Name: _____

Date of lab: _____ Section number: M E 345._____

Lab 5

Precalculations – Individual Portion
Digital Data: Data Acquisition and FFTs

Precalculations Score (for instructor or TA use only):	_____ / 20

1. **(2)** The digital data acquisition system in this lab utilizes an A/D converter that is 16-bit, and has an input range of −10 to 10 V. How many bins or levels of A/D output are available?

2. **(2)** What is the resolution of this system (give your answer both as an overall resolution and as a +/- resolution, i.e., ***quantization error*** in millivolts to 3 significant digits).

3. **(3)** For the system used in this lab, suppose the voltage to be measured in a certain experiment is always positive, and never exceeds 2.5 V. How many of the 16 bits of the A/D converter are effectively utilized?

4. **(4)** Consider a pure sine wave signal at 50 Hz. 256 discrete data points are taken at f_s = 200 Hz, and an FFT is calculated. Calculate the total sampling time, the frequency resolution, and the maximum *useful* frequency of the resulting frequency spectrum.

5. **(4)** Suppose a signal from a vibrating motor is to be sampled discretely with a digital data acquisition system to determine the RPM of the motor. It is known that the maximum possible RPM is 1800. What is the minimum sampling frequency (in Hz) with which the signal can be sampled in order to measure the RPM of the motor? For full credit, show all your work and explain.

6. **(2)** For the same conditions as above, suppose also that the RPM needs to be known to a resolution of 2 RPM (±1 RPM). Calculate the minimum number of discretely sampled data points necessary to achieve this resolution.

7. **(3)** In this lab you will be asked to build a mechanism that will be actuated by a small Lego motor. You will use Lego Technic pieces to build this mechanism. Legos are simple to build with, but if you have never used them before, you should do a quick search on the Internet and familiarize yourself with what pieces are available and how to connect them. A good starting point is www.legoengineering.com. Search for NXT Constructopedia or NXT Building Tips to find some introductory building instructions. Read the section of the lab procedure titled "Lego Mechanism Acceleration" and provide a sketch of an experimental setup that you believe will meet the objective specified. Be sure to indicate where the motor will be connected to your mechanism, where you will measure acceleration and how you think your setup will move.

Cover Page for

Lab 5

Lab Report – Group Portion

Digital Data: Data Acquisition and FFTs

Name 1: _____ Section M E 345._____

Name 2: _____ Section M E 345._____

Name 3: _____ Section M E 345._____

[Name 4: _____ Section M E 345._____]

Date when the lab was performed: _____

Group Lab Report Score (For instructor or TA use only):

Lab experiment and results, plots, tables, etc.	_____ / 50
Discussion	_____ / 30
TOTAL	_____ **/ 80**

Lab Participation Grade and Deductions – The instructor or TA reserves the right to deduct points for any of the following, either for all group members or for individual students:

- Arriving late to lab or leaving before your lab group is finished.
- Not participating in the work of your lab group (freeloading).
- Causing distractions, arguing, or not paying attention during lab.
- Not following the rules about formatting plots and tables.
- Grammatical errors in your lab report.
- Sloppy or illegible writing or plots (lack of neatness) in your lab report.
- Other (at the discretion of the instructor or TA).

Name	Reason for deduction	Points deducted	Total grade (out of 80)

Comments (for instructor or TA use only):

Digital Data: Data Acquisition and FFTs

Author: John M. Cimbala; also edited by Michael Robinson and Savas Yavuzkurt, Penn State University
Latest revision: 03 October 2014

