Synthesis of Musical Notes

1. Timing

These values could serve as a pacemaker for the musical notes, which could control the speed of playing. This experiment firstly established the relationship between the pulses, beats and samples:

```
bpm = 100;              % beats/min
bps = bpm/60;           % betas/sec
seconds_per_beat = 1/bps;
seconds_per_pulse = seconds_per_beat/4;
```

Considering the sampling theorem and the resolution of the Digital to Analog signal convertor; therefore, this report has choose the sample rate as 11025 Hz.

According to the relationship between the sample and pulse:

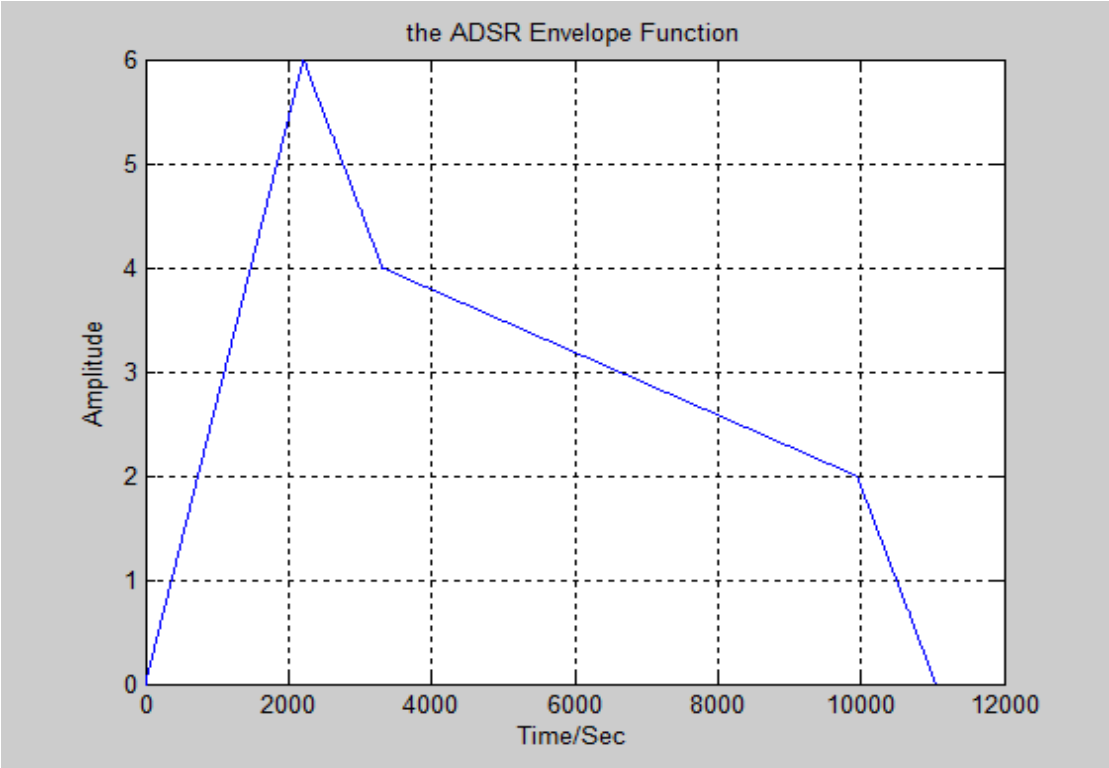
```
samples_per_pulse = 11025 * seconds_per_pulse;
```

