**Group ID: 5**

**Project Topic: Talking Piano**

**Bi-weekly Report 1**

**Discrete-time Fourier Transform (DTFT)**

Discrete-time Fourier transform is a form of Fourier analysis, it’s used to analyze samples of continuous function.

Discrete-time means that the transform operates on discrete data, the samples are uniformly spaced, from which it produces a function of frequency.

Under certain conditions of the sampling theorem, the original continuous function can be recovered from the DTFT and thus form the original discrete samples.

P5

**Definition**

DTFT is a discrete sequence of real or complex numbers is a Fourier series, which produces a periodic function of a frequency variable.

一張含有 文字, 時鐘 的圖片

自動產生的描述

**Inverse transform**

An operation that recovers the discrete data sequence from the DTFT function is called an inverse DTFT.

The standard formulas for the Fourier coefficients are also the inverse transforms:

一張含有 文字 的圖片

自動產生的描述P6

**Group ID: 5**

**Project Topic: Talking Piano**

**Bi-weekly Report 2**

**DFT (Discrete Fourier Transform)**

If we sample (divide) one Time period of a DTFT at a finite number of frequency points, we get DFT:

1)Used only for finite Sequence

2)Practical. Commonly used in computer science and audio analytics.

3)Non periodic and non-continuous.

4)Denoted by

P7

**1.** Every periodic audio wave can be seen as a composition of sinusoidal waves of different frequency and amplitude, which can be written as

A tune of piano is a periodic wave, which is approximately a sinusoidal wave of a certain frequency. By attaining the frequency and amplitude data of a given audio signal, we can mimic the audio signal using piano.

Without the original one, we may not distinguish words clearly

P3, only 5 seconds of video play

**2.** DTFT is difficult to implement and evaluate on a computer, since a computer works only on finite amount of data. Hence, in computer science, DFT is more applicable than DTFT. Generally, DFT convert segment of discrete audio data into discrete data related to frequency and amplitude of the components of sinusoidal waves, while DTFT generate a continuous function from discrete periodic set of data.

Also P7

**3.** Choose a finite number of frequency points. This is similar to **sampling** the Fourier transform at a certain number of points. Using DFT, however, reduces the computational complexity drastically.

一張含有 文字, 室內 的圖片

自動產生的描述

*Fig.1 demonstration of DFT to a simple curve*

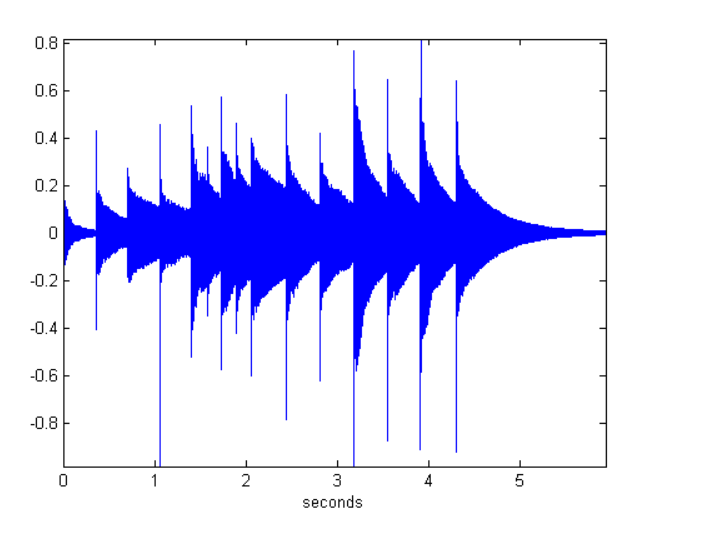
This is a demonstration of DFT data set. The data points shown are modulus of DFT data. The horizontal axis is (which is in the picture), the vertical axis is relative intensity. You can see that there are two pulses in the data points, which corresponds to two different frequencies of the component of . Note that height the spike at  is half of that at , indicating that the amplitude of  is half of that of .

