


## 1 Abstract

This lab introduces some basic tools that you will use throughout the course. The goals of this lab are: (1) To gain the ability to apply the basics of `Matlab` *that will be used in Engin 100*; (2) To learn the basics of sinusoidal signals, and (3) To learn the basics of sampling continuous-time signals. Projects later in the course will involve simple `Matlab` programs that process sampled continuous-time signals (music) to determine their sinusoidal components.

## 2 Background

### 2.1 Embedded audio and links

Throughout the PDF files for the digital signal processing (DSP) lecture notes and labs and projects there are audio examples that you can hear by using Adobe Acrobat Reader and clicking the play buttons. Other PDF readers may not be able to play the embedded audio. Here is an example: . You should hear a 2 second long 440 Hz tone when you click play, assuming you have a suitable reader, and ear buds connected to the sound card, and the system volume high enough, etc.

Also, throughout the PDF files are embedded URLs (uniform resource locator, aka web address), annotated using cyan boxed words. Clicking on any of them should open an Internet browser window related to the topic for more information. Try clicking on “Adobe Acrobat Reader” above and your browser should open to a corresponding Wikipedia page.

### 2.2 Matlab

`Matlab` is a computer program developed and sold by Mathworks. It is a commonly used mathematics computer program in signal processing, and it is used extensively in all engineering fields. As described in Section 3 below, UM students may obtain a free copy of the program for educational use. `Matlab` is available in all CAEN labs, and in central campus labs. There are open source alternatives such as `octave` and NumPy/Matplotlib, but they may not have all the functionality needed for this course.

“`Matlab`” is an abbreviation of MATrix LABoratory. It provides a convenient interactive environment for scientific computing. `Matlab` is particularly convenient for manipulating vectors and matrices, *i.e.*, arrays of numbers, and we will see that (digital) music signals are simply arrays of numbers.

A `Matlab` program is a list of `Matlab` commands executed in succession. This lab will teach you enough about `Matlab` to be able to use it. You will learn more in later Engin 100 labs, and also in Engin 101.

*Bring ear buds or headphones to lab, if possible. No speakers in CAEN labs.*

## 2.3 Sinusoids

Sinusoids have a central role in Engin 100. They are the building blocks of musical sounds.

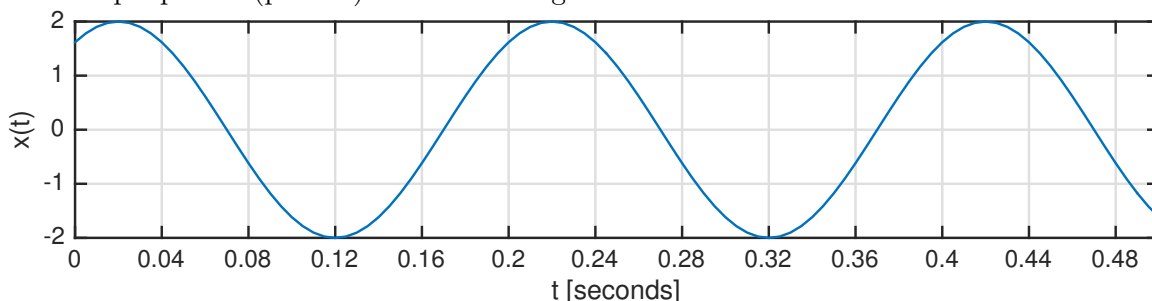
A *sinusoid* or *sinusoidal signal* or *sine wave* is a function or signal of the form

$$x(t) = A \cos(2\pi ft + \theta), \quad -\infty < t < \infty. \quad (1)$$

This sinusoid is a function of time  $t$ , typically measured in seconds. The signal  $x(t)$  is a function of  $t$ ; it associates a value of  $x$  with each value of  $t$ . A sinusoidal signal is defined in terms of the following three constants:

- $A$  = amplitude. The amplitude  $A$  and the signal  $x(t)$  have the same units (*e.g.*, “volts”).  
 $A \geq 0$  always. In music, if  $A$  increases then the tone sounds louder.
- $f$  = frequency in Hertz =  $\frac{\text{CYCLES}}{\text{SECOND}}$ .  
 $x(t)$  has *period*  $T = 1/f$  seconds, so  $x(t) = x(t + T) = x(t + 1/f)$  for all  $t$ .  
In music, this frequency is also called the *pitch*.
- $\theta$  = phase or phase shift, in radians or degrees. Phase =  $\theta$  is equivalent to time delay  $\tau = -\frac{\theta}{2\pi f}$  seconds.  
Another (less common) way of writing the sinusoid in terms of a time delay  $\tau$  is  $x(t) = A \cos(2\pi f(t - \tau))$ .  
Note that  $\sin(\alpha) = \cos(90^\circ - \alpha) = \cos(\alpha - 90^\circ)$ , so a sine is a cosine with phase  $\theta = -90^\circ$ .