2. Musical Tweaks

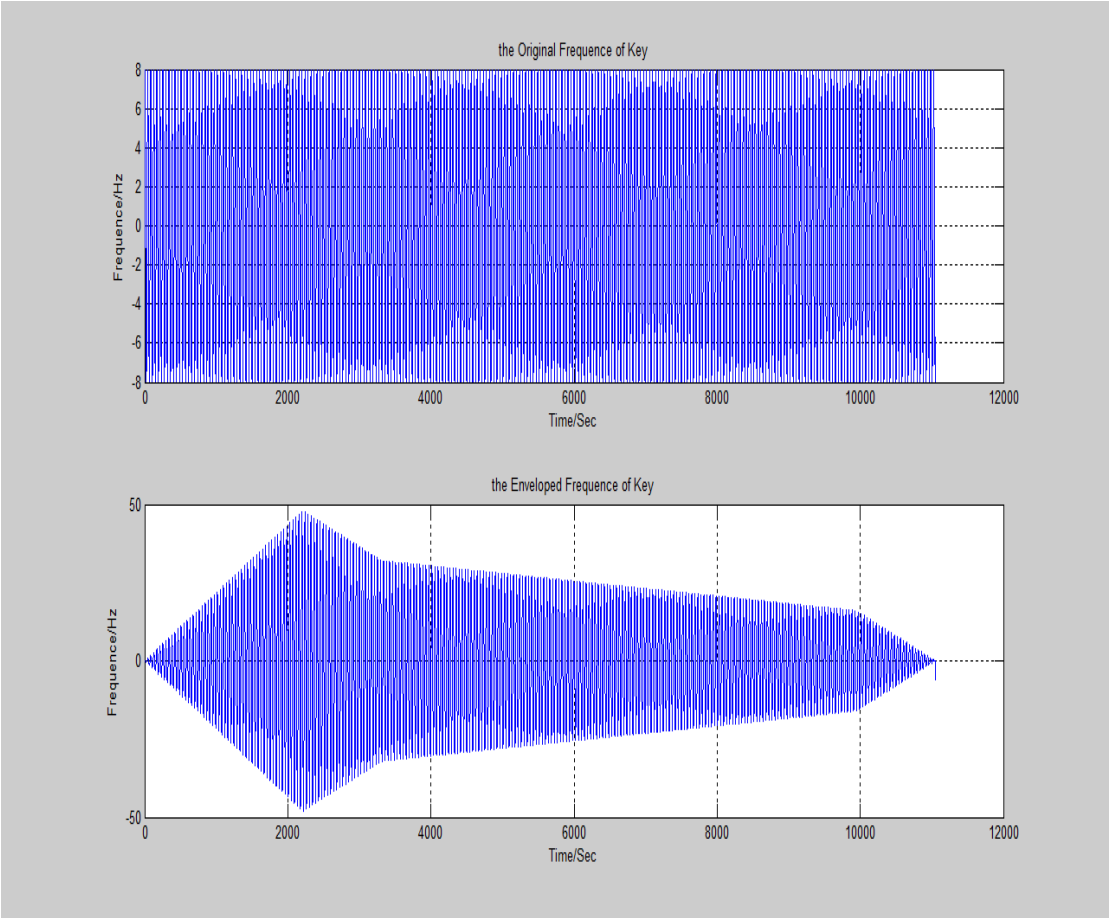
The *Key2note* function file could realize this, which used the ADSR envelope function to modify the frequency for every single key. It would make the note sound similar like instruments rather than the artificial machine.

```
function xx1 = Key2note( key,dur,fs )
    tt = 0:(1/fs):dur;
    freq = 440*2^((key-49)/12);
    xx = real(100*exp(1j*2*pi*freq*tt));
    L=length(tt);
    eva=zeros(1,L);
    eva1=[linspace(0,6,0.2*L)linspace(6,4,0.1*L)linspace(4,2,0.6*L)linspace(2,0,0.1*L)];
    for iter=1:length(eva1);
        eva(1,iter)=eva1(1,iter);
    end
    xx1=xx.*eva;
end
```

The ADSR profile for this certain envelope function (*eva* in this experiment):



The Key number equals to 40 for example, the original and enveloped signal respectively:



3. The processing of Data

① *How the data input*

The MATLAB provide the *load* command to read the data structure from the out of world, and in this experiment:

```
load('C:\Users\Administrator\Desktop\ELG4177\LABs\Lab2\Bach_Fugue\bach_fugue.mat');
```

