DSP_Assignment1 報告

711281114 通訊碩一 蘇品瑒

1. 數學推導

$x(t) = e^{j\Omega t}$

$$e^{j\Omega t} = I(t)\cdot R + y(t)$$

$$I(t) = \frac{dQ(t)}{dt} = C\cdot\frac{dy(t)}{dt}$$

$$e^{j\Omega t} = C\cdot\frac{dy(t)}{dt}\cdot R + y(t)$$

$$= RC\frac{dy(t)}{dt} + y(t)$$

$$e^{j\Omega t} = RC\lim_{\tau\to 0}\frac{y(t)-y(t-\tau)}{\tau} + y(t)$$

Assume τ is very small

$$e^{j\Omega t} \simeq RC\left(\frac{y(t)-y(t-\tau)}{\tau}\right) + y(t)$$

Setting $t = n\cdot\tau$ $(n \in \mathbb{Z})$ Discretize

圖（一）：算式推導

$$e^{j\Omega(nt)} = RC\left(\frac{y(n\tau)-y(n\tau-\tau)}{\tau}\right) + y(n\tau)$$

Let $x[n] = x(n\tau)$, $y[n-1] = y(\tau(n-1))$, $y[n] = y(n\tau)$

So, $e^{j\Omega(n\tau)} = e^{j\Omega[n]} = x[n]$

$$e^{j\Omega[n]} = \frac{RC}{\tau}y[n] - \frac{RC}{\tau}y[n-1] + y[n]$$

$$y[n] = \frac{RC}{RC+\tau}y[n-1] + \frac{\tau}{\tau+RC}e^{j\Omega[n]}$$

discrete system

圖（二）：算式推導

Q1: Find $y(t)$ => continuous

According to previous mathmatic manipulation:

$$e^{j\Omega t} = C \cdot \frac{dy(t)}{dt} \cdot R + y(t)$$

$$= RC \frac{dy(t)}{dt} + y(t)$$

Let $y(t) = Ae^{j\Omega t}$

$$e^{j\Omega t} = RC \cdot A \cdot j\Omega \cdot e^{j\Omega t} + Ae^{j\Omega t}$$

$$A = \frac{1}{1+j\Omega RC} \quad , \quad y(t) = \frac{1}{1+j\Omega RC} e^{j\Omega t} \quad \cancel{e^{j\phi}}$$

$$\frac{1}{1+j\Omega RC} = \frac{1}{\sqrt{1+(\Omega RC)^2}\, e^{j\phi}} \quad , \quad \phi = \tan^{-1}\left(\frac{\Omega RC}{1}\right) = \tan^{-1}(\Omega RC)$$

$$y(t) = \frac{1}{\sqrt{1+(\Omega RC)^2}} e^{-j\tan^{-1}(\Omega RC)} e^{j\Omega t} = \frac{1}{\sqrt{1+(\Omega RC)^2}} e^{j(\Omega t - \tan^{-1}(\Omega RC))}$$

#

圖（三）：問題一算式

$$Q2: \quad x = e^{j\Omega t}, \quad R = 1000, \quad C = \frac{1}{2\pi} \times \frac{1}{400} \times \frac{1}{1000}$$

$$\Omega = 2\pi f, \quad f = 100, 400, 3000$$

According to previous question:

$$y(t) = \frac{1}{\sqrt{1+(\Omega RC)^2}} e^{j(\Omega t - \tan^{-1}(\Omega RC))}$$

$$RC = \frac{1}{800\pi} \qquad \cos(\omega t)$$

Case1: $f = 100$

$$\Omega RC = 200\pi \times \frac{1}{\cancel{800\pi}_{4}} = \frac{1}{4}$$

$$y(t) = \frac{1}{\sqrt{1+\frac{1}{16}}} e^{j(200\pi t - \tan^{-1}(\frac{1}{4}))}$$

$$= \frac{4\sqrt{17}}{17} e^{j(200\pi t - \tan^{-1}(\frac{1}{4}))}$$

$$\simeq 0.9701 \, e^{j(200\pi t - \tan^{-1}(\frac{1}{4}))}$$

圖（四）：問題二算式

Case 2: $f = 400$

$$\Omega RC = 800\pi \times \frac{1}{800\pi} = 1$$

$$y(t) = \frac{\sqrt{2}}{2} e^{j(800\pi t - \tan^{-1}(1))}$$

$$\simeq 0.7071 e^{j(800\pi t - \frac{\pi}{4})}$$

Case 3: $f = 3000$

$$\Omega RC = \overset{15}{\cancel{6000\pi}} \times \frac{1}{\underset{2}{\cancel{800\pi}}} = \frac{15}{2}$$

$$y(t) = \frac{1}{\sqrt{1^2 + \frac{225}{4}}} e^{j(6000\pi - \tan^{-1}(\frac{15}{2}))}$$

$$= \frac{2}{\sqrt{229}} e^{j(6000\pi - \tan^{-1}(\frac{15}{2}))}$$

$$\simeq 0.1322 \; e^{j(6000\pi - \tan^{-1}(\frac{15}{2}))}$$

圖（五）：問題二算式

2. 程式設計、模擬：

根據前面的數學推導，可以得知下面式子：

$$y(t) \approx \frac{\tau}{\tau + RC} e^{j\Omega t} + \frac{RC}{\tau + RC} y(t - \tau)$$

在程式裡面，我們一定要用離散的方式下去做模擬，所以又可以把上式的連續函數透過下式轉換成離散式子：

$$\text{Setting } t = n\tau \;\; (n \in \mathbb{Z})$$

這個式子代表著將原本時間為 t 的 function 分成許多等分，每一等分佔τ極短的時間($\tau \to 0$)，總共有 n 等分，所以可以轉換成下式：

$$y(n\tau)=\frac{\tau}{\tau+RC}e^{j\Omega(n\tau)}+\frac{RC}{\tau+RC}y(n\tau-\tau)$$

最後，透過下面的式子轉換成離散版的原式：

$$\text{Let } x[n]=x(n\tau), y[n]=(n\tau), y[n-1]=y(\tau(n-1))$$

$$y[n]=\frac{\tau}{\tau+RC}e^{j\Omega[\mathrm{n}]}+\frac{RC}{\tau+RC}y[n-1]$$

上面的式子裡，有一個 exponential 項，為了方便進行模擬，透過尤拉公式將此項化為 cos 和 sin

$$e^{j\Omega[\mathrm{n}]}=cos(\Omega[\mathrm{n}])+jsin(\Omega[\mathrm{n}])$$

3. 程式架構：

```
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include <time.h>
#include <stdint.h>
#define M_PI 3.14159265358979323846


void simulate(int N,double freq, double tao, double RC,char txt[]){⋯

void gen_original(int N,double freq,char txt[]){⋯


int main(int argc, char *argv[]){

    double R_C = 1000.0*(1.0/4.0*M_PI)*(1.0/160000.0);                    //R*C
    int N = 20000;                                                         //Total points

    gen_original(N,100.0,"100hz.txt");
    simulate(N,100.0,(1.0/4000.0),R_C,"100hz{4000}.txt");
    simulate(N,100.0,(1.0/8000.0),R_C,"100hz{8000}.txt");
    simulate(N,100.0,(1.0/16000.0),R_C,"100hz{16000}.txt");

    gen_original(N,400.0,"400hz.txt");
    simulate(N,400.0,(1.0/4000.0),R_C,"400hz{4000}.txt");
    simulate(N,400.0,(1.0/8000.0),R_C,"400hz{8000}.txt");
    simulate(N,400.0,(1.0/16000.0),R_C,"400hz{16000}.txt");

    gen_original(N,3000.0,"3000hz.txt");
    simulate(N,3000.0,(1.0/4000.0),R_C,"3000hz{4000}.txt");
    simulate(N,3000.0,(1.0/8000.0),R_C,"3000hz{8000}.txt");
    simulate(N,3000.0,(1.0/16000.0),R_C,"3000hz{16000}.txt");

}
```

圖（六）：模擬程式簡略圖

首先，我在 main function 內先計算好 R*C 的數值，並且我定義一個 N，這個 N 代表著我要將弦波分割成幾等份，這邊我用 20000 下去跑。假設 N 設定的不夠，出來的數值會不夠細節，也就是說後面在 matlab 畫圖時會不像弦波。

