Nano-Hexapod - Test Bench

Dehaeze Thomas

June 14, 2021

Contents

This document is dedicated to the experimental study of the nano-hexapod shown in Figure 0.1.

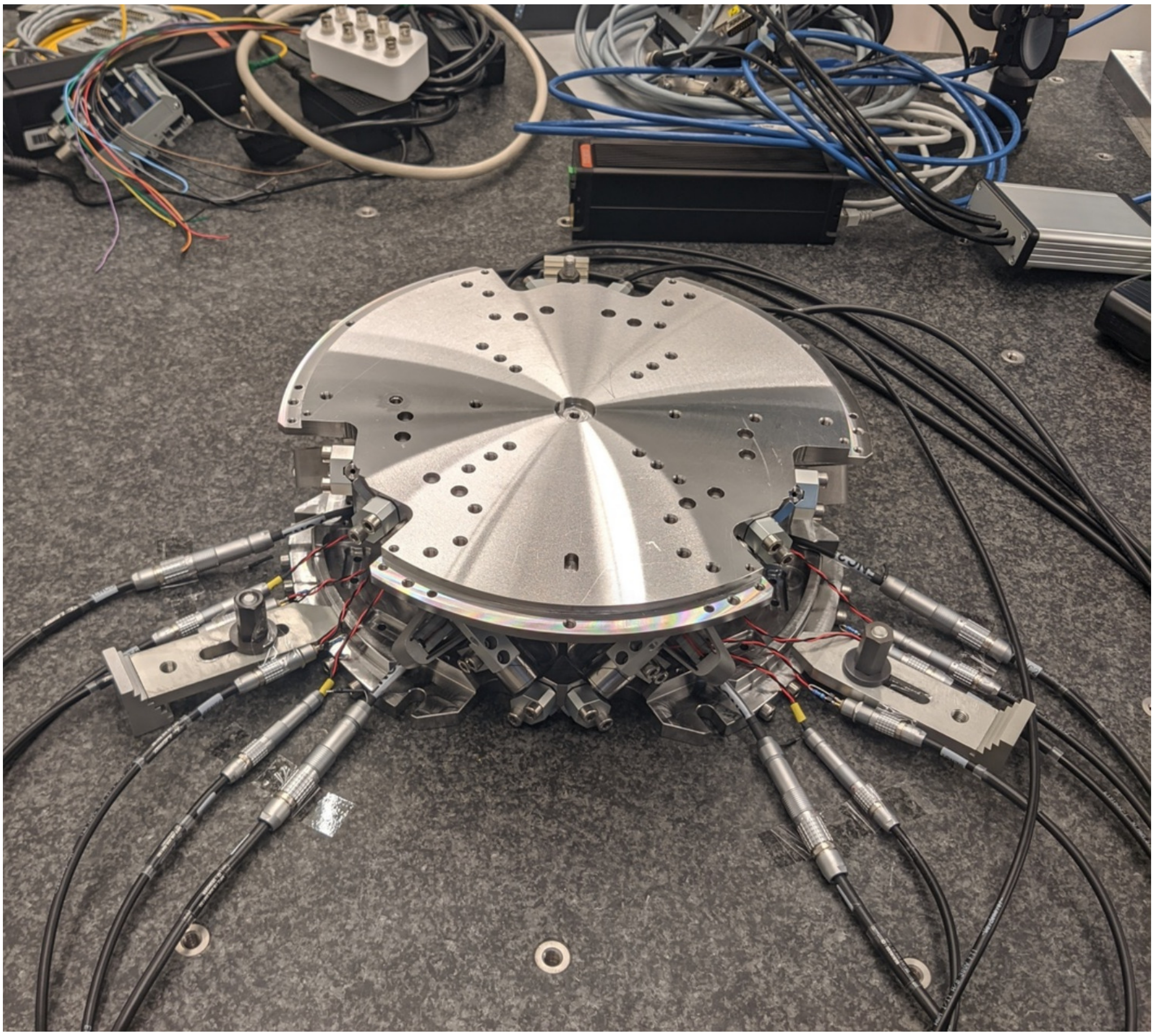

Figure 0.1: Nano-Hexapod

Note

Here are the documentation of the equipment used for this test bench (lots of them are shwon in Figure 0.2):

- Voltage Amplifier: PiezoDrive PD200
- Amplified Piezoelectric Actuator: Cedrat APA300ML
- DAC/ADC: Speedgoat IO313
- Encoder: Renishaw Vionic and used Ruler
- Interferometers: Attocube

In Figure 0.3 is shown a block diagram of the experimental setup. When possible, the notations are consistent with this diagram and summarized in Table 0.1.