**Group ID: 5**

**Project Topic: Talking Piano**

**Bi-weekly Report 3**

Audio signal is aperiodic since amplitude of component frequencies changes with time. As a result, merely implementing DFT to the entire audio file cannot provide us the sufficient information of amplitude-time relationship. Therefore, we should slice an audio signal to many pieces with same short time interval (say 0.1s. The actual time interval is to be determined through implementation of code and experimentation.), and by assuming the time interval is short enough to be periodic (i.e., amplitude of different component frequencies stays constant over time), we can do DFT to each piece of audio separately to get frequency-amplitude information of different time intervals.

****

*Fig.1 Example of an aperiodic audio signal*

In Scilab, we can use FFT instead of DFT to extract the audio signal information. Computing DFT directly from the definition is still to slow in practical. Instead, an FFT rapidly computes such transformations and hence reduce the complexity of computing DFT from to , where is the data size. Basically, FFT evaluates the result faster and is as accurate as DFT.

The difference between piano and real-world audio is that piano only have 88 keys, thus only 88 different frequencies, in contrast to thousands of different component frequencies of real-world audio. In order to mimic the audio, we also need to map the thousands of frequencies to 88 frequencies. We can do it by mapping similar frequencies to a single one.

P8

**Steps of producing Talking Piano (in Scilab)**

1. Read an audio file. Save it as an array.
2. Slice the array with respect to time into many different pieces of equal time interval.
3. Implementing FFT to each of the pieces. Map the frequencies to 88 frequencies of piano.
4. Construct the Talking Piano audio by inverse FFT into a single array after adding timbre of piano. This can be done in single step if we know the function of piano sound wave.
5. Save the new array as audio file. It is the Talking piano audio file.

P11

Here, I'd like to mention Nyquist frequency...

**Nyquist frequency**

Nyquist frequency is the minimum sampling frequency to present the unbroken original signal. It is a characteristic of a sampler that converts a continuous function or signal into a discrete sequence. The value of Nyquist frequency is at most one-half of the sampling rate. Suppose a sound signal consists of several frequencies, the highest frequency of which should be less than the Nyquist frequency so that the result is free of distortion.

