Assignment 8
The Digital Fourier Transform

March 19, 2019

You are all studying DSP this semester, and you know the following from this course and the signals course:

- A time function of the type $f(t)u_0(t)$ that grows slower than $e^{\alpha t}$ for some α has a Laplace transform, $F(s)$. The Laplace transform has an inverse transform which involves integrating vertically in the complex plane along a path that is to the right of all poles.
- A finite energy function of the type $f(t)$ has a Fourier Transform (and its inverse):

$$\begin{aligned} F(j\omega) &= \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt \\ f(t) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} F(j\omega)e^{j\omega t}d\omega \end{aligned}$$

- If $f(t)$ is periodic with period 2π, the Fourier Transform collapses to the Fourier Series

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{jnt}$$

where

$$c_n = \frac{1}{2\pi}\int_{t_0}^{t_0+2\pi} f(t)e^{-jnt}dt$$

for any t_0. We can take $t_0 = 0$ for convenience.

- We can invert this picture and say, suppose $f[n]$ are the samples of some function $f(t)$, then we define the Z transform as

$$F(z) = \sum_{n=-\infty}^{\infty} f[n]z^{-n}$$

Replacing z with $e^{j\theta}$ we get

$$F(e^{j\theta}) = \sum_{n=-\infty}^{\infty} f[n]e^{-jn\theta}$$

So clearly $F(z)$ is like the periodic time function that gives rise to the fourier series whose coefficients are the samples $f[n]$.

$F(e^{j\theta})$ **is called the Digital Spectrum of the samples** $f[n]$**. It is also called the DTFT of** $f[n]$

- $F(e^{j\theta})$ is continuous and periodic. $f[n]$ is discrete and aperiodic. Suppose now $f[n]$ is itself periodic with a period N, i.e.,

$$f[n+N] = f[n] \; \forall n$$

Then, it should have samples for its DTFT. This is true, and leads to the Discrete Fourier Transform or the DFT:

> Suppose $f[n]$ is a periodic sequence of samples, with a period N. Then the DTFT of the sequence is also a periodic sequence $F[k]$ with the same period N. (You will prove this in the DSP course). So we have
>
> $$\begin{aligned} F[k] &= \sum_{n=0}^{N-1} f[n] \exp\left(-2\pi \frac{nk}{N} j\right) = \sum_{n=0}^{N-1} f[n] W^{nk} \\ f[n] &= \frac{1}{N} \sum_{k=0}^{N-1} F[k] W^{-nk} \end{aligned}$$
>
> Here $W = \exp\left(-2\pi j/N\right)$ is used simply to make the equations less cluttered. The values $F[k]$ are what remains of the Digital Spectrum $F(e^{j\theta})$. We can consider them as the values of $F(e^{j\theta})$ for $\theta = 2\pi k/N$, since the first N terms in the expression for $F(e^{j2\pi k/N})$ yield
>
> $$F(e^{j2\pi k/N}) = \sum_{n=0}^{N-1} f[n] \exp\left(-\frac{2\pi kn}{N} j\right) + \ldots$$
>
> which is the same as the DFT expression. What about the remaining terms? They are repetitions of the above sum and help build up the delta function that is needed to take us from a continuous transform to discrete impulses.

- What this means is that the DFT is a sampled version of the DTFT, which is the digital version of the analog Fourier Transform

In this assignment, we want to explore how to obtain the DFT, and how to recover the analog Fourier Tranform for some known functions by the proper sampling of the function.

The DFT in Python

There are two commands in Python, one to compute the forward fourier transform and the other to compute the inverse transform. They are

```
numpy.fft.fft()
numpy.fft.ifft()
```

If you import pylab, both functions are imported into the local namespace. Let us try this on a random function:

```
from pylab import *
x=rand(100)
X=fft(x)
y=ifft(X)
c_[x,y]
print abs(x-y).max()
```

- When you run this code, the print statement will give you a value of about 10^{-15}. The starting vector, x, and the vector got by running fft and ifft on x are essentially the same vector.

- There is another thing to note. The command c_[x,y] tells Python to make a 100×2 matrix by using x and y as the columns. This is a quick and dirty way to print both out side by side. And what this shows is that x is pure real while y is very slightly complex. This is due to the finite accuracy of the CPU so that the ifft could not exactly undo what fft did.
- Note that I used a 100 point vector, but it is much better to use a number that is 2^k.

