MATLAB Hands-on Exercise Session
Report submission due: (Sunday) March 26, 2023

In this exercise session, we will practice some of the MATLAB commands and functions we learnt in the MATLAB lecture. We will also learn about the concept of "frequency" through practical examples, which is a fundamental concept in electrical engineering.

Note: In this session you need to go through the whole exercise set in the given order, since exercises build on each other. However, you are asked to submit the answers to only some selected questions for ease of reporting, which are highlighted below.

Part I: Overview of basic tools

In this first part, we will remember and use some of the basic commands we saw in the MATLAB lecture for warming up. You can do the exercises in this part by typing your commands in the command window. Remember that, any time you need, you can clean the command window by typing `clc`, erase all your variables and workspace by typing `clear all`, and close all your figures by typing `close all`.

1. Generate a variable `L` that is equal to 6.
2. Create a vector `a` of length `L` which consists of only 0's:

   ```
   a=zeros(1,L);
   ```

   Click on the variable `a` in the workspace to observe its entries. Try also typing

   ```
   a=zeros(1,L)
   ```

   without the semicolon and observe what difference it makes.
3. It is also very easy to generate vectors or matrices that consist of only 1's in MATLAB.
   - Type `help ones` in the command window to learn how you can use the `ones` function.
   - Generate an `L` $\times$ `L` matrix `A` that consists of 1's.
4. Now use the `randn` command to generate an `L` $\times$ `L` matrix `A` with random entries. If you don't know how to use the `randn` command, you can get help from the command window by typing `help randn`.
5. We will now modify some of the entries of `A`.
   - Suppose that we would like to set the first row of `A` to 0. You can do this by typing

     ```
     A(1,:)=0
     ```

   - Similarly, now set the last three columns of `A` to -1. You can do this with a short single command. (Hint: Remember how intervals can be specified in the format `J:K` in MATLAB. You may also type "`help :`" if you need guidance.)

6. Now, we would like MATLAB to find the indices of the `A` matrix that contains 0's. Try the command

```
[m n]=find(A==0)
```

and observe its output. Write down the values of `m` and `n` and briefly explain what they represent.

7. Now form a $1 \times \mathtt{L}^2$ vector `b` containing the integer values $[1\ 2\ \cdots\ \mathtt{L}^2]$, by using the "`:`" operator (such as `J:K`). Reshape the vector `b` into an $\mathtt{L} \times \mathtt{L}$ matrix `B` as

```
B=reshape(b,L,L);
```

8. We will now take the product of `A` and `B` using two different product operators. Try

```
C=A*B;
D=A.*B;
```

and observe their effect. Comment on the difference between the two product operators " * " and ".* ". A very common beginner's mistake in MATLAB is to use * instead of .*, which typically leads to dimension mismatch and other errors.

9. - Reshape the $\mathtt{L} \times \mathtt{L}$ matrix `D` back to a $\mathtt{L}^2 \times 1$ vector `d` by using the colon " : " operator. (Type "`help :`" if you don't know how to do this.)
   - Try the commands

```
d(end)
mean(d)
abs(d)
```

and observe their effect.

10. Generate a new figure box with the `figure` command, and then plot the entries of the `d` vector using the `stem` command. Name the x-axis as `n` and the y-axis as `d[n]`. Also give a title to your figure. You can use the `xlabel`, `ylabel`, and the `title` commands for these. Save your figure and provide your plot in your report.

Part II: Frequency analysis of sinusoidal signals

In this second part, we will work on sinusoidal signals of the form

$$x(t) = \sin(w_0 t).$$

Here t is a time variable, and $x(t)$ is a sine signal that oscillates with angular frequency w_0 as t increases. Mathematically speaking, a "signal" means nothing but a function of time t.