Introduction and Background (*Note*: To save paper, you do *not* need to print this section for your lab report.)
Prior to the 1980s, the oscilloscope and strip-chart recorder represented the most common methods for measurement of time-varying signals. However, with the advent of the personal computer and the introduction of PC-compatible data acquisition cards, PC-based digital data acquisition became standard in most laboratories by the late 1980s. By combining high speed data acquisition cards with graphical software, it is now possible to design complex data acquisition systems with real-time data analysis and plotting features, with minimal programming. The data acquisition hardware converts analog inputs into the digital domain at the specified sampling rate, and the software manipulates and displays the desired output. The most important difference between digital data acquisition and analog data acquisition is that while an analog device captures a signal continuously, a digital device samples *discretely* at some specified sampling rate – information in between two discrete data points is *not* recorded. Computers are not the only devices that employ digital data acquisition. Digital multimeters, digital oscilloscopes, and many other specialized laboratory instruments also employ analog-to-digital (A/D) converters and digital data acquisition. Audio devices like MP3 players convert the other way (digital-to-analog, or D/A) to drive the speakers. Communications equipment such as digital cellular phone systems use *both* A/D and D/A converters.

In this lab we use both ***MATLAB*** and ***LabVIEW*** (Laboratory Virtual Instrumentation Engineering Workbench) for digital data acquisition. In MATLAB, some simple subroutine commands are run in order to acquire digital data. In LabVIEW, ***virtual instruments*** (VI's) are designed graphically by arranging icons on a worksheet. Each icon represents some element of the system, and the icons are connected by "wires" to control the data flow. Once the virtual instrument is constructed, instructions are issued to the data acquisition hardware, either inside the PC or external to the PC (the hardware we use in our lab is connected through the USB port).

Precautions about using digital data acquisition
Due to the nature of the A/D conversion process, it is necessary to take several precautions to ensure that the analog signal is adequately represented once it is converted to the digital domain. An experimenter who is not careful can encounter problems such as *poor resolution*, *clipping*, and *aliasing*. The most important factors when sampling a signal are *dynamic range*, *input range*, and *sampling rate*.

The ***dynamic range*** (***DR***) of an A/D converter is the ratio of the largest voltage (V_{max}) to the smallest *change* in voltage (ΔV) that can be measured, but *DR* is usually expressed in decibels. *DR* is also related to the number of discernible levels or bins that the A/D converter can assume. Since there are 2^N available levels or bins for an *N*-bit A/D converter, *DR* is expressed as: monopolar - $DR = 20\log_{10}\left(2^N\right)$, bipolar - $DR = 20\log_{10}\left(2^{N-1}\right)$. For example, a 12-bit A/D converter with a range of -10 to 10 V is *bipolar*, and can measure a maximum voltage of 10 V. The smallest change in voltage that can be measured is equal to the resolution of the A/D, which is $\Delta V = \dfrac{\left(V_{max} - V_{min}\right)}{2^N}$. Here, $\Delta V = \dfrac{\left(10-(-10)\right)}{2^{12}} = 0.0048828$ V. The dynamic range is therefore $DR = \dfrac{V_{max}}{\Delta V} = \dfrac{10 \text{ V}}{0.0048828 \text{ V}} = 2048$, which in decibels is $DR = 20\log_{10}\left(2048\right) = 66.2$ dB. Or, using the above equation for a bipolar A/D, we get the same result, $DR = 20\log_{10}\left(2^{12-1}\right) = 66.2$ dB. Note that this particular A/D has 2^{12}, or 4096 bins. If this same A/D converter were *monopolar*, its dynamic range would be larger, namely, $DR = 20\log_{10}\left(4096\right) = 72.2$ dB.

It is important to utilize as much of the available dynamic range as possible. For example, if the ***input range*** of a 12-bit data acquisition card is set for 0 to 10 V, but the input signal varies between 0 to 100 mV (0.1 V), only 1 percent (0.1 V divided by 10 V) of the possible dynamic range, or only about 41 of the possible 4096 levels, are utilized – this is very poor resolution of the signal! When the full dynamic range of the A/D converter is not utilized, the resolution of the measured signal suffers. In this example, we should either *amplify* the signal before sending it to the data acquisition system, or adjust the input range of the A/D converter to something more in line with the expected data. For example, the input range of many data acquisition cards can be set to 0 to 1 V. Even in this case,

only about 10% of the dynamic range of the A/D converter is utilized in the above example, but the resolution of the signal is ten times better for the 0 to 1 V case than for the 0 to 10 V case.