Here is an example plot of (part of) a sinusoidal signal:



Let us “reverse engineer” this plot to express the sinusoid mathematically.

- Amplitude:  $A = 2$ . This is the maximum value of the signal because the maximum value of  $\cos$  is 1. (In contrast, the *peak-to-peak* amplitude = 4).
- Period: examine the time difference between two peaks:  $T = 0.22 - 0.02 = 0.42 - 0.22 = 0.2$  seconds.  
So the frequency is  $f = 1/T = 5$  Hertz.
- Time delay: normally the first peak of  $\cos$  is at 0, but here it is at  $\tau = 0.02$  seconds. That is the time delay.  
The phase is thus  $\theta = -2\pi 5\tau = -2\pi 5(0.02) = -\pi/5$  radians.
- So a mathematical formula for this sinusoidal signal is:  $x(t) = 2 \cos(2\pi 5(t - 0.02)) = 2 \cos(10\pi t - \pi/5)$ .

Note that “ $x$ ” is on the vertical axis here; get used to this!

## 2.4 Reading questions

You must read the description of each lab and project before the start of the corresponding lab section. To help ensure that, each lab/project writeup includes a few easy “reading questions” that you must answer on Canvas no later than 24 hours in advance of the start of your lab session. Here are the first two such questions. There are more later in the document. Submit your answers to Canvas after you finish reading this lab.

**RQ Lab1.1.** A sinusoidal signal has amplitude=8, frequency=100Hz, and phase  $\theta = \pi/3$ . What is the value of the signal at  $t = 0.02$  seconds? Because this is a warm-up problem, here is the answer. The problem is asking for  $x(0.02)$  where the sinusoidal signal is  $x(t) = 8 \cos(2\pi 100t + \pi/3)$  so we simply substitute  $t = 0.02$  into the formula:  $x(0.02) = 8 \cos(2\pi(100)0.02 + \pi/3) = 8 \cos(4\pi + \pi/3) = 8 \cos(\pi/3) = 8/2 = 4$ .

Even though the answer is given, you still must enter it into Canvas to verify you read the lab in advance!

**RQ Lab1.2.** A sinusoidal signal has frequency=50Hz and phase  $\theta = 2\pi/3$ . The value of the signal at  $t_1 = 0.01$  is  $x(t_1) = 7$ . What is the amplitude of the sinusoid?

## 2.5 Sampling

Unless you have an analog computer, continuous-time signals such as music must be *sampled* or *discretized* into a list of numbers that can be processed with a (digital) computer. The numbers themselves must also be *quantized* or *digitized* into a finite number of bits (finite precision). Engin 100 will not explicitly deal with quantization much, although we will see some of its effects in the next lab<sup>1</sup>. In signal processing, the term “sampling” does *not* mean incorporating a short segment of one recording into another, as is often done in certain music styles today.

Sampling is performed by analog-to-digital (A/D) converters. These are very sophisticated electronic components present in any digital audio system. Fortunately, all we need here is a simple mathematical model. The usual way to sample a continuous-time signal  $x(t)$  is to set  $t = n\Delta$  for integers  $n$  for a desired *sampling interval*  $= \Delta$  seconds. The corresponding *sampling rate* is  $S = \frac{1}{\Delta} \frac{\text{Sample}}{\text{Second}}$ . The units of the sampling rate  $S$  is “Hertz” or  $\frac{\text{Sample}}{\text{Second}}$ . When the continuous-time signal  $x(t)$  is the input to an A/D converter, the output is the following list of numbers that can be stored digitally:

$$\begin{aligned} &\{x(0), x(\Delta), x(2\Delta), x(3\Delta), \dots\} = \\ &\{x[0], x[1], x[2], x[3], \dots\}, \end{aligned}$$

where we relate the digital signal  $x[n]$  and the analog signal  $x(t)$  as follows:

$$x[n] = x(n\Delta) = x(t) \Big|_{t=n\Delta} = x\left(\frac{n}{S}\right) = x(t) \Big|_{t=n/S}. \quad (2)$$

If  $x(t)$  has finite *duration* (length), *i.e.*, it starts at some time and stops at another, as all music does (*e.g.*, musical chairs), then  $x[n]$  is a finite list of numbers. This makes  $x[n]$  suitable for computer processing.

