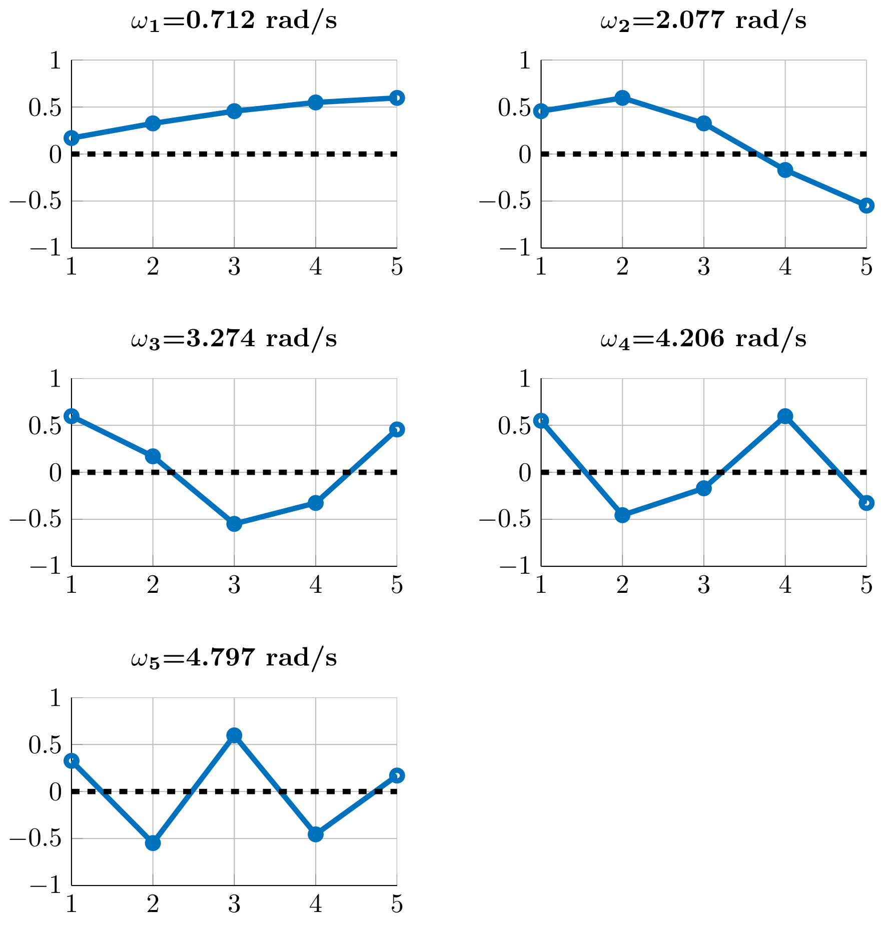

Figure 3: Mode Shapes n=5

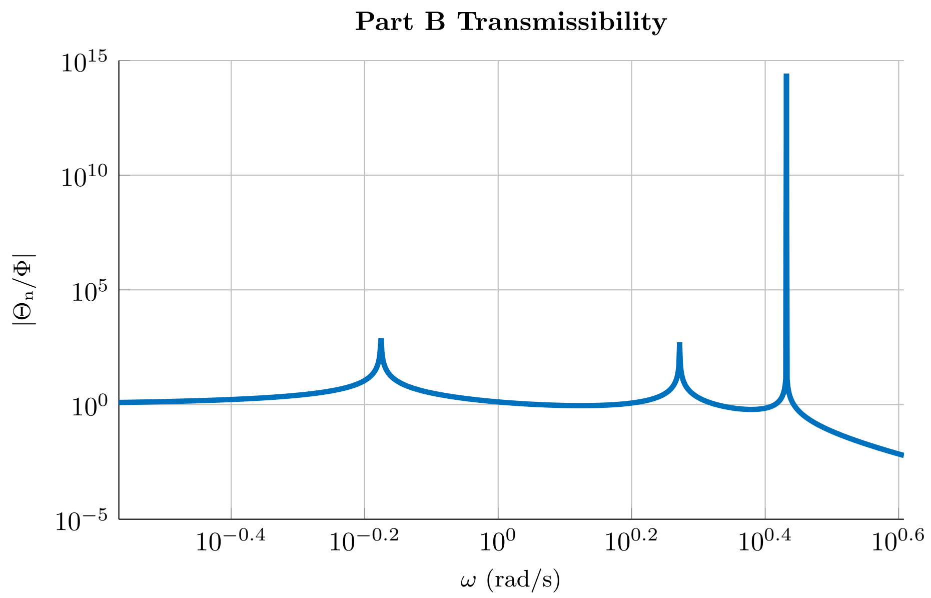

Figure 4: Undamped n=3

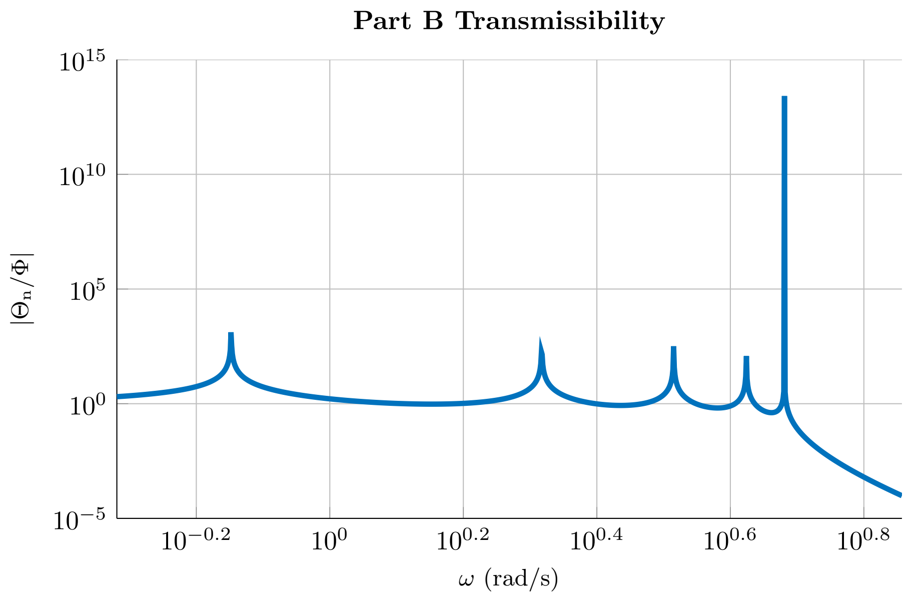

Figure 5: Undamped n=5

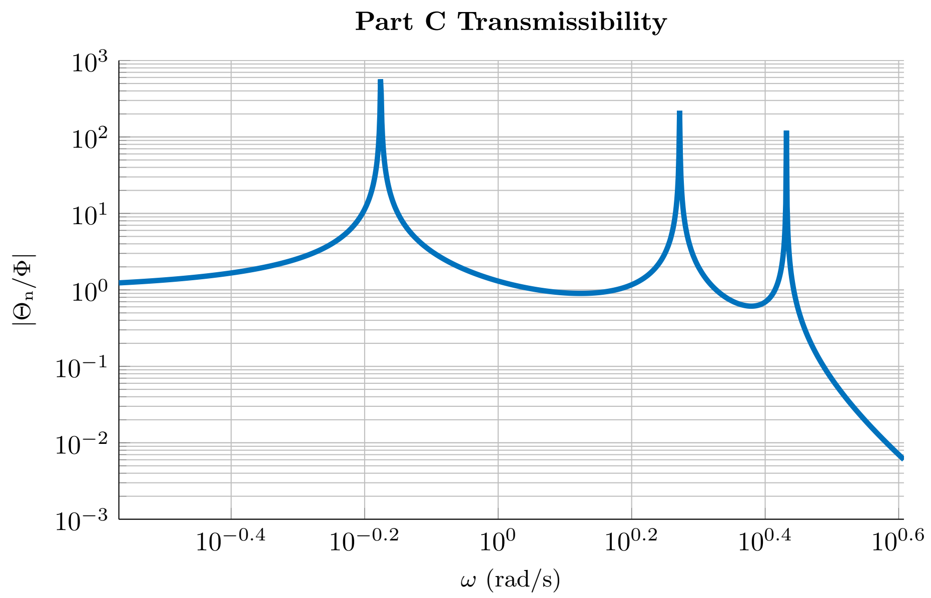

Figure 6: Absorbers at the First and Last Disks n=3 u=0.3