P9

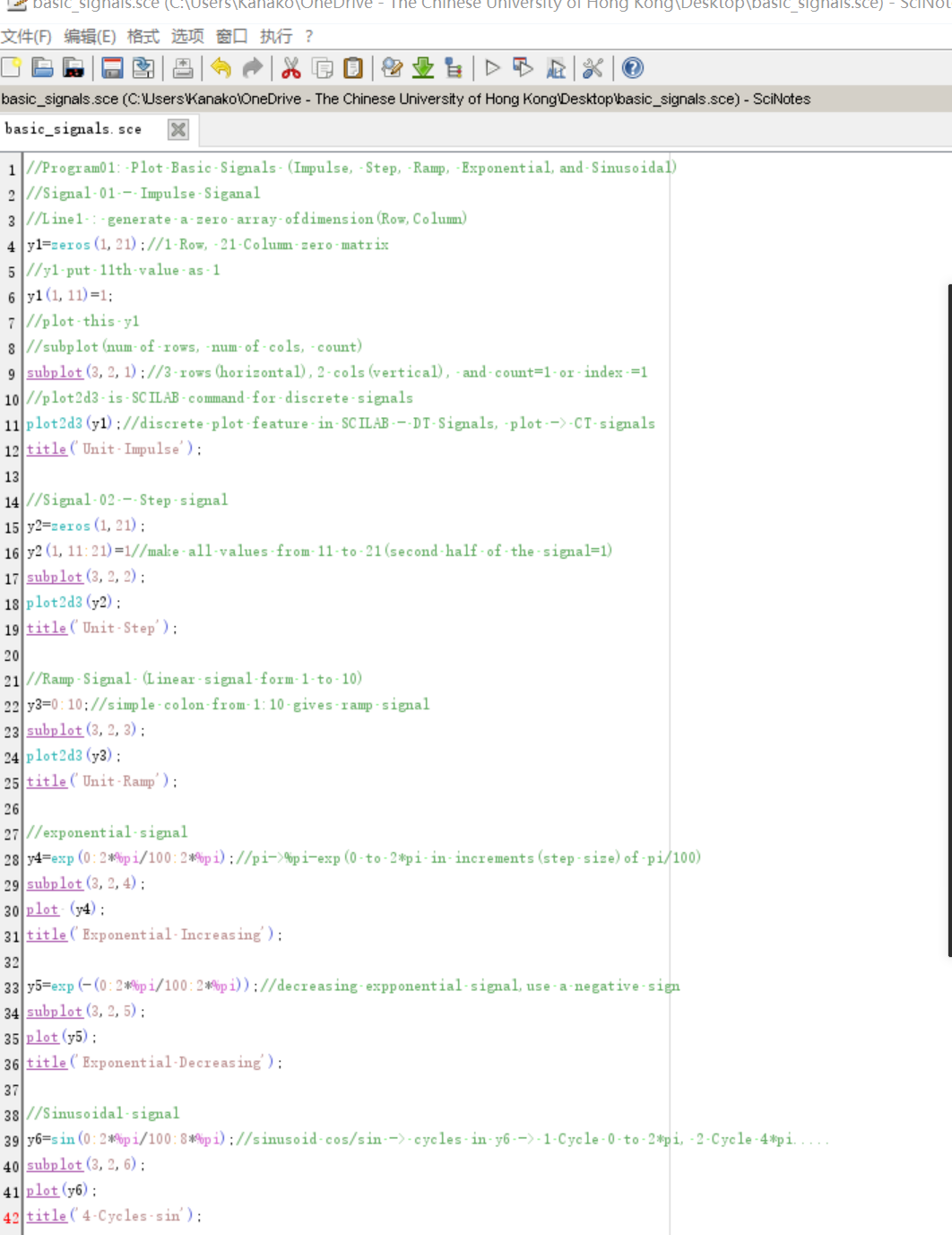
**Group ID: 5**

**Project Topic: Talking Piano**

