BYTES AND BEATS

An Introduction to Programming with MATLAB

Instructor Guide

Module 13: Importing and Editing Sounds

Prerequisite Domain Knowledge: Functions, Figures

Expected Completion Time: 30 minutes

Table of Contents

Importing sounds into MATLAB®

Expected Duration: 15 minutes

Learning Objectives

- Import sound files to MATLAB
- Understand that sound files are imported as simple numeric vectors

Motivation

Sound files can be easily imported into MATLAB and edited. Students will have access to a library of sound files of notes from different instruments.

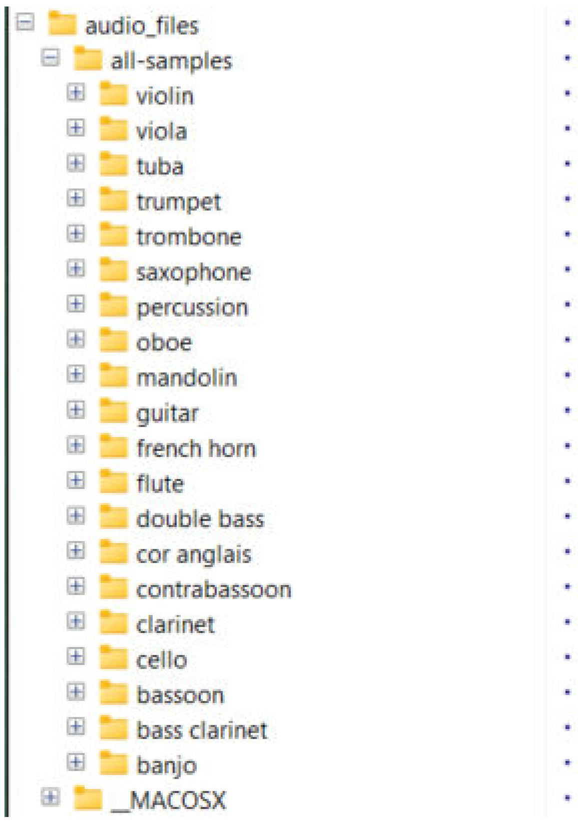
This library was recorded by an orchestra in the UK.

Materials

- MATLAB

Steps

- Ask the students to click on the little plus sign ⊞ next to the *audio_files* folder in Current Folder section to view the other music folders in it. They can use the ⊞ and ⊟ buttons to navigate the folder structure and take a look at all the different musical notes and percussions available.



```
⊟ audio_files
  ⊟ all-samples
    ⊞ violin
    ⊞ viola
    ⊞ tuba
    ⊞ trumpet
    ⊞ trombone
    ⊞ saxophone
    ⊞ percussion
    ⊞ oboe
    ⊞ mandolin
    ⊞ guitar
    ⊞ french horn
    ⊞ flute
    ⊞ double bass
    ⊞ cor anglais
    ⊞ contrabassoon
    ⊞ clarinet
    ⊞ cello
    ⊞ bassoon
    ⊞ bass clarinet
    ⊞ banjo
  ⊞ _MACOSX
```

- Ask the students to use MATLAB documentation to see if they can find a function in MATLAB to read audio files. The MATLAB documentation can be opened by typing `doc` at the prompt in the **Command Window**. Let the students search for a bit.
- If they type "read audio files" the first result is the `audioread` function, which is the function we will be using!
- Explain that from the documentation it looks like `audioread` has the syntax:

```
filename = "guitar_G3_very-long_forte_normal.mp3";
[y,Fs] = audioread(filename)
```

`filename` – is the input argument – the file you want to import.

`y` and `Fs` - are the outputs:

- `y` is the sample data of the audio
- `Fs` is the rate at which the samples were collected, or in other words, the number of samples in 1 second of data. The `audioread` function reads the file and provides the variables `y` and `Fs` as outputs.
- Ask the students to import one of the WAV files:

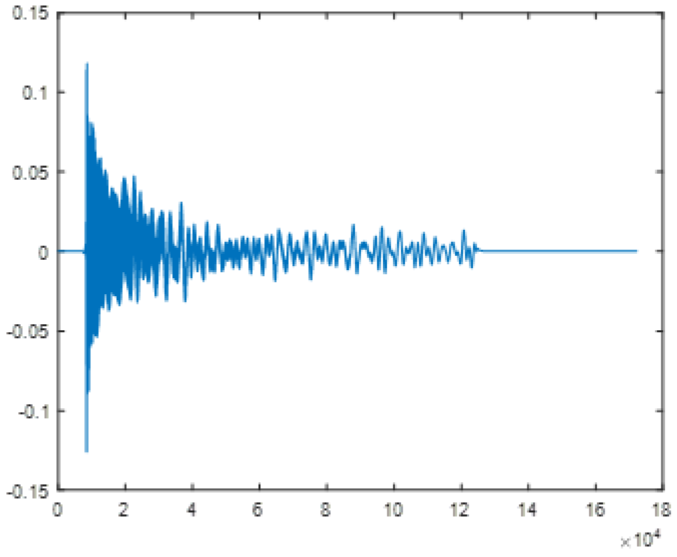
```
[y,Fs]= audioread("guitar_F4_very-long_forte_normal.wav");
```

