Using the MATLAB Data Acquisition Toolbox

1. Introduction

This document will describe some of the general usage of MATLAB's Data Acquisition Toolbox. The software in the toolbox allows MATLAB to acquire data from sensors and to send out electrical signals that can be used to control or drive external devices. We will be using this toolbox with two different pieces of hardware. One is the sound card built into your laptop, the microphone and speaker will serve as the data acquisition (and output) devices. The sound card provides convenience because it will allow you to test the DAT outside of the lab. We will also be using a commercial USB data acquisition box that will plug into your laptop, these devices will be in the class laboratory. The data acquisition (DAQ) box allows your computer to acquire data from many different types of sensors.

The documentation on the MATLAB web site provides a reasonable amount of information on the different commands in the toolkit. This tutorial will provide you an overview of some of the basics of data acquisition and show you how to use the toolkit. Much of the information used in this tutorial is translated from the MATLAB web-site, we have just distilled down some of the key points through a few simple examples.

The focus of this tutorial is to provide examples that use your built in sound card: at the end we will show you how to use the USB DAQ as well.

2. Some definitions and concepts

Here are a few definitions and concepts, in no particular order, that might be new for you.

Analog to Digital (A/D) conversion Most measurement sensors are analog, that is they provide a continuous electrical signal of an voltage. Your computer is digital, it can only store discrete numbers. The hardware that we will be using has electronics that can take discrete, digital measurements of a continuous analog signal.

Sample rate The number of times the A/D hardware takes a discrete measurement each second is called the sample rate. You want to sample your data faster than the quantity you are measuring is changing.

Triggering Triggering is when you tell the data acquisition hardware to begin taking data. You can trigger the data collection manually (i.e. you *tell* the device when to start collecting data) or you can trigger the device off the signal you are measuring. For example, you can tell the hardware to start taking data (trigger) only after the signal exceeds one volt.

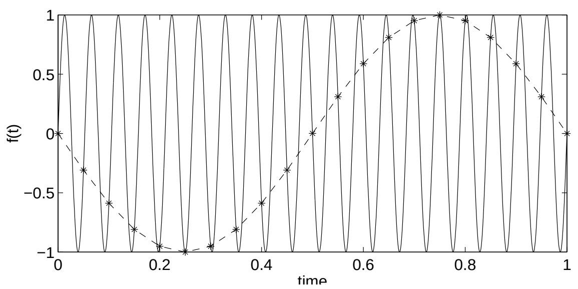

Figure 1. Example of alias error. The signal is 19 Hz, but we are sampling at 20 Hz. This sample rate is too slow for the signal and the result is a bogus 1 Hz signal. To capture the signal we would need to sample at 40 Hz or faster.

Alias Error When a signal has a frequency component that is greater than *half sample rate*, you will obtain a bogus low frequency signal. To visualize this phenomena use MATLAB to plot a high frequency sine wave with a small number of points, i.e `>> x = [0:0.05:1]; >>plot(x,sin(2*pi*19*x));`. You will see one cycle of the wave rather than 19: the result and the desired signal are shown in figure 1.

3. Getting Started

First you need to make sure that you have all the software installed and that the hardware is working properly. You need MATLAB installed with the Data Acquisition Toolbox (DAT)

To make sure that the Data Acquisition Toolbox is installed you can go to the MATLAB prompt and type the command,

```
>>ver
```

This command will return which components and version of MATLAB that you have installed, hopefully the Data Acquisition Toolbox is installed. It is part of the standard MATLAB install, so unless you have done something weird to your computer this should work fine.

Your laptop has a sound card built in and MATLAB should be able to talk to this device in the same way that it will talk to the USB DAQ. Most of this tutorial will be done with your built in sound card (speakers and microphones) as your data acquisition device.

4. Analog Input

Now you are ready to start acquiring actual data and processing it with your laptop. The sensors that you will use send out a continuous, analog, electrical signal. The A/D hardware samples this signal at the rate you specify and sends values of voltage at each sample time to your laptop. The continuous, analog signal has been converted into a list of numbers, corresponding to the voltage at certain instances in time. Digital sampling is represented in Figure 2. When using your computer's sound card as the input device you will be sampling the output from