For a sinusoidal signal sampled at  $S$  “Hertz”  $= S \frac{\text{Sample}}{\text{Second}}$ , (*i.e.*, every  $\Delta = \frac{1}{S}$  seconds), the sampled signal is

$$x[n] = x(n\Delta) = x(t) \Big|_{t=n\Delta} = A \cos(2\pi f \Delta n + \theta) = A \cos\left(2\pi \frac{f}{S} n + \theta\right). \quad (3)$$

Note that  $f/S$ ,  $f\Delta$  and  $n$  are all dimensionless. It is usually desirable to put equations in dimensionless form.

It might seem that we will lose information about  $x(t)$  by storing only its samples  $x[n] = x(n\Delta)$ . How can we recover  $x(t)$  from  $x(n\Delta)$ , *e.g.*, for audio playback? Amazingly, we will see later in Engin 100 that we can reconstruct  $x(t)$  *exactly* from its samples  $x[n] = x(n\Delta) = x(n/S)$  as long as  $S > 2f_{\max}$ , where  $f_{\max}$  denotes the maximum frequency of  $x(t)$ . This fact was discovered by Claude Shannon, a UM alumnus. His bust is outside the EECS building.

To illustrate these concepts, consider a standard music CD that holds 64 minutes of music. The original analog musical signal is *sampled* at  $S = 44100 \frac{\text{Sample}}{\text{Second}}$  and *quantized* using 16 bits per sample (providing  $2^{16} = 65536$  possible different signal values per sample). There are two channel components because it is stereo (although early Beatles recordings were monophonic, among others). So we can determine how many bytes are stored on the CD as follows (watching the units):

$$\left[ \frac{1 \text{ byte}}{8 \text{ bits}} \right] \left[ 16 \frac{\text{bits}}{\text{channel}} \right] \left[ 2 \frac{\text{channel}}{\text{sample}} \right] \left[ 44100 \frac{\text{sample}}{\text{second}} \right] \left[ 60 \frac{\text{second}}{\text{minute}} \right] [64 \text{ minutes}] = 677 \text{ MBytes}.$$

**RQ Lab1.3.** A sinusoidal signal has amplitude=6, frequency=2000 Hz, and phase  $\theta = 0$ . The signal is sampled at rate  $S = 8000 \frac{\text{Sample}}{\text{Second}}$ . What is the value of the sampled signal  $x[n]$  when  $n = 4$ ?

Now we introduce `Matlab` so that we can plot signals like sinusoids, manipulate them, and listen to them. But first we start with the basics.

---

<sup>1</sup> If you buy a music CD (does anyone still do that?) and import it to another digital format such as MP3, then the import software (*e.g.*, iTunes) will likely give you options to set the encoding quality to high or low; this choice influences quantization.

### 3 Getting started with Matlab

UM students can obtain a copy of Matlab software at no cost:

<http://caenfaq.engin.umich.edu/10378-Free-Software-for-Students/matlab-for-students>

<https://www.itcs.umich.edu/sw-info/math/MATLABStudents.html>

If you have your own computer, it may be convenient for you to have a free copy.