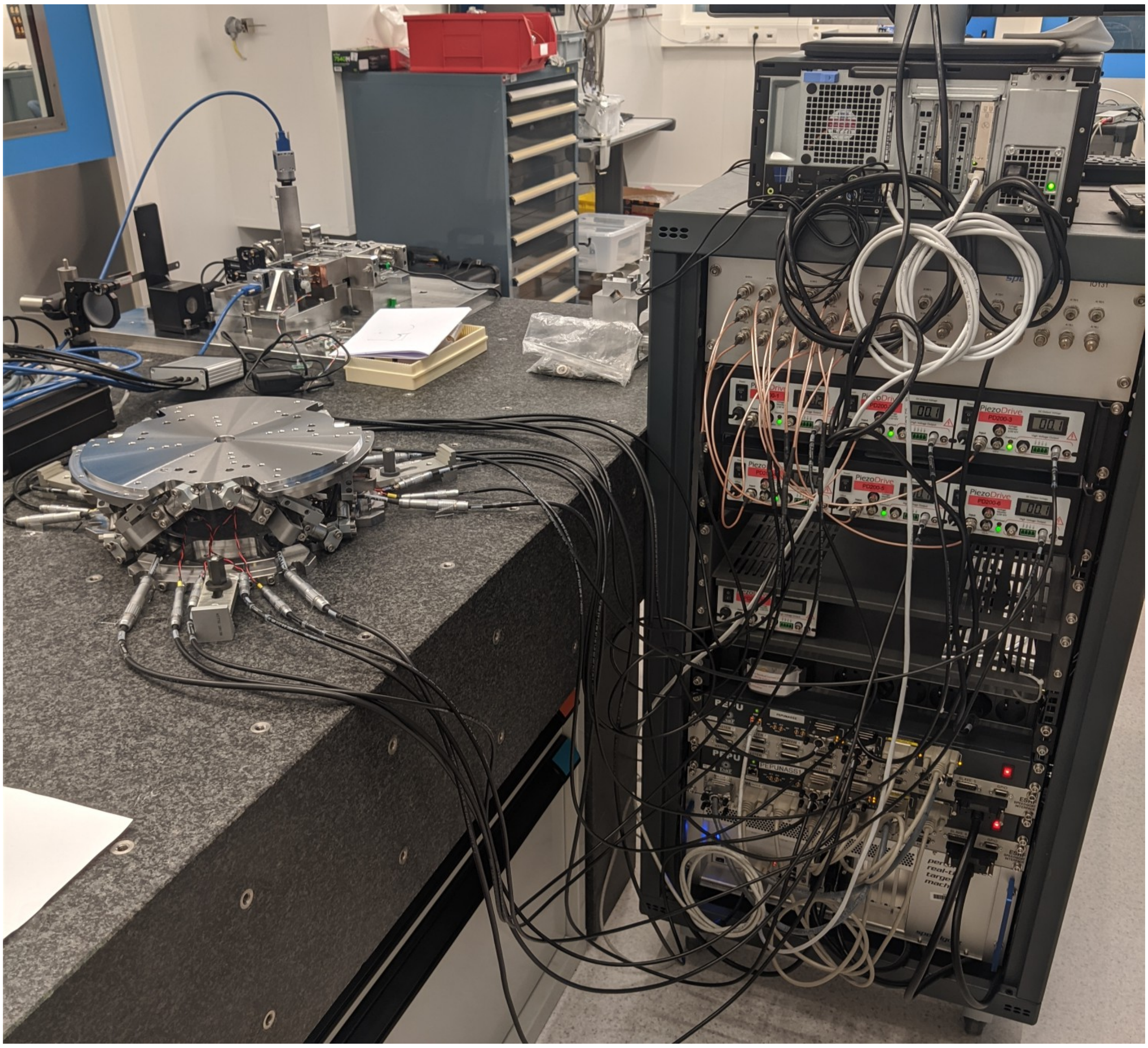

Figure 0.2: Nano-Hexapod and the control electronics

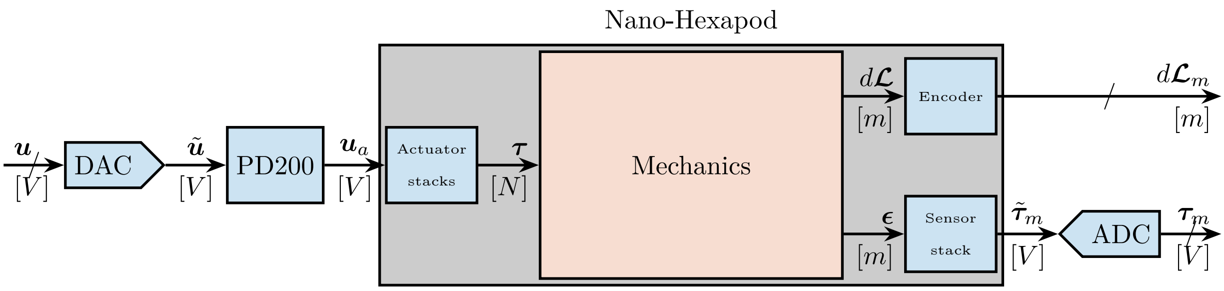

Figure 0.3: Block diagram of the system with named signals

Table 0.1: List of signals

	Unit	Matlab	Vector	Elements
Control Input (wanted DAC voltage)	`[V]`	`u`	$\boldsymbol{u}$	u_i
DAC Output Voltage	`[V]`	`u`	$\tilde{\boldsymbol{u}}$	$\tilde{u}_i$
PD200 Output Voltage	`[V]`	`ua`	$\boldsymbol{u}_a$	$u_{a,i}$
Actuator applied force	`[N]`	`tau`	$\boldsymbol{\tau}$	τ_i
Strut motion	`[m]`	`dL`	$d\boldsymbol{\mathcal{L}}$	$d\mathcal{L}_i$
Encoder measured displacement	`[m]`	`dLm`	$d\boldsymbol{\mathcal{L}}_m$	$d\mathcal{L}_{m,i}$
Force Sensor strain	`[m]`	`epsilon`	$\boldsymbol{\epsilon}$	ϵ_i
Force Sensor Generated Voltage	`[V]`	`taum`	$\tilde{\boldsymbol{\tau}}_m$	$\tilde{\tau}_{m,i}$
Measured Generated Voltage	`[V]`	`taum`	$\boldsymbol{\tau}_m$	$\tau_{m,i}$
Motion of the top platform	`[m,rad]`	`dX`	$d\boldsymbol{\mathcal{X}}$	$d\mathcal{X}_i$
Metrology measured displacement	`[m,rad]`	`dXm`	$d\boldsymbol{\mathcal{X}}_m$	$d\mathcal{X}_{m,i}$

This document is divided in the following sections:

- Section 1: the encoders are fixed to the struts
- Section 2: the encoders are fixed to the plates

1 Encoders fixed to the Struts

1.1 Introduction

In this section, the encoders are fixed to the struts.

It is divided in the following sections:

- Section 1.2: the transfer function matrix from the actuators to the force sensors and to the encoders is experimentally identified.
- Section 1.3: the obtained FRF matrix is compared with the dynamics of the simscape model
- Section 1.4: decentralized Integral Force Feedback (IFF) is applied and its performances are evaluated.
- Section 1.5: a modal analysis of the nano-hexapod is performed

1.2 Identification of the dynamics

1.2.1 Load Data

Matlab
```matlab
%% Load Identification Data
meas_data_lf = {};

for i = 1:6
    meas_data_lf(i) = {load(sprintf('mat/frf_data_exc_strut_%i_noise_lf.mat', i), 't', 'Va', 'Vs', 'de')};
    meas_data_hf(i) = {load(sprintf('mat/frf_data_exc_strut_%i_noise_hf.mat', i), 't', 'Va', 'Vs', 'de')};
end
```

1.2.2 Spectral Analysis - Setup

Matlab
```matlab
%% Setup useful variables
% Sampling Time [s]
Ts = (meas_data_lf{1}.t(end) - (meas_data_lf{1}.t(1)))/(length(meas_data_lf{1}.t)-1);

% Sampling Frequency [Hz]
Fs = 1/Ts;

% Hannning Windows
win = hanning(ceil(1*Fs));
```

```
% And we get the frequency vector
[~, f] = tfestimate(meas_data_lf{1}.Va, meas_data_lf{1}.de, win, [], [], 1/Ts);

i_lf = f < 250; % Points for low frequency excitation
i_hf = f > 250; % Points for high frequency excitation
```

1.2.3 DVF Plant

First, let's compute the coherence from the excitation voltage and the displacement as measured by the encoders (Figure 1.1).

Matlab
```
%% Coherence
coh_dvf_lf = zeros(length(f), 6, 6);
coh_dvf_hf = zeros(length(f), 6, 6);

for i = 1:6
    coh_dvf_lf(:, :, i) = mscohere(meas_data_lf{i}.Va, meas_data_lf{i}.de, win, [], [], 1/Ts);
    coh_dvf_hf(:, :, i) = mscohere(meas_data_hf{i}.Va, meas_data_hf{i}.de, win, [], [], 1/Ts);
end
```

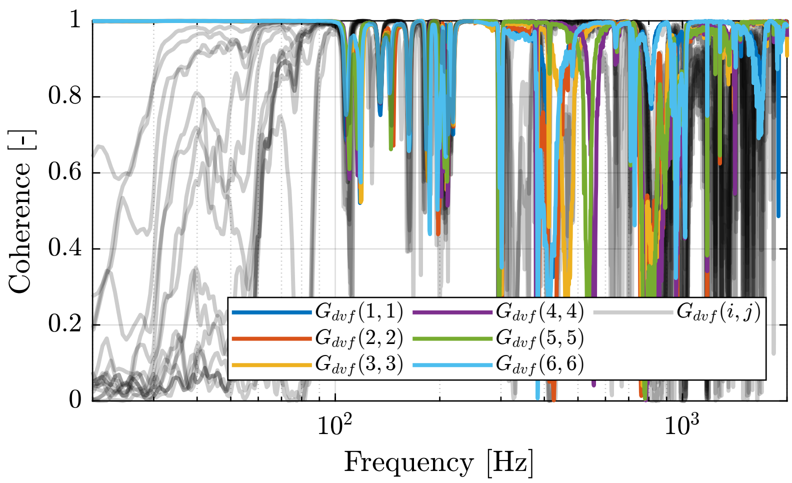

Figure 1.1: Obtained coherence for the DVF plant

Then the 6x6 transfer function matrix is estimated (Figure 1.2).

Matlab
```
%% DVF Plant (transfer function from u to dLm)
G_dvf_lf = zeros(length(f), 6, 6);
G_dvf_hf = zeros(length(f), 6, 6);

for i = 1:6
    G_dvf_lf(:, :, i) = tfestimate(meas_data_lf{i}.Va, meas_data_lf{i}.de, win, [], [], 1/Ts);
    G_dvf_hf(:, :, i) = tfestimate(meas_data_hf{i}.Va, meas_data_hf{i}.de, win, [], [], 1/Ts);
end
```

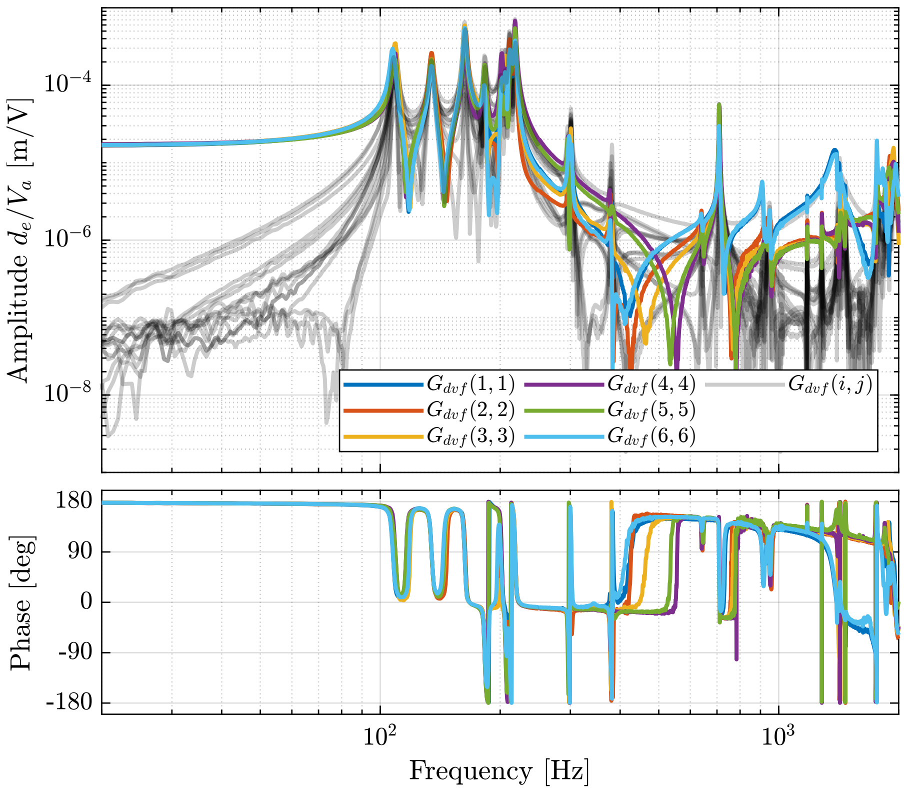

Figure 1.2: Measured FRF for the DVF plant

1.2.4 IFF Plant

First, let's compute the coherence from the excitation voltage and the displacement as measured by the encoders (Figure 1.3).