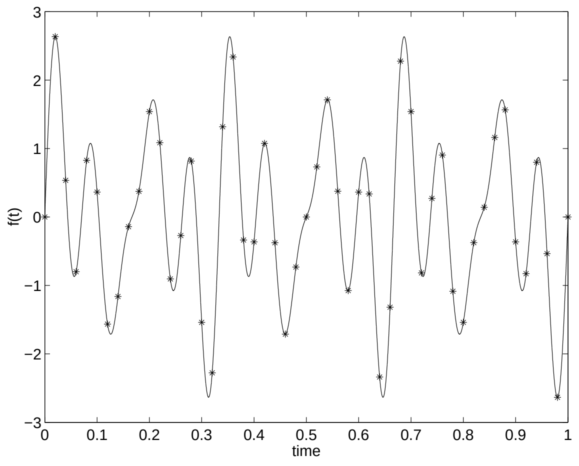

Figure 2. Digital sampling of a continuous analog signal. The sample rate was 50 Hz for data that was acquired over one second.

your computer's microphone. Later we will be using hardware which allow us to sample data from many different types of sensors (rather than only a sound sensor).

There are a few steps that you must always follow; the steps initialize your hardware device and the channels you would like to sample. The first thing you need to do is create an analog input device object and open up channels for data input. At the MATLAB prompt, try typing

```
>>AI = analoginput('winsound',0);
>>chan = addchannel(AI,1);
```

You have now defined the windows sound card to be the analog input device and have opened channel 1 for data input. Your sound card only has two channels: channel 1 and channel 2; if you specify other channels MATLAB will provide an error. The zero in the `analoginput` command is a hardware specific flag. For a list of supported hardward you should see the MATLAB help.

The variable `AI` is an object that has many associated properties. You can change these properties so that the device will behave in the manner you would like. The card has defaults that you can accept, though they may not be optimal for your application. Simply typing

```
AI
```

with no semi-colon will return the general properties of the device such as the sample rate, the number of samples to acquire, and trigger conditions. (Note that typing a variable name with no semi-colon simply displays the contents of that variable). The properties that are listed at this point will be the defaults. You should also notice when you type `AI` that the associated hardware device is defined to be your sound card. To see *all* the properties of AI, type

```
>>get(AI).
```

The `get` command will provide you with a long list of properties for the analog input object that you are free to change. The commands `get` and `set` are used to find out the complete list of properties and to change them, respectively. The `set` and `get` commands are general to any MATLAB object. The exhaustive list and what all the AI properties do can be found on the MATLAB help page for the DAT.

To view the value of a particular object property you can type either `get(AI, 'PropertyName')` or `AI.PropertyName`. An example might be; `get(AI, 'SampleRate')` will tell you the current sample rate. The format `AI.SampleRate` is the general format for referencing any data structure, a topic we will cover later. The way to read the code is that SampleRate is a variable that belongs to the object AI.

To provide an example of changing object properties, the following commands will set the sample rate, the number of samples to acquire, and the trigger condition. In this example, we will take 44,100 samples in one second (this is a standard audio rate). We will trigger the hardware manually, which means that when we would like the sound card to start collecting data from the microphone when we type the command `trigger(AI)`.

```
AI
duration = 1;
SampleRate = 44100;
set(AI,'SampleRate',SampleRate);
set(AI,'SamplesPerTrigger',duration*SampleRate);
set(AI,'TriggerType','Manual');
AI
```

Notice that before and after changing the properties of the AI object we are displaying the basic object properties by typing `AI`. Notice how the properties have changed before and after the set commands have been issued. The use of typing `AI` in the above example serves no purpose other than to display the object properties at those times.

Now that you have configured the card you can begin acquiring data. Notice that when you type `AI`, MATLAB returns the Engine Status which should say that the `AI` object is 'waiting for START'. We will insert `AI` in the following commands so you can see what the Engine is doing at different stages. The order of commands that you should perform are the following

```
AI
start(AI);
AI
trigger(AI);
AI
data = getdata(AI);
```

These commands work as follows: After the start command is issued the card is collecting data and recycling it through a temporary buffer. Viewing `AI` following the start should now display the Engine Status as 'waiting for TRIGGER 1 of 1'. After the trigger command is issued, the card is triggered and collects 44,100 samples for one second. Depending on how fast you type `AI` following the trigger command the Engine Status will either say 'LOGGING data' -or- if you wait for the acquisition to finish `AI` will tell you that you acquired 44,100 samples, that 44,100 samples are available for GETDATA and the Engine is 'waiting for START'. The latter message means that the data acquisition is complete. The data acquired at the trigger command is saved in the temporary buffer. The data is pushed into the

MATLAB workspace by issuing the `getdata` command. Notice that the variable AI is an argument to each command. The handle to the object must be specified because multiple objects can be initialized at a given time: you must be specific about which device you are using. You now plot the data that you acquired by typing, what else,

```
plot(data).
```

MATLAB can also be a little fussy if you leave the device open. The best thing to do is to wait until the device stops and then close it and clear the object from memory. Specifically you should add the following lines to your m-file script.

```
wait(AI,duration+2)
delete(AI)
clear AI
```

Typing help on any of these commands will clarify what each command is doing. The wait command prevents your script from continuing until the time needed for data collection is passed. The extra 2 seconds is just to add a little extra time for things to finish cleanly. Also, note that sometimes MATLAB gets a little confused and weird things will happen. Sometime a piece of code that has worked one hundred times will all of a sudden stop working correctly. If something seems weird, you can try the command `daqreset`. Otherwise close MATLAB, restart and try again. Sorry.