#### 3.1 Running Matlab

Matlab is available for Mac, Linux and Windows computers. You can use any such computer provided it has a sound card for audio. The instructions for logging into a CAEN computer and starting Matlab change frequently with operating system (OS) and software upgrades. Here is a link to the F15 instructions: <http://goo.gl/MXDsS9> Your lab instructor will demonstrate how to login and start Matlab during your first lab session. There is also a lot of documentation on the CAEN web site <http://caen.engin.umich.edu/>

- After you login and start Matlab, eventually the Matlab desktop window will open, and you will see “command window” with a >> prompt.
- You will type commands at that prompt. Hereafter we use **this font** to denote Matlab commands that you type at the prompt. (To save typing, you can “cut and paste” text from this document into the Matlab window.)
- After you type one or more commands, if you want to execute one of them again, just tap the up arrow key until the command is shown again. You can also use the arrow keys and mouse to edit (modify) one of those previous commands and then hit enter (return) to run it.
- You can use Matlab remotely, *e.g.*, via the *Windows Remote Desktop Service*, following instructions here: <http://caen.engin.umich.edu/connect/overview>  
Of course, if you are doing that from your residence then you probably have your own computer so installing Matlab is probably preferable.

## 3.2 Vectors and Arrays

Because **Matlab** is based on linear algebra, when using **Matlab** you must think in terms of vectors and arrays, aka *matrices*. Examples of a *row vector*, a *column vector*, an *array*, and the array's *transpose* in mathematical notation are, respectively:

$$x = [ 3 \quad 1 \quad 4 ] \quad y = \begin{bmatrix} 3 \\ 1 \\ 4 \end{bmatrix} \quad Z = \begin{bmatrix} 3 & 7 & 4 \\ 1 & 5 & 9 \end{bmatrix} \quad Z' = \begin{bmatrix} 3 & 1 \\ 7 & 5 \\ 4 & 9 \end{bmatrix}.$$

In **Matlab** syntax you type these in as: `x = [3 1 4]; y = [3;1;4]; Z = [3 7 4; 1 5 9]; Z'`

You could also type `y=x'` where the **transpose** operation `x'` means convert rows to columns, and vice-versa. Note the convenient similarity between the mathematical notation and the **Matlab** syntax.

A common mistake in using **Matlab** is thinking you that have a row vector when in fact you have a column vector. You can check by using `size(x)`; here that gives `ans = 1 3` which tells you that `x` is a  $1 \times 3$  array, which we call a row vector.

Try the following commands and note what they are doing (**reshape** will be *very* useful later):

- `A = [1 2 3;4 5 6]`, `B = A(:)` (stacks A by columns).
- `reshape(B,3,2)` (converts the column B into a  $3 \times 2$  array).
- `x = [3 1 4 1 5 9 2 6 5 3 5]`; for `n=1:10`; `d(n) = x(n+1) - x(n)`; `end`; `d`

This takes differences between the elements of `x`.

What is the size of the vector `d`?

- `x = [3 1 4 1 5 9 2 6 5 3 5]`; `c = x(2:11) - x(1:10)`

This is a faster way to take differences of `x`.

A more elegant way is: `c = x(2:end) - x(1:end-1)`

The simplest way of all uses the built-in **Matlab** command `diff` as follows: `c = diff(x)`

- Try to predict what the result will be before you enter the following: `x = [10 20 30 40]`; `s = sum(x)`
- `D = [1 2 3]`; `E = [4 5 6]`; `F = [D E]`  
The operation `[D E]` *concatenates* vectors D and E, *i.e.*, it appends E after D to get the longer vector F.
- The colon is useful for generating arrays of values. Try these commands and think about the output. You will need to use such commands frequently:  
`x = 3:7`, `t = 2:1/10:3`, `y = 10:2:20`, `z = 40:-1:30`
- At any point if you want to display the value of an array you can either just type the array name, *e.g.*, `x`, and press enter, or use the display command: `disp(x)`
- The command `whos` shows you all the variables you have defined. If you want to start over, type `clear`.

## 3.3 Plotting

A particularly common **Matlab** task you will perform is plotting (samples of) functions.

Try running the following examples. (No, this is not part of the lab you will turn in yet):

- `t=[1:100]`; `x=cos(2*pi*t/30)`; `plot(x)` This plots a sinusoidal signal with period=30.
- `t=[1:100]`; `y=2.^(t/20)`; `plot(y)` This plots an exponential function growing in time.
- `t=[1:100]`; `z=2.^(-t/20) .* cos(2*pi*t/30)`; `plot(z)` This plots a decaying sinusoid.

Typing `t` and `x` will display for you a row vector of 100 numbers. You can get column vectors by typing `t'` and `x'`. You can display them as side-by-side columns by typing `[t' x']` (think in terms of vectors or arrays).

