BYTES AND BEATS

An Introduction to Programming with MATLAB

Instructor Guide

Module 14: Working with Sound Files

Prerequisite Domain Knowledge: Importing and editing sound files, Vector manipulation, indexing

Expected Completion Time: 50 minutes

Table of Contents

Creating a Sound File for Silence

Expected Duration: 10 minutes

Learning Objectives

- Create a sound file that represents silence in MATLAB®.

Motivation

Students will create a file that represents silence, and they will use this file in an activity. Silence is the absence of sound, which is an important aspect of music.

Materials

- MATLAB

Steps

- Ask the students to think of a song where silence is used. An example could be the song “Hello” by Adele. You could play them the song and the students can then have a discussion on the effect silence has on the feel of the song.
- Sound files have the information stored in them as samples of their wave. Import a sound file in MATLAB and go over this in the with the students.

```
[y1,Fs]=audioread("guitar_F3_very-long_forte_normal.mp3");
```

Clicking on `y1` in the workspace opening the numeric vector in MATLAB to view. It is a vector with 1 column and 40000 rows.

- Ask the students what the corresponding numeric vector for silence would look like. The numeric vector should be composed of all zeros.
- To make a numeric vector of zeros in MATLAB, we can do:

```
a = [0 0 0 0 0 0]
```

To make a bigger vector of about a thousand length, entering the zeros can be tedious. In MATLAB there is a built-in function that lets us create such a large numeric vector.

```
b = zeros(1,6)
```

Students can see in the **Workspace** that a and b have the same value. Both are 'row vectors' with 1 row and 6 columns.

Similarly, we can create a 'column vector' of zeros with 1 column and 6 rows:

```
c = zeros(6,1)
```

Now, let the students observe the **Workspace** and ask them how they would create the same vector without using the function zeros. The answer is by using semi-colons as separators just like they see in the **Workspace**. Try it.

```
d = [0; 0; 0; 0; 0; 0]
```

We will use column vectors for storing information in our music files just like `y1,` which is a column vector.

- Let us make a numeric column vector of the reference size 40000.

```
silence_vector = zeros(40000,1);
```

- Once we have this numeric vector, we can create a sound file from it by using the `audiowrite` command in MATLAB.

```
audiowrite("silence.wav",silence_vector,44100);
```

The first input is the sound file to create, the second input is the numeric vector of the samples, and the third is the sample frequency of 44100.

- Execute the commands the students should see the WAV file created in the current folder. Ask them to play the wave file and verify that it is indeed silence.

Concatenating Sound Files

Expected Duration: 20 minutes

Learning Objectives

- Play notes one after the other
- String notes together, or “concatenate vectors”
- Get familiar with the instrument notes (guitar, trumpet, and violin) in the music library **audio_files**

Motivation

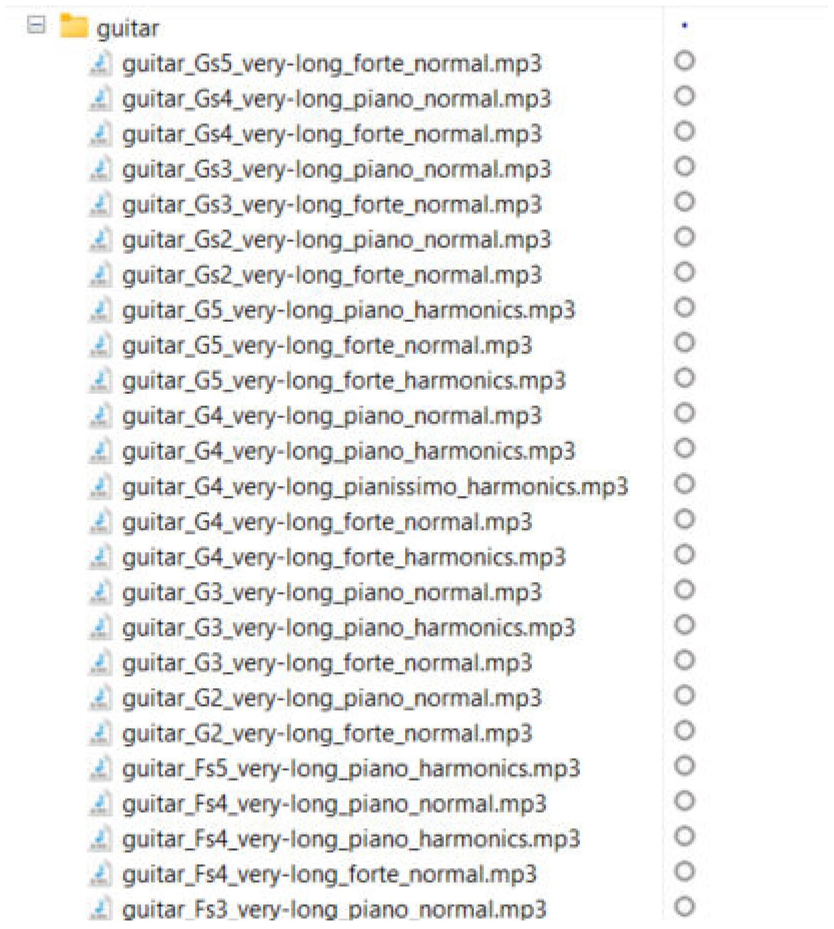
Audio mixing is an important aspect of music. It involves combining, mixing and editing sound files to make new creations or enhance original compositions.