- Once the amplitude values are imported into MATLAB, we can hear the sound by using `sound` function:

```
sound(y,Fs)
```

- We can also view the sound file by using the `plot` command

```
plot(y)
```



- Ask the students to try importing a file and get them to understand that `y` is the sampled data from the sound wave and is simply a large numeric vector. The command `length(y)` will tell them how long the vector is or how many numbers the vector contains. Ask them to import, listen to and plot a few different sound files and find out their lengths.
- We can make changes to the numbers of this numeric vector to make changes to the sound, which we will see in the next section.

Editing sounds in MATLAB

Expected Duration: 15 minutes

Learning Objectives

- Edit imported sound files in MATLAB
- Write the edited sound to an audio file

Motivation

Sometimes the audio files you have are not in a format that would suit your application, so you might need to edit them. In our case, they have extended periods of silence that need to be removed.

Materials

- MATLAB

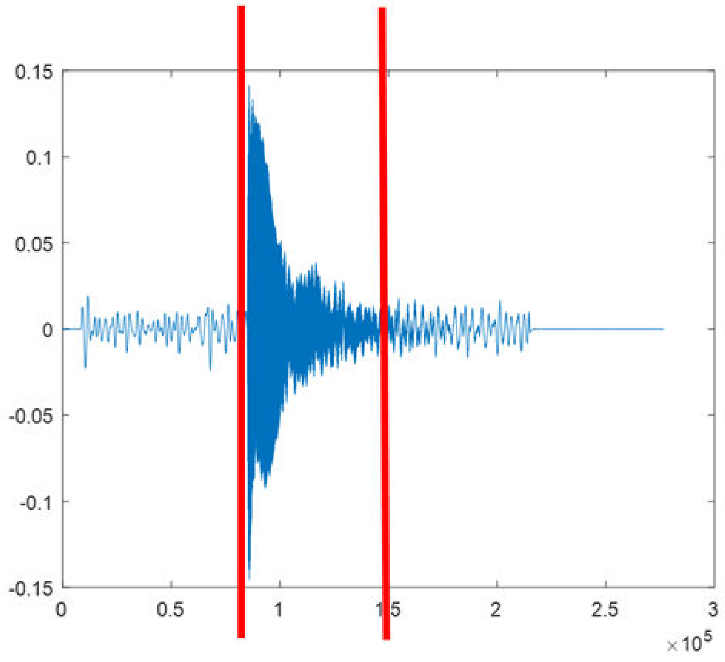
Steps

- After listening to the sound files, we notice that there are long durations of silences in the files. So, if the students were to string together the audio files like they did in the loops exercise, there would be a lot of silence. We are now going to use MATLAB to edit these files.
- The `plot` command can be used to visualize the sound file and find out where most of the amplitudes lie and what silence periods can be snipped. For instance, the sound files may have amplitudes (useful information about the note) between the samples 20,000 and 60,000.
- Ask the students to follow along with you for the rest of this lesson.
- Use `audioread` to read in the audio file below into variables `y`, `Fs` just like in the previous activity and play the sound:

```
[y,Fs] = audioread("guitar_G3_very-long_forte_normal.wav");
sound(y, Fs)
```

- Find out the length of the vector `y` and plot it

```
length(y)
plot(y)
```



- We can see that the data of interest mainly lies within the two vertical red lines, from around samples 80000 (0.8 * 10^5) to 150000 (1.5 * 10^5). We can then snip out the relevant portion of the sound file by using indexing.

```
y1 = y(80000:150000);
```

- Use the `length` function to find the number of samples in `y1`.

```
length(y1)
```

- Play the truncated sound in MATLAB using the functions `sound`:

```
sound(y1, Fs)
```

- Now we can use the function `audiowrite` to create our own file. Ask the students to check out the documentation for the function to find the correct syntax. The first input argument is the file name to create. The second is the sample data and the sample frequency, which can be the same as the original file.

```
audiowrite("guitar_G3_70001.wav",y1,Fs);
```

This will create `guitar_A3_70001.wav` in the current folder.