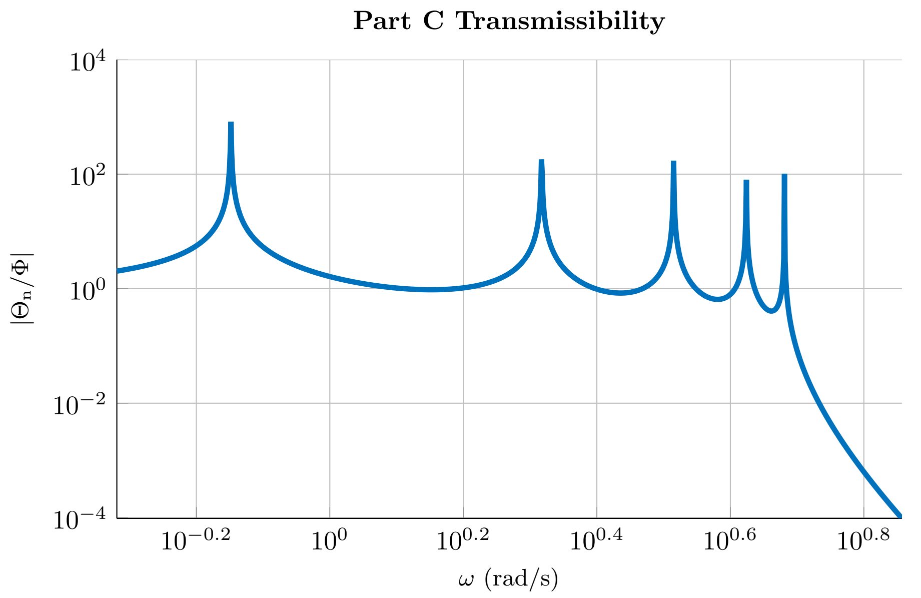

Figure 7: Absorbers at the First and Last Disks n=5 u=0.3

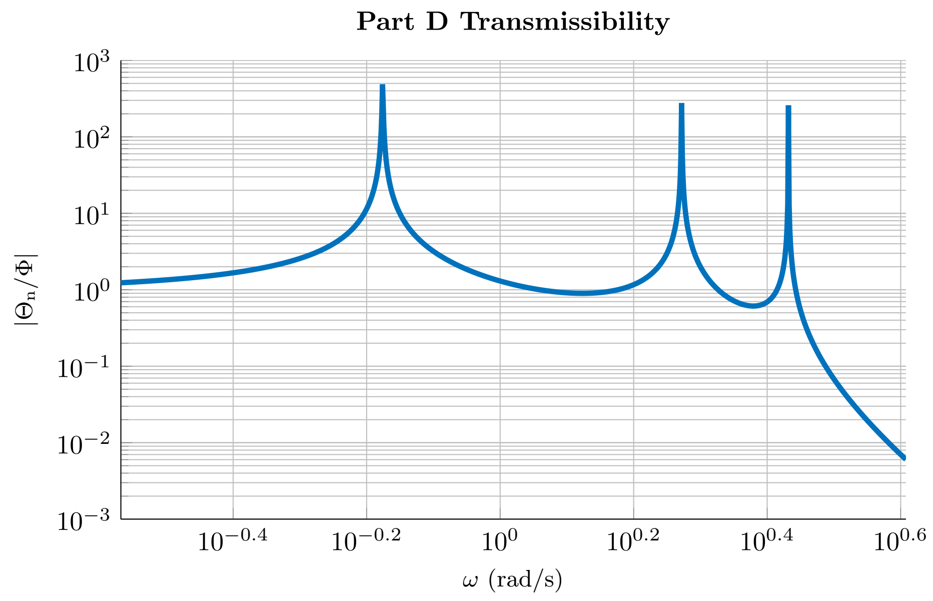

Figure 8: Freely Placed Absorbers n=3 u=0.3

axis as expected, and the peaks are close to each other in the magnitude but not exactly the same. There is a room for improvement if the constraint on the positions of the absorbers is removed. Figure 8 and Figure 9 show the solution of the last part, where the magnitude of the transmissibility peaks are very close to each other and the maximum transmissibility of the final design is lower as can be seen in Table 1.