Matlab

```matlab
%% Coherence for the IFF plant
coh_iff_lf = zeros(length(f), 6, 6);
coh_iff_hf = zeros(length(f), 6, 6);

for i = 1:6
    coh_iff_lf(:, :, i) = mscohere(meas_data_lf{i}.Va, meas_data_lf{i}.Vs, win, [], [], 1/Ts);
    coh_iff_hf(:, :, i) = mscohere(meas_data_hf{i}.Va, meas_data_hf{i}.Vs, win, [], [], 1/Ts);
end
```

Then the 6x6 transfer function matrix is estimated (Figure 1.4).

Matlab

```matlab
%% IFF Plant
G_iff_lf = zeros(length(f), 6, 6);
G_iff_hf = zeros(length(f), 6, 6);

for i = 1:6
    G_iff_lf(:, :, i) = tfestimate(meas_data_lf{i}.Va, meas_data_lf{i}.Vs, win, [], [], 1/Ts);
    G_iff_hf(:, :, i) = tfestimate(meas_data_hf{i}.Va, meas_data_hf{i}.Vs, win, [], [], 1/Ts);
end
```

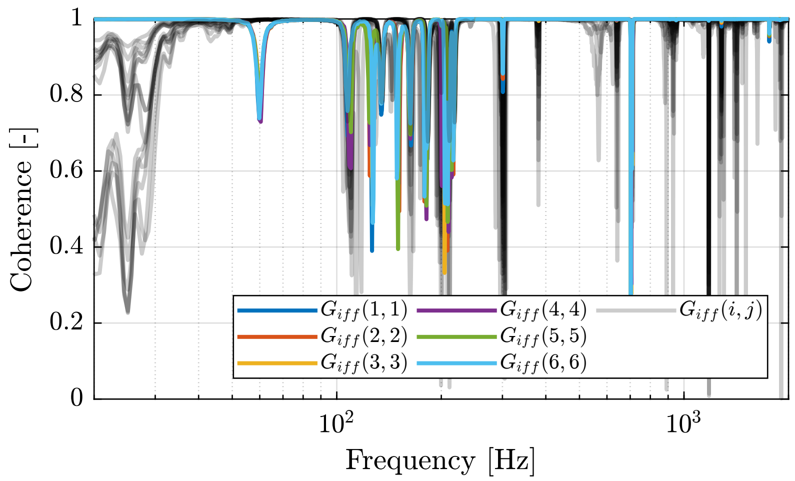

Figure 1.3: Obtained coherence for the IFF plant

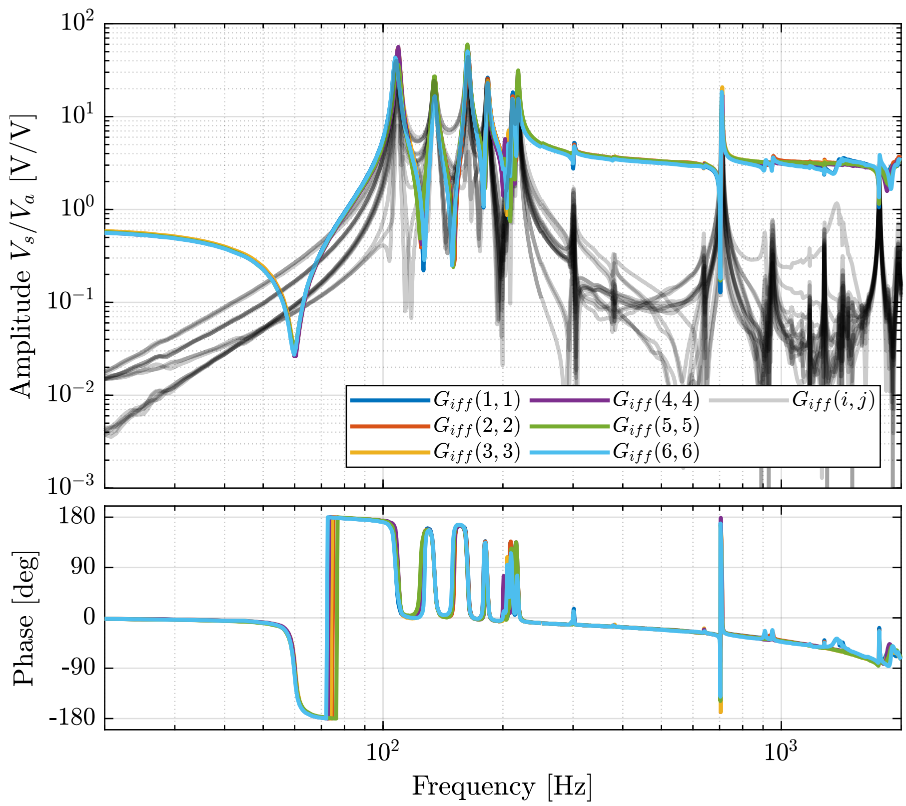

Figure 1.4: Measured FRF for the IFF plant

1.3 Comparison with the Simscape Model

In this section, the measured dynamics is compared with the dynamics estimated from the Simscape model.

1.3.1 Dynamics from Actuator to Force Sensors

Matlab
```
%% Initialize Nano-Hexapod
n_hexapod = initializeNanoHexapodFinal('flex_bot_type', '4dof', ...
                                       'flex_top_type', '4dof', ...
                                       'motion_sensor_type', 'struts', ...
                                       'actuator_type', '2dof');
```

Matlab
```
%% Identify the IFF Plant (transfer function from u to taum)
clear io; io_i = 1;
io(io_i) = linio([mdl, '/F'],  1, 'openinput');   io_i = io_i + 1; % Actuator Inputs
io(io_i) = linio([mdl, '/Fm'],  1, 'openoutput'); io_i = io_i + 1; % Force Sensors

Giff = exp(-s*Ts)*linearize(mdl, io, 0.0, options);
```

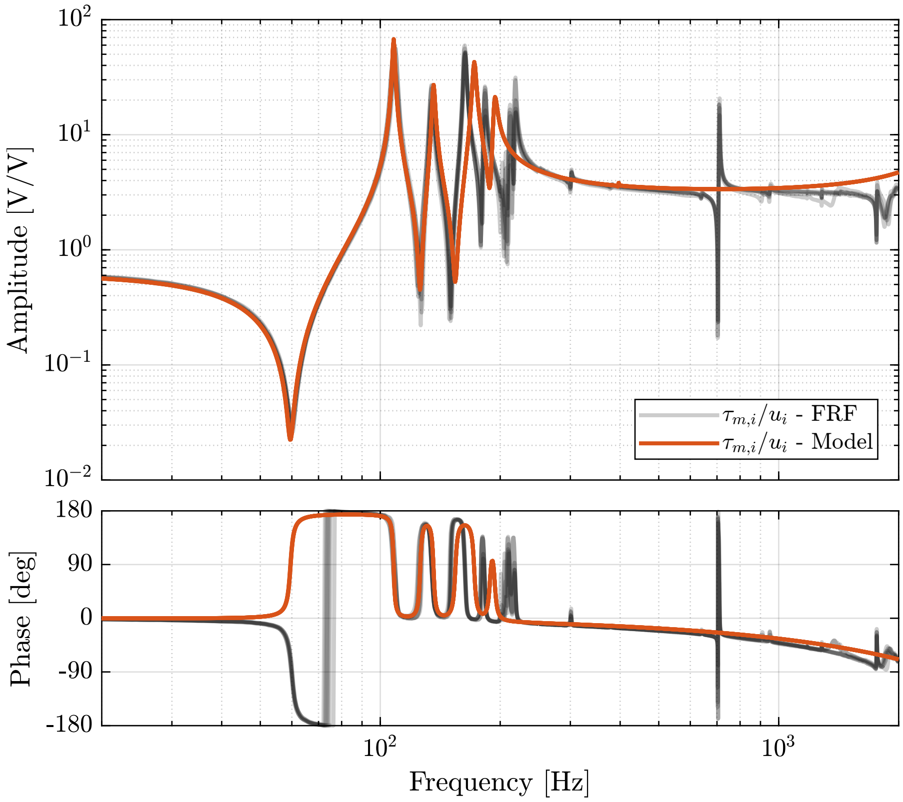

Figure 1.5: Diagonal elements of the IFF Plant

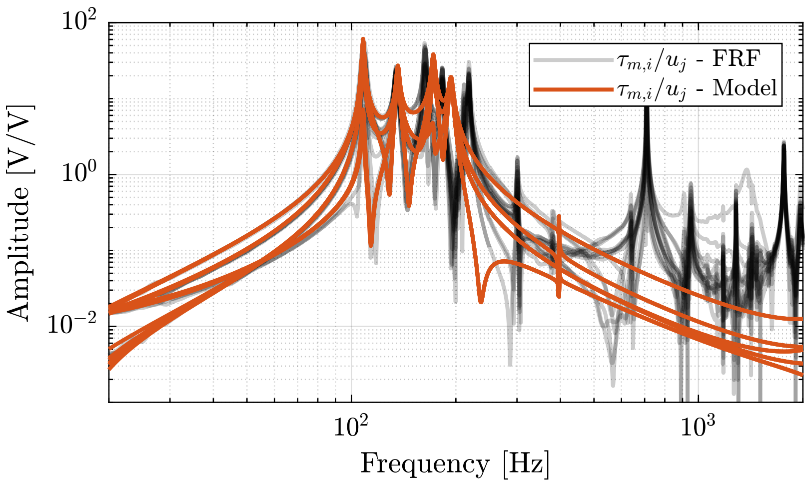

Figure 1.6: Off diagonal elements of the IFF Plant

1.3.2 Dynamics from Actuator to Encoder

Matlab
```
%% Initialization of the Nano-Hexapod
n_hexapod = initializeNanoHexapodFinal('flex_bot_type', '4dof', ...
                                       'flex_top_type', '4dof', ...
                                       'motion_sensor_type', 'struts', ...
                                       'actuator_type', '2dof');
```