這裡我總共寫了兩個 function。首先是 simulate，此 function 負責跑上面推導出的算式並且我用一個二維陣列 yn 記錄實部和虛部的數值並且輸出成一個

txt 檔案。

```c
    double *cos_array = (double*)malloc(sizeof(double)*N);
    double *sin_array = (double*)malloc(sizeof(double)*N);

    for(int a = 0;a<N;a++){                                   //Create cos and sin data (Represent x[n] = e^(j*omega*n/N) in euler's form)
        cos_array[a] = cos(2*M_PI*freq*(double)a/N);
        sin_array[a] = sin(2*M_PI*freq*(double)a/N);
    }

    double **yn = (double**)malloc(sizeof(double*)*2);
    int z;
    for (z = 0; z < 2; z++)
        yn[z] = (double*)malloc(sizeof(double)*(N+N/2));      //To simulate truncated sinusoid, zero-padding N/2
    double result = 0.0;

    for(int i = 0;i < 2 ; i++){
        for(int k =0; k<(N+N/2);k++){
            yn[i][k] = 0.0;
        }
    }
```

圖（七）：模擬程式 function: simulate 局部圖

這裡我為了還原現實的訊號，我在 cos 和 sin 陣列的尾端在補上一些 0 來模擬訊號突然中斷的狀況，並且在最後利用 matlab 畫圖實的確有發現暫態(Transient)的現象發生。

```c
    FILE *txtfp;                                              //Open a txt file
    txtfp=fopen(txt,"wb");
    if(txtfp==NULL){
        printf( "open failure" );
    }
    else{

        for(int a = 0; a < (N+N/2); a++){
            if(a==0){
                yn[0][a] = (tao/(tao+RC))*cos_array[a];
                yn[1][a]= (tao/(tao+RC))*sin_array[a];
            }
            else{
                if(a < N){
                yn[0][a] = (tao/(tao+RC))*cos_array[a]+(RC/(tao+RC))*yn[0][a-1];
                yn[1][a] = (tao/(tao+RC))*sin_array[a]+(RC/(tao+RC))*yn[1][a-1];
                }
                else{
                    yn[0][a] = (RC/(tao+RC))*yn[0][a-1];
                    yn[1][a] = (RC/(tao+RC))*yn[1][a-1];
                }
            }
            fprintf(txtfp,"%lf\t%lf\n",yn[0][a],yn[1][a]);      //write into txt
            fflush(txtfp);                                     //clear cache
        }
        for (int i = 0; i < 2; i++)                            //free 2-dim malloc
            free(yn[i]);
        free(yn);
        free(cos_array);
        free(sin_array);
        fclose(txtfp);                                         //close txt file
        fflush(txtfp);
    }
}
```

圖（八）：模擬程式 function: simulate 局部圖

而 gen_original 則是直接印出$e^{j\Omega[n]}$的值，代表著原本訊號，並且一樣用二維陣列記錄實部和虛部並且輸出成 txt 檔案。根據題目要求，我們有三種不同的頻率(100Hz、400Hz、3000Hz)以及三種不同的τ值(1/4000、1/8000、1/16000，單位為秒)，加上 3 種原本的$e^{j\Omega[n]}$數值，所以總共有 12 個 txt 檔案輸出。

```c
void gen_original(int N,double freq,char txt[]){
    double *cos_array = (double*)malloc(sizeof(double)*N);
    double *sin_array = (double*)malloc(sizeof(double)*N);

    for(int a = 0;a<N;a++){                          //Create cos and sin data (Represent x[n] = e^(j*omega*n/N) in euler's form)
        cos_array[a] = cos(2*M_PI*freq*(double)a/N);
        sin_array[a] = sin(2*M_PI*freq*(double)a/N);
    }
        FILE *txtfp;                                 //Open a txt file
    txtfp=fopen(txt,"wb");
    if(txtfp==NULL){
        printf( "open failure" );
    }
    else{

        for(int a = 0; a < N; a++){
            fprintf(txtfp,"%lf\t%lf\n",cos_array[a],sin_array[a]);     //write into txt
            fflush(txtfp);                           //clear cache
        }

        free(cos_array);
        free(sin_array);
        fclose(txtfp);                               //close txt file
        fflush(txtfp);
    }
}
```

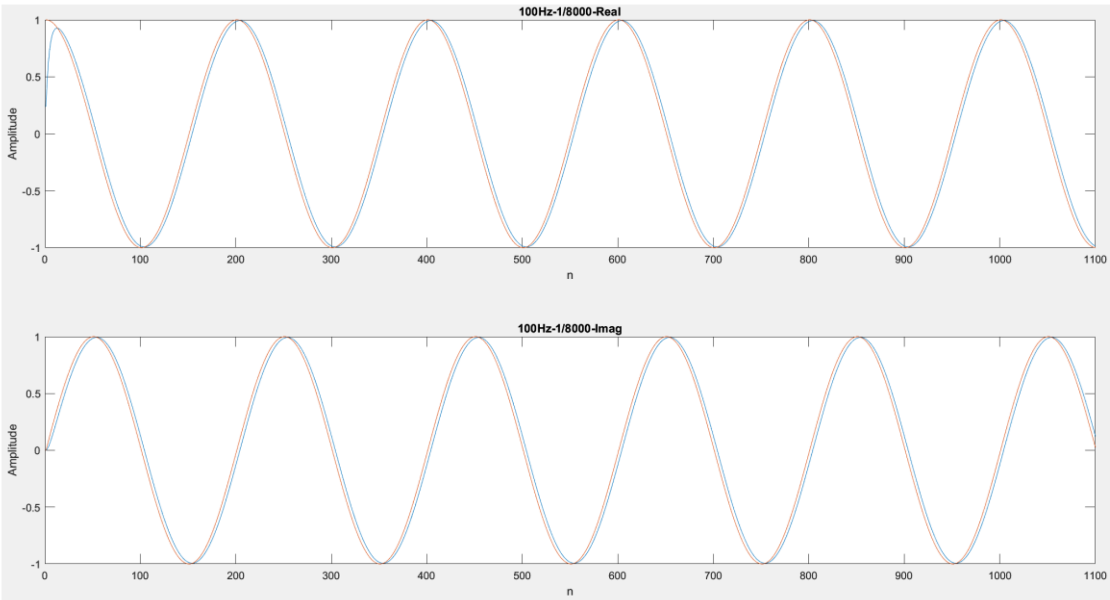
圖（九）：模擬程式 function: gen_original

有了這 12 個 txt 檔，為了方便觀察它的變化情形，我使用 matlab 讀取 txt 檔案的功能將這些數值讀取至 matlab 之後並對數值做 plot 的動作。