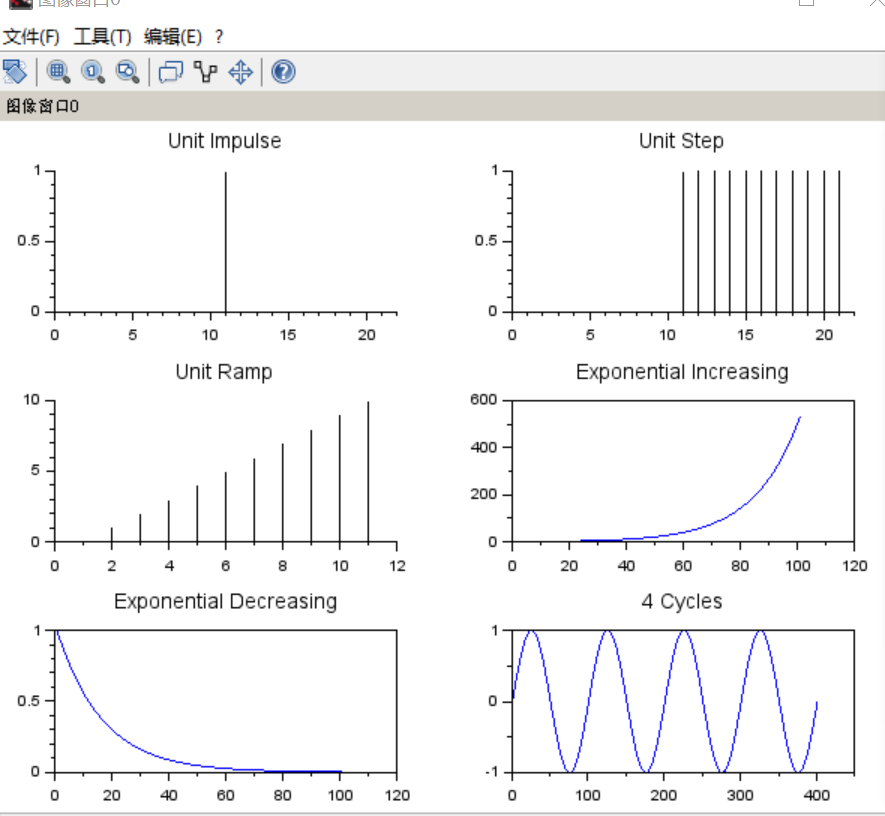
**Bi-weekly Report 4**

**Using SCILAB**

Some basic signal:

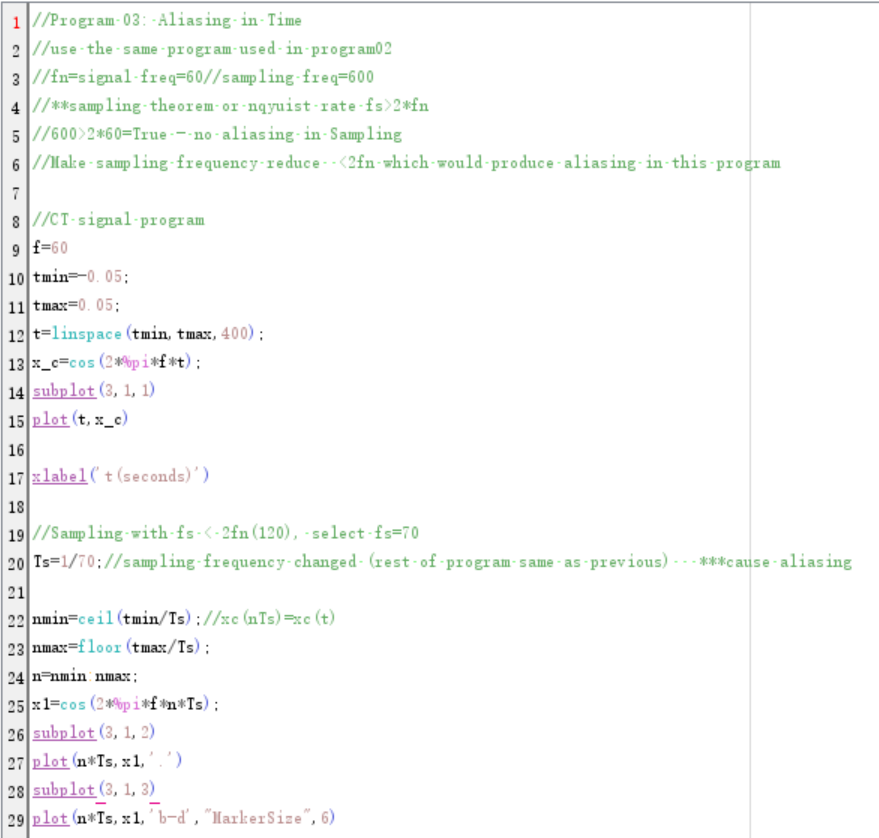


Code↑

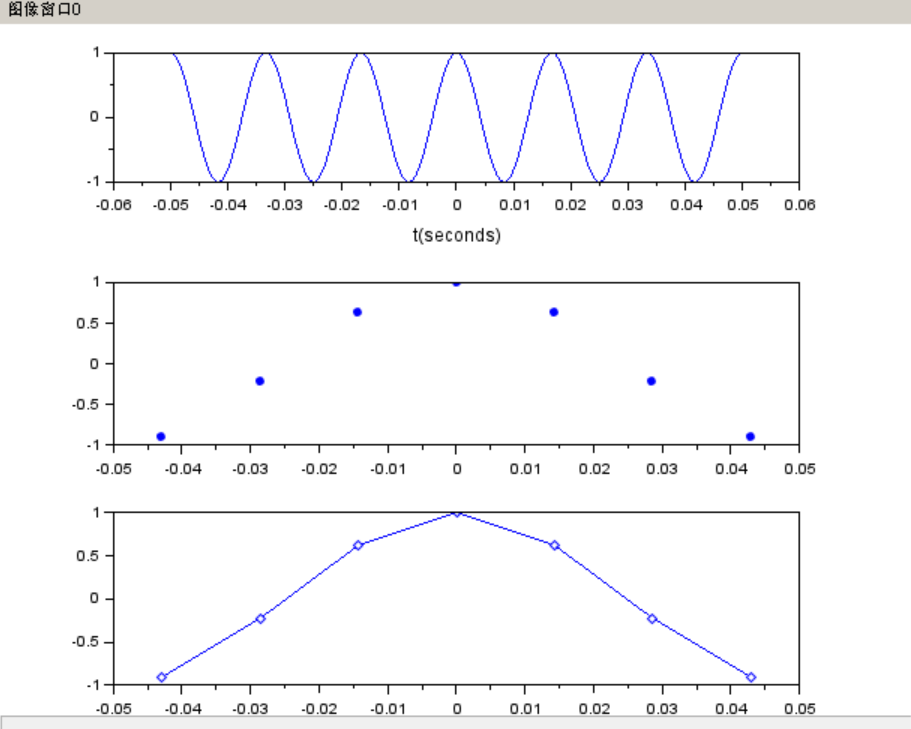


Plot↑

A program related to Nyquist frequency in Report 3:



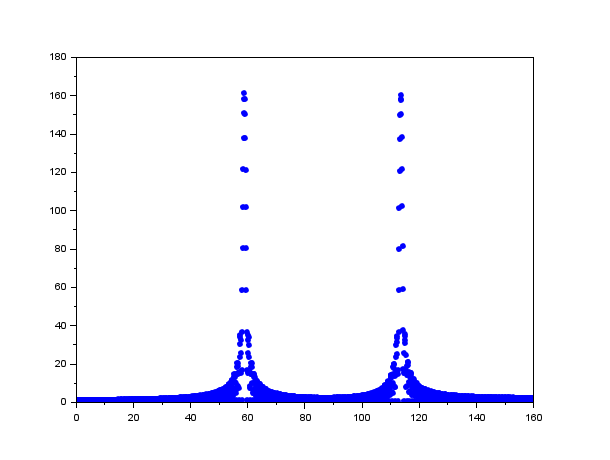
As annotated, sampling frequency that is lower than doubled signal frequency would produce aliasing in the plot of this program.



We can observe that the curve is not smooth(aliasing).

ALIASING, that is image flaws and unpleasant artifacts caused by improper sampling.

