Functions in MATLAB and the Groove Station

Version 1.25: Oct 5, 2007 2:32 pm GMT-5

University Of Washington Dept. of Electrical Engineering

Abstract

Learn about simple functions and representing signals and use them to create your own groove.

1 Introduction

In this lab, you will learn about MATLAB function files, which we will refer to as m-files and have the file extension `*.m`, the same as script files. You will create a number of functions for manipulating sound signals to create a groove (a short song).

The difference between a function and a script is that functions can return values and take parameters, while scripts are simply a collection of commands. Unlike script files, any variables created in the function are not available after the function has run, that is, the variables are active inside the scope of the function, and the variables cannot be accessed out-of-scope. The MATLAB on-line help system has a nice write-up about functions and how to handle various things like returning more than one value, checking the number of arguments, etc. To learn more, type the following commands and read the online help:

```
≫ help function
≫ help script
```

You need to use a text editor to create function files. MATLAB has an internal editor that you can start by clicking "File" and then "New" "m-file". You can also use other editors such as `Notepad` on the PC's, and `pico`, `emacs`, or `vi` on Unix machines.

NOTE: Remember that in order to run custom scripts or functions, the MATLAB working directory needs to be set to the location of those files.

[*]http://creativecommons.org/licenses/by/2.0/

2 Sound in MATLAB

Download the sound samples from the sound resources[1] page and save them to your working directory. Use `wavread` to load `.wav` files, and use `load` to load `.mat` files. Plot each one in turn, and try to guess what it will sound like (the name might help).

On a computer, sounds are represented digitally, which means that only samples of the signal at fixed time intervals are stored. We'll learn more about this later. For now you just need to know that the time interval Ts (or equivalently the sampling rate Fs=1/Ts) is something you need to keep track of for playing sounds.

Now play each sound. The goal is to learn how the time domain signal sounds. Use the sound command to play a sound signal. You must specify the playback sample rate (Fs), which will be the same as the sample rate of the sound samples on the web site (they are 8000 Hz). For example, if you wanted to play a sound called bell and its sample rate was 8000 Hz, then you would enter the following command,

```
   ≫ Fs=8000;
≫ sound(bell, Fs);
```

If you use a different value for Fs, you will effectively be doing time scaling.

When working with sound in MATLAB, it is important to remember that the values of the audio signals are in the range [-1, 1]. Keep this in mind when you are writing your functions. Your functions should expect inputs with values in the range [-1, 1] and anything out of that range will be clipped when you play the sound.

3 Function Files

Before we can create our groove, we need to make functions that will allow us to modify the sound signals in various ways. After all, wouldn't it be boring to make a groove out of the same note over and over again? Let's create some functions that will let us time scale, reverse, delay, fade, and repeat a sound, and mix two sounds together.

There are many functions built into MATLAB. One that will be useful here is `fliplr`, which is a one step way of time reversing a signal. Try this with the bell sound.

Another function that we created for you is timescale.m[2] , which you can use to speed up or slow down a signal. Download it and give it a try. Notice that it also changes the pitch of a sound – why?

Download the function fade.m[3] , make sure you save it as `fade.m`. Start MATLAB, and go to the directory where you saved the function. You can see and change your current directory at the top of the MATLAB screen. Enter "help fade" at the MATLAB prompt. If you did everything correctly, you should see the help text (in the `.m` file) in response to `help fade`. Notice that we've now added a new command to MATLAB that can be used as if it were a built-in function.

Enter the following commands at the MATLAB prompt:

```
   ≫ time = 0:.01:1;
≫ y = cos(time .* pi .* 25);
≫ plot(time, fade(y));
```

You can see in the plot that fade does indeed fade-out the cosine wave. You can use this function on audio signals as well.

Exercise 1 Fader

[1] "Sound resources" <http://cnx.org/content/m13854/latest/>
[2] http://cnx.org/content/m13555/latest/timescale.m
[3] http://cnx.org/content/m13555/latest/fade.m

1. Modify the fade function so that you can adjust the slope of the ramp which will affect the level of the fade. Use the variable `level` (which is already in the parameter list for you in the function) to represent the strength of the fade as a decimal fraction. The function should make sure that the value is between 0 and 1.
2. Like in the code example above, plot your function with the cosine wave to see its effect. Throughout this lab you may find it helpful to plot functions (use the plot command).
3. Demonstrate to the TA that your fade function works.

Exercise 2 Repeater

1. Create a function that repeats a sound N times. Use a `for` loop for this. Inside the `for` loop you will need to concatenate sound signals. For example, if you have two vectors `x` and `y`, you can concatenate them like this:

```
≫ x = [1 4 2 2 3];
≫ y = [5 8 3 9 0];
≫ x = [x y];
```

The first line of your function might look like this:

```
function [ out ] = repeat(in, N)
```

2. Demonstrate your repeater using an N specified by the TA.
3. Optional: Add an argument that let's you insert silence in between each repetition.

Exercise 3 Delay (Shift)

1. Create a function to time-delay a signal. Because we are working with digital data, you can do this by simply adding zeros (zero pad) in the front. The inputs to the function should be the signal and the amount of time-delay. The number of zeros to add will depend on the time-delay and the sample rate. The sound signals from the resource page have a sample rate of 8,000 Hz, but it is good coding style not to assume this and to still have the sample rate (Fs) be an input to the function in case you wanted to change it later.
2. Demonstrate to the TA that your delay function works by plotting the original and delayed signal together with the subplot command.

Exercise 4 Mixer

1. Create a function that adds two sound vectors together; your function should be able to handle inputs that are not the same size. The output values cannot be outside of the range [-1, 1], so you will have to re-scale them. One option is to re-scale the summed sound if it goes out of this range. You may want to look at the source code to the `soundsc` function for a way to do this. What happens if you let the sounds go out of this range and you try to play them with the `sound` command?

   NOTE: You can view the source code to most MATLAB functions by using the command `type function_name`.

2. Demonstrate to the TA that your mixer function works by playing a mix of two sounds.

4 Groove Station

In order to create a groove, you're going to need some instruments. The groove will be made up of some sound samples modified in any way you want and concatenated together to make one long sound vector. Use only the sound samples from the sound resources[4] page.

Exercise 5 Make Your Groove

1. Create a script (not a function) to build your groove. You can use any combination of the above functions, or even create additional functions if you want. Use concatenation to combine the sounds together to make your groove. When you are finished save your groove with the `wavwrite` command (Remember to specify the sample rate (Fs), which for the sounds on the resource page is 8000 Hz).
2. Your groove should be between 10 and 30 seconds long.
3. Plot your newly created groove signal.
4. Demonstrate your groove to the TA. Explain how you created it.

[4]"Sound resources" <http://cnx.org/content/m13854/latest/>