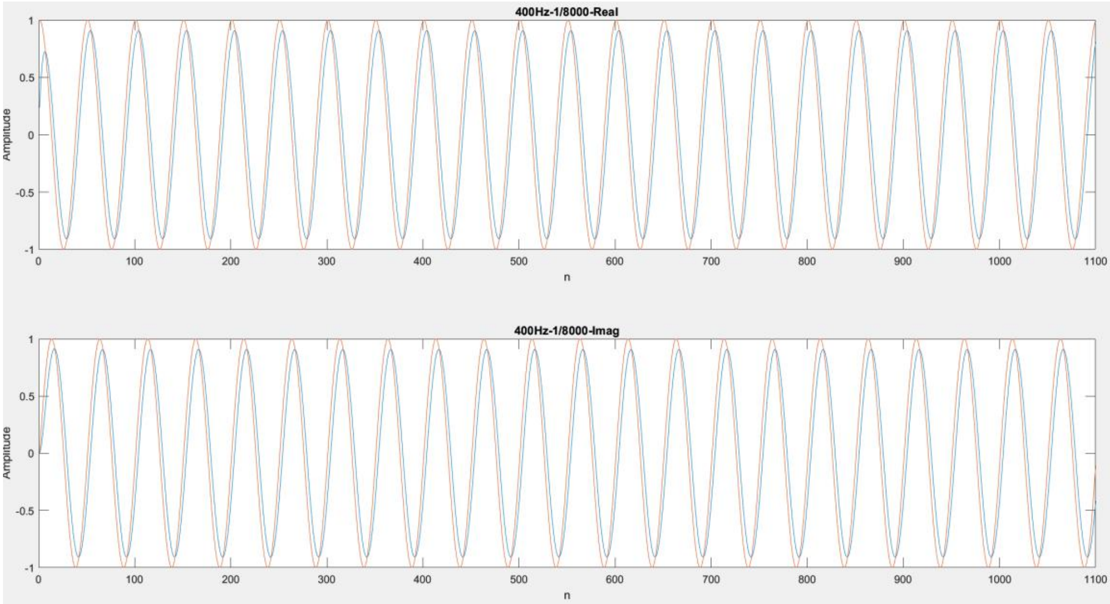
4. 結果討論：

4.1 相同τ、不同頻率的情況

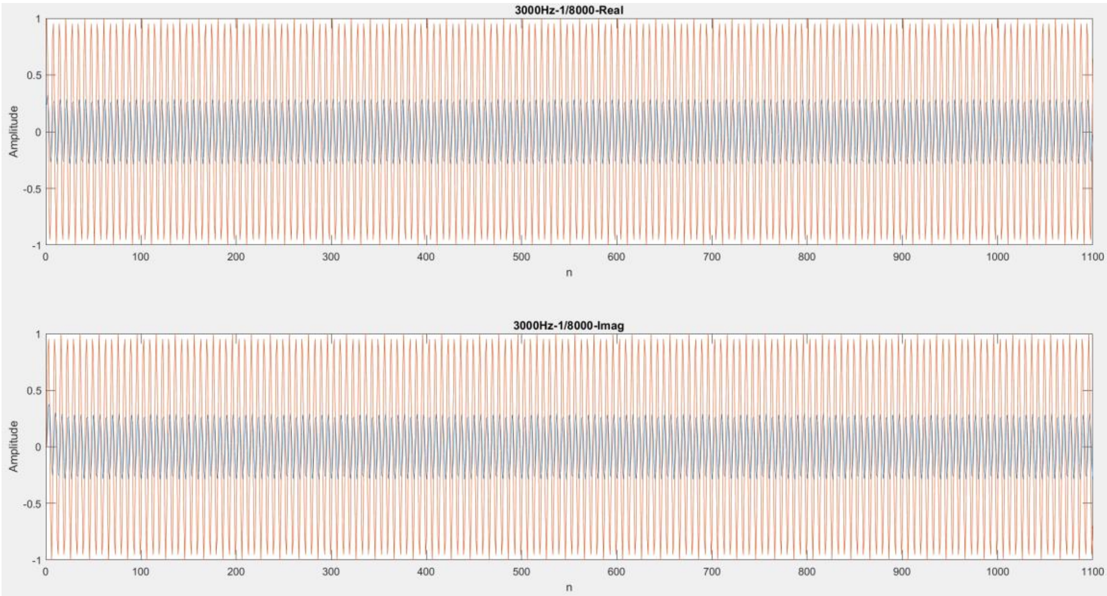
這邊我固定選定$\tau = 1/8000$的圖形。



圖（十）：100Hz、1/8000 實部(上)虛部(下)前段波型



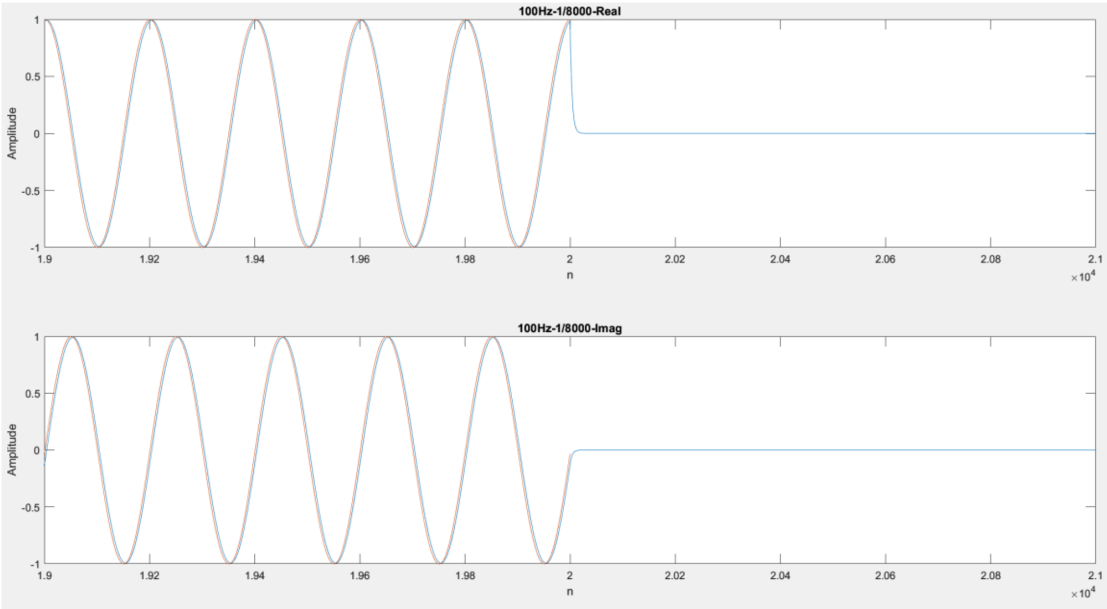
圖（十一）：400Hz、1/8000 實部(上)虛部(下)前段波型



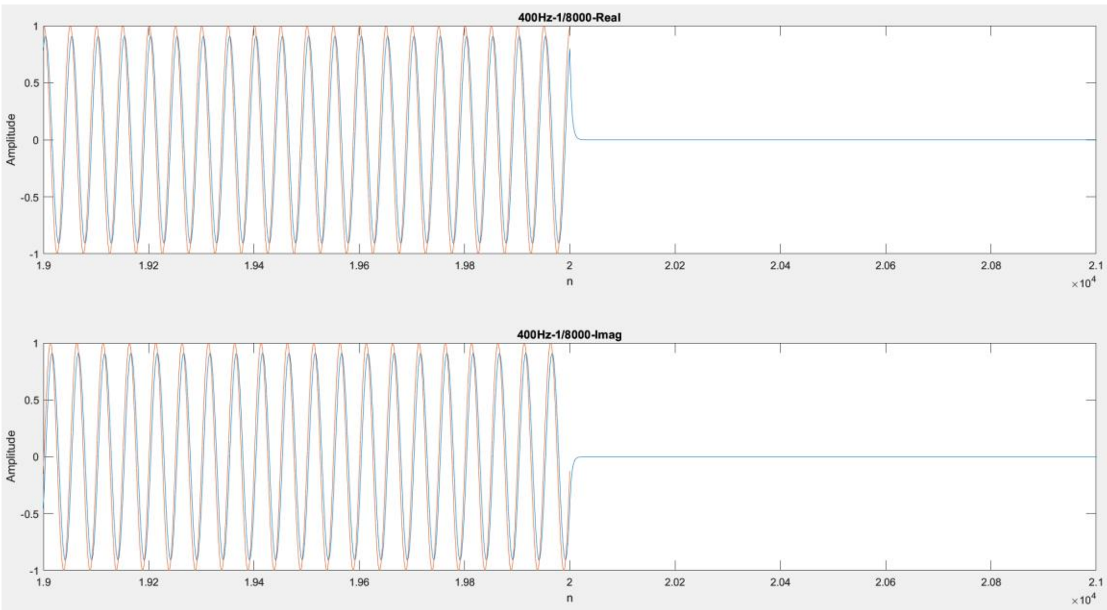
圖（十二）：400Hz、1/8000 實部(上)虛部(下)前段波型

由上面三張圖，我觀察到不論是哪個頻率，都有濾波的效果，也就是將原訊號(橘色訊號)變成變化率較低的藍色訊號，並且都會差一些相位。而在這三者之中，又以 3000Hz 的濾波效果最為明顯。仔細觀察這三張圖也可以發現藍色訊號在剛開始時圖型感覺有些「跑掉」，是因為訊號還在暫態，必須要一段時間才可到達穩態。