In the last part, the inertia distribution between the absorbers is also freely decided by the optimizer. The result shows that all the inertia concentrated on the last disk. Basicly, the results says that all the vibration absorption effort should be spent on the disk which the transmissibility needs to be reduced. This is the expected answer. This is true regadless of the total viscosity. Comparing Figure 9 and Figure 10 shows this. In both cases all the absorber inertia is concentrated on the last disk. As the one with a bigger u has more absorber inertia the maximum transmissibility is smaller for that case as can be seen in Table 1.

In conclusion, the amount of disks between the input and output has less effect on the transmissibility compared to the absorber inertia. The disks in the middle just act as a channel for the energy to travel, the important design is having a absorber on the last disk that has a tuned damping coefficient. These can be read from Table 1.

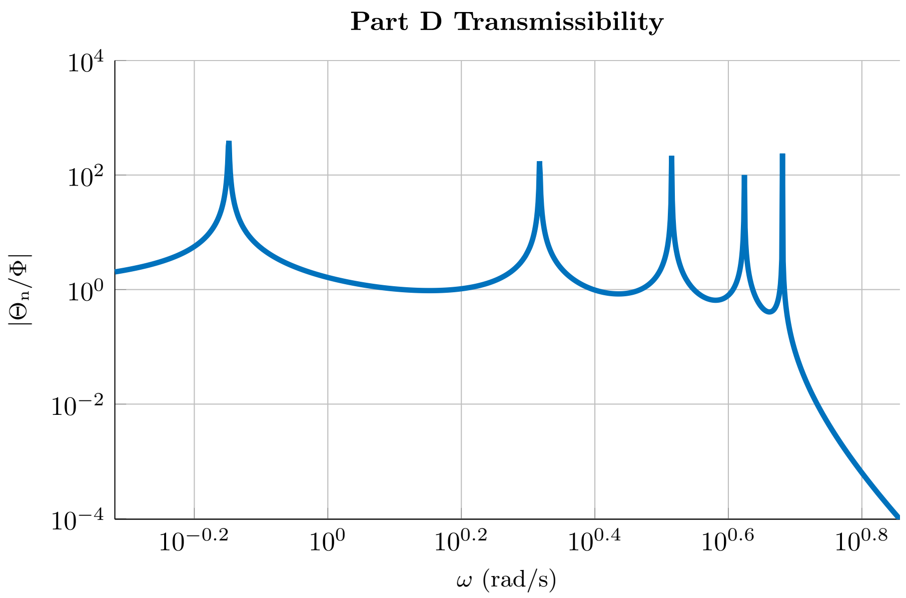

Figure 9: Freely Placed Absorbers n=5 u=0.3

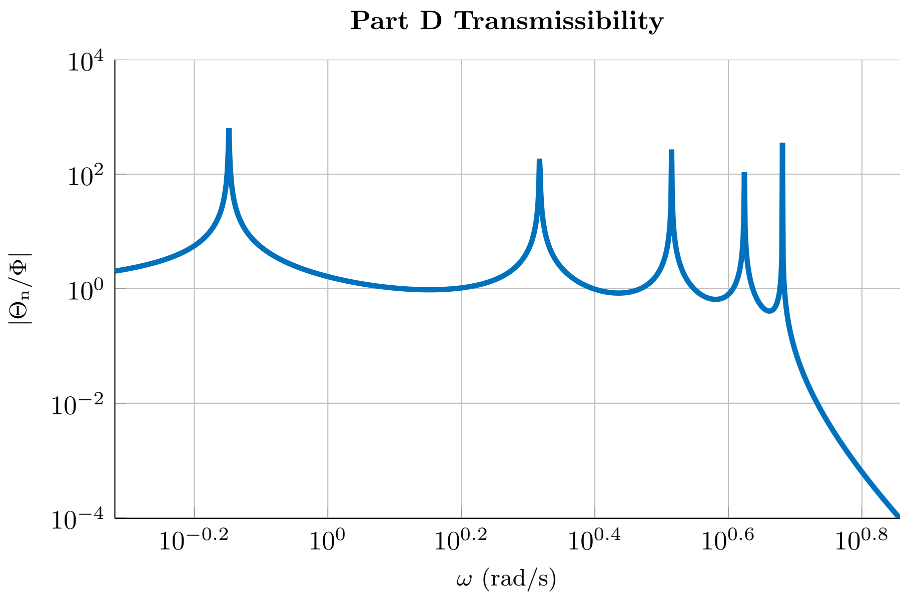

Figure 10: Freely Placed Absorbers n=5 u=0.2

Table 1: Maximum Transmissibilities

n	u	Part 2	Part 3	Part 4
5	0.3	12743.996	867.723	469.213
3	0.3	7441.508	834.714	500.153
5	0.2	12743.996	1299.804	703.329
4	0.2	8343.719	1284.744	720.641
5	0.1	12743.996	12739.701	1405.481