1. Instead of writing our codes in the command window, we will write them in script files from now on.
   - Create a new MATLAB script file by following the main menu options Home →New Script. This should open a blank matlab file.
   - Save the file in a folder of your choice in your computer.
   - Please note that it is common practice to start MATLAB scripts with the

```
clc
clear all
close all
```

commands to reset the work environment.

- When implementing and testing new scripts, it is important to know how to use debugging tools. Debugging is very easy in MATLAB: Clicking on the hyphen "-" in the gray zone next to a line will put a red dot called a "breakpoint". You can then use the menu tools conveniently: Click on "Run" to start the code, "Step" to proceed line by line in a code, "Continue" to go straight until the next line with a breakpoint, and "Step in" to enter inside a function. You can quit the code by clicking on "Quit Debugging" any time you want.

2.
- In your script file, generate an angular frequency variable `w0` and set its value to 2π. Also generate a time duration variable `T` and set its value to 10 (measured in seconds, we will assume).
- We wish to study the signal $x(t) = \sin(w_0 t)$ in the time interval $t \in [0, \mathtt{T}]$. However, in MATLAB it is not possible to exactly represent continuous intervals such as $[0, \mathtt{T}]$. What we can do instead is to take many samples from the time interval $[0, \mathtt{T}]$ and put them in a vector `t`. Let's assume that we will take `fs` $= 50$ samples per second to approximately represent the 10-second time interval $[0, \mathtt{T}]$. We can do this by generating a vector

```
t=0:1/fs:T;
```

- Now generate a vector `x` that consists of the values of the signal $x(t) = \sin(w_0 t)$ evaluated at the t values in your vector `t`.
- Create a new figure box and plot the signal $x(t)$ as a function of t using the `plot` command:

```
plot(t,x);
```

Please label the horizontal axis as `t (seconds)` and the vertical axis as `x(t)` in your plot. Also give a title to your figure. Use the `xlabel`, `ylabel`, and the `title` commands. Save your figure and provide your plot in your report.

3. Our next job is to implement a function that analyzes the frequency content, or the "spectrum" of a given input signal. The "spectrum" of a signal essentially means its frequency content, i.e., how much it oscillates at each frequency. In particular, a simple sinusoidal signal such as our signal $x(t) = \sin(w_0 t)$ oscillates only at the angular frequency w_0; hence its spectrum is very simple. On the other hand, an audio signal such as a song is typically made up of many different frequency components. In fact, each frequency corresponds to a different musical note.

   We will now learn how to analyze and display the spectrum of an arbitrary input signal.

- Let us implement our frequency analyzing function `frequencyAnalyzer` in a separate file. Follow the options New →Function to create a new MATLAB function script. Modify the title of the function as `frequencyAnalyzer`.
- The function should take as inputs the signal `x` and the sampling parameter `fs`. Modify the input arguments accordingly.
- You can remove the output argument for the moment. Then, save your function file in the same folder as your main script file. Make sure that the file name is "frequencyAnalyzer.m" (the same as the name of the function).

4. We can now start implementing inside our `frequencyAnalyzer` function.

- In signal processing, we compute the spectrum (i.e. frequency content) of a given signal with a technique called the Fourier transform. You will learn about this (a lot !) in your EE301 Signal Processing course.
- In MATLAB we take the Fourier transform using the `fft` function. It is common practice to represent the spectrum with capital letters, so let us call the spectrum of our signal as `X`. Type:

```
X=fft(x);
```

- The spectrum `X` often contains complex entries. It is easier to work with real numbers. Form a new real vector `magX` that contains the magnitude of each entry of `X`. You can use the `abs` function for this.
- Next, obtain the length of the signal `X` and store it in a variable called `N`. You can use the `length` function for this.
- We would now like to plot the spectrum magnitude `magX`, but we wish to represent frequencies in units of Hz (Hertz). Type

```
f=fs*(0:N-1)/N;
```

  which creates a frequency variable `f` in units of Hz. (It's not at all obvious to see why this should give you something in Hz at this point. The two-year-older "you" will know why.)
- Before plotting the spectrum, we need to take care of a last detail. The second half of the vector `magX` is in fact a symmetric replica of its first half when the input signal is real. This is for mathematical reasons, which you will learn in the EE301 and EE430 courses. For the moment, let us just take the first halves of the vectors `magX` and `f`, and get rid of the second halves. You can do this with the commands

```
magX=magX(1:round(N/2));
f=f(1:round(N/2));
```