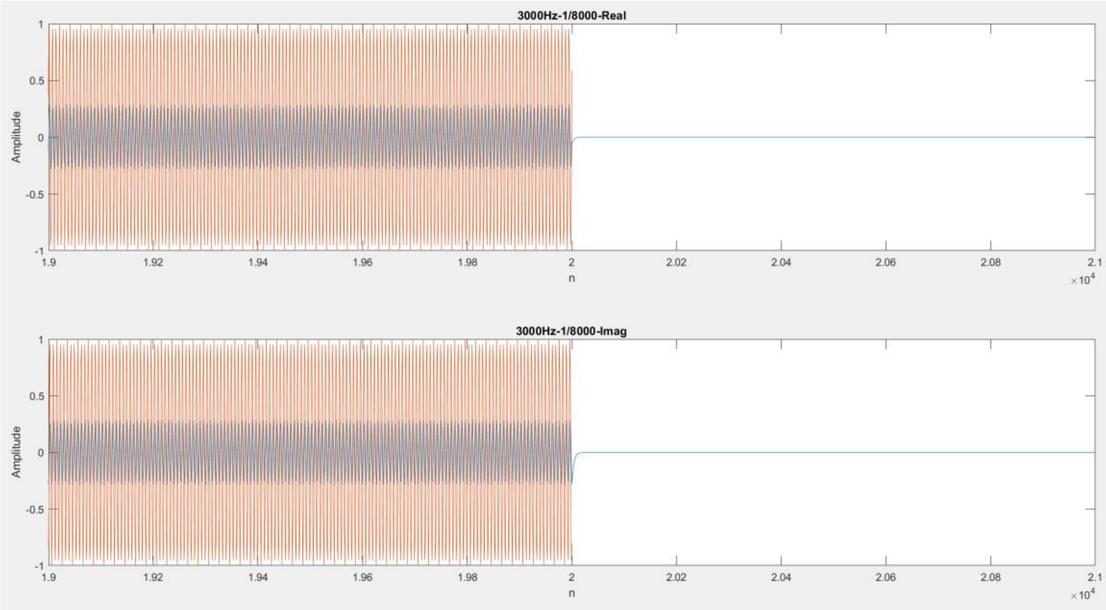
前面程式有提到，為了模擬現實非無限的訊號，我在 cos 和 sin 陣列的尾端在補上一些 0 來模擬訊號突然中斷的狀況，這點在 matlab 畫圖之後也可以觀察到。



圖（十三）：100Hz、1/8000 實部(上)虛部(下)後段波型



圖（十四）：400Hz、1/8000 實部(上)虛部(下)後段波型

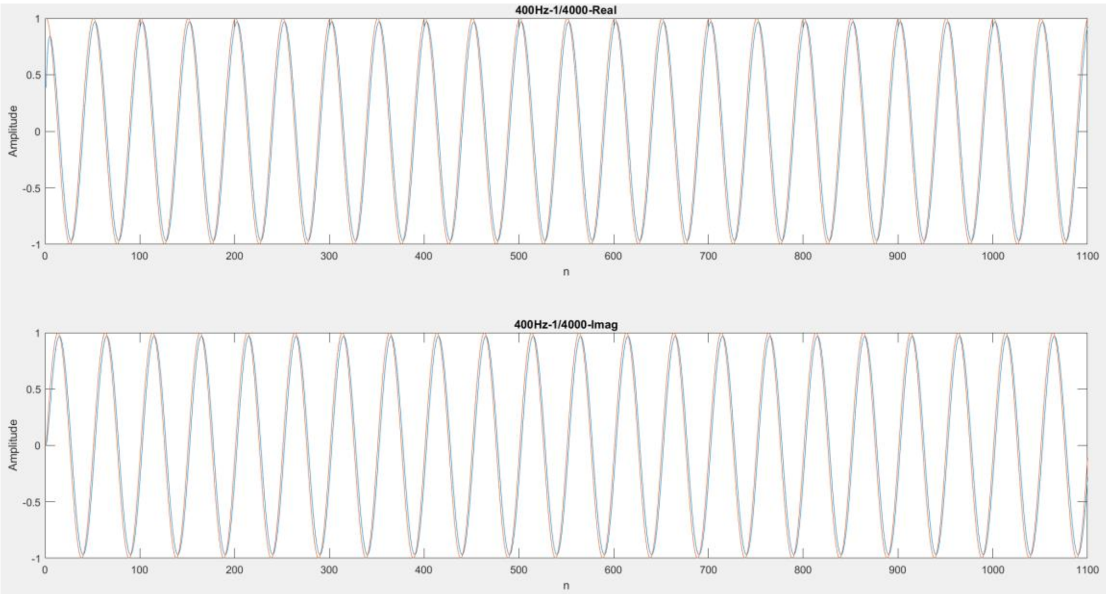


圖（十五）：3000Hz、1/8000 實部(上)虛部(下)後段波型

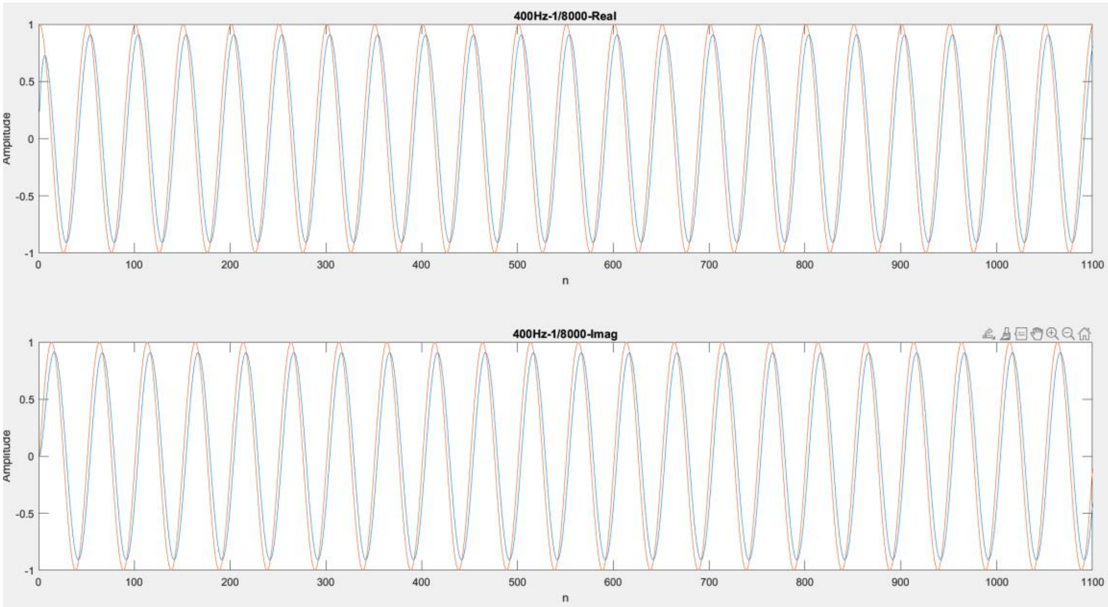
可以看到在三種不同頻率的藍色訊號尾端都會由一個值慢慢趨近於 0，我認為這就是暫態時候的情況。