Another potential problem with digital data acquisition is ***clipping***. If a voltage lies beyond the input range of the A/D converter, the signal is clipped. For example, an A/D converter with an input range of 0 to 10 V is not able to distinguish voltages above 10 V from voltages equal to 10 V – the signal is clipped at 10 V. Similarly, any negative voltages are indistinguishable from a zero volt signal – the signal is clipped at 0 V.

Significant measurement errors called ***aliasing*** errors are also possible if the waveform is not sampled at high enough frequency. To avoid aliasing, the ***sampling rate*** must be at least twice the maximum frequency of the measured signal. This restriction is called the ***Nyquist criterion***. Signal aliasing occurs when waveforms are sampled at frequencies *below* the Nyquist frequency. Aliased signals appear to have frequencies (and possibly even waveform *shapes*) that differ from those of the actual signal. For adequate resolution of the waveform shape, data should be sampled at a much higher frequency – typically at least five times the Nyquist frequency, if possible.

Digital PC-based data acquisition will not totally replace oscilloscopes, at least not in the near future. The reason is sampling frequency. The maximum sampling frequency of modern PC A/D systems is typically less than a MHz (megahertz). By comparison, a good digital oscilloscope may sample as high as several GHz (gigahertz)!

The Fast Fourier Transform (FFT)

The ***fast Fourier transform*** (***FFT***) is a computationally efficient form of the more general ***discrete Fourier transform*** (***DFT***), which is itself a discretized version of the even more general ***Fourier transform*** (***FT***). Like Fourier series analysis, FFT analysis enables us to calculate the frequency content of a signal. Fourier series analysis is useful for continuous, periodic, *analog* signals of known fundamental frequency. FFT analysis, on the other hand, is useful for discretely sampled (*digital*) data, and can be applied even if the signal is not periodic. With FFT analysis, the fundamental frequency of a periodic signal does not have to be known a priori. Both MATLAB and LabVIEW have built-in FFT features, which are utilized in this lab.

For N sampled data points at ***sampling frequency*** f_s, the most useful output of an FFT calculation is the ***frequency spectrum*** or ***amplitude spectrum***, which is a plot of modified FFT amplitude versus frequency. The frequency spectrum shows the relative importance or contribution of discrete frequencies, which range from zero to $f_s/2$. (The factor of two is a direct result of the Nyquist criterion.) The number of discrete frequencies on the frequency spectrum plot is $N/2 + 1$. This is half of the number of discretely sampled data points in the original signal, plus one extra since we typically plot *both* extreme values – from zero Hz (DC component) to the folding frequency $f_{folding}$. Other details about FFTs, along with equations and examples, are provided in the lecture notes. Here, a brief summary is provided of the parameters that are important when using FFTs in a laboratory experiment:

- N is the ***number of discrete data points*** taken. N is an input parameter, chosen by the user.
- f_s is the ***sampling frequency***, in Hz. f_s is an input parameter, chosen by the user.
  All other properties of the FFT, including sampling time, maximum frequency, frequency resolution, etc., are determined solely from these two inputs, N and f_s.
- T is the ***total sampling time***, and is calculated as $T = N/f_s$. To increase the sampling time, we must either *increase* the number of data points, or *decrease* the sampling frequency (or both).
- $f_{folding}$ is the ***folding frequency***, also called f_{max}, the ***maximum frequency***. $f_{folding} = f_s/2$. $f_{folding}$ is the maximum frequency plotted on the frequency spectrum plot, since $f_{folding}$ is the maximum frequency at which reliable information about the signal can be calculated, due to the Nyquist criterion. The *only* way to increase $f_{folding}$ is to increase the sampling frequency.
- Δf is the ***frequency resolution*** or ***frequency increment*** of the frequency spectrum. $\Delta f = 1/T = f_s/N$. On the frequency spectrum plot, amplitudes of the FFT are plotted at $N/2 + 1$ discrete frequencies, each separated by Δf. In other words, the discrete values of f are 0, Δf, $2\Delta f$, $3\Delta f$, ... , $[(N/2 - 1)]\Delta f$. [The amplitude at exactly $f_{folding}$, i.e., at $(N/2)\Delta f$, is *also* plotted; this results in a total of $(N/2) + 1$ discrete frequencies, counting both $f = 0$ and $f = f_{folding}$.] The *only* way to increase the frequency resolution is to increase sampling time.

