Synthesis of Sinusoidal Signals using Tuning Forks

Tuning forks are physical systems that generate sinusoidal signals. When a tuning fork is exposed to vibration, it disturbs nearby air molecules, creating regions of higher-than-normal pressure (called compressions) and regions of lower-than-normal pressure (called rarefactions).

Pre-Lab: Read the lab handout and section 2.7 in the textbook on tuning forks. Consider the following question. This lab uses tuning forks with frequencies ranging from 128 Hz to 4096 Hz. Assuming that the stiffness of the tuning forks is the same, comment on the mass of the tuning forks as the frequency increases. Verify your hypothesis when you get to the lab.

1 Overview

In this lab, you will use a microphone and a computer equipped with an Analog to Digital (A to D) converter to digitize a sinusoidal signal from different tuning forks. The microphone will convert the sound signal generated by the tuning fork to an electrical signal, which in turn is converted to a sequence of numbers stored in a digital file which can be displayed on computer screen as a sinusoidal curve. Characteristics of the sound wave, such as its period T and frequency can be determined from this curve. Knowing the wave's period, its frequency f is easily computed using the formula

f = 1 /T

2 Matlab Preliminaries

In this section, you will write an M-File function that takes input sound signal at a specific sampling rate and stores it as an output file in wav format. The input signal will be the tuning fork signal when you strike it and expose to vibration.

2.1 Introduction to Using Functions in Matlab

Functions are M-files that can accept input arguments and return output arguments. The names of the M-file and of the function should be the same. Functions operate on variables within their own workspace, separate from the workspace you access at the MATLAB command prompt. The first line of a function M-file starts with the keyword function. It gives the function name and order of arguments. A simple function, called compmag5 that computes the five times the magnitude of a complex number of the form, x+jy is given below. In this case, there are two input arguments and one output argument.

```
function [z] = compmag5(x,y);
% % % % % % % % %
% this function computes the magnitude of the complex number x+jy and returns
it in variable z.
% Inputs:
% x – real part of complex number
% y – imaginary part of complex number
%Outputs:
%z – magnitude of complex number times 5
%written by J.A. Dickerson, January, 2005

a=5
% compute magnitude and multiply
z =a∗ sqrt(x^2+y^2);
```

The next several lines, up to the first blank or executable line, are comment lines that provide the help text. These lines are printed when you type

```
>>help compmag5
```

The first line of the help text is the H1 line, which MATLAB displays when you use the lookfor command or request help on a directory. The rest of the file is the executable MATLAB code defining the function. The variable a introduced in the body of the function, as well as the variables on the first line, m, x and y, are all local to the function; they are separate from any variables in the MATLAB workspace. This means that if you change the values of any variables with the same name while you are in the function, it does not change the value of the variable in your workspace.

If no output argument is supplied, the result is stored in ans. You can terminate any function with an end statement but, in most cases, this is optional. Functions normally return when the end of the function is reached. The function is used by calling the function within Matlab or from a script. For example, the commands below can all be used to call compmag5 and get a result of m=10.

```
>>x=2;y=2; m=compmag5(x,y);
```

2.2 Audio Input using a Matlab function

Matlab inputs the signal from a microphone using the audiorecorder commands. Type doc audiorecorder for more information. To use audiorecorder, follow the steps below.

```
% set up and audiorecorder object to control the audio input. This object
% controls your sampling rate, Fs, the number of bits, nbits, used to
% represent the sample and which device is used to collect the sample.

%recObj = audiorecorder(Fs,nbits,device)
Fs = 11025, nbits = 16, device = 1,
recObj = audiorecorder(Fs,nbits,device)
```

The next step is to record a signal. The code below first gives a cue to start data collection and then records 5 seconds of audio. Note, you may need to lengthen the recording time for some tuning forks.

```
disp('Start Recording')
recordblocking(recObj, 5);
disp('End of Recording.');
```

To play back the recording.

```
play(recObj);
```

Store data in double-precision array, `tuningfork1`.

```
tuningfork1 = getaudiodata(recObj);
```

Plot the audio samples with a properly labeled time axis. First set up a vector with the same length as your audio signal, tuningfork1, then scale it based on the sampling interval (1/Fs):

```
T = (0:length(tuningfork1)-1)* (1/Fs);
plot(T,tuningfork1);
```

To save the audio file, save the array `tuningfork1`. To play it back later, use the function `sound`.

```
sound(tuningfork1, Fs);
```

2.3 Labeling Plots in MATLAB

It is very important to be able to label all plots and graphs turned in with your lab reports. Basic functions for labeling axes and putting titles on your graphs are given in this section.
`xlabel, ylabel:` these commands put a text label on the x and y axes of a plot. For example, to label the x- axis as time in msec: `xlabel('time (msec)')`; The characters within the quotes are plotted on the active graph.