Let us now take a function we know about: y=sin(x). Let us use the fft and see what spectrum we get. We know that

$$y = \sin(x) = \frac{e^{jx} - e^{-jx}}{2j}$$

So the expected spectrum is

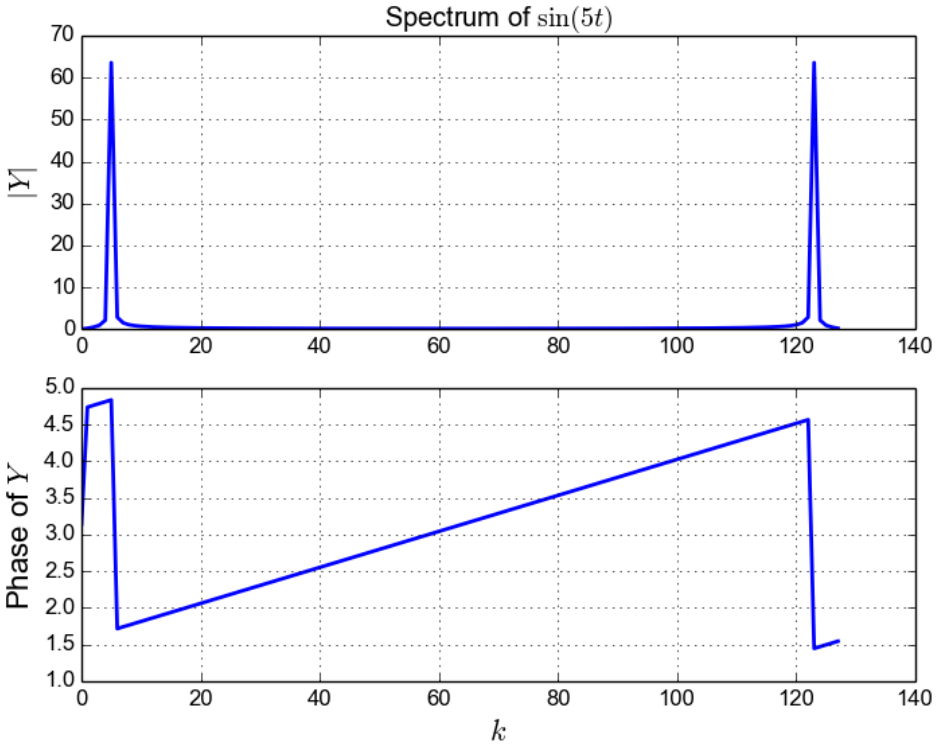
$$Y(\omega) = \frac{1}{2j}\left[\delta(\omega - 1) - \delta(\omega + 1)\right]$$

What do we get? We will use 128 points between 0 and 2π.

```
from pylab import *
x=linspace(0,2*pi,128)
y=sin(5*x)
Y=fft(y)
figure()
subplot(2,1,1)
plot(abs(Y),lw=2)
grid(True)
subplot(2,1,2)
plot(unwrap(angle(Y)),lw=2)
grid(True)
show()
```

3 ⟨*eg1* 3⟩≡

```
from pylab import *
x=linspace(0,2*pi,128)
y=sin(5*x)
Y=fft(y)
figure()
subplot(2,1,1)
plot(abs(Y),lw=2)
ylabel(r"$|Y|$",size=16)
title(r"Spectrum of $\sin(5t)$")
grid(True)
subplot(2,1,2)
plot(unwrap(angle(Y)),lw=2)
ylabel(r"Phase of $Y$",size=16)
xlabel(r"$k$",size=16)
grid(True)
savefig("fig9-1.png")
show()
```



- We get spikes as expected. But not where we expected. There is energy at nearby frequencies as well.
- The spikes have a height of 64, not 0.5. We should divide by N to use it as a spectrum.
- The phase at the spikes have a phase difference of π, which means they are opposite signs, which is correct
- The actual phase at the spikes is near but not exactly correct.
- We haven't yet got the frequency axis in place

What could be going wrong? The problem is that the DFT treats the position axis as another frequency axis. So it expects the vector to be on the unit circle starting at 1. Our position vector started at 0 and went to 2π, which is correct. The fft gave an answer in the same value. So we need to shift the π to 2π portion to the left as it represents negative frequency. This can be done with a command called *fftshift*.