Here is a summary of some useful techniques and rules to remember when calculating FFTs:

- To get better frequency resolution for a fixed sampling frequency, increase the number of data points.
- To get better frequency resolution for a fixed number of data points, decrease the sampling frequency. (But be careful here not to let f_s fall below the Nyquist criterion limit.)

- To get frequency component information at higher frequencies, increase the sampling frequency.
- To reduce ***leakage*** in the frequency spectrum (see lecture notes for a discussion about leakage), do one or more of the following:
  - Increase the number of sampled data points N (at the cost of more computer time).
  - Decrease the sampling frequency f_s (but do not sample at such a low frequency that the Nyquist criterion is violated).
  - Multiply the time signal by a ***windowing*** function prior to taking the FFT (at the cost of throwing away a significant portion of the signal, in particular data points near the start and finish of the time trace).

A healthy human ear can hear sounds in the range of about 20 Hz to about 20,000 Hz. Therefore, the standard data acquisition rate for digitally recorded music (CD audio and MP3 or MP4) is 44100 Hz – to avoid aliasing at even the very highest frequencies that can be heard. If music were sampled at a significantly lower frequency than this, aliasing would occur, which would distort the sound and lower the quality of the recording.

Accelerometers come in a number of different configurations and are used to measure acceleration. The Lego accelerometer can measure two types of acceleration; static and dynamic. Static acceleration is caused by gravitational force. The object may be stationary, but the accelerometer will still detect an acceleration in the direction gravity is acting. For example, if you point the accelerometer down, it will measure –9.8 m/s^2. If you point the accelerometer up, it will measure +9.8 m/s^2. *Dynamic* acceleration, on the other hand is the acceleration associated with a moving object. In this lab you will use an accelerometer to measure the motion of a mechanism you will design, and then you will use the acceleration data to estimate the rotational velocity of a motor. A picture of a Lego accelerometer and a Lego motor can be seen in Figures 1 and 2. The program you will be given will log acceleration data in the x direction indicated in Figure 1.

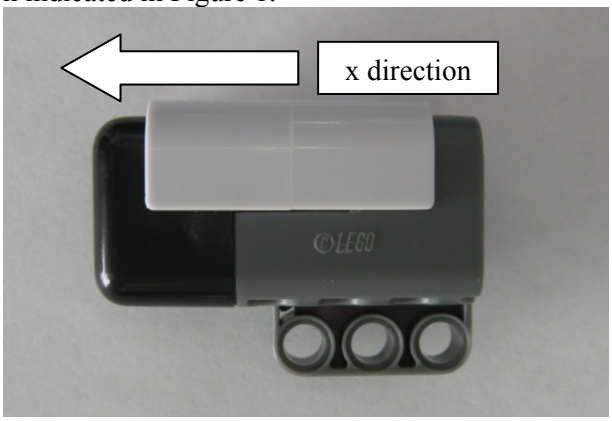

Figure 1: Lego accelerometer with positive *x* direction shown

Figure 2: Lego motor. Note that the entire orange piece rotates with the motor.

You will also be using the Lego Mindstorms brick, known as the NXT, a labeled image of which is shown in Figure 3. The NXT has internal A/D (analog to digital) converters and can collect data from a wide variety of sensors. Sensors attach to the bottom of the device through the ports labeled 1, 2, 3 and 4. The NXT can also control motors through the three ports labeled A, B, and C on the top of the device. By connecting a USB cable to the top of the device, a computer can communicate with the system.

Figure 3: Lego NXT brick with buttons and connections indicated

Objectives

1. Become familiar with the basics of data acquisition using both MATLAB and LabVIEW.
2. Experience music distortion due to aliasing.
3. Experience clipping, aliasing, and other pitfalls of digital data acquisition.
4. Experiment with FFT analysis of signals, and become familiar with the parameters critical to FFTs.