- You can plot multiple functions at once (note the colors): `plot(t,x, t,z)`
- You can plot `x` vs. `y`: `plot(x, y, '-o')`

In such cases the number of elements in `t` and `x`, or `x` and `y`, must be the same.

### 3.4 Miscellaneous

These simple examples illustrate several important points about `Matlab`.

- Defining `x` as a function of `t` produces a vector of `x` values corresponding to the vector of `t` values. (In many other programming languages you would have to write a loop that is less concise.)
- Putting a semicolon “;” at the end of a command suppresses output; without it `Matlab` will type the results of the computation on your screen. This is harmless, but can be annoying.
- The operation “`.*`” multiplies two vectors *element-by-element*, whereas “`*`” multiplies vectors as in linear algebra.  
Try these: `[1 2] * [3 4]'` and `[1 2]' * [3 4]` and `[1 2] .* [3 4]` and `[1 2] * [3 4]`  
(The last one gives an error message about dimensions.)
- Compare `[0:.1:1]` (11 numbers spaced by 0.1) and `linspace(0,1,10)` (10 numbers spaced by 0.111). This difference is sometimes called the *picket fence problem*. If you have a 30 foot fence with pickets every 3 feet, how many pickets are there? Answer: 11 (not 10). Programmers often introduce bugs by disregarding this distinction.

### 3.5 Getting help for Matlab

- You can get help for a Matlab command, say, `linspace` by typing `help linspace` or `doc linspace`.
- See <http://web.eecs.umich.edu/~aey/eecs451/whywont.html> for ideas on what is wrong with your program.
- See <http://web.eecs.umich.edu/~aey/eecs451/matlab.pdf> for a tutorial on Matlab (for another course).

### 3.6 Programs (.m files)

You can access previously-typed commands using up-arrow and down-arrow on your keyboard. But typing gets old. Instead, you can write (and save) a *program* or *script* (a sequence of Matlab commands) using *.m files*. At the upper left of the Matlab window, click:

**File** → **New** → **m-file**

This opens a window with a text editor. Type in commands (just like you would at the prompt) and then:

**File** → **Save as** → `lab1.m` (or whatever you want to name your file).

Make sure you save it with an `.m` extension. Then you can run the file by typing its name at the prompt: `>> lab1`  
Make sure the file name is not the same as a Matlab command!

To *download a file* from a web site, right-click on it, select **save target as**, and use the menu to select the proper file type (specified by its file extension). All m-files should have a `.m` extension (such as `lab1.m`).

### 3.7 Printing

You can print out the current figure (the one in the foreground; click on a figure to bring it to the foreground) by typing `print` at the prompt or by clicking the printer icon. But usually instead of printing to paper you will be saving figures electronically. To save a figure as a PNG file, type `print -dpng lab1fig1.png` for example. Type `help print` for a list of printing options. Alternatively, click on the “save figure” icon (floppy disk) and then select PNG format.

You can insert a `.png` image file into a Google Doc by dragging the figure file icon into the document, or by using the menu **Insert**→**Image**.... You will do this frequently in your lab reports and presentations.

Printing one plot on a page usually wastes paper and money. You can put several plots on a page using the `subplot` command. `subplot(mni)` produces an `m`×`n` array of plots; `i` is a number between 1 and `mn` that specifies where in the array the next plot is to be placed. Try (during the same Matlab session as above):

```
clf, subplot(311), plot(t,x), subplot(312), plot(t,y), subplot(313), plot(t,z)
clf, subplot(321), plot(t,x), subplot(323), plot(t,y), subplot(326), plot(t,z)
```

To exit Matlab, type `quit`.

**RQ Lab1.4.** What Matlab command shows you all the variables you have defined?

## 4 Lab 1: What you must do

### 4.0 Start a Google Doc for your lab results

After completing this lab you will submit a PDF file of your results to Canvas for grading. You will generate the PDF file using Google docs. So before starting, point your browser to <http://drive.google.com> and login (with your umich account) and create a new document. Put your name, date, and section number at the top of the document. Put all all of the requested figures and answers into this document. Ask your lab instructor for help if needed. If you have not used it already, Google Docs is worth learning and using for this report because it allows collaborative editing that will be useful later for the team projects. (This lab is an individual assignment.)

### 4.1 Sinusoids

[5] Use `subplot` to make a figure consisting of the four plots described below.