The second thing that is wrong is that both 0 and 2π are the same point. So our 128 points need to stop at the poing just before 2π. The best way to do this is to create a vector of 129 values and drop the last one:

```
x=linspace(0,2*pi,129);x=x[:-1]
```

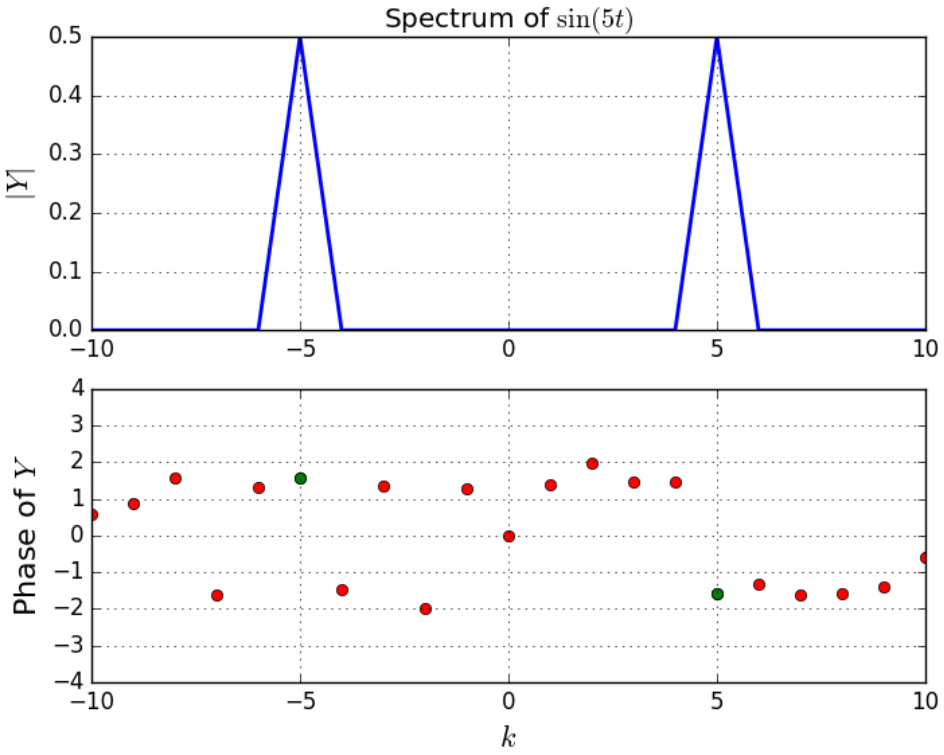
The ω axis has a highest frequency corresponding to N. This is because we have N points between 0 and 2π. So $\Delta x = 2\pi/N$, and the sampling frequency becomes $N/2\pi$. Thus $\omega_{max} = N$. The actual frequencies go upto half that frequency only.

4 ⟨*eg2* 4⟩≡

```
from pylab import *
x=linspace(0,2*pi,129);x=x[:-1]
```

```
y=sin(5*x)
Y=fftshift(fft(y))/128.0
w=linspace(-64,63,128)
figure()
subplot(2,1,1)
plot(w,abs(Y),lw=2)
xlim([-10,10])
ylabel(r"$|Y|$",size=16)
title(r"Spectrum of $\sin(5t)$")
grid(True)
subplot(2,1,2)
plot(w,angle(Y),'ro',lw=2)
ii=where(abs(Y)>1e-3)
plot(w[ii],angle(Y[ii]),'go',lw=2)
xlim([-10,10])
ylabel(r"Phase of $Y$",size=16)
xlabel(r"$k$",size=16)
grid(True)
savefig("fig9-2.png")
show()
```

The plot is now as follows:



- Things have improved! The peaks are exactly at 0.5, and the remaining values are zero to machine precision.
- Better, their position along the x axis is correct as our function is $\sin(5x)$ and we expect a spike at $\omega = \pm 5$
- Only the phase at the peak is meaningful and it is $\pi/2$ and $-\pi/2$ for the two peaks. The peak for $\omega = 5$ should be $0.5/j$, or $0.5\exp(-j\pi/2)$.

