**2. Discrete Fourier Transform**

**Q1: Why do we need to use "(1:64)-1"?**

Matlab vectors start from 1, not zero, but what we need for calculating sin(wt) is from 0 to 63. Therefore, we use n-1 to refer to the appropriate element.

**Q2: What values of w give the cleanest dft?**

I tried w=2, 8, 12.5, 16, 32

Then I found that when w=12.5, the plot give the cleanest result:

we can find that there are two pulse in imaginary domain, one is positive and the other is negative (odd function). And in the real domain, it is almost all zero. That is what we expect for the DFT of sin function.

That is because when we do the sampling for the signal, we have to guarantee the sampling points must cover n (integer) times of period, so that the inverse function wouldn’t produce leakage.

That is, 64\*2\*pi\*w/100=2\*pi\*n(n=1,2,3……)

w=100n/64

w= 1.5625, 3.125, 6.25, 12.5, 25……

**Q3: What happens if we use "cos"?**

I tried w=2, 8, 12.5, 16, 32

Then I also found that when w=12.5, the plot give the cleanest result.

When w=12.5, we can find that there are two pulse in Real domain, both are positive (even function). And in the imaginary domain, it is almost all zero. That is what we expect for the DFT of cos function.

But for the Abs domain, the cos and sin function looks the same.

**Q4: Why doesn’t this look symmetrical?**

The Fourier transform e^(-jwt) =cosx +jsinx.

So the real part of S(K) would be s[n] convolute with cos, while the imaginary part would be s[n] convolute with sin.

since cos and the s[n] are both even function, the real part is symmetrical.

However, the sin is odd function, so the imaginary part is not necessarily symmetrical. And the Abs part (real/imag) is not symmetric too.

Therefore, S(k) is not symmetric as a whole.

**Explain your results in terms of what you know about the Fourier Transform.**

1)DFT of uniform function:

X[0] =s(n)\*64=64

When k ><0, X[K]=s[n]\*(1-e^(-2jk\*pi/N)\*N)/(1-e(-2jk\*pi/N)))=0

So that is corresponds to the Real, Imag, Abs, Angle plot of uniform function.

2) DFT of delta function:

X[K]=s[0]\*1=1

So that is corresponds to the Real, Imag, Abs, Angle plot of uniform function.

3) sin& cos

It has been answered in Q2 and Q3

4) Symmetrical function

It has been answered in Q4

**3. Comparison with Matlab’s FFT function**

**Q1: Compare the results of your dft against the built-in fft. Are the result the same?**

Yes. Since the FFT is the Fast transform method of DFT, so they look the same in general.

**Q2: How long does it take to perform the operation.**

Firstly, I use *“tic; dft(ones(1,4)); toc”*, and I found the recorded time is not accurate since the time interval is too small.

Then, I repeat this for 10^4 times: “*tic; for (i=1:1e4) dft(ones(1,4)); end; toc”*

T(dft)=0.136175/10000=0.00001862

T(fft)=0.054053/10000=0.00000541

**Explain what this tells you about the DFT compared to the FFT in real applications**

From the loglog plot, we can know that when the n gets larger, the time taken for DFT appears to be n^2, while the time taken for FFT appears to be nlogn. So that tells us FFT can practically save time in calculation.

**4. Single Windowed Fourier Transform**

**Q1: Plot the magnitude of FFT, explain what this tells you about the waveform**

From the FFT magnitude plot of dbarrett.wav, we can determine that the frequency of this audio concentrate at 0.1kHZ, that correspond to frequency of the man’s voice.

And the smaller segments of frequency which lies around 0.1KHZ, reflects the leakage of FFT.

And from the FFT magnitude plot of the vmatthew2.wav, we can see the concentrated frequency is around 1KHZ, which is higher than that of dbarrett.wav. That correspond the female’s voice, which is accord with the common sense that the female’s voice has higher frequency compared to male’s.

**Q2: Explain the difference between these results.**

I tried various values of t and n.

N=64, T=500, 2048, 5096,9911

N=128, T=500, 2048, 5096,9911

N=256, T=500, 2048, 5096,9911

N=512, T=500, 2048, 5096,9911

And generally speaking, all the Hanning window diminish the frequency leakage of FFT.

N is the window width. We can see when n=512, the frequency plot gives the most accurate result. Because according to the uncertainty theorem, the frequency will become more accurate while time range become larger.

And the t reflects the window centered at different time. Since the audio’s frequency is almost the same all time (it can be seen at Task 5’s spectrogram), so the frequency plot did not change too much as t changes.

**5. STFT and Spectrogram**

**Q1: For fastest results on long files, use powers of 2, Why?**

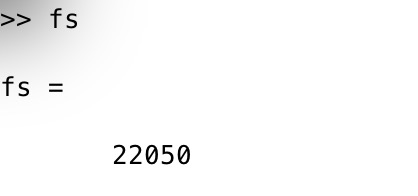
For the long files, we may want to use FFT to accelerate the calculation. Since we have to split the signal into half, and then 1/4, then 1/8 …… and finally split to 2 points, the number of points have to be the power of 2.

**Q2: Record what values of window size give best visualization results for different files.**

The best visualization exists when both the frequency and time are accurate.

‘piccolo.wav’, when w=512, we can get best visualization; For ‘dbarrett2.wav’, when w=256, we can get best visualization; For ‘vmatthew2.wav’, when w=256, we can get best visualization; For ‘bwv8272b.wav’, when w=512, we can get best visualization.

**Q3: Record the sampling frequency**



**Q4: estimate the fundamental frequencies**

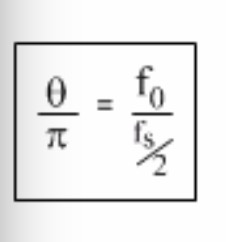
1. by calculating the frequency range displayed by specgram

From the “piccolo.wav NFFT=512 Normalized.bmp”, we can read that

F1(spec)=0.45 rad/sample

F2(spec)=0.109 rad/sample

F3(spec)=0.181 rad/sample

Since 

F(note1)=0.45\*fs/2=1600Hz,

F(note2)=0.109\*fs/2=1200Hz

F(note3)=0.181\*fs/2=2000Hz.

1. by supplying specgram with the correct value fs when you call it.

From the‘piccolo 512.bmp’，we can read that:  
F(note1)= 1700Hz,

F(note2)=0.109\*fs/2=1200Hz

F(note3)=0.181\*fs/2=2100Hz.