FFT Workshop 2019

September 27th 2019; Additions: 14-10 2019

---- begin - changed upon request --

Due date: Extended to Monday October 21st 2019 23.59h.

Note: If using MATLAB proved to be too challenging, it is allowed to use a programming language + libraries of your choice, i.e., C++, Python, etc.

----- end - added upon request ---

Introduction

These assignments should be made using MATLAB®. MATLAB is available on the Unix machines only. Please restart the computer and select the Unix/Linux operating system. Regretfully, sound does not always work on the Unix machines. To listen to the original wav-files and the results you may want to open another machine under Windows and use a directory for your files that can be seen on both machines.

Further Preparations:

1. Download Voicebox.zip from the API2019 web site and save it in your Unix home directory. Unzip the file in your home directory by opening a console window and typing: 'unzip Voicebox.zip'. This will make a new subdirectory *Voicebox* with all the Voicebox MATLAB routines and data files. Note that, further information on the Voicebox toolkit functionality can be found on the following website: www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html
2. You are ready to start matlab. In the console window type: 'matlab &', this will start MATLAB. After starting MATLAB a window appears with several sub-windows. If not, select <Desktop><Desktop Layout><Default> to initiate the default layout. In the command window you can issue all the Matlab commands.
3. Under <Help><Product Help> an extensive Help-environment can be found for references, tutorials, and examples on the complete functionality of MATLAB.
4. In MATLAB select <File><Set Path>. In the <Set Path><Dialog> that appears you can add folders and their subfolders to the MATLAB search paths. MATLAB scripts and data in these folders can directly be used from the MATLAB command line. Click on the <Add with Subfolders> button and browse to the just created *Voicebox* directory and select it. Click <close> and <yes> to apply your changes for following MATLAB sessions.
5. This step (5) only works if sound drivers are installed. You should use the example wave files that are available in the *Voicebox/data* directory or take new samples under Windows 7. Connect your microphone and headphones. Record a wave file in the API2019 folder, and check if it contains data. In the console window type:

```
cd API2019
rec -r 21000 -t wav sound.wav
play sound.wav
```

6. Now you are ready to start using the Voicebox functions. After Step 4 they are automatically recognized by MATLAB. You can see them in the left top window, if you browse to the directory under Voicebox that contains them. If you need help, go to the Voicebox website mentioned above. Now have a quick look at the different functions present in the Voicebox toolkit.
7. Let's do a small example. We will load the just recorded wave file, or another example wave file from the Voicebox/data directory. Then convert it to the frequency domain, filter out some frequencies and transform the result back to the time domain. Finally, we save the file and listen to the result and observe the resulting data. In the matlab console (rightmost window) type:

```
[y,fs,wmode,fidx]=readwav('sound.wav','r',-1,0);
```

8. Variable <y> now contains the stereo sample data, y(:,1) contains the left channel. Type the following to put the left channel in <left> and view the contents of the left channel:

```
left=y(:,1);
```

9. Now let’s have a look at the sound data and the frequency spectrum (we have to tell the function that we used a 16000 sampling rate). Type:

```
plot(left);
figure;
spgrambw(left,16000);
```

10. Observe the data. Let's slice up the left channel in 6 equal parts and show the individual power spectra using rfft (fast fourrier transform) on all 6 parts. Type:

```
frames=enframe(left, uint16(length(left)/6));
frames=transpose(frames);
fftdata=rfft(frames);
fftdata=fftdata.*conj(fftdata);
plot(fftdata);
```

11. This results in a combined plot of the power spectra of the six consecutive parts of the sound file. You should be able to see that the power of the different frequencies is different in each of the parts. These kinds of differences thus characterize the sound data in terms of average frequency amplitudes. If you cannot see these differences, plot the different spectra in different windows, by typing:

```
plot(fftdata(:,1))
figure
plot(fftdata(:,2))
figure
etc...
```

12. It should be clear that the earlier drawn figure of the power spectrum (step 8) actually is a finer grained (and better visualized) version of the different plots you have just created.

Assignment 1 Feature Vectors

In the introduction we showed how we can compute the Fourier transform of the sound signal. Implement a procedure that takes as input a wav file and gives as output for every part of 512 samples the energy of 8 frequency bands (for example: [0Hz,1kHz), [1kHz, 2kHz), … , [7kHz, 8kHz) , please note you may have to use other bands) in the Fourier transformed signal. Calculate these features for the piano.wav file. Use these features to calculate the following code (C_i) for the piano.wav, where

- C_i = up, if the current max energy band is of a higher frequency as the previous max energy band
- C_i = down, if the current max energy band is of a lower frequency as the previous max energy band
- C_i = empty, otherwise

This codes the signal as a sequence of pitch tendencies, which can be used to recognize melodies. In some respect it resembles the so called Pearson's code. Send your coding-routine and your 'up/down'-code for the piano.wav in a single zip file to erwin@liacs.nl before Wednesday October 16th 2019 23.59h.

Appendix: Quick-List for using MATLAB for this assignment.

Some Basic Matrix Vector Operations

```
>> A = fix(10*rand(3,3))      // creates a random 3x3 matrix with integer values in [0,10).
>> b = fix(10*rand(3,1))      // creates a random column vector of dimension 3.
>> c = A*b                    // calculates A applied to b
>> det(A)                     // calculates the determinant of A
>> b = b’                     // gives the transpose of vector b
>> x = A\b                    // calculates the solution of the linear system of equations:
                              // Ax = b’, if a solution exists. (Try, some other matrices.)
>>lambda = eig(A)             // calculates the eigenvalues λ1 λ2 λ3 of A
>> A = fix(10*rand(512,1));   // creates a random 512x1 matrix with integer
>>                            // values in [0,10). NB Without the ‘;’ you get the whole
>>                            // listing of values of A.
>> E = fft(A)                 // calculates the Fourier Transform of A
>> E = abs(E)                 // calculates the Power Spectrum
>> Plot(E)                    // and plots it.
```

Useful Functions and Features

- size_y = size(y) will put the size information of the variable y into the variable size_y. If, for example y is a 200x400 matrix, size_y(1) is equal to 200, and size_y(2) is equal to 400. Use this to parameterize your for-loops etc.
- MATLAB has all the basic control structures like if (Boolean) statement else statement end; for-statement, while-statement, etc. See <Help><MATLAB><Programming> for further information.
- For Boolean expressions a kind of C-like format is used: a ==b, a<b, etc. For strings the well known basic functions as strcmp(name1, name2), etc. are available.

Implementing your own functions/scripts in MATLAB

In the directory Voicebox/Myroutines you will find two examples PlotWav.m, and PlotFFT.m. These simple examples show you the basics in defining your own MATLAB routines/functions, and parameter passing in MATLAB. From the command line you can now issue PlotWav(‘sound.wav’), PlotWav(), etc.
Use these examples to write your own routines.