Suppose we now look at AM modulation. The function we want to analyse is

$$f(t) = (1+0.1\cos(t))\cos(10t)$$

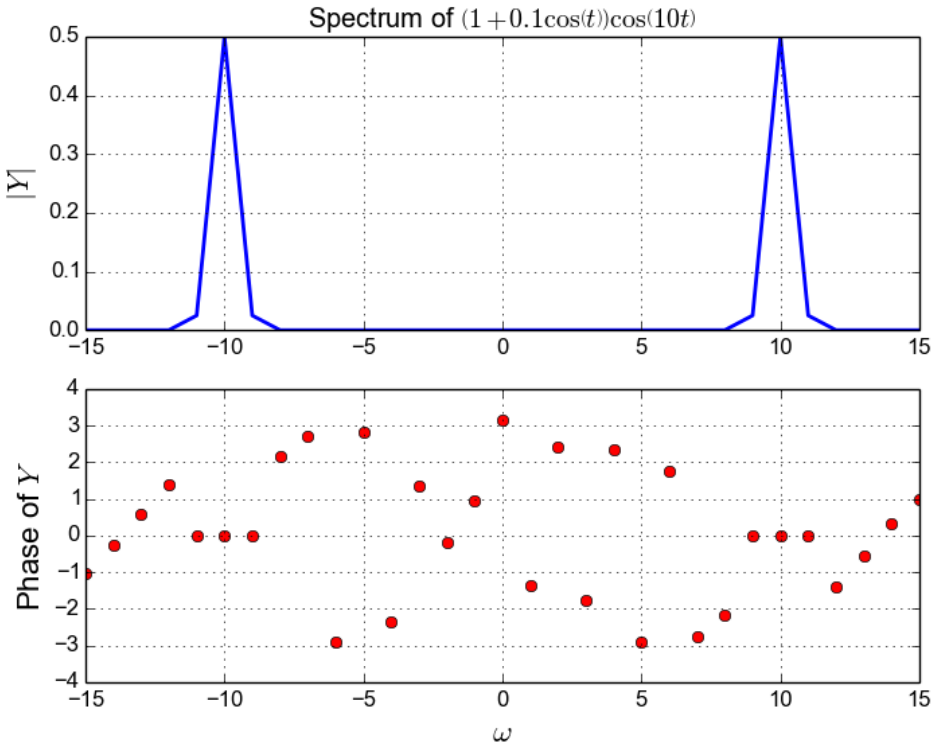
Again, we use 128 points and generate the spectrum using the above code, only changing the definition of $f(t)$.

6 ⟨*eg3* 6⟩≡

```
from pylab import *
t=linspace(0,2*pi,129);t=t[:-1]
y=(1+0.1*cos(t))*cos(10*t)
Y=fftshift(fft(y))/128.0
w=linspace(-64,63,128)
figure()
subplot(2,1,1)
plot(w,abs(Y),lw=2)
xlim([-15,15])
ylabel(r"$|Y|$",size=16)
title(r"Spectrum of $\left(1+0.1\cos\left(t\right)\right)\cos\left(10t\right)$")
grid(True)
```

```
subplot(2,1,2)
plot(w,angle(Y),'ro',lw=2)
xlim([-15,15])
ylabel(r"Phase of $Y$",size=16)
xlabel(r"$\omega$",size=16)
grid(True)
savefig("fig9-3.png")
show()
```

And we get ...



Not what we expected!

- There are three non-zero amplitudes on both sides, as expected.
- All have zero phase, since we have cosines.
- But not three spikes. Just a broader single spike

What went wrong? Well, we did not allow for enough frequencies. Let us use 512 samples instead. But how?

- If we put more points between 0 and 2π,

  ```
  t=linspace(0,2*pi,513);t=t[:-1]
  ```

  it will give the same spacing but a high sampling frequency.