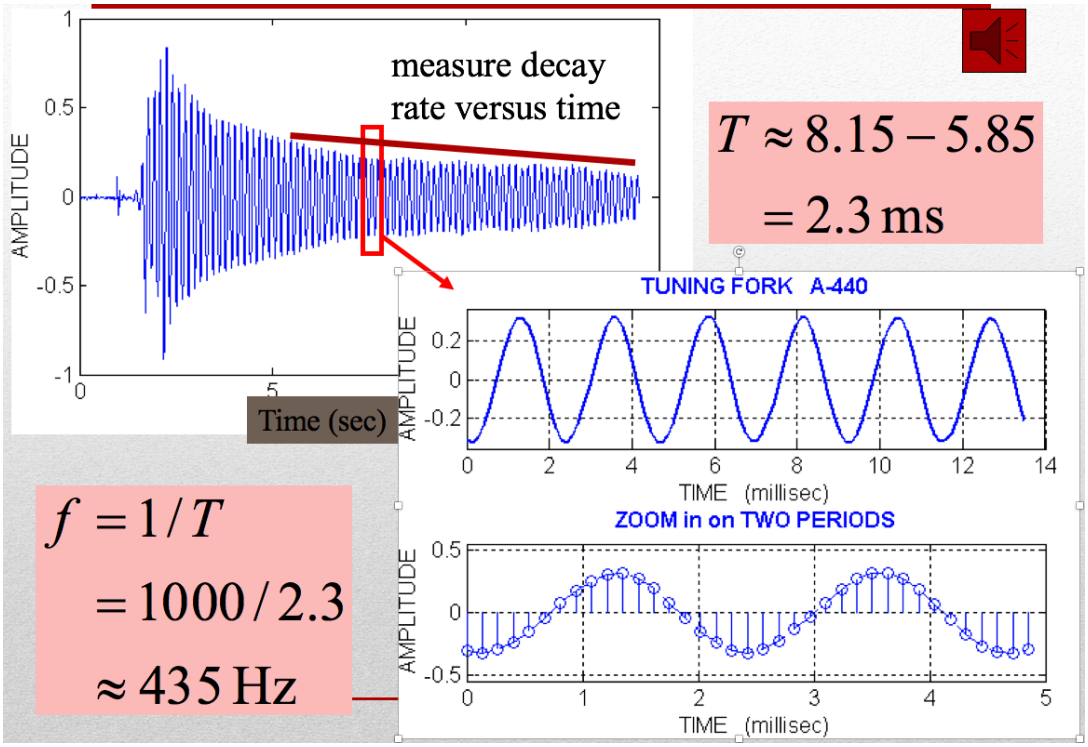
title: this command puts a title above the graph. For example, to label a plot 'Time vs. Tuning Fork Response', use: `title('Time vs. Tuning Fork Response');`
axis: this command sets the beginning and end values for the graph axes. For example, if the time should start at 2 sec and end at 5 sec and the recorded signal should range between 0 and 10 volts, use the command: `axis([2, 5, 0 10]);` `axis([xmin xmax ymin ymax])`.

3. Frequency view of signals in Matlab

Signals can also be viewed in terms of frequency as well as time. The Fourier Transform is used to show what frequencies are present in a signal. We will cover exactly how this works in class in the next few weeks. The command below plots the frequency spectrum of a signal where Fs is the sampling frequency:
`freqz(tuningfork1,[1],Fs);`

4 Lab Procedure Signal Collection

A. Collect tuning fork signals. Use a microphone and collect signals for two different tuning forks at frequencies at or under 2048 Hz using a sampling frequency of 11,025 samples/sec. Make sure that you give each signal a different name to avoid overwriting.



B. Measure the frequency of these signals using a plot of the signal. Include the plots that you used in your lab report. Be sure to label your graph and all the time units using the methods given in section 2.3 above. Show how you measured the signal. In order to have the proper time index on the plot, the sampling interval (the space between samples) used in this signal must be calculated (Ts=1/Fs). The time vector must then be scaled to give the proper time values.

C. Compare the measured frequency with that on the tuning fork. What are possible sources of error? Use the zoom and cursor functions in Matlab figures.

D. Measure how the energy in the signal falls off over time for each of the tuning forks. Provide an annotated figure that shows how you measured the signal.

E. Repeat steps A-C using a sampling frequency of 44,100. Are your estimates of frequency more accurate? Why or why not? Give plots.

F. Frequency Analysis Compute the magnitude spectrum of the signals in step A using the method give in section 3 of this lab. Verify that the peaks occur at the frequency of the tuning fork. Turn in your plots. (Note: If your signal is not close, then your time window for collection is probably too long.) What is the amplitude in decibels of the signal in each case? Compare the tuning fork frequency estimated by the frequency spectrum with the one estimated using the time signal.

G. Comment on the other signals present in the spectrum. Why don't you see a nice clean delta function like in the notes?

5. Lab Report

A. Answer all questions in section 4 of this lab and include all requested plots and calculations of the frequency and magnitude decay estimation with clearly labeled graphs. Put them in a pdf file and submit electronically using the appropriate lab account. Fill in the table below as part of your report:

Signal Source	Frequency Estimate from Time Plot	Magnitude Decay Rate V/sec	Frequency Estimate from frequency plot
Tuning fork labeled 426.6 at 11,025 samp/sec	430 Hz	.02	428

B. Include all your MATLAB code in an appendix of your report. You do not need to include the tuning fork signals.