4.2 相同頻率、不同τ的情況

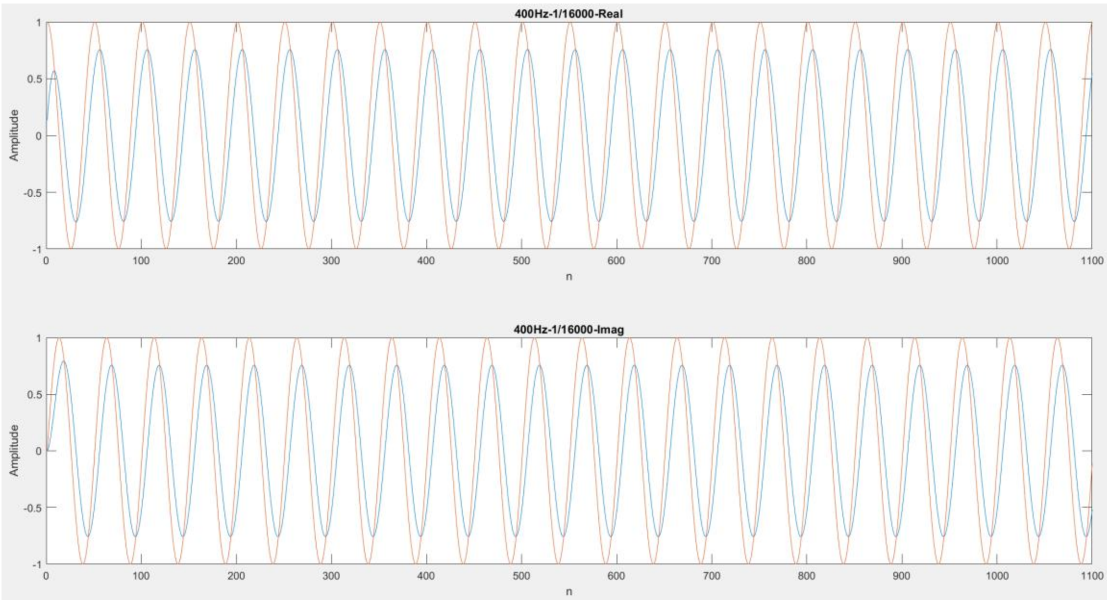
這裡我使用 400Hz 的圖型當作例子



圖（十六）：400Hz、1/4000 實部(上)虛部(下)前段波型

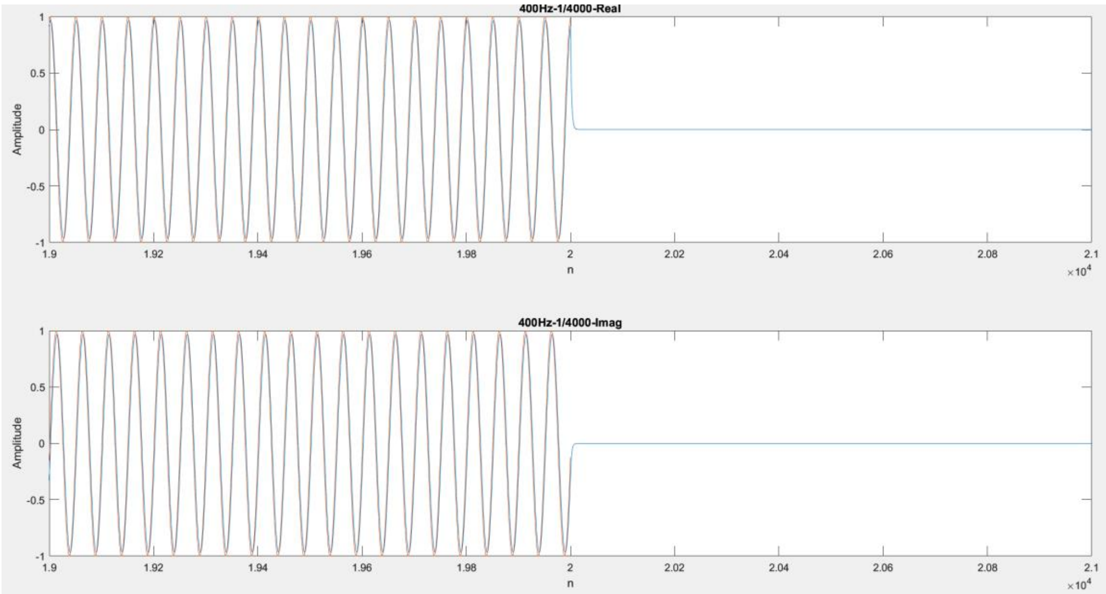


圖（十七）：400Hz、1/8000 實部(上)虛部(下)前段波型

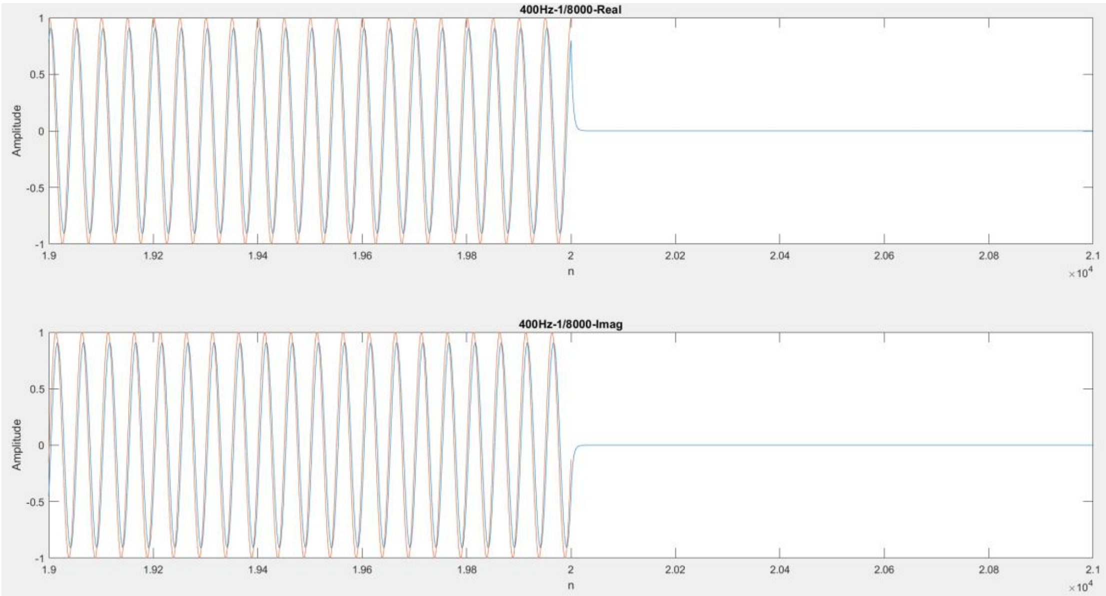


圖（十八）：400Hz、1/16000 實部(上)虛部(下)前段波型

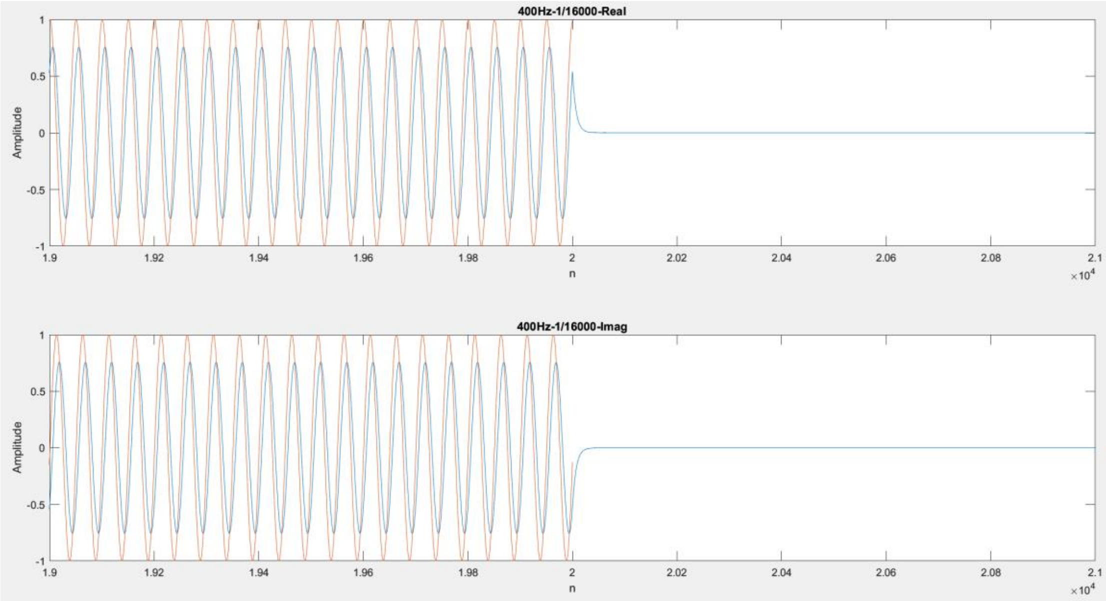
從上面三張圖可以觀察到實部的藍色波型似乎都穩定，但虛部的藍色波型是隨著τ越來越小越穩定，τ可以當作是將資料切成 N 等份，每個等份之間的時間相差多少，τ越小代表每個等份之間的時間相差很小，那麼兩筆資料間的差異也會越小，所以才有上面三張圖型的差異。