- What we need to do is to stretch the t axis.

  ```
  t=linspace(-4*pi,4*pi,513);t=t[:-1]
  ```

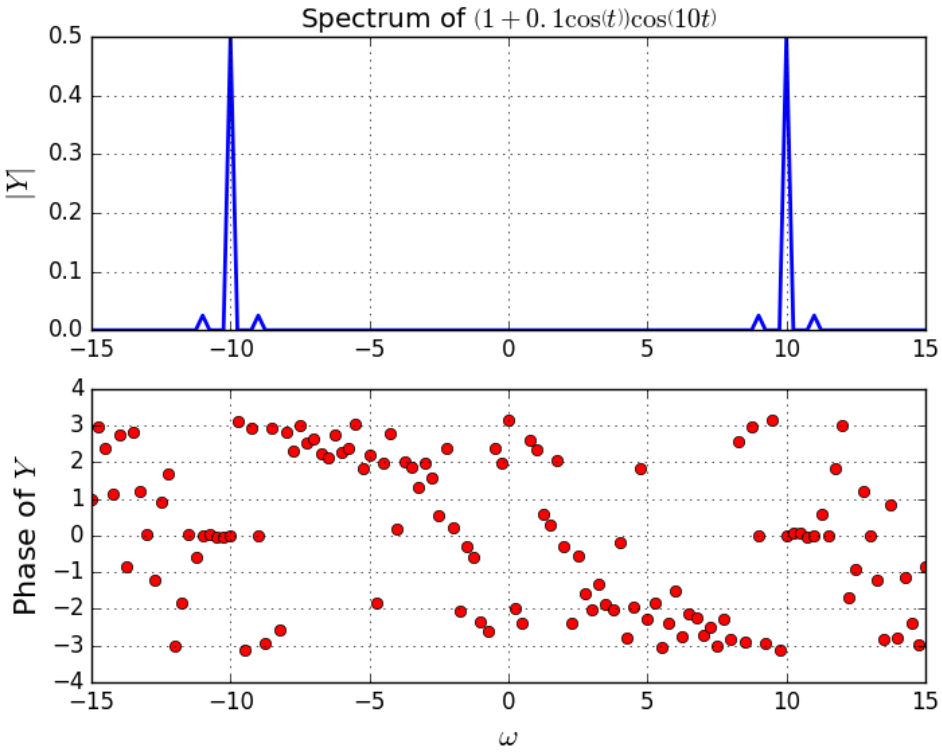
  This will give us tighter spacing between frequency samples, but the same time spacing - i.e., the same sampling frequency.

8 ⟨*eg4* 8⟩≡

```
from pylab import *
t=linspace(-4*pi,4*pi,513);t=t[:-1]
y=(1+0.1*cos(t))*cos(10*t)
Y=fftshift(fft(y))/512.0
w=linspace(-64,64,513);w=w[:-1]
```

```
figure()
subplot(2,1,1)
plot(w,abs(Y),lw=2)
xlim([-15,15])
ylabel(r"$|Y|$",size=16)
title(r"Spectrum of $\left(1+0.1\cos\left(t\right)\right)\cos\left(10t\right)$")
grid(True)
subplot(2,1,2)
plot(w,angle(Y),'ro',lw=2)
xlim([-15,15])
ylabel(r"Phase of $Y$",size=16)
xlabel(r"$\omega$",size=16)
grid(True)
savefig("fig9-4.png")
show()
```

Finally we are there:



Perfect! We have three clear spikes. The phases at those spikes are zero to machine precision. The location of the spikes are 9, 10 and 11 radians per sec. The height of the side bands come from

$$\begin{aligned} 0.1\cos(10t)\cos(t) &= 0.05\left(\cos 11t + \cos 9t\right) \\ &= 0.025\left(e^{11tj} + e^{9tj} + e^{-11tj} + e^{-9tj}\right) \end{aligned}$$

All the amplitudes are real and positive as seen in the phase plot.

Assignment

1. Work through the examples above.
2. Generate the spectrum of $\sin^3 t$ and $\cos^3 t$. Compare with what is expected.
3. Generate the spectrum of $\cos\left(20t + 5\cos(t)\right)$. Plot phase points only where the magnitude is significant (say greater than 10^{-3}). What do you think is happening?!
4. The Gaussian $\exp\left(-t^2/2\right)$ is not "bandlimited" in frequency. We want to get its spectrum accurate to 6 digits. Try different time ranges, and see what gets you a frequency domain that is so accurate. (The Fourier Transform of a Gaussian is a Gaussian in ω. Look up the transform pair to confirm)