A Code

```
% me425 spring2022 project
% atalay gecgel sipahioglu

close('all');
clear();
clc();

print("me425 spring2022 project");
print("atalay gecgel sipahioglu");

% -------------------------------------------------------
% INPUT -------------------------------------------------
% -------------------------------------------------------

% Get the n and u from the user.

% Number of Disks (Min: 2, Max: 5)
n = 4;

% Total Houdaille Damper Viscosity (Min: 0.1, Max:
   0.3)
u = 0.1;

% Print
print("");
print("Input:");
print("~~~~~~");
print("[-] n = %4.0f", n);
print("[-] u = %4.2f", u);

% -------------------------------------------------------
% INITIAL CALCULATIONS ----------------------------------
% -------------------------------------------------------

% Find I and k.
% Note: using `n`, `u` from the previous part.
```

```
% Rotational Inertia of a Disk
I = 100 / n;

% Torsional Stiffness Between Disks
k = 25 * n;

% Print
print("");
print("Initial Caclculations:");
print("~~~~~~~~~~~~~~~~~~~~~~~");
print("[-] I = %5.1f", I);
print("[-] k = %5.1f", k);

% -----------------------------------------------------
% PART A ----------------------------------------------
% -----------------------------------------------------

% Find M and K.
% Use modal analysis to find the natural frequencies
  and mode shapes.
% Note: using `n`, `I`, `k` from the previous part.
% Note: using functions `f_M`, `f_K` that are defined
  at the end.

% Timer Start
c_start = tic();

% Inertia Matrix
M = f_M(n, 0, I, []);

% Stiffness Matrix
K = f_K(n, 0, k);

% First Transformation
M_ = M^(-1/2);
K_ = M_ * K * M_;

% Eigenvector and Eigenvalue Matrix
% Eigenvectors are the mode shapes.
% Eigenvalues are the squares of the natural
  frequencies.
[P, L] = eig(K_);

% Natural Frequencies in rad/s
w = zeros(n, 1);
for j = 1:n
    % Natural frequency is found by taking the square
      root of the eigenvalues.
    w(j) = L(j, j)^(1/2);
```

```
    % Mode shapes are found by making the first
       element of the eigenvectors positive.
    P(:, j) = P(1, j) / abs(P(1, j)) * P(:, j);
end

% Plot
figure();
tiledlayout(n, 1);
xlabel('n');
for j = 1:n
    nexttile();
    hold('on');
    grid('on');
    title(sprintf("\\omega_%.0f=%5.3f rad/s", j, w(j))
       );
    ylim([-1 1]);
    plot(1:n, P(:, j), '-o', 'LineWidth', 2);
    plot(1:n, zeros(1, n), 'k--', 'LineWidth', 2);
end

% Elapsed Time
c_elapsed = toc(c_start);

% Print
print("");
print("Part A:");
print("~~~~~~~~");
print("[-] Elapsed Time: %5.2f s", c_elapsed);
prmat("[-] M", M, "%9.3f");
prmat("[-] K", K, "%9.3f");
prmat("[-] M_", M_, "%9.3f");
prmat("[-] K_", K_, "%9.3f");
for j = 1:n
    print("[*] w_%1.0f = %5.3f rad/s", j, w(j));
    prvec(sprintf("[*] v_%1.0f", j), P(:, j), "%5.1f")
       ;
end

% ----------------------------------------------------
% PART B ---------------------------------------------
% ----------------------------------------------------

% Construct the range of excitation frequencies and
   find the transmissibilities
% using the repectance matrix.
% Note: using `w`, `n`, `M`, `K`, `k` from the
   previous part.
% Note: using functions `f_C`, `f_T_peaks` that are
   defined at the end.
```

```
% Timer Start
c_start = tic();

% Excitation Frequency Range
w_e_range = max(w) * (10.^(-1:0.001:log10(1.5)));

% Damping Matrix
% This is just a zero matrix in this part.
