# SE 165/265 Homework 5
**Assigned May 1st, 2024; Due May 8th, 2024 (11:59 pm Pacific Time)**

**Assignment Goal**:

1. Look at damage-sensitive features that identify the transition of the system from one that can be accurately modeled as a linear system in its undamaged state to one that exhibits nonlinear response characteristics as a result of damage.

2. Reinforce material presented in lectures 9 and 10.

**Background Reading:** *Structural Health Monitoring: A Machine Learning Perspective, Chapter 8: Features Based on Deviations from Linear Response*

**Introduction**

We will continue to use experimental data from the 4-story structure. You can refer to Homework #1 to recall how the data were acquired. Recall from the signal processing lecture that the coherence function is a measure of how linearly related a response measurement is to an input measurement.

With the 4-story structure, the first 9 states correspond to a system that is linear, but with different stiffness and mass values to simulate operational and environmental variability one might encounter in real-world applications.

Damage that is introduced with the bumper mechanism in states 10-17 results in a nonlinear system response somewhat analogous to a crack opening and closing. Therefore, we would expect the coherence function to change because of this transition from a linear to nonlinear system, which results from the introduction of the simulated damage.

## Task 1: Load data and create arrays.

Download the file **data3SS2009.mat** from the SE 165-265 CANVAS Data File folder, 4-story structure data. **Load** this file in Matlab. This file contains measurements from all 5 sensors.

Loading this file will give you a 3-D matrix called **dataset** ($\mathbf{8192 \times 5 \times 850}$), where 8,192 corresponds to the data points, 5 corresponds to channels 1 to 5, and 850 is the number of measurements (50 measurements each for 17 states). Recall from HW 1 that states 1-9 are considered undamaged, and states 10-17 are damaged.

In this assignment, **we will work with data from all the channels: Channel 1** (shaker input), **Channels 2 - 5** (response data from the accelerometers on the floors 1-4).

Although not necessary, I first saved the data from channels 1-5 in new 8192x850 arrays called:

Channel 1 = **input**
Channel 2 = **response1**

Channel 3 = **response2**
Channel 4 = **response3**
Channel 5 = **response4**

## Task 2: Calculate and plot coherence functions.

Calculate the coherence between the input (channel 1) and the corresponding response signals (channels 2 to 5) for all the data sets (States 1-17). Form the coherence estimate using 16 averages, zero overlap and with a Hanning window applied to each average.

For each input-sensor pair (input and Channels 2-5) you should have 850 coherence functions.

You will need to generate a vector of frequencies to plot the coherence function, noting the sampling parameters for these data: 8192 pts, T=25.6 s, dt=0.003125s, sampling frequency = 320 Hz. You can generate this frequency vector directly, or you can generate it with the MATLAB **mscohere** function:

cxy=**mscohere(x,y,window,noverlap,nfft)**, or
[cxy,f] = **mscohere(x,y,window,noverlap,nfft,fs)**

Note, in the subsequent analyses, we will compare the coherence functions from the same input-sensor pair, so you should store your coherence functions in 4 arrays corresponding to the different input-sensor pairs (or everything in a 3D matrix if you prefer). For simplicity, in the following tasks, we will refer to these input-sensor pairs as "**sensors**".

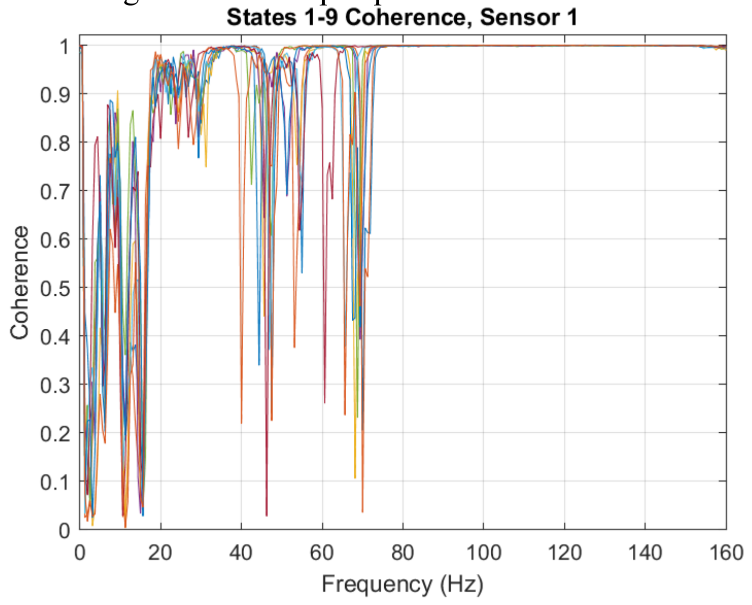
The rows in each array correspond to the coherence functions. You should have four arrays that have dimension of 850 x 257, where 850 corresponds to the 50 measurements from each system state, and 257 is the number of the frequency values in our coherence function when it is estimated from 16 averages of an 8192-pt record.