Matlab
```
%% Identify the DVF Plant (transfer function from u to dLm)
clear io; io_i = 1;
io(io_i) = linio([mdl, '/F'],  1, 'openinput');  io_i = io_i + 1; % Actuator Inputs
io(io_i) = linio([mdl, '/D'],  1, 'openoutput'); io_i = io_i + 1; % Encoders

Gdvf = exp(-s*Ts)*linearize(mdl, io, 0.0, options);
```

1.4 Integral Force Feedback

1.4.1 Root Locus and Decentralized Loop gain

Matlab
```
%% IFF Controller
Kiff_g1 = (1/(s + 2*pi*40))*... % Low pass filter (provides integral action above 40Hz)
          (s/(s + 2*pi*30))*... % High pass filter to limit low frequency gain
          (1/(1 + s/2/pi/500))*... % Low pass filter to be more robust to high frequency resonances
          eye(6); % Diagonal 6x6 controller
```

Then the “optimal” IFF controller is:

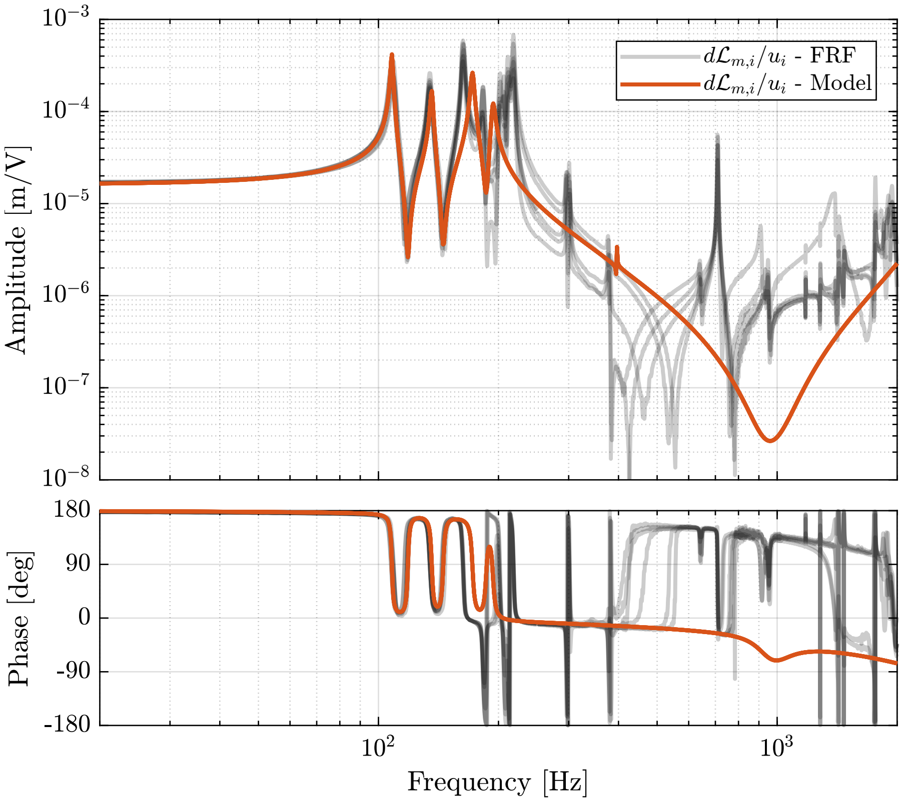

Figure 1.7: Diagonal elements of the DVF Plant

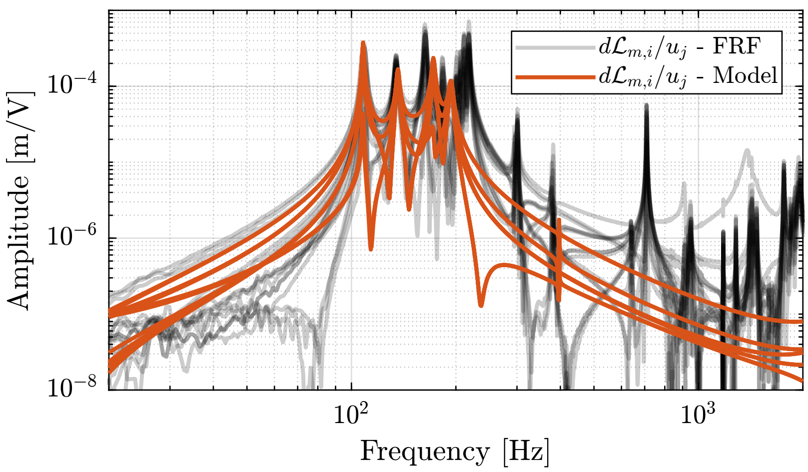

Figure 1.8: Off diagonal elements of the DVF Plant

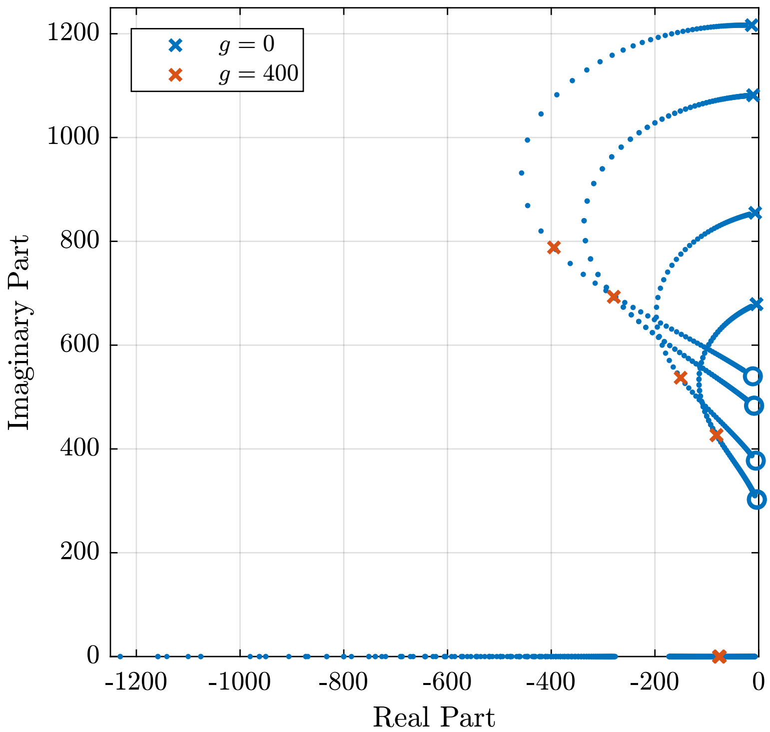

Figure 1.9: Root Locus for the IFF control strategy

Matlab
```
%% IFF controller with Optimal gain
Kiff = g*Kiff_g1;
```

1.4.2 Multiple Gains - Simulation

Matlab
```
%% Tested IFF gains
iff_gains = [4, 10, 20, 40, 100, 200, 400];
```

Matlab
```
%% Initialize the Simscape model in closed loop
n_hexapod = initializeNanoHexapodFinal('flex_bot_type', '4dof', ...
                                       'flex_top_type', '4dof', ...
                                       'motion_sensor_type', 'struts', ...
                                       'actuator_type', '2dof', ...
                                       'controller_type', 'iff');
```

Matlab
```
%% Identify the (damped) transfer function from u to dLm for different values of the IFF gain
Gd_iff = {zeros(1, length(iff_gains))};

clear io; io_i = 1;
io(io_i) = linio([mdl, '/F'],  1, 'openinput');  io_i = io_i + 1; % Actuator Inputs
io(io_i) = linio([mdl, '/D'],  1, 'openoutput'); io_i = io_i + 1; % Strut Displacement (encoder)
```