圖（十九）：400Hz、1/4000 實部(上)虛部(下)後段波型



圖（二十）：400Hz、1/8000 實部(上)虛部(下)後段波型



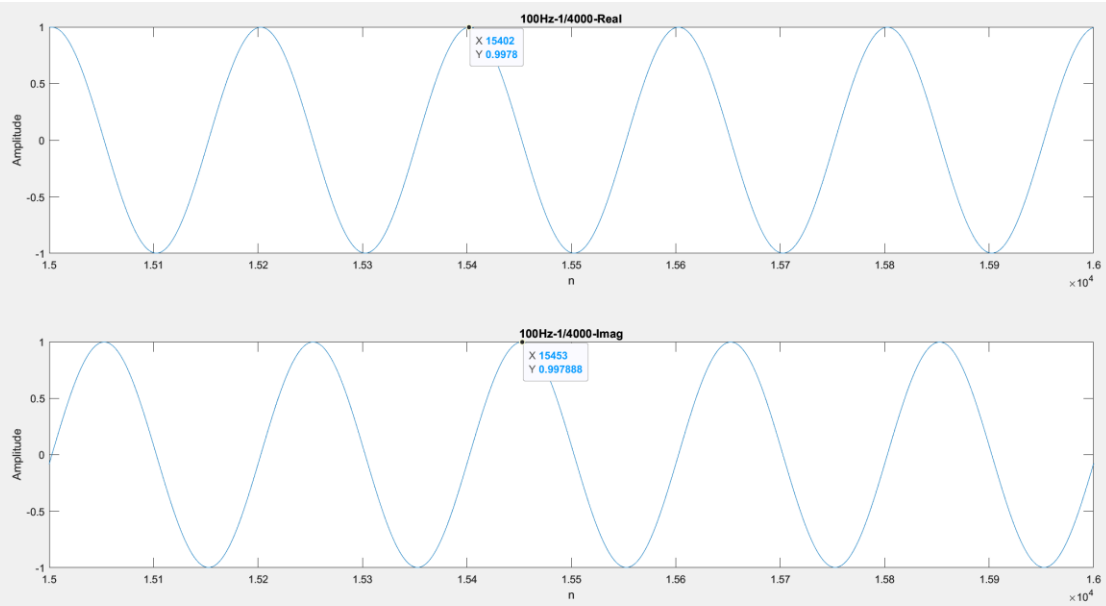
圖（二十一）：400Hz、1/16000 實部(上)虛部(下)後段波型

如果從波型尾段來觀察的話，也可以得到相同的結論：當τ越小，藍色波型可以更快的進入趨近於 0 的穩態。

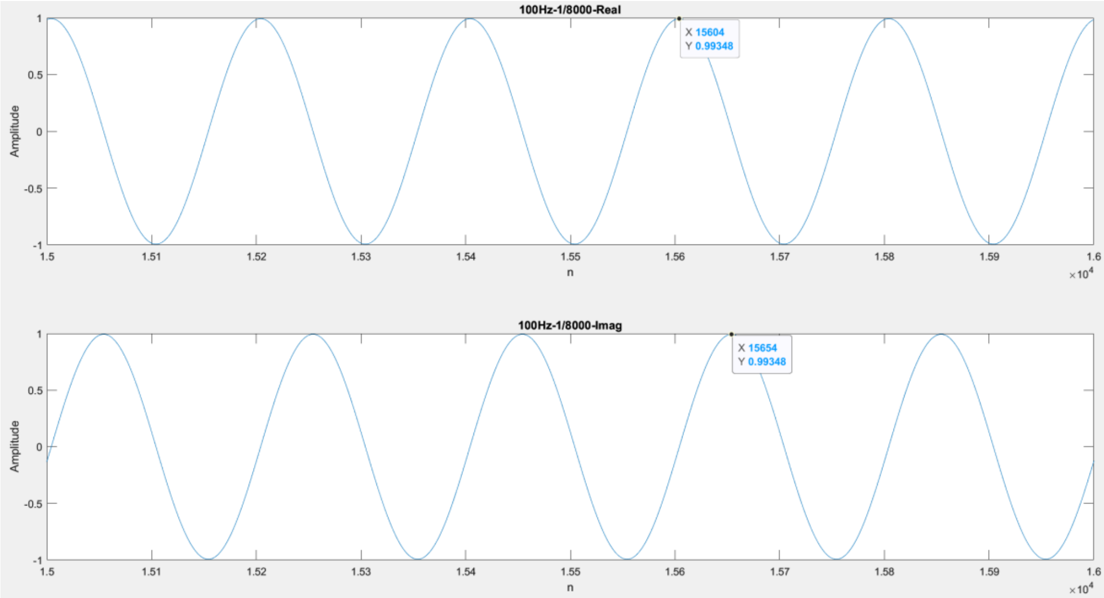
4.3 與理論值之偏差

以下將列出 9 種不同的狀況之穩態數值

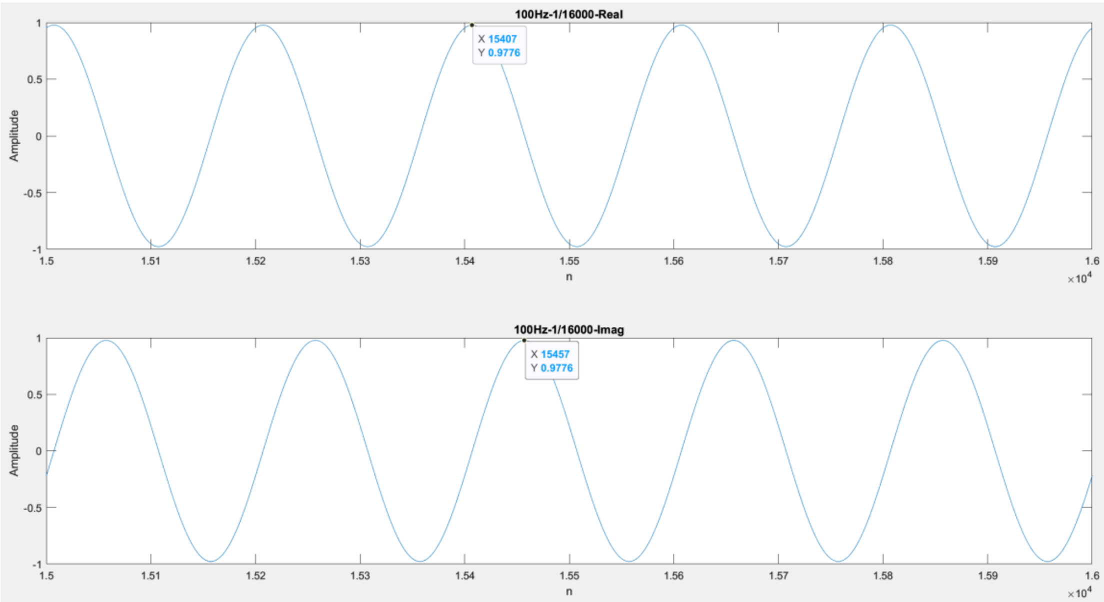
4.3.1 頻率為 100Hz



圖（二十二）：100Hz、1/4000 實部(上)虛部(下)穩態波型



圖（二十三）：100Hz、1/8000 實部(上)虛部(下)穩態波型