These data have 1*3 structure, which is 1 to 3, and each one structure has three fields:

```
theVoices(i).noteNumbers;
theVoices(i).startPulses;
theVoice(i).durations;
```

② *Create a container to store the synthesis of music*

In this step, we need a big enough container to make storage space for the coming synthesis music tone. The length of this matrix could be measured according to the number of notes, and the last note's starting pulse, duration as well. This experiment generate a one-dimensional matrix to claim space for notes, and use *ones* to initialized the matrix in order to improve the efficiency of the program:

```
for loop_of_voice = 1:numVoices,
    numNotes(loop_of_voice) = length(theVoices(loop_of_voice).noteNumbers);
    final_pulse(loop_of_voice)=theVoices(loop_of_voice).startPulses(numNotes(loop_of_voice));
    final_duration(loop_of_voice)=theVoices(loop_of_voice).durations(numNotes(loop_of_voice));
end
sound_box=ones(1,ceil(samples_per_pulse*(max(final_pulse)+max(final_duration)) ) );
```

The point in this part is to make sure that the container would be big enough to hold all the coming data. So that exaggeration strategy would be used, such as the *ceil*.

③ *Transform and Broadcast*

There will be three fields to broadcast altogether, so the circulation would be used to traverse each field. Furthermore, the individual notes. A double *for* circulation should be equipped. The outer circulation control the traversing of the field and the inner circulation is going to transform each key number into frequency, process them with the pre-set pulse and insert them into the container sequentially.

The following MATLAB code illustrated that idea:

```
for loop_of_voice = 1:numVoices,
  for kk = 1:numNotes(loop_of_voice),
      duration_in_seconds=seconds_per_pulse*theVoices(loop_of_voice).durations(kk);
      the_tone=Key2note(theVoices(loop_of_voice).noteNumbers(kk),duration_in
```

```
        _seconds,fs);
        tone_length = length(the_tone);
        start_sample=round((theVoices(loop_of_voice).startPulses(kk)-
        1)*samples_per_pulse + 1);
        sound_box(start_sample:start_sample+tone_length+1)=sound_box(start_sa
        mple:start_sample+tone_length-1)+the_tone;
    end
end
```

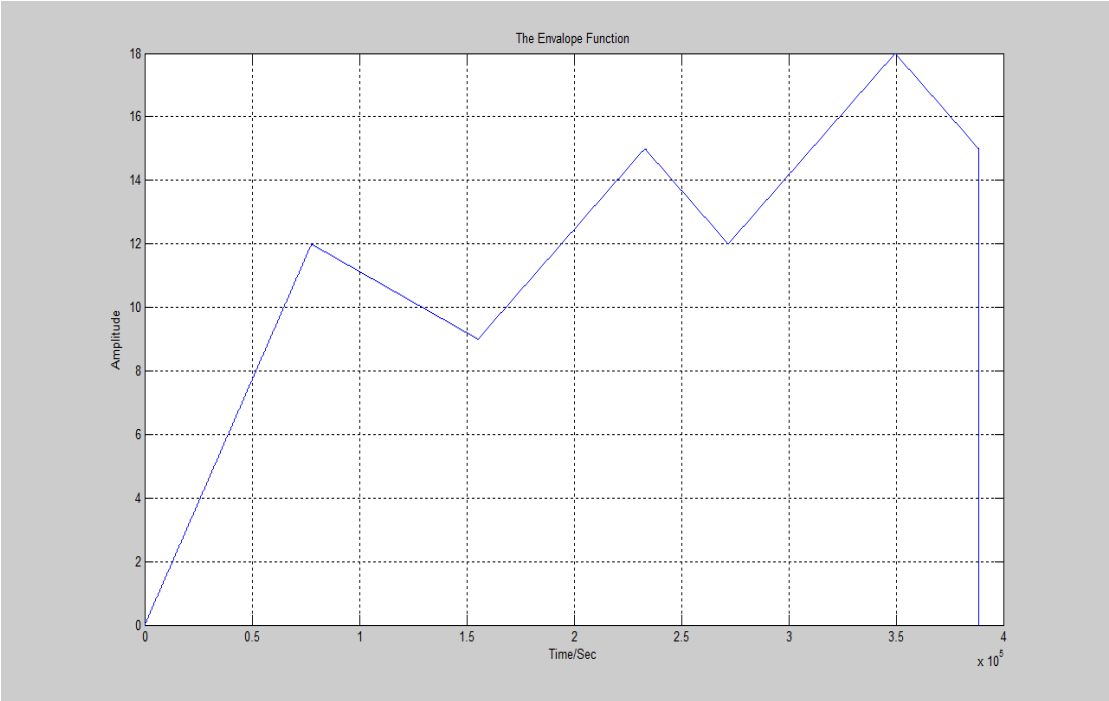
The processing generates the actual sinusoid each key by making call to the function *Key2note()* written previously.