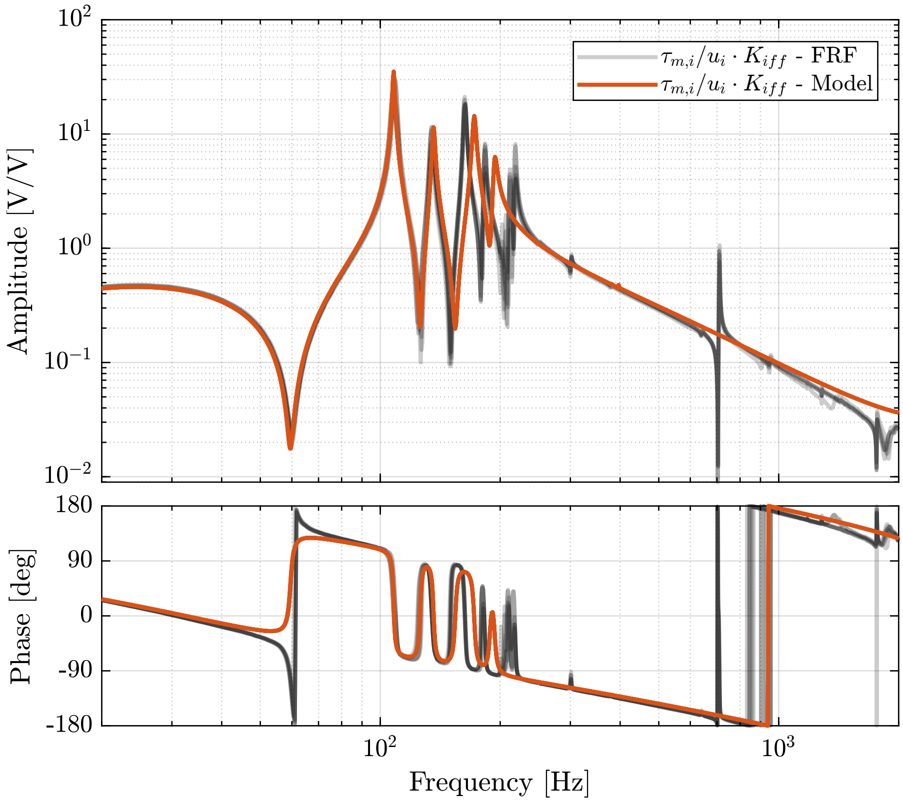

Figure 1.10: Bode plot of the "decentralized loop gain" $G_{\text{iff}}(i,i) \times K_{\text{iff}}(i,i)$

```
for i = 1:length(iff_gains)
    Kiff = iff_gains(i)*Kiff_g1*eye(6); % IFF Controller
    Gd_iff(i) = {exp(-s*Ts)*linearize(mdl, io, 0.0, options)};

    isstable(Gd_iff{i})
end
```

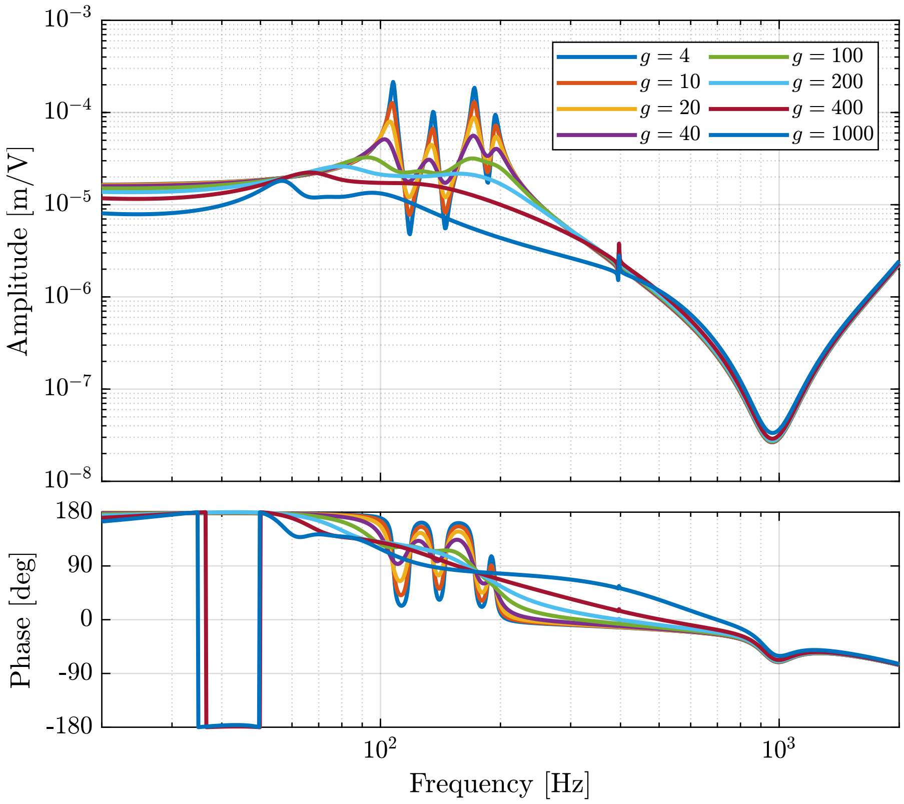

Figure 1.11: Effect of the IFF gain g on the transfer function from $\boldsymbol{\tau}$ to $d\boldsymbol{\mathcal{L}}_m$

1.4.3 Experimental Results - Gains

Let's look at the damping introduced by IFF as a function of the IFF gain and compare that with the results obtained using the Simscape model.

Load Data

```matlab
%% Load Identification Data
meas_iff_gains = {};

for i = 1:length(iff_gains)
    meas_iff_gains(i) = {load(sprintf('mat/iff_strut_1_noise_g_%i.mat', iff_gains(i)), 't', 'Vexc', 'Vs', 'de', 'u')};
end
```

Spectral Analysis - Setup

```matlab
%% Setup useful variables
% Sampling Time [s]
Ts = (meas_iff_gains{1}.t(end) - (meas_iff_gains{1}.t(1)))/(length(meas_iff_gains{1}.t)-1);

% Sampling Frequency [Hz]
Fs = 1/Ts;

% Hannning Windows
win = hanning(ceil(1*Fs));

% And we get the frequency vector
[~, f] = tfestimate(meas_iff_gains{1}.Vexc, meas_iff_gains{1}.de, win, [], [], 1/Ts);
```

DVF Plant

```matlab
%% DVF Plant (transfer function from u to dLm)
G_iff_gains = {};

for i = 1:length(iff_gains)
    G_iff_gains{i} = tfestimate(meas_iff_gains{i}.Vexc, meas_iff_gains{i}.de(:,1), win, [], [], 1/Ts);
end
```

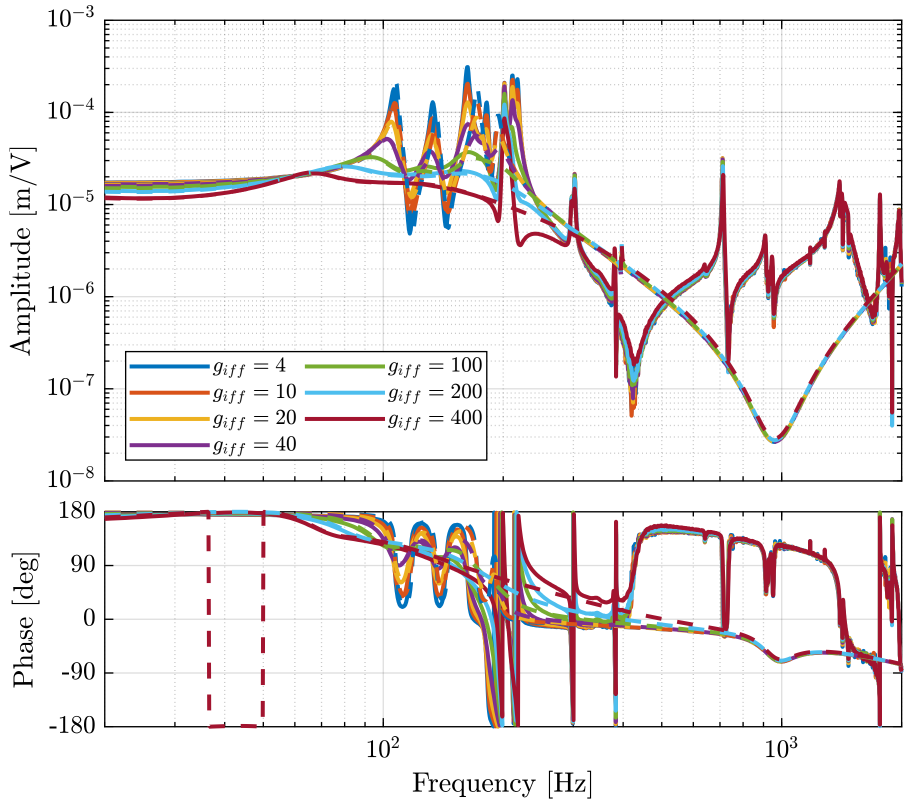

Figure 1.12: Transfer function from u to $d\mathcal{L}_m$ for multiple values of the IFF gain