Equipment

- personal computer with digital data acquisition, sound card, and speakers
- software: LabVIEW virtual instrumentation software, MATLAB, and Microsoft Office
- function generator, along with appropriate cables
- digital oscilloscope
- Music player (your cell phone or your MP3/MP4 player, the lab computer, etc)
- 3 stereo patch cords (1/8-inch) and 1 headphone splitter jack
- Microphone
- Lego NXT Kit

Procedure

Note: When using the front microphone input jack to play music from your cell phone, iPod, MP3/MP4 player, or other device, you need to turn the microphone boost down to +0dB, otherwise the computer will clip the signal, and the sound will be distorted. When using the actual microphone, you need to increase the gain back up to around +30dB, otherwise you'll get almost no signal. Note also that on some of the lab computers, you may need to use the *rear* microphone input jack instead of the front one.

Introduction to aliasing using MATLAB, the computer sound card, and music

1. Set the computer playback and recording volumes to their maximum levels: Right click on the Speakers icon in the task bar and select Playback devices (this should open a window Sound). In the *Playback* tab, select *Speakers* and click on *Properties*. In this window select the *Levels* tab and drag volume to the highest level. Return to the Sound window and select the *Recording* tab, select *Mic* and click on *Properties*. In this window select the *Levels* tab and drag volume to the highest level. Be sure to note which microphone and speaker jacks are being used (default).
2. Open a new m-File in MATLAB: Start-All Programs-MATLAB(version)-File-New-Script.
3. Copy and Paste the provided MATLAB code, MATLAB_Music_Aliasing.m, into the blank m-file.
4. Connect the output of a music player (cell phone, MP3 player, computer's headphone jack, etc) to the microphone jack on the front of the desktop computer using the Stereo Plug-to-Stereo Plug cable.
5. The speakers should already be connected. If not, connect the speakers to the headphone output jack on the front or back of the desktop computer. If you are using the computer as your music player, you may need to connect a splitter to the headphone jack and plug both wires into the splitter.
6. Play a song [*a song with good range (both high notes and low notes) is best*] on your music player and run the MATLAB m-File (Cell-Evaluate Current Cell or simply press <Ctrl>Enter with your cursor inside the m-File). MATLAB will record and then play back 10 seconds of the music. The default sampling rate is 44,100 Hz, and when MATLAB plays back the music, it should sound normal (no distortion).
7. **(5)** Repeat the same part of the song, but at lower sampling frequencies: Change variable *Fs* in the m-File to 20000, 10000, 5000, 2500, and 1000 Hz. [You may also change the recording time if desired.] Record and play back the music for each case. Record your observations below. In particular, what happens when music is sampled at too low of a sampling frequency?

Data acquisition using MATLAB *Note*: In the previous section, you used the computer's sound card to acquire data. Here we use the USB data acquisition system (DAQ) in the lab.

1. Connect the output of the function generator to the oscilloscope and *also* to channel 0 on the USB data acquisition system.
2. Turn on the function generator and the oscilloscope.
3. Adjust the function generator so that a sine wave of approximately 50 Hz is being produced. Hit the Autoset button on the scope to monitor the signal.
4. Using the oscilloscope as a monitoring tool, adjust the amplitude to around 2 V peak to peak so that the signal will not cause clipping in the A/D board when the virtual instrument starts to sample data.
5. Turn the DC offset knob to its middle position, so that there is approximately zero DC offset (for some function generators, the DC offset knob must be either pushed in or pulled out in order to have any effect).
6. Download the file USB1608G_Sample.m from the class website.
7. **(5)** Acquire data each time you adjust the amplitude, frequency, and DC offset of the function generator. You will need to adjust the sampling rate and number of samples as necessary to produce nice-looking plots. Play with the settings until you can observe clipping and aliasing. Record your observations below, with sketches or copy and paste plots from the screen.