C = f_C(n, 0, [], []);

% Plot Transmissibility Range
plot_T_range(n, w_e_range, M, C, K, k, "Part B
   Transmissibility");

% Elapsed Time
c_elapsed = toc(c_start);

% Print
print("");
print("Part B:");
print("~~~~~~~~");
print("[-] Elapsed Time: %5.2f s", c_elapsed);
print("[-] T_max = %.3f", max(f_T_peaks(n, w, M, C, K,
    k)));

% ------------------------------------------------------
% PART C -----------------------------------------------
% ------------------------------------------------------

% Optimize the maximum of the peaks in the
   transmissibility for the damping
% coefficients of the absorbers which are connected to
    1 and 5.
% Note: using `u`, `n`, `I`, `k`, `w`, `w_e_range`
   from the previous part.
% Note: using functions `f_M`, `f_K`, `f_T_peaks`, `
   f_C` that are defined at the
% end.

% Timer Start
c_start = tic();

% Number of Absorbers
m = 2;

% Absorber Inertias
Ia = zeros(m, 1);
Ia(1) = u / 2;
Ia(2) = u / 2;
```

```
% Absorber Positions (Assumed to be unique for each
   absorber.)
na = zeros(m, 1);
na(1) = 1;
na(2) = n;

% Inertia Matrix
M = f_M(n, m, I, Ia);

% Stiffness Matrix
K = f_K(n, m, k);

% Optimization Parameter Vector
% [ca1, ca2]
f_ca = @(x) [x(1); x(2)];

% Initial Value
x_0 = [0.5, 0.5];

% Lower Bound
x_lb = [0, 0];

% Optimized Function
x_f = @(x) f_T_peaks(n, w, M, f_C(n, m, na, f_ca(x)),
   K, k);

% Optimization Options
x_options = optimoptions('fminimax');
x_options.MaxIterations = 100;
x_options.MaxFunctionEvaluations = 1000;
x_options.Display = 'off';

% Optimization Results
[x, ~, x_maxfval] = fminimax(x_f, x_0, [], [], [], [],
    x_lb, [], [], x_options);

% Absorber Dampings
ca = f_ca(x);

% Minimized Maximum Transmissibility
T_min = x_maxfval;

% Damping Matrix
C = f_C(n, m, na, ca);

% Plot Transmissibility Range
plot_T_range(n, w_e_range, M, C, K, k, "Part C
   Transmissibility");

% Elapsed Time
```

```
c_elapsed = toc(c_start);

% Print
print("");
print("Part C:");
print("~~~~~~~~");
print("[-] Elapsed Time: %5.2f s", c_elapsed);
prvec("[-] Ia", Ia, "%7.3f");
prvec("[-] na", na, "%7.0f");
prmat("[-] M", M, "%9.3f");
prmat("[-] K", K, "%9.3f");
prvec("[-] x_0", x_0, "%7.3f");
prvec("[-] x", x, "%7.3f");
prvec("[-] ca", ca, "%7.3f");
prmat("[-] C", C, "%9.3f");
print("[-] T_max = %.3f", T_min);

% ---------------------------------------------------------
% PART D ---------------------------------------------------
% ---------------------------------------------------------

% Optimize all the possible combinations of the
   absorber positions over the
% damping coefficients and absorber inertias.
% Select the one with the smallest transmissibility.
% Note: using `m`, `n`, `w`, `u`, `I`, `k`, `w_e_range
   ` from the previous part.
% Note: using functions `f_T_min`, `f_M`, `f_C`, `f_K`
    that are defined at the
% end.

% Timer Start
c_start = tic();

% Absorber Positions (Assumed to be unique for each
   absorber.)
na = zeros(m, 1);

% Absorber Inertias
Ia = zeros(m, 1);

% Absorber Dampings
ca = zeros(m, 1);

% Minimum Maximum Transmissibility
T_min = Inf;

% For all possible combinations...
na_combinations = nchoosek(1:n, m);
for j = size(na_combinations, 1)
```

```
    na_j = na_combinations(j, :);

    % Optimize
    [Ia_j, ca_j, T_min_j] = f_T_min(n, m, w, na_j, u,
       I, k);

    % ... replace if better.