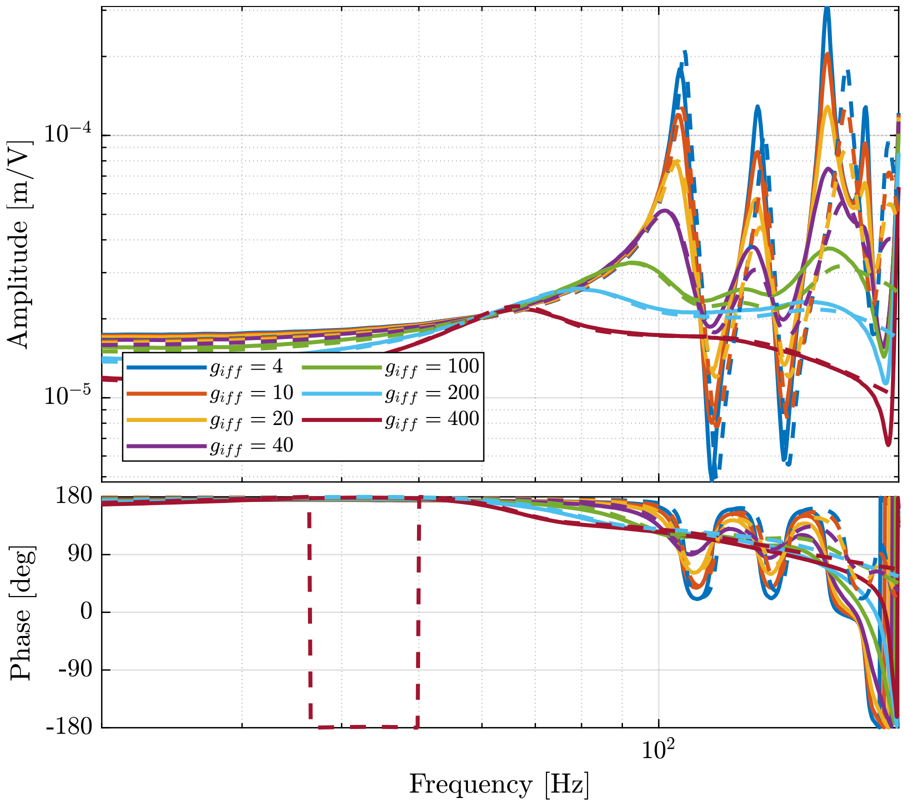

Figure 1.13: Transfer function from u to $d\mathcal{L}_m$ for multiple values of the IFF gain (Zoom)

Important

The IFF control strategy is very effective for the damping of the suspension modes. It however does not damp the modes at 200Hz, 300Hz and 400Hz (flexible modes of the APA). This is very logical.
Also, the experimental results and the models obtained from the Simscape model are in agreement.

Experimental Results - Comparison of the un-damped and fully damped system

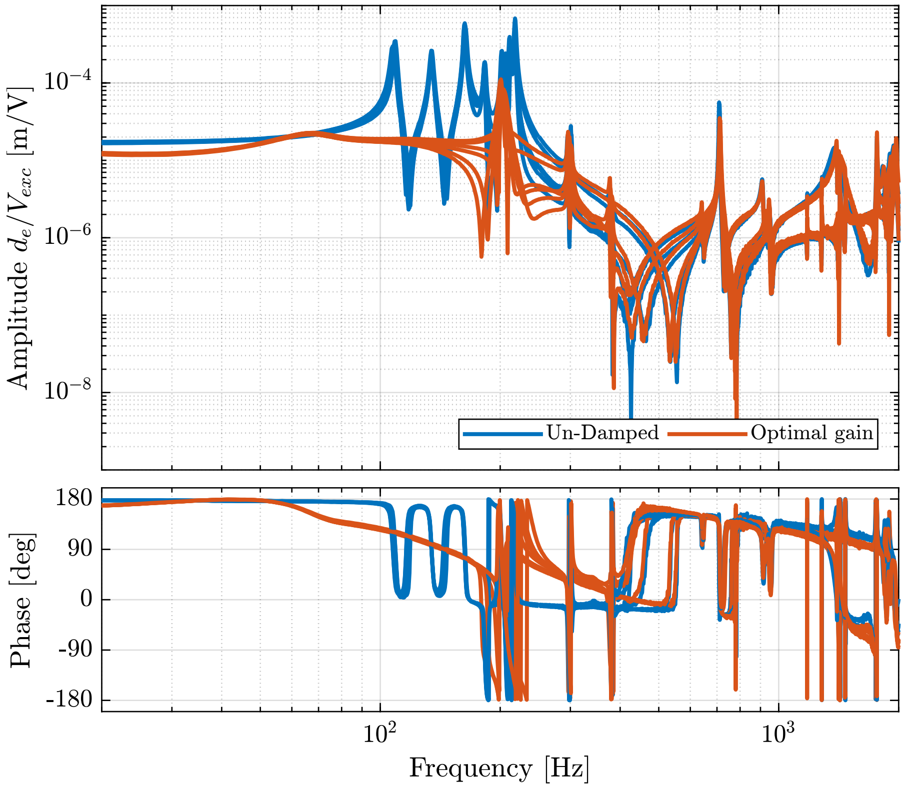

Figure 1.14: Comparison of the diagonal elements of the tranfer function from $\boldsymbol{u}$ to $d\boldsymbol{\mathcal{L}}_m$ without active damping and with optimal IFF gain

Question

A series of modes at around 205Hz are also damped.
Are these damped modes at 205Hz additional "suspension" modes or flexible modes of the struts?

1.4.4 Experimental Results - Damped Plant with Optimal gain

Let's now look at the 6×6 damped plant with the optimal gain $g = 400$.

Load Data

```matlab
%% Load Identification Data
meas_iff_struts = {};

for i = 1:6
    meas_iff_struts(i) = {load(sprintf('mat/iff_strut_%i_noise_g_400.mat', i), 't', 'Vexc', 'Vs', 'de', 'u')};
end
```

Spectral Analysis - Setup

```matlab
%% Setup useful variables
% Sampling Time [s]
Ts = (meas_iff_struts{1}.t(end) - (meas_iff_struts{1}.t(1)))/(length(meas_iff_struts{1}.t)-1);

% Sampling Frequency [Hz]
Fs = 1/Ts;

% Hannning Windows
win = hanning(ceil(1*Fs));

% And we get the frequency vector
[~, f] = tfestimate(meas_iff_struts{1}.Vexc, meas_iff_struts{1}.de, win, [], [], 1/Ts);
```

DVF Plant

```matlab
%% DVF Plant (transfer function from u to dLm)
G_iff_opt = {};

for i = 1:6
    G_iff_opt{i} = tfestimate(meas_iff_struts{i}.Vexc, meas_iff_struts{i}.de, win, [], [], 1/Ts);
end
```

Important

With the IFF control strategy applied and the optimal gain used, the suspension modes are very well damped. Remains the undamped flexible modes of the APA (200Hz, 300Hz, 400Hz), and the modes of the plates (700Hz).
The Simscape model and the experimental results are in very good agreement.

1.5 Modal Analysis

Several 3-axis accelerometers are fixed on the top platform of the nano-hexapod as shown in Figure 1.19.

The top platform is then excited using an instrumented hammer as shown in Figure 1.18.

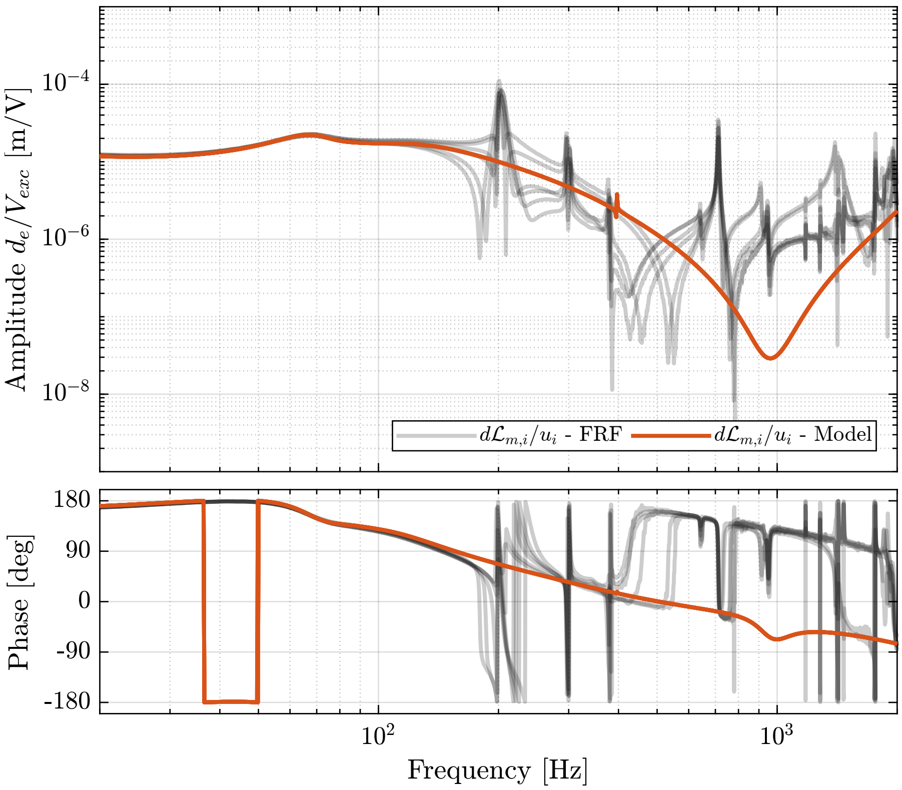

Figure 1.15: Comparison of the diagonal elements of the transfer functions from $\boldsymbol{u}$ to $d\boldsymbol{\mathcal{L}}_m$ with active damping (IFF) applied with an optimal gain $g = 400$