- We can now plot the spectrum inside our `frequencyAnalyzer` function. Create a new figure box and plot the spectrum magnitude `magX` as a function of the frequency `f`. Label the horizontal axis as `Frequency (Hz)`, and the vertical axis as `Spectrum magnitude`.

5. We are now ready to test our `frequencyAnalyzer` function.

   - Go back to your main script file and call the `frequencyAnalyzer` function for your input signal `x`. Remember that you need to specify the sampling parameter `fs` as well.
   - You may often get an error or warning message when testing a new script. If this happens, make sure to use the debugging tools to spot where the problem has occurred. Undefined variables, syntax errors, dimension mismatches are among the common causes for mistakes.
   - If you managed to get your function up and running, you should be able to see the spectrum plot of the input signal now. Save your figure and provide the spectrum plot in your report.
   - Inspect the plotted spectrum. Can you approximately spot the dominant frequency f_0 in the signal `x`? Interpret this by remembering that our signal is $x(t) = \sin(w_0 t)$. What is the relation between w_0 (the angular frequency in radians per second) and f_0 (the frequency in Hertz)?

6. Let us now develop our function `frequencyAnalyzer` further by adding an output argument.

   - Let the output of the function be a vector named `freq_x`, which will return a list of the dominant frequencies we will detect in the input signal `x`.
   - In order to determine the frequency or frequencies that are dominant inside our input signal, we will compare the magnitude `magX` of the spectrum to a threshold value. The idea is that the frequencies where `magX` lies above the threshold will be detected as dominant frequencies.
   - Inside your function script "frequencyAnalyzer.m", please write a code that determines a suitable threshold value automatically. It is often a good idea to adapt the threshold to your signal: For instance, find the peak value of `magX`, and then set the threshold to be a certain percentage (such as 10%) of the peak value. You may use the `max` function for finding the peak value.
   - Once you set your threshold, find the indices of `magX` where `magX` lies above the threshold.

- Extract the values of the frequency vector `f` corresponding to these indices, and store them in the output vector `freq_x`. These are the frequency values (in Hz) where the spectrum `magX` exceeds the threshold.

7. Now in your main script file, call the `frequencyAnalyzer` function again. Assign the output to a variable called `freq_x`. Display the dominant frequencies on the command window using the `disp` command as follows:

```
disp(['Dominant frequencies in the input signal: ' num2str(freq_x) ' Hz']);
```

   Report the dominant frequencies in the signal and comment on the result. Is the function output consistent with the dominant frequency f_0 you determined in part 5?

8. Let us now generate a different input signal.

   - In your main script file, generate a new vector `y` that represents the following signal:

$$y(t) = \sin(w_0 t) + \sin(2w_0 t) - 4\sin(3w_0 t)$$

     Take $w_0 = 2\pi$ as before, and generate the signal over the same time interval $t \in [0, \mathtt{T}]$.
   - Plot the signal $y(t)$ and compare it to $x(t)$. Label the axes suitably. Provide the plot in your report. In what way do $x(t)$ and $y(t)$ have similar and different characteristics?

9. Lastly, let us analyze the spectrum of our input signal $y(t)$. Provide `y` as input to the `frequencyAnalyzer` function. Inspect the spectrum of $y(t)$ and comment. What are the dominant frequencies in $y(t)$ in units of Hz? Comment on how these are related to the angular frequencies w_0, $2w_0$, and $3w_0$ of the three sinusoids making up the signal.