Hint: type `subplot(221)` before doing part (a).

(a) Type `t=[0:10]; x=3*cos(t)+4*sin(t); plot(t,x, '-')` This should be a jagged-looking plot.  
The reason is that it is only sampled at integers, and samples are connected by straight lines.

(b) Type `t=[0:0.1:10]; x=3*cos(t) + 4*sin(t); plot(t,x)` This should be a smoother plot.  
Are you surprised that the sum of a sin and a cos is a pure sinusoid? See the Appendix below for why.  
Did you forget to use `subplot` here? If so, rather than retyping, use the up arrow.

(c) Type `n=[1:4000]; x = cos(2*pi*440*n/8192); sound(x)` This should sound like tone note “A.”

(d) Type `plot(x)` This should be a blue smear! It is about 200 cycles squished together.

(e) Type `x(1:8)`  
This extracts the first 8 elements of the vector `x` and displays them.  
You will use frequently this way of extracting certain elements of a vector.

(f) Type `plot(x(1:100))`  
This extracts the first 100 samples of `x` and then plots them so we can better see (part of) the sinusoid.

(g) Type `sound(x(1:2:end))` This extracts every other value of `x`, halving its length. The sinusoid is squished together and effectively is sped up by a factor of two, so the tone is now “A” one octave higher.

(h) Add a title using `title('Lab 1 Figure 1')` and save this four-plot figure as a PNG file and import it into your Google Doc.

### 4.2 Non-sinusoidal signals

[5] Use `subplot` to make a figure consisting of the four plots described below.

(a) Type `load train; whos; sound(y)`  
This signal (included in `Matlab`) should sound like a train whistle. The `whos` command shows you that the file `train` contains two variables: `y` contains the signal and `Fs` contains the sampling frequency `S`.

(b) Type `plot(y)` (blue smear) and `n = 1501:1700; plot(n, y(n), '-')` to zoom in.  
Note that this signal is approximately periodic.

[2] Q1: How many samples are displayed in the second plot? (The answer is not 1700!)

(c) Type `sound(y(1:2:end))` Again this doubles the pitch and halves the length.

(d) Type `z = y' .* cos(2*pi*1000*[1:length(y)]/8192); sound(z)`  
This is another way to alter pitch, called *modulation*.  
Multiplying by this cosine shifts all of the frequencies of `y` both up and down by 1000 Hertz.

(e) Type `plot(z)` (blue smear) and `n = 1501:1700; plot(n, z(n), '-')` to zoom in.  
Note how this signal `z` differs from the original signal `y`.

(f) Add an appropriate title, save as PNG, and import to your report document.



### 4.3 Sum of two sinusoids

[5] Use `subplot` to make a figure consisting of the four plots described below.

(a) Type `t=[1:8192]/8192; z=cos(2*pi*440*t) + cos(2*pi*444*t); sound(z)`

Describe this sound. Does it sound like two tones close together in pitch? Or like a single tone with a time-varying amplitude?

(b) Type `plot(z)` and `n = 1501:1700; plot(n, z(n), '-')` to zoom in on `z`.

You need both of these plots. Now can you *see* why this sounds the way it does? See Appendix below for why.

(c) Type `t=[0:.01:.99]; f=2; x=cos(2*pi*f*t); plot(x, '-')` Just a sampled 2 Hz sinusoid.

(d) Type `t=[0:.01:.99]; f=98; x=cos(2*pi*f*t); plot(x, '-')` Just a sampled 98 Hz sinusoid.

(e) Notice something unsettling? Try this:

`t=[0:.01:.99]; x=cos(2*pi*98*t) - cos(2*pi*2*t); plot(x)` (Do not print this plot).

Sampling 2 Hz and 98 Hz sinusoids at  $\Delta = 0.01$  seconds, *i.e.*,  $S = 100$  Hz, yields the same result! This phenomena is called *aliasing*. We will learn later that we must sample much faster than 100 Hz to distinguish a 98 Hz signal from lower frequency signals.

(f) Add an appropriate title, save as PNG, and import to your report document.

(g) Type `t=[1:8192]/8192; y = 0.6*cos(2*pi*440*t) + 0.6*sin(2*pi*440*t); sound(y)`

Does this sound like one tone or two? (See Appendix.)

