MUS4831, 12.10.2017, Olivier Lartillot, olivier.lartillot@imv.uio.no

Matlab Tutorial

Aims: To get familiar with the basic concepts of *Matlab*. See the slides for explanations about *Matlab*.

1. Random composer.

Let's compose randomly a very simple "score" that we will represent as a matrix of numbers.

1.1. Generate a random chord of 5 notes as a column vector ***Chord*** containing numbers between 0 and 36 (the chromatic scale spanning three octaves).

> For that purpose we can use the function ***rand***, that generate real numbers between 0 and 1. See ***help rand***.
>
> ***r* = *rand*(5,1)** produces a 5-by-1 matrix (hence a column vector) of random numbers.
>
> ***r* * 36** produces random numbers from 0 to 36.
>
> In order to obtain integers instead of real numbers, uses the function ***round***:
>
> ***Chord* = *round*(*r* * 36)**

1.2. Write the whole process in a script. Run the script.

1.3. One problem with this method is that the notes are not sorted, and we can have duplicate pitches for one chord. Let's try another method starting from the specification of the interval distance between successive notes:

> Generate a column vector of 4 intervals (between +1 and +12 semi-tones, for instance).
>
> ***r* = *rand*(4,1)**
>
> ***Intervals* = *round*(*r* * 11) + 1**
>
> From the set of intervals ***Intervals***, the actual set of notes can be generated using the cumulative sum ***cumsum***. (Example: ***cumsum*(1:5) = [1 3 6 10 15**]).

1.4 Generate a random sequence of three notes ***BassLine*** following the strategy 1.1.

1.5. Generate a sequence of three transpositions of the same chord ***Chord*** such that the bass notes follow the sequence ***BassLine***. Model it as a 5-by-3 matrix ***ChordProgression***.

1.6. Suppose the synthesizer used for the audio rendering is a little broken and cannot play the *G* (i.e., 7). Let's first detect the position of the *G*s using the ***find*** function. You should use the syntax

> **[*raw col*] = *find* (...)** (see ***help find***).

1.7. Change now all the *G* in the chord progression into *F#* (i.e., 6).

2. Simple synthesizer.

2.1. Generate one second of a sinusoid of frequency 440 Hz, sampled at rate 44100 Hz.

```
f = 440;

sr = 44100;

t = 0:1/sr:1;

y = sin (2 * pi * f * t);
```

2.2. Display the waveform curve, add title and axis legends using the functions ***plot***, ***title***, ***xlabel***, ***ylabel***. Zoom in the figure in order to observe the sinusoid.

2.3. Play the waveform:

```
soundsc (y, 44100)
```

2.4. Add some noise of some pre-specified amplitude ***A***:

```
y2 = y + A*rand (1, length (y));
```

2.5. Write all the operations in a script ***mysound.m***. Run the script.

2.6. Modify the sound waveform in the script by applying a half-wave rectification:

```
y3 = abs(y2);
```

Display and play the resulting waveform.

2.7. Add fade-in and fade-out envelopes of one second duration each. For that purpose, we can use a *Hanning* window of length half a second for instance:

```
h = hann (round (sr * 0.5));
```

The new envelope can be constructed using the two halves of the *Hanning* window, separated by a series of ones.

2.8. Transform the script into a function with two inputs: the frequency ***f*** of the sinusoid, and the amplitude ***A*** of the noise.

2.9. Generalize the function such that when the first input ***f*** is a column vectors of ***N*** numbers, it outputs a chord, i.e. a mixture of ***N*** sinusoids.

2.10. Generalize the function such that when the first input ***f*** is a matrix of ***N***-by-***L*** numbers, it outputs a succession of chords.

3. Random music

Finally, play the composition generated in exercise 1 using the synthesizer constructed in exercise 2. You might need the ***midi2hz*** routine from *MIDItoolbox*. Enjoy.