> **Exercise 1:** Type up the previous commands into an m-file script. Run the commands and make sure that everything in MATLAB is working. Make sure that the script runs, that data is acquired for the given time and acquires the correct number of samples. Read the MATLAB help and figure out how to take simultaneous measurements on two channels, modify your m-file to do so. Remember that the sound card only has two channels, 1 & 2. When you run the command whistle, clap, or make noise. When you plot the data your should see spikes in in the microphone amplitude at the times that you have clapped.

5. Analog Output

In addition to acquiring data from external sensors with MATLAB, you might want to send data out to an external device, for example you way want to control a small motor, turn on a light, or power a speaker (in the case of the sound card). The commands for sending out an analog signal are very similar to those we used for acquiring data. The following program will send a 500 Hz sine wave to analog output channel 1 for one second.

```
%%Open the analog device and channels
AO = analogoutput('winsound',0);
chan = addchannel(AO,1);

%% Set the sample rate and how long we will send data for
%% 44,100 Hz, 1 seconds of data
duration = 1;
SampleRate = 44100;
set(AO,'SampleRate',SampleRate)
set(AO,'TriggerType','Manual')
NumSamples = SampleRate*duration;

%% Create a signal that we would like to send, 500 Hz sin wave
x = linspace(0,2*pi*500,NumSamples);
data = sin(x)';

%% Put the data in the buffer, start the device, and trigger
putdata(AO,data)
start(AO)
trigger(AO)

%% clean up, close down
wait(AO,duration+1)
delete(AO)
clear AO
```

The program works as follows: We start by opening the device and defining the channels that we will be using. Channels 1 & 2 correspond to the left and right speakers on your laptop. Next we define how long we would like to send data date for and at what rate (44100 Hz sample rate for 1 second). We set the trigger to manual. Next, we create data that we would like to send, a 500 Hz sine wave that will last 1 second. Once the data is created we put the data in the output buffer, next we start the device running, and trigger the output. It is only at the trigger command that the signal is sent out of the sound card. Finally, we clean up and shut down the device as before.

Exercise 2: Write a MATLAB program that will send a signal to the speaker while recording with the microphone. You will need to initialize both the analog input and analog output devices in the same program. Set the duration and sample rate to be the same for the input and output. Try triggering the input device before the output and the output device before the input. Comment on the time delay, i.e. the input and the output device cannot be triggered simultaneously, if you want fast measurements. Try changing the amplitude of the output signal to see how the volume is influenced.

6. Setting Trigger Conditions

There are many times that you want to take data only when an particular event happens (for example, the system will respond only when the input signal exceeds a certain level). In the previous exercise you saw that with the manual trigger you were not able to start the devices simultaneously and there was a dead time when the signal was being sent but the input channel was not yet triggered. One way to remedy this delay is to use software triggers rather than manual triggers.

In setting a software trigger condition you must first set the trigger type to be `Software`, rather than `Manual`. You must also specify which channel you will be triggering off, if more than one channel is defined. Finally, you must specify the condition that must be met to invoke the trigger. As an example we initialize two channels and use the second one as the trigger. We will trigger when the signal passes through 0.1 volts from below (i.e. you must specify whether the signal should fall through one volt or rise through it).

```
AI = analoginput('winsound',0);
chan = addchannel(AI,[1 2]);
set(AI,'TriggerChannel',chan(2))
set(AI,'TriggerType','Software')
set(AI,'TriggerCondition','Rising')
set(AI,'TriggerConditionValue',0.1)
```

You may want to experiment with the 'Timeout' property. This property tells the program how long to hold before the program gives up waiting for the trigger condition. The default is only one second, so you may want to make this time longer.

> **Exercise 3:** Using the program written in Exercise 2 as a starting point, you will modify the program to use software triggers. Modify the program to send a sinusoidal signal to the output while setting a trigger condition on the input channel. Plot the input data. Did the trigger work? Did the command time out? If the command timed out try setting a lower threshold or making the amplitude of the output signal louder. Try setting the trigger condition value higher and use a clap to initiate data acquisition.