一張含有 文字 的圖片

自動產生的描述

Example of DFT in Scilab as illustrated in desmos in report before

**Group ID: 5**

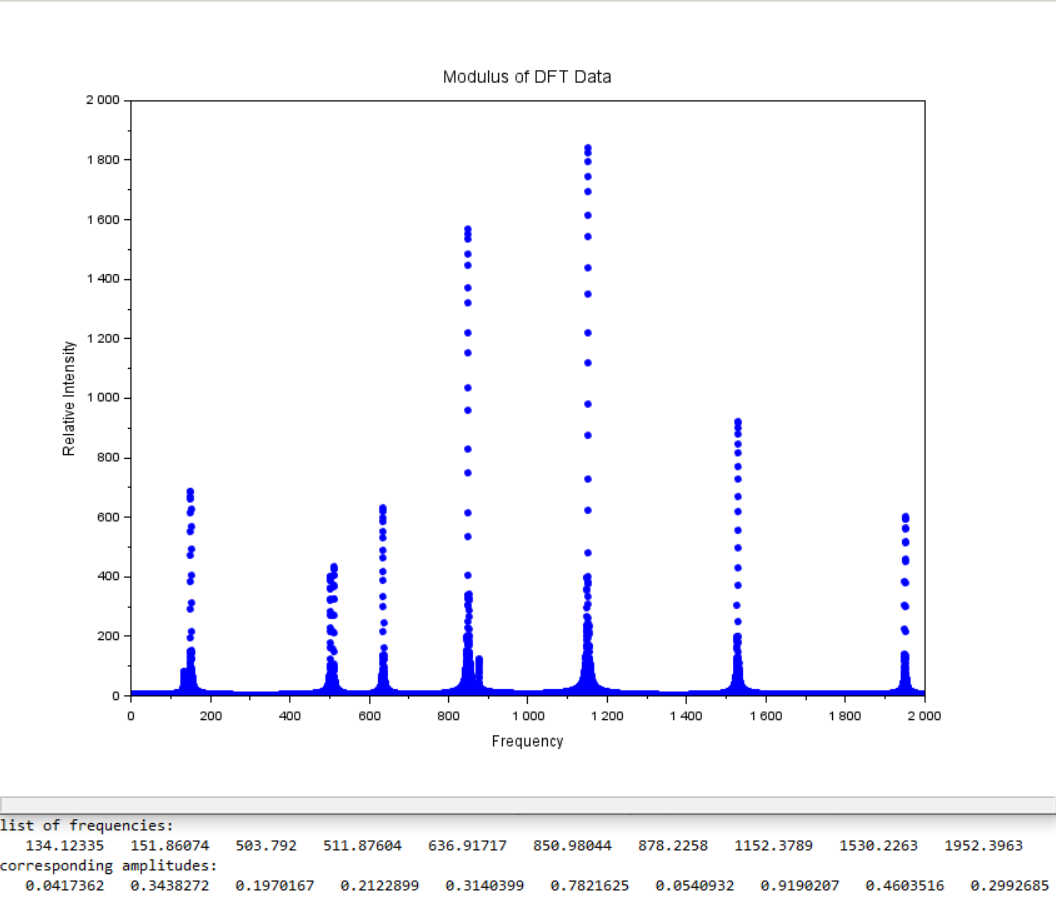
**Project Topic: Talking Piano**

**Bi-weekly Report 5**

**Explanation for the code last week**

**一張含有 文字 的圖片

自動產生的描述**

****

***The revised version of codes in report 4***

***For the latest update, please visit:***[***https://github.com/dmddjack/ESTR1005\_Project***](https://github.com/dmddjack/ESTR1005_Project)

Define frequency data of the sinusoidal wave:

Define amplitude data of the corresponding frequency:

Modify the amplitude data to a diagonal matrix:

Define vector which consists of entries of uniformly distributed time in ascending order:

, whereis sampling rate.

Define vector which consists of entries of uniformly distributed frequencies in ascending order, which is the sample of horizontal axis:

, where, according to Nyquist limit.

Define matrix operation for any as

According to function of sinusoidal wave:

Define sample data of the function of sinusoidal wave:

Sum up each row:

The equation of DFT is:

According to Euler’s formula:

We have:

,

Define sample data of the real part and imaginary part of the exponential part function in DFT separately:

Thus,

Plot into a single graph, which is the graph generated by the code in Scilab in report 4.

Appendix: In the above sample code, we take

All the frequency and amplitude data of the sinusoidal wave is generated randomly.

P10

**Group ID: 5**

**Project Topic: Talking Piano**

**Bi-weekly Report 6**

**Linear convolution of Filtering of signals**: suppose there’s a music audio, if we want to get rid of some of the noise or smoothen the voice, we may filter it with another mask or signal, and this filtering is done using convolution.

The **Convolution Theorem** states that under suitable conditions the Fourier transform of a convolution of two functions (or signals) is the pointwise product of their Fourier transforms.

F[g(x) \* f(x)] = F[f(x)] F[g(x)]

x(n) = [1 0 1 0] as a 4-point signal

h(n) = [1 0] is filter of length 2

What convolution does is scan x(n) with h(n), it erases the signal and smoothens it.

Linear convolution of x(n) with h(n) is represented as x(n)\*h(n) and calculated as:

1 0 1 0

1 0

1×1+1×0=0

-------------------------

1 0 1 0

1 0

1×1+0×0=0

-------------------------

1 0 1 0

1 0

0×1+1×0=0

-------------------------

1 0 1 0

1 0

1×1+0×0=1

-------------------------

1 0 1 0

1 0

0×1+0×0=0

-------------------------

Convolution of x(n)\*h(n) = [0 1 0 1 0]

The size of the output in linear convolution is sum of length of two signals – 1.