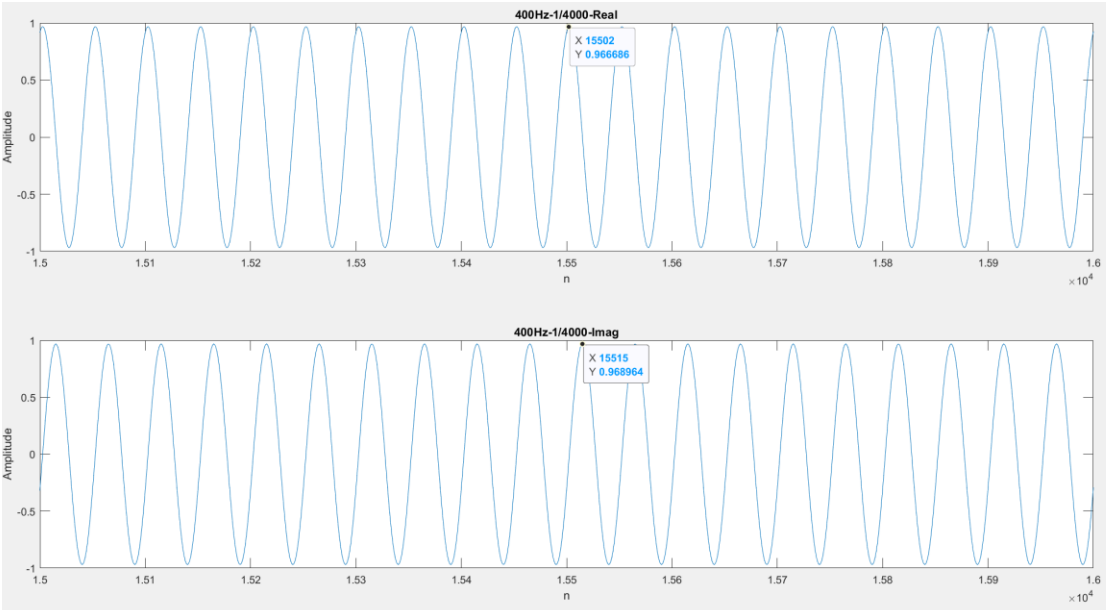
圖（二十四）：100Hz、1/16000 實部(上)虛部(下)穩態波型

根據第一部分數學推導的問題二，我們可以得知理想的濾波圖形為：

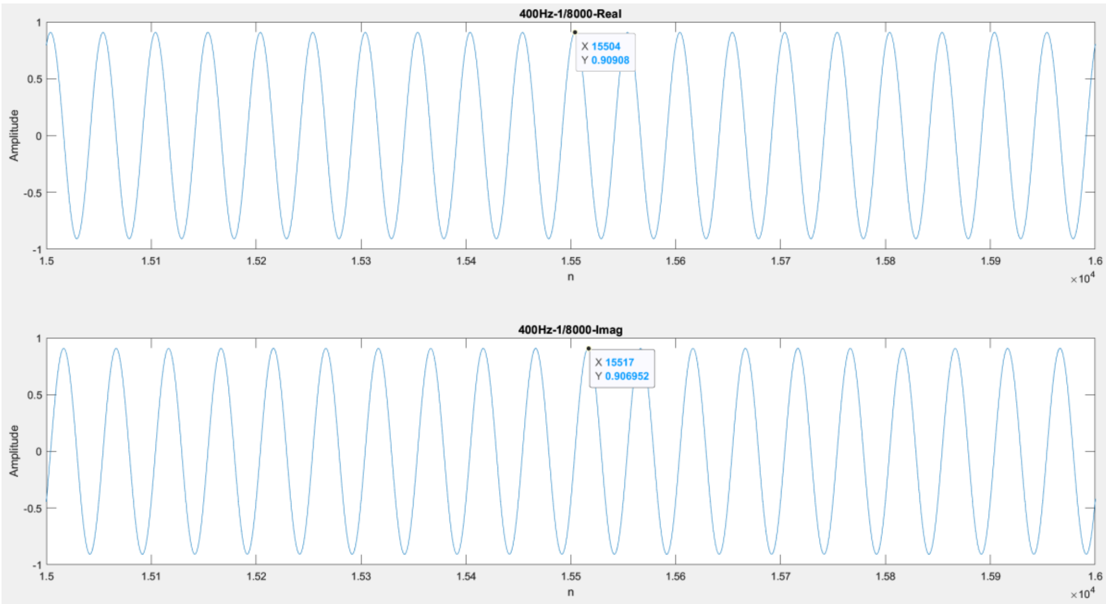
$$y(t) \approx 0.9701e^{j(200\pi t-\tan^{-1}(\frac{1}{4}))}$$

由上式可得知理想振幅為 0.9701，但會有誤差，原因是當我們在數學推導時，為了從連續轉成離散(離散才能夠模擬)，會把微分的 limit 拿掉，因此模擬時會有誤差。當τ值越小時，振幅越能接近理想值，原因如同 4.2，τ可以當作是將資料切成 N 等份，每個等份之間的時間相差多少，τ越小代表每個等份之間的時間相差很小。τ值越小，越能夠代表原本 limit 趨近於 0 的條件，也就越接近理想值。