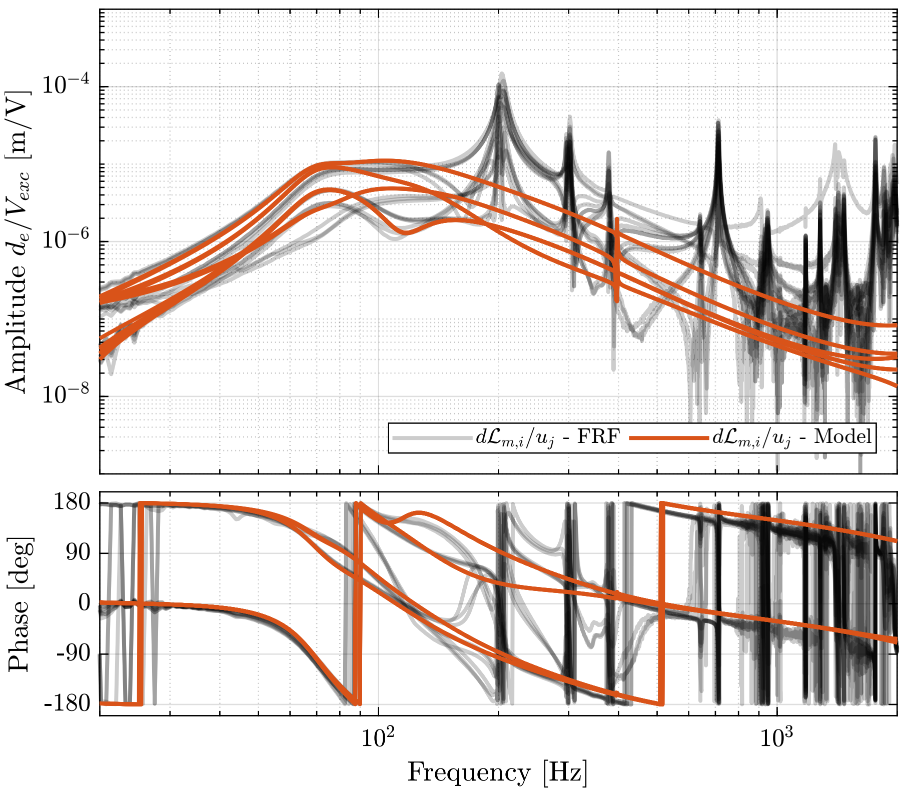

Figure 1.16: Comparison of the off-diagonal elements of the transfer functions from $\boldsymbol{u}$ to $d\boldsymbol{\mathcal{L}}_m$ with active damping (IFF) applied with an optimal gain $g = 400$

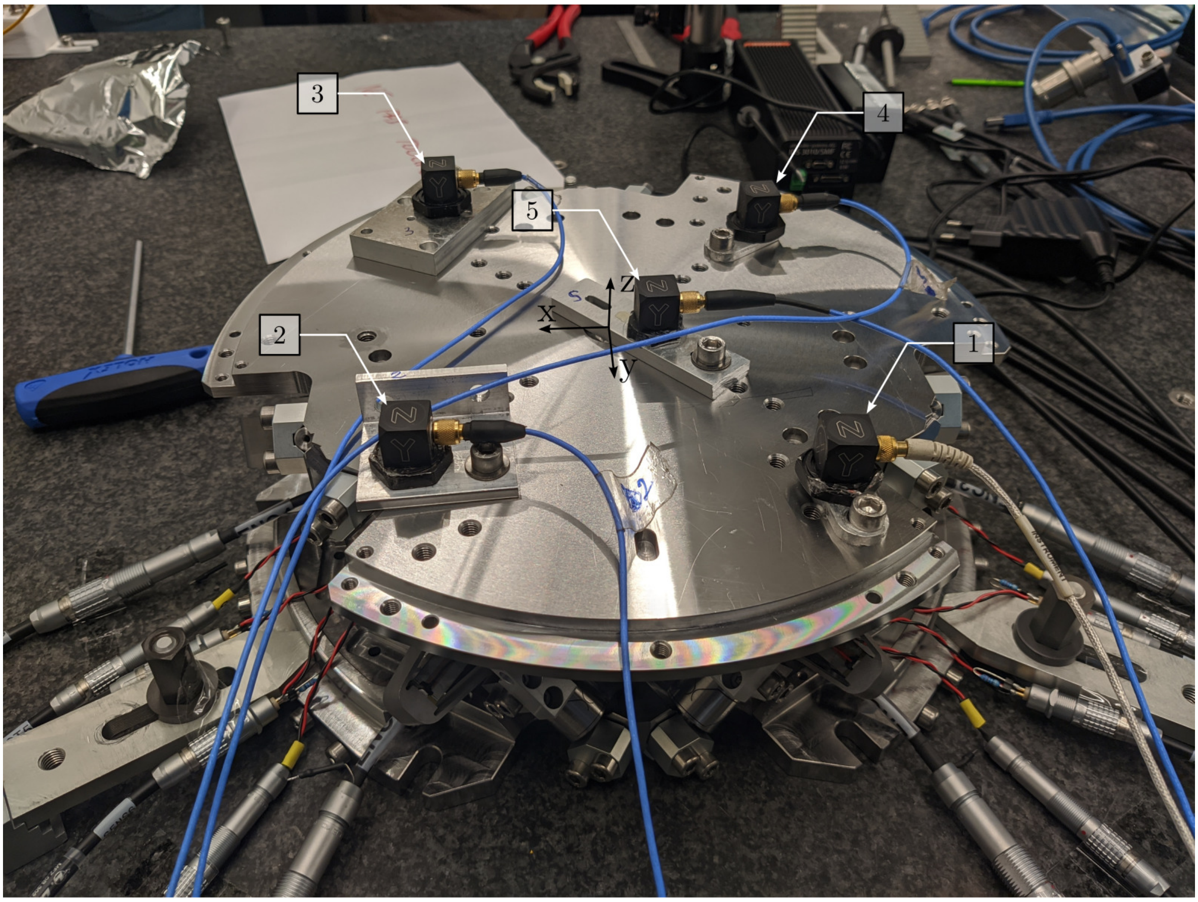

Figure 1.17: Location of the accelerometers on top of the nano-hexapod

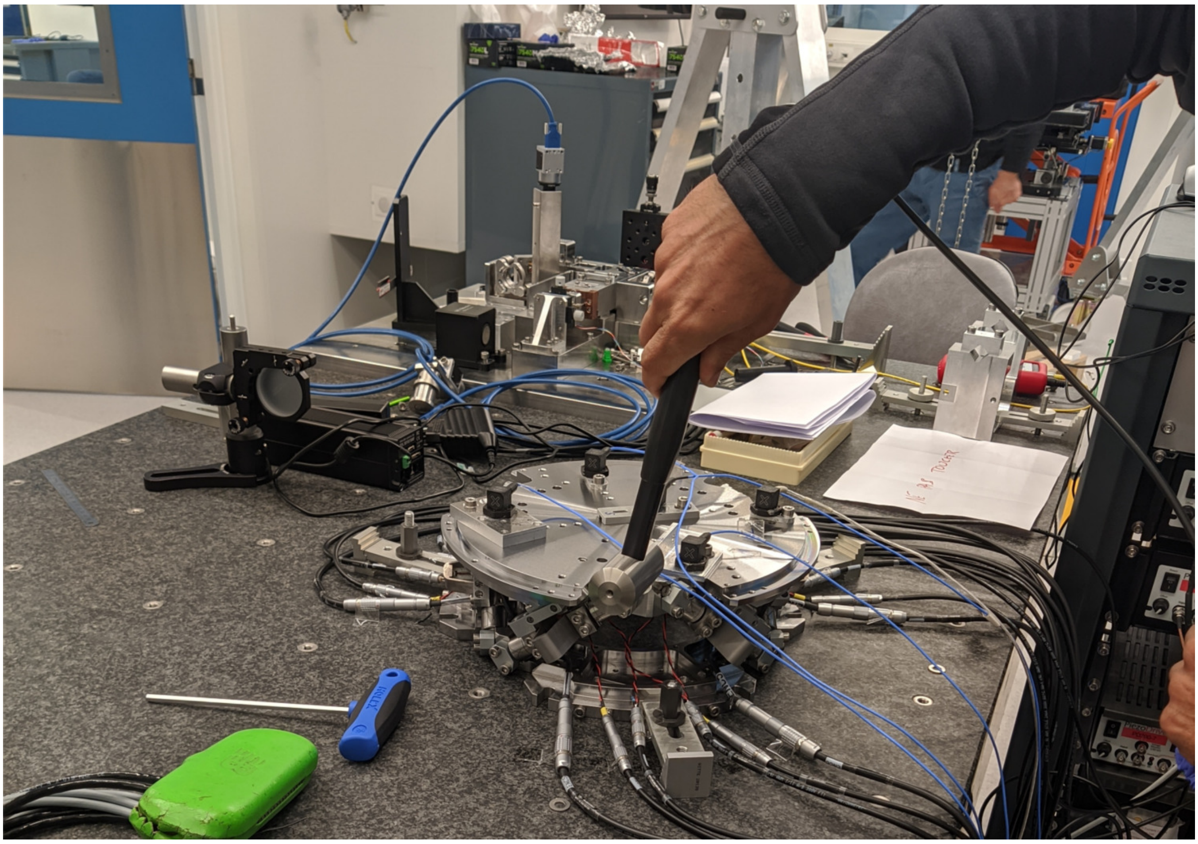

Figure 1.18: Example of an excitation using an instrumented hammer

1.5.1 Effectiveness of the IFF Strategy - Compliance

In this section, we wish to estimated the effectiveness of the IFF strategy concerning the compliance.