④ *Overall envelopment*

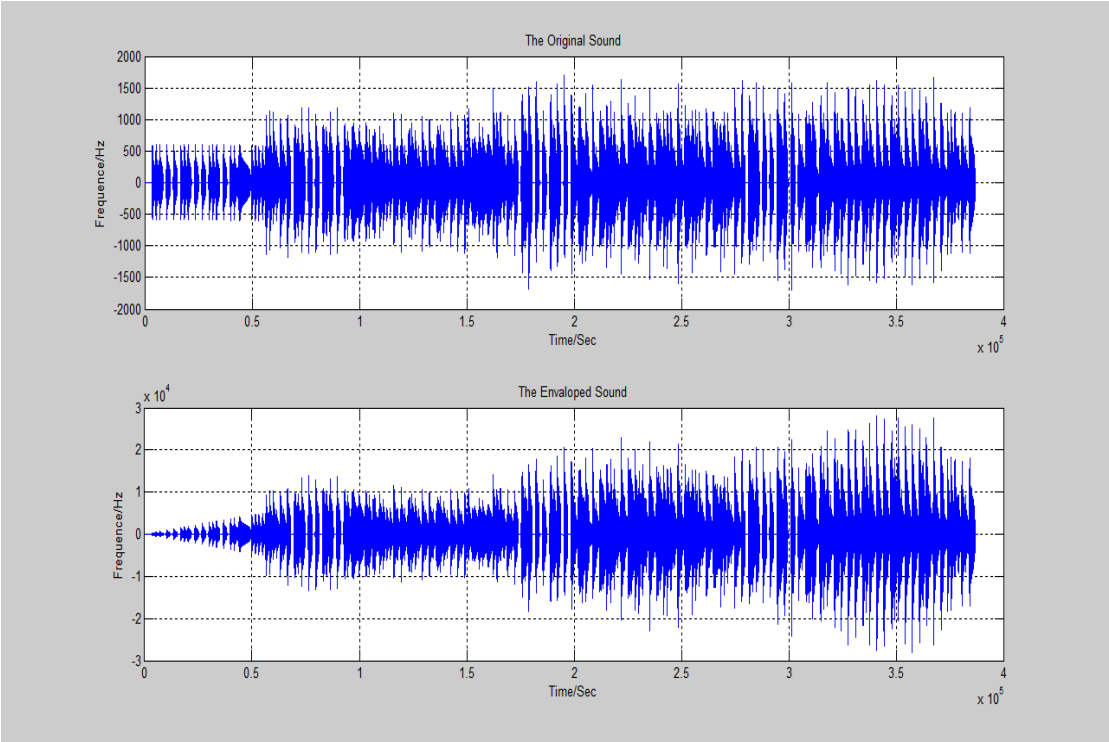
As to pursue better quality of the synthesized sound, this experiment implement the macroscopic modification.

```
Len=length(sound_box);
enve=zeros(1,Len);
enve1=[linspace(0,12,0.2*Len)  linspace(12,9,0.2*Len)  linspace(9,15,0.2*Len)
linspace(15,12,0.1*Len) linspace(12,18,0.2*Len) linspace(18,15,0.1*Len)];
for i=1:length(enve1)
    enve(1,i)=enve1(1,i);
end
sound_new_box=sound_box.*enve;
```