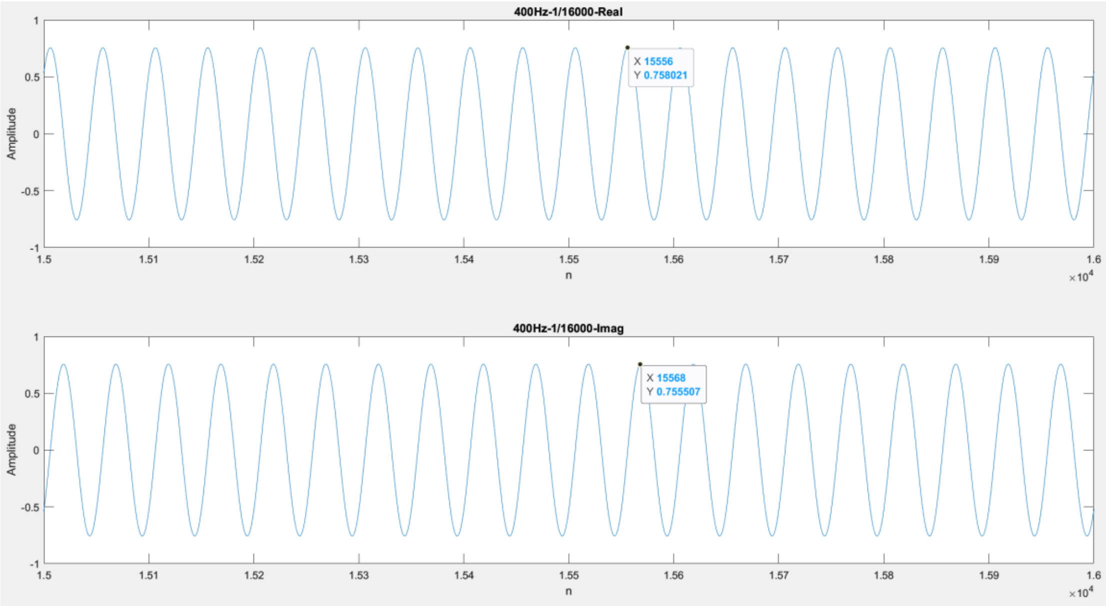
4.3.2 頻率為 400Hz



圖（二十五）：400Hz、1/4000 實部(上)虛部(下)穩態波型



圖（二十六）：400Hz、1/8000 實部(上)虛部(下)穩態波型

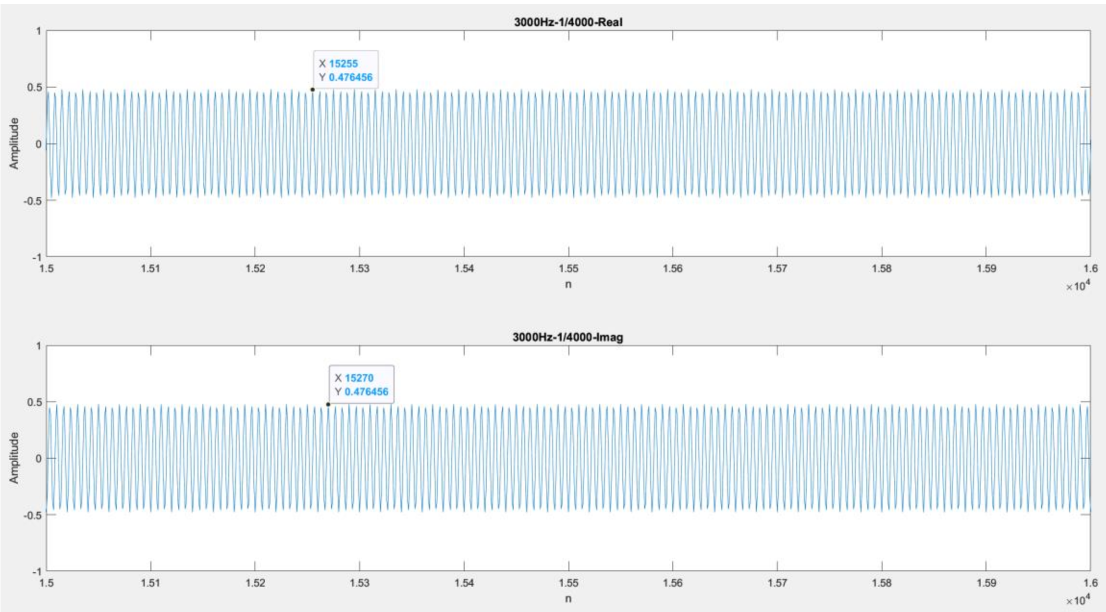


圖（二十七）：400Hz、1/16000 實部(上)虛部(下)穩態波型

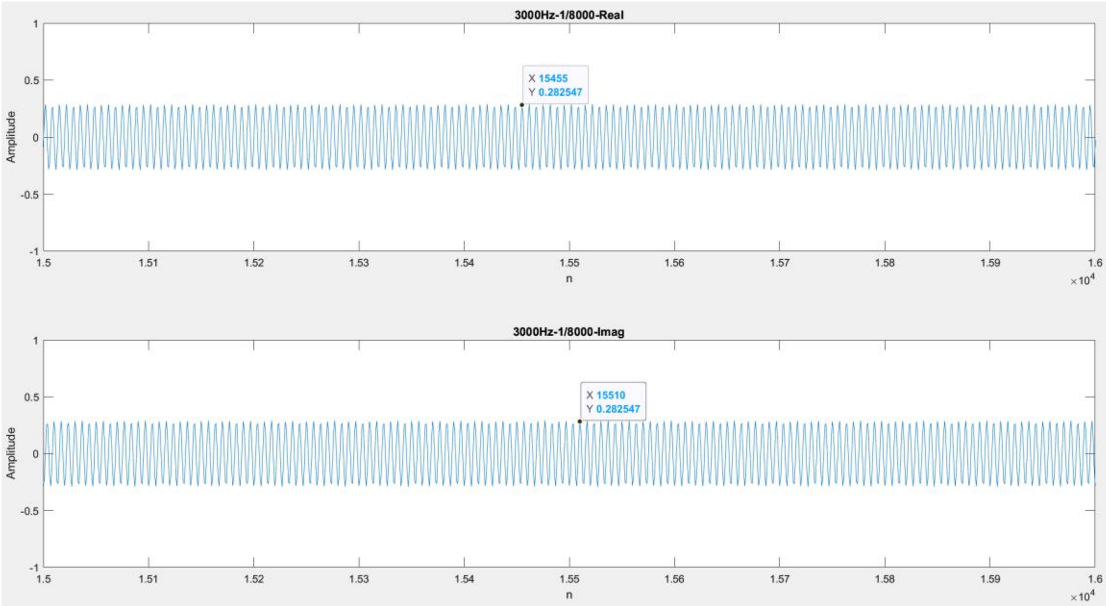
和 100Hz 一樣，根據第一部分數學推導的問題二，我們可以得知理想的濾波圖形為：

$$y(t) \approx 0.7071e^{j(800\pi t-\tan^{-1}(1))}$$

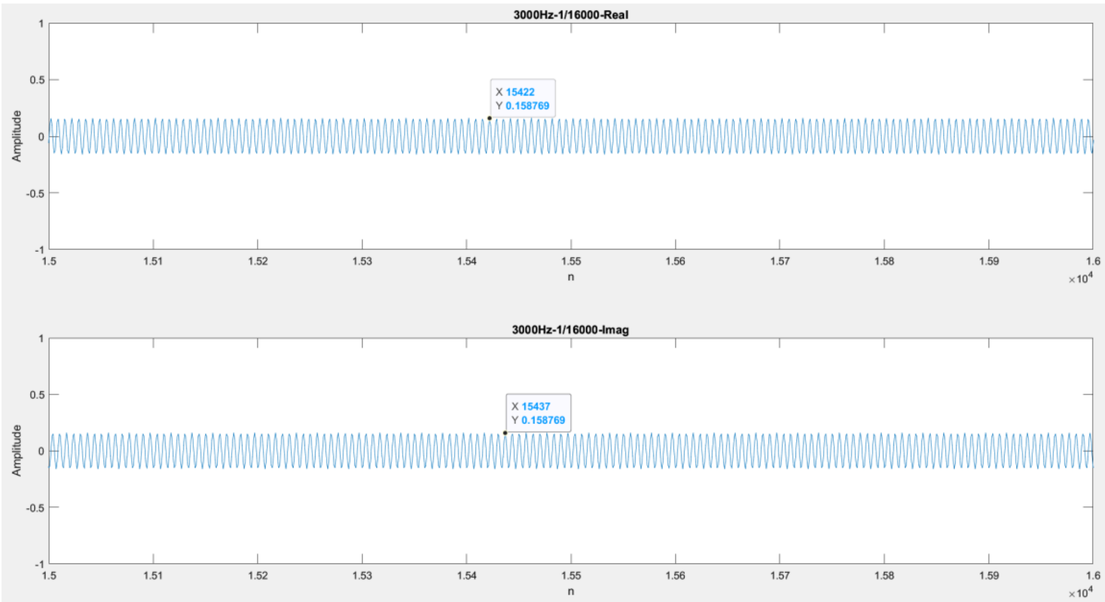
4.3.3 頻率為 3000Hz



圖（二十八）：3000Hz、1/4000 實部(上)虛部(下)穩態波型



圖（二十九）：3000Hz、1/8000 實部(上)虛部(下)穩態波型



圖（三十）：3000Hz、1/16000 實部(上)虛部(下)穩態波型

和上面一樣，根據第一部分數學推導的問題二，我們可以得知理想的濾波圖形為：

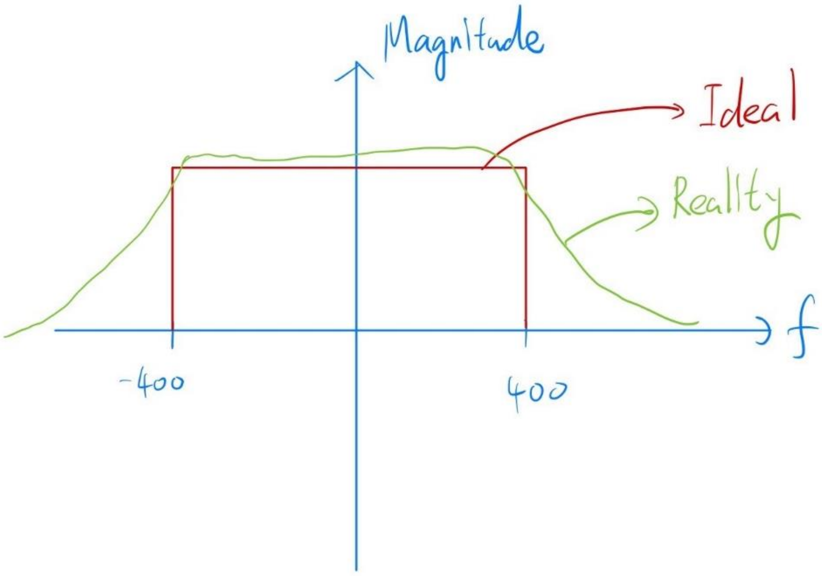
$$y(t) \approx 0.1322e^{j(6000\pi t - \tan^{-1}(\frac{15}{2}))}$$

4.3.4 綜合討論

根據三種不同 Hz 的理想波型公式，可以看到當頻率越高時，振幅會越小，原因是此次模擬的 RC 電路為低通濾波器，截止頻率為：

$$f_{stop} = \frac{1}{2\pi RC} = \frac{1}{2\pi \times 1000 \times (\frac{1}{2\pi} \times \frac{1}{400} \times \frac{1}{1000})} = 400Hz$$

當頻率超過$f_{stop} = 400Hz$時，這個系統就會把該頻率過濾掉，這也就是為甚麼 3000Hz 的振幅下降最多的原因(原本為 1 的弦波下降成 0.13 左右的弦波)。而 400Hz 仍然會下降一點點的原因，我認為是這個濾波的 window 在頻域軸上並不是非常的理想，而是在 400Hz 處左右開始緩慢下降，所以才會有振幅稍微下降的情形。



圖（三十一）：頻域 Window 示意圖