    if T_min > T_min_j
        na = na_j;
        Ia = Ia_j;
        ca = ca_j;
        T_min = T_min_j;
    end
end

if ~isinf(T_min)
    % Inertia Matrix
    M = f_M(n, m, I, Ia);

    % Damping Matrix
    C = f_C(n, m, na, ca);

    % Stiffness Matrix
    K = f_K(n, m, k);

    % Plot Transmissibility Range
    plot_T_range(n, w_e_range, M, C, K, k, "Part D
       Transmissibility");
end

% Elapsed Time
c_elapsed = toc(c_start);

% Print
print("");
print("Part D:");
print("~~~~~~~~");
print("[-] Elapsed Time: %5.2f s", c_elapsed);
prvec("[-] na", na, "%7.0f");
prvec("[-] Ia", Ia, "%7.3f");
prvec("[-] ca", ca, "%7.3f");
if ~isinf(T_min)
    prmat("[-] M", M, "%9.3f");
    prmat("[-] C", C, "%9.3f");
    prmat("[-] K", K, "%9.3f");
    print("[-] T_max = %.3f", T_min);
else
    print("[!] Could not found even a single finite
       solution!");
end
```

```
% -----------------------------------------------------
% CONSTRUCTION FUNCTIONS ------------------------------
% -----------------------------------------------------

% For creating the inertia matrix. In the case with no
    absorbers give `m` as
% `0`, `Ia` as `[]`.
function M = f_M(n, m, I, Ia)
    % Inertia Matrix
    M = zeros(n + m);
    for j = 1:n
        M(j, j) = I;
    end
    for j = 1:m
        M(n + j, n + j) = Ia(j);
    end
end

% For creating the damping matrix. In the case with no
    absorbers give `m` as
% `0`, `na` and `ca` as `[]`. The resulting matrix
   will be just zeros.
function C = f_C(n, m, na, ca)
    % Damping Matrix
    C = zeros(n + m);
    for j = 1:m
        C(n + j, n + j) = ca(j);
        C(n + j, na(j)) = -ca(j);
        C(na(j), n + j) = -ca(j);
        C(na(j), na(j)) = ca(j);
    end
end

% For creating the stiffness matrix. In the case with
   no absorbers give `m` as
% `0`.
function K = f_K(n, m, k)
    % Stiffness Matrix
    K = zeros(n + m);
    for j = 1:n
        if j > 1
            K(j, j - 1) = -k;
        end
        if j < n
            K(j, j + 1) = -k;
            K(j, j) = 2 * k;
        else
            K(j, j) = k;
        end
    end
end

% -----------------------------------------------------
% TRANSMISSIBILITY FUNCTIONS -------------------------
% -----------------------------------------------------

% For designing the absorbers in part D. Minimizes the
    maximum of the peaks
% using `fminimax` and returns the optimum parameters.
    Only needs to know the
% positions of the absorbers. The combination of all
   possible absorber positions
% can be iterated over to get the best desing. The
   total absorber inertia is
% equated to viscosity by giving a linear equality
   constraint to `fminimax`.
function [Ia, ca, T_min] = f_T_min(n, m, w, na, u, I,
   k)
    % Stiffness Matrix
    K = f_K(n, m, k);

    % Optimization Parameter Vector
    % [Ia1, Ia2, ca1, ca2]
    f_Ia = @(x) [x(1); x(2)];
    f_ca = @(x) [x(3); x(4)];

    % Initial Value
    x_0 = [u / 2, u / 2, 0.5, 0.5];

    % Lower Bound
    x_lb = [0, 0, 0, 0];

    % Optimized Function
    x_f = @(x) f_T_peaks(n, w, f_M(n, m, I, f_Ia(x)),
       f_C(n, m, na, f_ca(x)), K, k);

    % Linear Equality Constraint
    x_Aeq = [1, 1, 0, 0];
    x_beq = u;

    % Optimization Options
    x_options = optimoptions('fminimax');
    x_options.MaxIterations = 100;
    x_options.MaxFunctionEvaluations = 1000;
    x_options.Display = 'off';

    % Optimization Results