Now we will generate some plots to get an idea of how the coherence varies at each sensor location. Remember, for all the tasks that requires **subplots**, when creating the figure (before the subplots), use:

```
      figure('Renderer', 'painters', 'Position', [10 10 1200 900])
```

**Plot 1:** For each sensor, overlay a plot of the coherence functions for the **first** measurement of each undamaged state (rows 1, 51,101…401). These four plots should be displayed as a **2 x2 grid of subplots**. For this figure only, you do not need to create a legend. We just want to observe the general trend.

The plots should give an idea of how much the simulated operational and environmental variability influence the coherence. Note poor coherence is to be expected below 20 Hz and above 150 Hz because a frequency band-limited excitation between 20-150 Hz was used. For sensor 3 there will be poor coherence above approx. 100 Hz because the turnbuckle on the bumper mechanism wasn't tightened in its open position and could rattle.



*Hint 1: Result of subplot(2,2,1)- Coherence measurements between input and response 1, first measurement of states 1 to 9.*

**Plot 2:** For each sensor, plot an overlay of the coherence from the first measurement of state 1 (row 1) and the first measurement of state 10 (row 451) (low damage level). These plots should be displayed in **a 2 x 2 grid** of subplots. Add a legend to identify the coherences functions corresponding to the two states.

**Plot 3:** For each sensor plot an overlay of the coherence from the first sample of state 1 (row 1) and the first sample of state 14 (row 651) (high damage level). These plots should be displayed in a 2 x 2 grid. Add a legend to identify the coherences functions corresponding to the two states.

## Task 3: Calculate a coherence comparison metric.

Examination of the plots generated in Task 2 shows qualitatively that the coherence deviates further from an amplitude of one across the spectrum as the damage level increases. The goal now is to come up with a metric that will allow us to quantify that observation.

Because we know the coherence function should take on a value of one for a linear system, we will define our metric as a "unity deviation metric, UDM". For each measurement, at each frequency value, $i$, we will simply subtract the coherence value, $COH_i$ from one, and then sum up the magnitude of these differences for all $n$ frequency values.

$$\text{UDM} = \sum_{i=1}^{n} \left| \left(1 - \text{COH}_i\right) \right|$$

We will also do some data cleansing as part of this task. We will use our knowledge of the 20-150 Hz frequency band limits on the excitations. We do not expect the coherence to be close to one outside of this frequency band because there is no input being applied outside this band and the resulting coherence calculation in these regions will be primarily based on noise in the data.

Therefore, rather than summing over $i = 1 - 257$, we will just use the portion of the coherence function that correspond to frequencies between 20 and 150 Hz. From your frequency vector, find the appropriate indexes $i$ to use for the UDM calculation.

For each sensor, for all 850 measurements, calculate the UDM value for the frequency range 20-150Hz.

**Plot 4:** In a 2x2 grid, create four **bar plots** (one for each sensor) using the UDM value from the first 10 measurements of states 1- 9 and the first 10 measurements of states 10 - 14. Therefore, these plots should have 140 vertical bars (90 corresponding to undamaged cases and 50 corresponding to damaged cases).

Use the same vertical scale on each of the four plots. Use the command **bar** instead of **plot**.

In addition, on each plot, add a **vertical red line** that separates the undamaged cases (1-90) and damaged cases (91-140).

Also, add a **horizontal red line** corresponding to the lowest value of all the **damaged** cases being analyzed for that sensor (i.e., 451 to 460, 501 to 510, and so on).

- Print this value on the command window. Make sure to indicate which sensor the printed value refers to.

The horizontal line will vary for each plot, while the vertical one is fixed at 90. These lines will give an idea of how well separated the damage cases are from the undamaged cases.

Your x-axis label can be named "measurement".