Measurement of whistling frequencies using MATLAB

1. In MATLAB, start a new m-file: File-New-Script.
2. Copy and Paste the provided MATLAB function, Matlab_Whistle_Data_Acquisition.m, from the website, into the blank m-file. [*Note*: This is just a *function*, not a fully operational MATLAB program.]
3. Save the new m-file as Matlab_Whistle_Data_Acquisition.m in your *My Documents* folder.
4. Change the current folder in MATLAB to your *My Documents* directory (Click Browse for Folder [look for the … icon] on the MATLAB Toolbar and select *Libraries* > *Documents* > *your CAC ID*) *Note*: If you do not make the folder in which you saved your m-file the current folder in MATLAB, the function will not run.
5. You may adjust the sampling time (the variable called duration in MATLAB; default = 0.2 s) and the sampling frequency (the variable called Fs in MATLAB; default = 10000 Hz) if necessary.
6. The typical frequency of human whistling ranges from a few hundred to a few thousand hertz. With this in mind, set the sampling frequency for the digital data acquisition system to around 10,000 Hz, which will enable you to resolve this frequency range without aliasing.
7. Connect the microphone output to the microphone jack on the back of the desktop computer (pink jack). Make sure the microphone switch is on. *Note*: A microphone's output is very small (millivolts), but the A/D converter in the computer's sound card is designed for such low voltages. However, it may be necessary to unplug and plug in the microphone several times to make it work right.
8. To run this MATLAB function, type Matlab_Whistle_Data_Acquisition('winsound',0,1) into the MATLAB Command Window. A figure with the Continuous Time History and FFT of the microphone signal should appear. *Note*: The command you are issuing is the title of the m-file you saved in Step 3, followed with instructions on where to record the signal. If you did not save the file appropriately, you will receive an error. By default, MATLAB will acquire and then display the data in a continuous loop.
9. Have one student (one who can whistle well) whistle into the microphone to provide a nearly pure tone. I have found that blowing *across* the microphone, rather than directly into it, produces a much purer tone.
10. If the scale on the plot(s) gets messed up, close (X) the figure, and then with the cursor in the *Command Window*, hit the up arrow and <Enter> using the keyboard. This will re-run and re-scale the plots.
11. **(2)** Using the FFT frequency spectrum of the virtual spectrum analyzer, record (in the table below) the *lowest* frequency that the student can produce by whistling. Also record the *highest* frequency. *Note*: If lots of low frequency noise is encountered in the signal, move the microphone away from your mouth. (Most of the low frequency noise comes from turbulent airflow around the microphone head – moving it away from the blowing jet reduces this noise significantly.)

Student name	**Lowest whistling frequency (Hz)**	**Highest whistling frequency (Hz)**

12. **(3)** Just for fun, swap the microphone with the music player and re-run. Observe the time trace and FFT of some music. For best results, increase the sampling frequency to 44,100 Hz to avoid aliasing of the music, and shorten the sample duration to 0.1 s. Using the headphone splitter jack, split the signal from the music player, plugging one patch cord into the sound card, and plugging the second patch cord directly into the computer's speaker input. That way you can hear the music and watch the time trace and FFT at the same time. Save the FFT trace and include it in your Word document, with proper labels.

Data acquisition and spectral analysis using MATLAB

1. Download the file USB1608G_Sample_with_FFT.m onto your desktop. This function is the same as the example code we have used before with the USB DAQ except that it plots both the time trace of the signal (from channel zero) *and* the frequency/amplitude spectrum.
2. Set the function generator to produce a 50 Hz sine wave, with nearly zero DC offset, and with a peak-to-peak amplitude of around 5 or 6 V.
3. Set the sampling rate in the code to 2000 Hz, and the number of samples to 200, so that 0.1 seconds of data will be displayed (5 full cycles of the 50 Hz waveform).
4. Run. The digitally acquired signal (the perceived signal) should accurately reproduce the input sine wave since approximately 40 discrete points per sine wave period are available (200 data points / 5 periods). The amplitude spectrum should also show a nice peak at the correct frequency, with very little leakage.
5. Adjust the DC offset and/or amplitude knobs of the function generator to see how the digital signal is clipped when the input signal is out of range.
6. When you have a display that clearly illustrates clipping, Stop.
7. Change the title of your plot to indicate the sampling rate and number of samples acquired.
8. Open a Microsoft Word document into which images and information will be copied and pasted.
9. **(2)** Insert the JPEG image of your clipped signal and amplitude spectrum.
10. **(2)** In the Word document, type in the frequency setting of the function generator.
11. **(3)** Record your observations below:

Test the virtual spectrum analyzer, and observe aliasing

1. With the function generator set to produce a 50 Hz sine wave, adjust the DC offset and/or amplitude of the function generator so that there is *no clipping*, but the signal occupies nearly the entire input range of the A/D converter (−10 to 10 V).
2. Sample at 2000 Hz with the virtual instrument. Use enough data points so that a nicely reproduced sine wave is seen, along with a frequency spectrum showing a peak at the proper frequency (no aliasing).
3. **(4)** While keeping the sine wave input at 50 Hz, decrease the sampling frequency of the virtual instrument to 200 Hz. Do you still see a nice sine wave? Is there aliasing? Explain and record your observations here.

4. **(4)** Decrease the sampling rate further, monitoring both the time trace and the frequency spectrum, until you clearly have ***aliasing***. Record your observations below, and insert jpeg images of your MATLAB figures (time trace and frequency spectrum) into your report.

See attached Figures _____ .

5. **(4)** As a special case, set the sampling rate as close as possible to *twice* the frequency of the signal. Switch the function generator between sine wave, square wave, and triangular wave. Monitor the signal on both the oscilloscope and the virtual instrument. Record your observations here.

6. **(2)** Play with the signal frequency and/or the sampling frequency to experience other examples of aliasing, as discussed in the related learning module. For example, generate a 50 Hz sine wave, but sample at 55 Hz. You should see a normal 50 Hz sine wave on the oscilloscope, but an aliased sine wave at around 5 Hz on the virtual instrument. The frequency spectrum should show a peak at around 5 Hz as well. Insert a JPEG image of the MATLAB figure into your lab report along with the settings used in your MATLAB code.

See attached Figures _____ .

7. **(4)** In your Microsoft Word document (which will become part of your lab report), add text identifying each of the plots copied from MATLAB, along with an explanation of each. Be sure to explain how each case was produced, indicating the parameters involved, such as the function generator frequency and type (sine wave vs. triangular wave), sampling frequency, number of data points, etc. Each figure must be *numbered* and *labeled* appropriately, and *referred* to by number in the text of the report.

Lego Mechanism Acceleration

Your goal in this section is to take what you have learned about collecting and analyzing data, and apply it to understanding the motion of a real mechanism that you will design and build; part of the experiment is to collect and analyze the data. Try to focus on understanding experimental design and the broader implications of data analysis rather than just finding "the correct answer." Your grade for this section will be based on effort and completeness; the actual experiment itself is somewhat ***open-ended***.

You have been provided with a program that will rotate a Lego motor and record data from an accelerometer. You need to determine the rotational speed in revolutions per second of this motor. To do this, build an experimental setup with Legos and attach your accelerometer in a way that you think will generate a roughly sinusoidal signal. Remember from the introduction that the accelerometer will only record accelerations in the x direction as shown in figure 1. Keep in mind that the accelerometer only measures the acceleration of the sensor itself; that is, if the sensor isn't accelerating, and is parallel to the direction gravity is acting, it will read roughly 0 meters per second squared. You will perform an FFT on the signal you obtain from the accelerometer to determine the rotational speed of the motor.

Consider as many ways of performing this task as possible. For example, you could build a mechanism that moves the accelerometer (you might Google 'four bar mechanism' or 'slider crank mechanism' as a starting point), or you could create a mechanism that moves the entire experimental setup with a roughly sinusoidal acceleration. You could also take advantage of the fact that these accelerometers can measure the acceleration from gravity. Don't be afraid to experiment with a new idea. Note that the NXT brick can be operated while connected to the computer, or it can be operated by battery with no computer connection. The "best" FFT you can create is the one that allows you to most accurately determine the motors rotational speed, so think about how different designs will affect your results.