Materials

- MATLAB

Steps

- Using the ⊞ and ⊟ buttons next to folder names, let’s navigate to the folder ***audio_files>all-samples>guitar*** which will have the files containing different guitar notes.



guitar
guitar_Gs5_very-long_forte_normal.mp3
guitar_Gs4_very-long_piano_normal.mp3
guitar_Gs4_very-long_forte_normal.mp3
guitar_Gs3_very-long_piano_normal.mp3
guitar_Gs3_very-long_forte_normal.mp3
guitar_Gs2_very-long_piano_normal.mp3
guitar_Gs2_very-long_forte_normal.mp3
guitar_G5_very-long_piano_harmonics.mp3
guitar_G5_very-long_forte_normal.mp3
guitar_G5_very-long_forte_harmonics.mp3
guitar_G4_very-long_piano_normal.mp3
guitar_G4_very-long_piano_harmonics.mp3
guitar_G4_very-long_pianissimo_harmonics.mp3
guitar_G4_very-long_forte_normal.mp3
guitar_G4_very-long_forte_harmonics.mp3
guitar_G3_very-long_piano_normal.mp3
guitar_G3_very-long_piano_harmonics.mp3
guitar_G3_very-long_forte_normal.mp3
guitar_G2_very-long_piano_normal.mp3
guitar_G2_very-long_forte_normal.mp3
guitar_Fs5_very-long_piano_harmonics.mp3
guitar_Fs4_very-long_piano_normal.mp3
guitar_Fs4_very-long_piano_harmonics.mp3
guitar_Fs4_very-long_forte_normal.mp3
guitar_Fs3_very-long_piano_normal.mp3

- If we want to create a new sound file which plays two notes of the guitar, say **D4** and **A4**, then the way to do it would be to play the **guitar_D4_very-long_forte_normal.mp3** file first and then **guitar_A4_very-long_forte_normal.mp3**.

Timing	1	2
Guitar	D4	A4

- Combining notes to play one after the other is called ***concatenation.*** Let's see how we can do concatenation in MATLAB.
- Read in the sound files:

```
[gd4,Fs]= audioread("guitar_D4_very-long_forte_normal.mp3");
[ga4,Fs]= audioread("guitar_A4_very-long_forte_normal.mp3");
```

- The `gd4` and `ga4` variables have samples of the sound wave and `Fs` contains the sample frequency information. Here `g` in the variable name stands for guitar, followed by the note. They can have any variable name but having names that are descriptive are useful.
- To concatenate the files, we would simply create a new vector and place the values one after the other. We can try this with two small vectors in the **Command Window**:

```
A = [1; 1; 1]
B = [2; 2; 2]
C = [A; B]
```

- In a similar fashion, to concatenate the samples, we can do:

```
y = [gd4; ga4];
```

- y now has 382560 samples (186096 samples from gd4 and 196464 samples from ga4)
- We then use soundto listen to y to confirm it is indeed playing the two notes one after the other.

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

- We can then use the `audiowrite` function to create a music file.

```
audiowrite("concatenate.wav", y, Fs);
```

- Ask the students to play this file and verify that they hear the two notes being played one after the other.
- Let the students concatenate some music notes on their own. They can use any of the instrument files to do so.

Adding Sound Files

Expected Duration: 20 minutes

Learning Objectives

- Students will learn how to add sound files together into one sound file.
- Adding essentially plays more than one note at the same time
- Familiarize with the percussion notes in the music library **audio_files**