### 4.4 Do not always believe computer software

- Is software like `Matlab` infallible? Type `roots(poly(ones(20,1)))`

This forms the polynomial  $(z - 1)^{20}$  having 20 multiple roots at  $z = 1$ , and then computes the roots of that polynomial. Did you get 20 ones? No!

- To see what happened, type `zplane(roots(poly(ones(20,1))))`

(This will not work in labs that do not have `Matlab`'s Signal Processing Toolbox installed, such as possibly the Central Campus computer labs.) Are computers infallible? Software like `Matlab` has numerical problems for polynomials with multiple roots. (You need not print or save this plot.)

### 4.5 Questions

[2] Q2. Determine how many elements are in the array `x` produced by the command `x = 3:0.01:5`

[2] Q3. Determine the spacing between the elements of the array `x = linspace(3,5,200)`

[3] Q4. Use `Matlab`'s `sum` command to determine the sum of the values  $300 + 301 + \dots + 400$ . Hint: use a colon. Report your command and the result.

[3] Q5. A sinusoidal signal with amplitude 6, frequency 10 Hz, and phase  $2\pi/15$  is sampled every 2 msec using the ideal sampling formula (2). What is the value of the digital signal  $x[n]$  when  $n = 5$ ?

Hint: the numbers have been designed here so that you could do this without a calculator, though some students may need to review the unit circle.

[5] Q6. Plot the signal  $\sin(2\pi 100t)$  for samples of  $t$  spaced by  $1/8192$  seconds over the interval  $[3, 3.1]$  seconds.

### 4.6 Lab report

Add to your Google Doc report the answers to the 6 questions above. Then use the Google Doc menu `File`  $\rightarrow$  `Download As`  $\rightarrow$  `PDF Document` to export a PDF file. Upload that PDF file to Canvas. If possible, show it to your lab instructor first.

If needed, you should keep seeking help from classmates and your lab instructor until you get all 30 points (and all the concepts). Hopefully you will finish it all during the first lab. If you need more time, you may have until the start of Lab 2 to complete it. Regardless, you must read Lab 2 thoroughly before coming to the next lab section.



## 5 Appendix: sum of sin and cos of same frequency

- Why is the sum of a sine and a cosine (of same frequency) in Section 4.3(g) a pure sinusoid rather than something more complicated?
- Why did the sum of two sinusoids at close frequencies in Section 4.3(a) appear to be one sinusoid with varying amplitude?

Answers to both questions follow from the trigonometric identity called the *cosine angle difference formula*:

$$\cos(\alpha - \beta) = \cos(\alpha) \cos(\beta) + \sin(\alpha) \sin(\beta).$$

- First, set  $\alpha = 2\pi ft$  and  $\beta = \theta$  and multiply by  $C$  to get

$$C \cos(2\pi ft - \theta) = C \cos(\theta) \cos(2\pi ft) + C \sin(\theta) \sin(2\pi ft) = A \cos(2\pi ft) + B \sin(2\pi ft).$$

Setting  $t = 0$  yields the equality:

$$A = C \cos(\theta).$$

Setting  $t = \frac{1}{4f}$  yields the equality:

$$B = C \sin(\theta).$$

Combining the formulas for  $A$  and  $B$  yields (think about polar coordinates):

$$C = \sqrt{A^2 + B^2}, \quad \tan \theta = \frac{B}{A}.$$

In other words, the sum of two sinusoids of the *same frequency* is just another sinusoid of that frequency but with a different amplitude and phase:

$$A \cos(2\pi ft) + B \sin(2\pi ft) = \underbrace{\sqrt{A^2 + B^2}}_{\text{amplitude}} \cos(2\pi ft - \underbrace{\arctan(B/A)}_{\text{phase}}) \quad (4)$$

Lab 3 uses this equation extensively.

- Second, set  $\alpha = 2\pi 442t$  and  $\beta = \pm 2\pi 2t$  and add and subtract the results to get:

$$\cos(2\pi 440t) + \cos(2\pi 444t) = 2 \cos(2\pi 2t) \cos(2\pi 442t).$$

So 440 Hz and 444 Hz sinusoids added together are identical to a 442 Hz sinusoid with a time-varying amplitude. Mathematically these two sides of the above equation are identical, but our ears “hear” the right-hand side. We will use this result (think tuning a piano) in Lab 2 to measure frequency.