7. Using the USB DAQ

Using the USB DAQ is fundamentally no different than the commands that were used for the sound card. There are a few differences in the details that we will elaborate here.

The DAQ that we are using has 8 channels for analog input, this means that we can acquire information from 8 simultaneous instruments in single-ended mode or 4 measurements in differential mode (the later mode is what we will typically use). Single ended just means the voltage is referenced to the same ground as on the USB DAQ and differential means a voltage difference between two wires is sensed. If you are operating the DAQ in differential mode the card channels range from Channel 0 to Channel 3.

To initialize our DAQ for analog input use the following command

```
AI = analoginput('nidaq','Dev1');
```

This command works the same as before, only we are defining a different piece of hardware. The arguments to this command are simply MATLAB conventions and are hardware specific. The argument nidaq refers to the maker of the hardware, National Instruments. The argument Dev1 is the device number of the USB DAQ. By default, the first serial number DAQ that you use with your laptop will be Dev1. If you come in tomorrow and use a different piece of hardware, it will by default be named 'Dev2'. You can see what the device id is by typing

```
daqhwinfo('nidaq');
```

If you do not get the reply that the installed board ID is something like Dev1, you may need to restart MATLAB. It seems from our experience with this hardware and windows 7, that you need to plug the hardware in, let it be recognized by your computer and then start MATLAB. It appears if you switch hardware or plug it in after MATLAB is started, that the device is not found.

To see what hardware is currently recognized by MATLAB, you can try,

```
HW = daqhwinfo
HW.InstalledAdaptors
```

These commands will tell you what hardware is installed. You should see winsound, parallel, and nidaq (if one of the USB boxes is plugged in). If you do not see the nidaq hardware, there is apparently a quirk with windows 7. The first time, you may need to close MATLAB, right click on the MATLAB icon and select run as administrator. Once open, type

```
daqregister('nidaq');
```

You should be able to close MATLAB, and open normally as your own user and then the nidaq hardware should be recognized. This is apparently a quirk with how windows 7 handles administrative privileges with hardware.

You can also check some of the specifications on the hardware by typing,

```
AI = analoginput('nidaq','Dev1');
daqhwinfo(AI);
```

To test the analog input on the DAQ you can use the function generator on your lab bench to generate a known voltage signal. Take wires from the function generator output signal and the ground and screw these wires into the terminal block into screws 2 & 3, respectively. The documentation that comes with the DAQ tells you what screws are attached to what channels and the device should also have the label on. Make sure the function generator is on, set the frequency of the wave and the amplitude. If you have done everything correctly and all the software/hardware is working then you will see the wave of known frequency and amplitude when you plot the data.

> **Exercise 4:** Modify the commands from Exercise 1 to work with the DAQ. Hook up the function generator to the terminal block as discussed above. Turn on the function generator and set the signal properties. Do you get a plot that looks as expected, i.e. the amplitude and frequency match those being output by the generator. Change the amplitude and frequency and make sure your result changes accordingly. Set a frequency higher than your sample rate and see if you can get alias error.

Acquiring data from measurement sensors, in theory, is no more difficult than hooking up the function generator. The differences, as you will see, is that measurement sensors often require a bit of deciphering of the sensor specifications to translate the voltage to a physical quantity. Also, while some sensors can be plugged directly into the DAQ, others cannot. We will work with the sensors during our lab time throughout the course, for now this is a detail for another day.

Analog output with the DAQ is not really any different than using the sound card and speaker. However, you should note that the hardware will not allow the output sample rate to be greater than 150 Hz with the 'nidaq' device. You can check this by typing `daqhwinfo(AO)` once you have created and analog output object. Further, if you try to run the USB DAQ with this toolbox in a continuous input/output control loop you will find that things run even slower than this.

To test that the device is sending a voltage out you can hook the output channel on the terminal block to the oscilloscope on your lab bench. Again, the default is to define the channel as differential so you must hook up the output and the ground to the scope. Another way to test the device is to connect wires between an input and output channel and write a program that will send a signal out and read it back in, which is the next exercise.

> **Exercise 5:** Hook wires between an analog output channel and an analog input channel. Modify the code in Exercise 2 to test sending a signal from the output channel to the input channel.

References

General MATLAB product documentation. *www.mathworks.com*

Data Acquisition Toolbox Product Documentation
http://www.mathworks.com/access/helpdesk/help/toolbox/daq/daq.shtml