Motivation

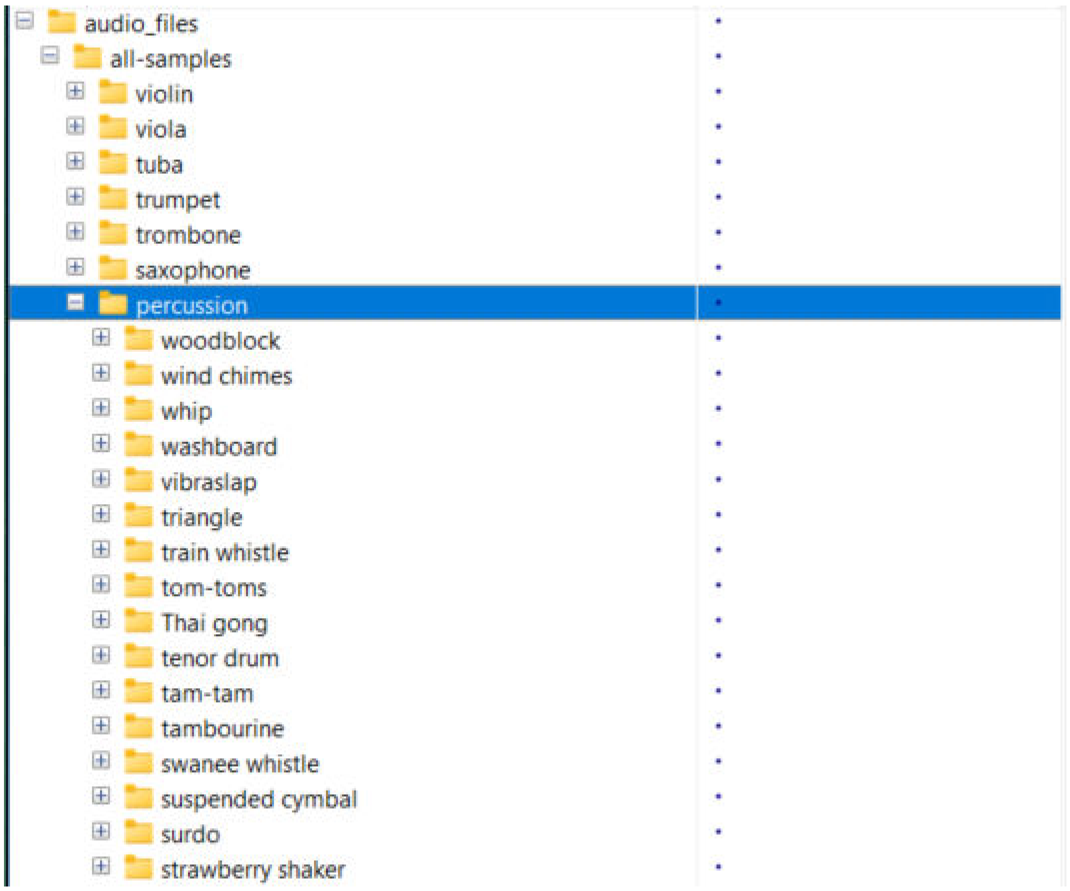
Music often has multiple instruments and notes playing simultaneously. Student will now learn how to add sounds together to play different instruments or percussions at the same time.

Materials

- MATLAB

Steps

- Using the ⊞ and ⊟ buttons next to folder names, let's navigate to the folder ***audio_files>all-samples>percussion*** which will have all of the percussion instruments available in the music library. Each percussion instrument has its own folder of files.



- If we want to create a new sound file that plays two percussions simultaneously, we simply need to add the values.

Timing	1
Percussion1	cabasa
	+
Percussion2	djundjun

- Playing beats simultaneously is achieved by ***addition.*** Let's see how we can do this in MATLAB.
- Read in the sound files:

```
[p1,Fs] = audioread("cabasa__phrase_mezzo-forte_effect.mp3");
[p2,Fs] = audioread("djundjun__phrase_mezzo-forte_rhythm.mp3");
```

As before, the `p1` and `p2` variables have samples of the percussion sound wave and `Fs` contains the sample frequency information.

- To add the files, we would simply need to add the values. Ask the students how they would do it (vector addition is covered in earlier modules). We can try this with two small vectors:

```
A= [1; 4; 5; 1; 2];
B= [2; 1; 3; 8; 2];
```

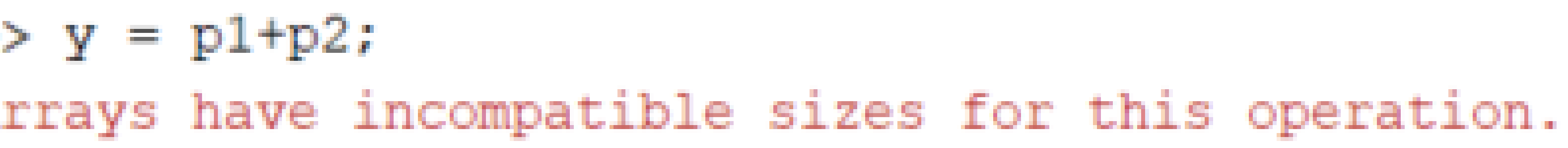
```
C= A+B
```

Note that each value in vector `A` gets added to the corresponding value in vector `B`.

- Similarly, to add the sound samples, we can do:

```
y = p1+p2;
```

Now, you will see an error regarding incompatible sizes for this operation:



```
> y = p1+p2;
rrays have incompatible sizes for this operation.
```

This is because the number of samples in each audio file is different! In order to merge these samples, we need to ensure that both files have the same number of samples.

Using our knowledge of arrays, combine the two variables by indexing just the first 30000 values of each together:

```
y = p1(1:30000)+p2(1:30000);
```

- We then use `sound` to listen to y and confirm it is indeed playing the two notes together

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

- We then use the `audiowrite` function to create a merged music file.

```
audiowrite("merge.wav",y,Fs);
```

- Ask the students to play this file and verify that they hear both instruments play simultaneously.
- Let the students come up with combinations of wave files of their own. For example, they can even try adding a percussion to a violin note.