1. **(2)** Attach a cell phone photo or a sketch of your mechanism to your lab report.

   See attached Figure _____ .

2. **(2)** In the space below, draw a rough sketch of what you think the acceleration signal will look like, and the resulting FFT. You may not know what the specific frequencies or amplitudes will be yet, so just focus on the shape of each graph.

3. Press the orange square on the front of the NXT to turn it on. It will take a few seconds for it to boot up.
4. Connect the NXT to your computer with the USB cable provided. If a message appears in the lower right corner about installing a driver for the NXT, wait a minute until another message indicates that the hardware is ready to use. You will also need to connect the accelerometer to port 1 and the motor to port A using the black Lego cables.
5. Open the LabVIEW vi file called Lab_5_acceleration_viewer.vi. This file will allow you to see the output of the accelerometer. Experiment with the sensor until you are confident that you know how it operates.
6. Open the LabVIEW vi file called Lab_5_acceleration_log.vi.
7. When you are ready, press the white 'run' arrow at the upper left of the LabVIEW screen. The motor will rotate for roughly 15 seconds, but will log data for only the first 10 seconds.
8. When the motor stops, in the top menu of LabVIEW go to Tools-NXT Application Browser and open Data Viewer.
9. Click the Add Data button on the left side of the window.
10. On the right side of the new window that appears, under the NXT heading, click the icon with the small red page; this loads your data that are stored in internal memory in the NXT brick.
11. You can now look at the data you obtained. If you are satisfied with the data, click Export, and save the file as "data.txt".
12. Open MATLAB with the file Lab_5_Lego_FFT.m; this MATLAB program reads in the file "data.txt".
13. Run the MATLAB file to see the FFT frequency spectrum and the original signal (time trace) of the acceleration data that you generated. The MATLAB file must be in the same folder as data.txt.
14. **(2)** Attach a plot of your acceleration signal and resulting frequency spectrum to this lab

See attached Figures _____ .

15. **(2)** What frequency (in rotations per second, or Hz) did you find?

Motor frequency from FFT = _____ .

16. **(2)** Now run the program again, and count how many times the motor rotates in ten seconds (use a stopwatch to keep track of the time). Use this number to estimate the rotational speed in rotations per second and write your result below.

Measured motor frequency = _____ .

17. When you are finished, press the dark gray rectangle on the NXT cube repeatedly until a message appears asking you if you want to turn off the NXT. Press the orange button to confirm that you want to turn the NXT off. The screen will go blank when the system shuts down.

Exit LabVIEW and save your files

1. Exit LabVIEW. *Note*: Select No to All if prompted about saving other things – you do not want to change your default settings.
2. Restore all equipment to its original location. *Before you leave, be sure to take an electronic copy of your Word document.*

Discussion Questions

1. **(5)** Discuss when and how *aliasing* was observed in your music recordings. Why is 44,100 Hz the standard sampling rate for music? Why is it important to sample at a high-enough frequency?

2. **(5)** For one of your cases with aliasing, use the equations given in the related learning module (and, if necessary the folding diagram or the general equation) to *predict* the aliasing frequency. Compare with your actual observation and discuss.

3. **(4)** Explain why the FFT of a guitar note looks the way it does and how what you hear when you listen to the note agrees with what the FFT tells you.

4. **(7)** Write a brief paragraph describing how you chose your mechanism, and what improvements you would make if you were going to perform this experiment again.

5. **(3)** How did you use the FFT to determine the motors rotational speed? Write a sentence about why your approach works, or may not work.

6. **(3)** Did your experimental FFT look like your prediction? If not, explain the differences?

7. **(3)** How close was the frequency you found from the FFT and the frequency you calculated by counting revolutions? Can you explain any differences?

   If time permits, play with other Lego configurations to see if you can generate a significantly different frequency spectrum.