The profile for this certain envelope function (*enve1* in this experiment):



And the overall signal frequency plot:

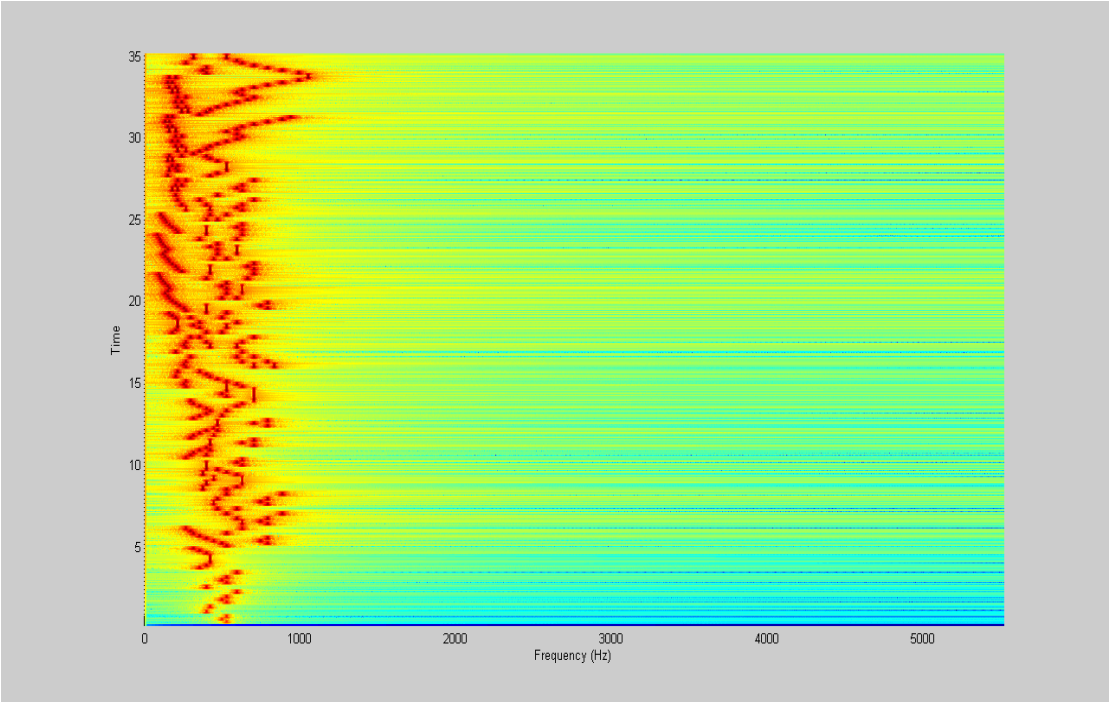


⑤ *Spectrogram of the Music*

This experiment use the *Spectrogram_show* function to present that the different frequencies, as the time-frequency representation.

Spectrogram_show(sound_new_box,fs);

And the spectrogram is show as following:



4. Conclusion

This experiment give us the insight of processing the digital signal, audio frequency signal for example. Different key of instruments corresponds to certain frequency, which would present a certain tone after D-A conversion.

The music synthesis is to be done with sinusoidal waveforms. For a certain type of data, we could make music synthesis by processing the individual signal and establish the connection between musical notes, the frequencies, and sinusoids. Meanwhile, the signal would be represented by spectral graph, which is in time-frequency domain.

Finally, the quality of the music would be improved by many kinds of methods, such as envelope function, sample frequency and so on.