The top plate is excited vertically using the instrumented hammer two times:

1. no control loop is used
2. decentralized IFF is used

The data is loaded.

```matlab
frf_ol  = load('Measurement_Z_axis.mat'); % Open-Loop
frf_iff = load('Measurement_Z_axis_damped.mat'); % IFF
```

The mean vertical motion of the top platform is computed by averaging all 5 accelerometers.

```matlab
%% Multiply by 10 (gain in m/s^2/V) and divide by 5 (number of accelerometers)
d_frf_ol = 10/5*(frf_ol.FFT1_H1_4_1_RMS_Y_Mod + frf_ol.FFT1_H1_7_1_RMS_Y_Mod + frf_ol.FFT1_H1_10_1_RMS_Y_Mod +
↪   frf_ol.FFT1_H1_13_1_RMS_Y_Mod + frf_ol.FFT1_H1_16_1_RMS_Y_Mod)./(2*pi*frf_ol.FFT1_H1_16_1_RMS_X_Val).^2;
d_frf_iff = 10/5*(frf_iff.FFT1_H1_4_1_RMS_Y_Mod + frf_iff.FFT1_H1_7_1_RMS_Y_Mod + frf_iff.FFT1_H1_10_1_RMS_Y_Mod +
↪   frf_iff.FFT1_H1_13_1_RMS_Y_Mod + frf_iff.FFT1_H1_16_1_RMS_Y_Mod)./(2*pi*frf_iff.FFT1_H1_16_1_RMS_X_Val).^2;
```

The vertical compliance (magnitude of the transfer function from a vertical force applied on the top plate to the vertical motion of the top plate) is shown in Figure 1.19.

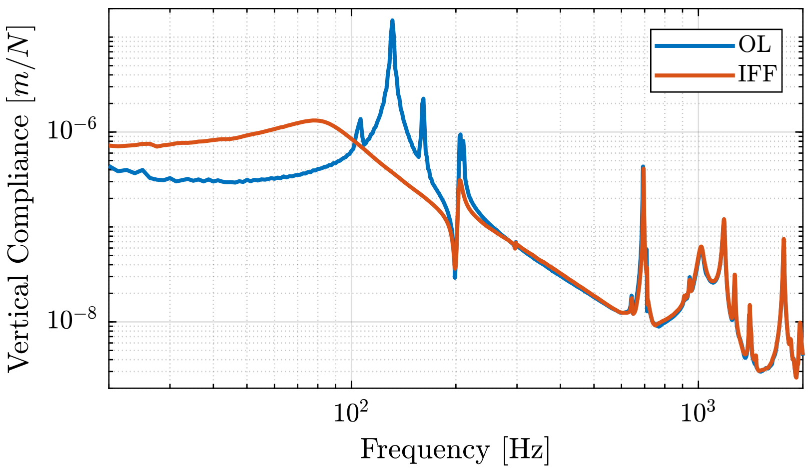

Figure 1.19: Measured vertical compliance with and without IFF

Important

From Figure 1.19, it is clear that the IFF control strategy is very effective in damping the suspensions modes of the nano-hexapode. It also has the effect of degrading (slightly) the vertical compliance at low frequency.
It also seems some damping can be added to the modes at around 205Hz which are flexible modes of the struts.

1.5.2 Comparison with the Simscape Model

Let's now compare the measured vertical compliance with the vertical compliance as estimated from the Simscape model.

The transfer function from a vertical external force to the absolute motion of the top platform is identified (with and without IFF) using the Simscape model. The comparison is done in Figure 1.20. Again, the model is quite accurate!

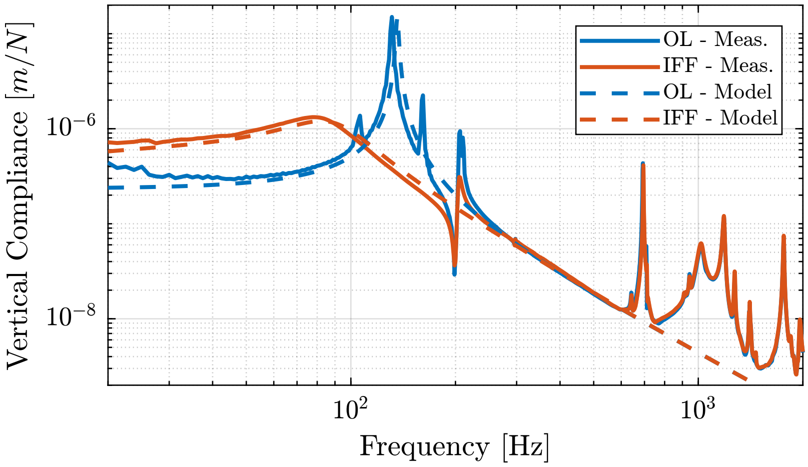

Figure 1.20: Measured vertical compliance with and without IFF

1.5.3 Obtained Mode Shapes

Then, several excitation are performed using the instrumented Hammer and the mode shapes are extracted.

We can observe the mode shapes of the first 6 modes that are the suspension modes (the plate is behaving as a solid body) in Figure 1.22.

Then, there is a mode at 692Hz which corresponds to a flexible mode of the top plate (Figure 1.22).

The obtained modes are summarized in Table 1.1.

Table 1.1: Description of the identified modes

Mode	Freq. [Hz]	Description
1	105	Suspension Mode: Y-translation
2	107	Suspension Mode: X-translation
3	131	Suspension Mode: Z-translation
4	161	Suspension Mode: Y-tilt
5	162	Suspension Mode: X-tilt
6	180	Suspension Mode: Z-rotation
7	692	(flexible) Membrane mode of the top platform

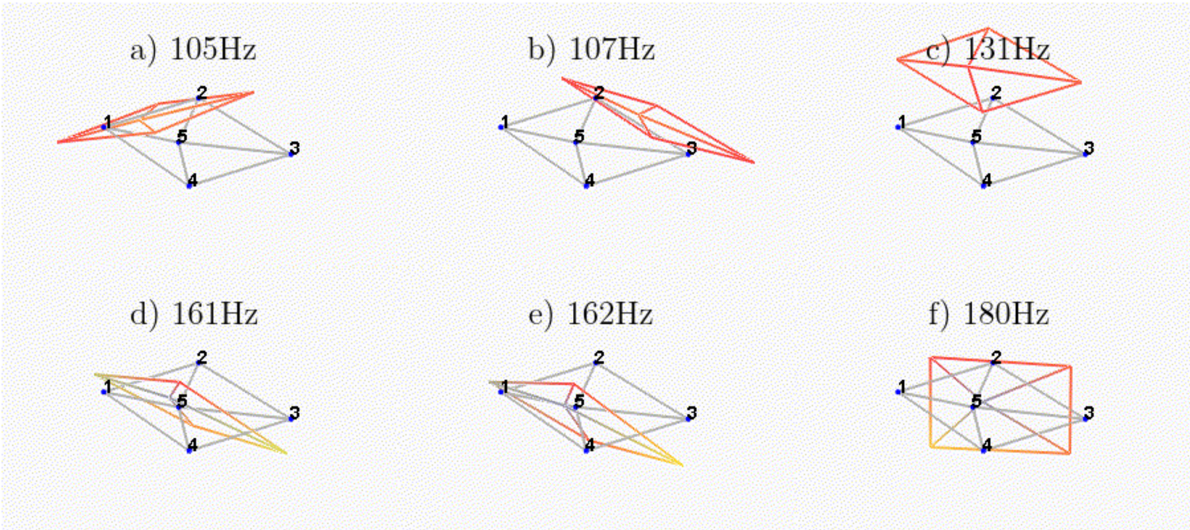

Figure 1.21: Measured mode shapes for the first six modes

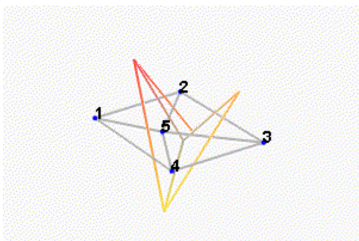

Figure 1.22: First flexible mode at 692Hz

2 Encoders fixed to the plates