    [x, ~, x_maxfval] = fminimax(x_f, x_0, [], [],
       x_Aeq, x_beq, x_lb, [], [], x_options);
    % Absorber Inertias
    Ia = f_Ia(x);

    % Absorber Dampings
    ca = f_ca(x);

    % Minimized Maximum Transmissibility
    T_min = x_maxfval;
end

% For finding the peaks in the transmissibility plot
   very fast and secure. It
% always returns the peaks because it uses the natural
    frequencies. With the
% damping the peaks shift to left slightly. Looks for
   the peaks via iterating
% over the natural frequencies and using `fminbnd`
   starting from the 95% of the
% natural frequency to 100% of it. Fast because it
   looks for a very small
% window. Guaranteed to find the peaks because it does
    not do grid search.
function T_peaks = f_T_peaks(n, w, M, C, K, k)
    % Transmissibility Peaks
    T_peaks = zeros(1, n);
    for j = 1:n
        % Optimization Parameter Vector
        % [w_e]
        % Lower Bound
        x_lb = w(j) * 0.95;

        % Upper Bound
        x_ub = w(j);

        % Optimized Function
        x_f = @(x) -f_T(n, x, M, C, K, k);

        % Optimization Options
        x_options = optimset('fminbnd');
        x_options.MaxIterations = 100;
        x_options.MaxFunctionEvaluations = 1000;
        x_options.Display = 'off';

        % Optimization Results
        [~, x_fval] = fminbnd(x_f, x_lb, x_ub,
            x_options);

        % Maximum Transmissibility
        T_peaks(j) = -x_fval;
    end
end

% For calculating a range of transmissibilities for
   the given range of
% excitation frequencies.
function T_range = f_T_range(n, w_e_range, M, C, K, k)
    % Transmissibility Range
    T_range = zeros(size(w_e_range));
    for j = 1:length(w_e_range)
        T_range(j) = f_T(n, w_e_range(j), M, C, K, k);
    end
end

% For calculating the transmissibility of the last
   disk for the given excitation
% frequency. Very fast because it only uses the
   necessary element of the
% receptance matrix.
function T = f_T(n, w_e, M, C, K, k)
    % Receptance Matrix
    a = (-w_e^2 * M + 1i * w_e * C + K)^ - 1;

    % Transmissibility
    T = abs(a(1, n) * k);
end

% ----------------------------------------------------
% OUTPUT FUNCTIONS -----------------------------------
% ----------------------------------------------------

% For plotting the transmissibility over a range of
   excitation frequencies.
function plot_T_range(n, w_e_range, M, C, K, k, name)
    % Plot
    figure();
    set(gca, 'YScale', 'log');
    set(gca, 'XScale', 'log');
    hold('on');
    grid('on');
    xlabel('\omega (rad/s)');
    ylabel('|\Theta_n/\Phi|');
    plot(w_e_range, f_T_range(n, w_e_range, M, C, K, k
       ), 'LineWidth', 2);
    title(name);
end

% For easier general printing. Puts new line at the
   start.
function print(varargin)
```

```
    fprintf('%s\n', sprintf(varargin{:}));
end

% For printing matrices.
function prmat(name, matrix, element)
    print("%s [%.0f, %.0f]: ", name, size(matrix, 1),
       size(matrix, 2));
    for k = 1:size(matrix, 1)
        for j = 1:size(matrix, 2)
            fprintf(element, matrix(k, j));
        end
        fprintf("\n");
    end
end

% For printing vectors.
function prvec(name, vector, element)
    fprintf("%s [%.0f]: ", name, length(vector));
    for k = 1:length(vector)
        fprintf(element, vector(k));
    end
    fprintf("\n");
end
```