Summary of Fourier Series / FFT Analysis

PRINT THIS AND BRING A COPY TO CLASS

From a practical standpoint, one can think of an FFT as a means of taking a sampled signal, $f(t)$, defined at a series of time points `ts`=[0, Δ,2Δ,3Δ,…(N-1)Δ][1] and estimating a set of complex amplitudes describing the response. If we define a vector containing the sampled time history as `fvec`=f(`ts`)=[$f(0)$, $f(\Delta)$,$f(2\Delta)$,…], then this can be done with the following Matlab command using the "fftp" function provided on the course website,

```
[F_DFT,ws]=fftp(fvec,ts);
```

where `F` is a vector giving the complex amplitude of each harmonic and `ws` is a vector defining the frequency for each complex amplitude. In doing this, we notice a few important things:

1. `length(F_DFT)=length(ws)`=N/2+1
2. `ws`=[0,ω_1, $2\omega_1$, $3\omega_1$, … , ω_{max}]

As explained in the book, the first makes sense because each complex amplitude has a real and imaginary part, so given N samples of a real time function, we can only expect to estimate N/2 complex quantities. In fact, we get N/2+1 quantities because the first and last end up being real. The second observation comes from the fact that the lowest frequency that we can estimate will have exactly one cycle per period, T, so:

$$\boxed{\omega_1 = \frac{2\pi}{T}} \qquad (3.7.1)$$

All of the other frequencies are integer multiples of ω_1, up to the highest frequency that can be extracted from the sampled signal, ω_{max}.

One of the most common uses of this is to estimate the steady-state response, $x(t)$ of an SDOF system with EOM: $m\ddot{x} + c\dot{x} + kx = f(t)$ by computing the response to each harmonic separately. This can be done with the following lines of Matlab code:

```
for k=1:length(ws);
      H(k)=1/(-m*ws(k)^2 + 1i*c*ws(k) + k);
      X_DFT (k)=H(k)*F_DFT (k);
end
[xvec,ts]=ifftp(X_DFT,ws);
```

A variety of problems can be solved simply knowing the procedure above, but to really understand how this works and its limitations, we must dive a little deeper into the theory, which is elaborated in the book. A few important points are summarized below.

1.) The complex amplitudes that we estimate using the FFT are not the complex amplitudes of a harmonic in the sense that we are used to. They follow the definition used in common software, so they are scaled differently.

---

[1] Note that we include the sample at t=0 but we omit the sample at t=T since $f(T)$=$f(0)$.

When we think of a complex amplitude, we typically imagine a signal being expressed as $f(t)=\mathrm{Re}(Fe^{i\omega t})$. A Fourier Series allows us to generalize this to decompose an arbitrary periodic signal, $Q(t)$, into a sum of harmonics,

$$\boxed{Q(t) = \frac{1}{2}F_0 + \mathrm{Re}\left[\sum_{n=1}^{\infty} F_n \exp(in\omega_1 t)\right]} \qquad (3.7.2)$$

or the following form is more convenient in some cases:

$$\boxed{Q(t) = \frac{1}{2}\sum_{n=-\infty}^{\infty} F_n \exp(in\omega_1 t),\ F_{-n} = F_n^*} \qquad (3.7.4)$$

If we know the time function, $Q(t)$, in closed form, we can compute the exact Fourier coefficients using the following.

$$\boxed{F_k = \frac{2}{T}\int_{-T/2}^{T/2} Q(t)\exp(-ik\omega_1 t)\,dt} \qquad (3.7.8)$$

The coefficients that we obtain from the DFT are the estimates of F_k that we would obtain if we replaced $Q(t)$ above with its sampled representation `fvec` and used the rectangle rule to evaluate the integral. The book calls these estimates $\hat{G}_n$ and explains that they are also scaled differently so,

$$F_n \cong \boxed{\hat{G}_n \approx \frac{2}{N}G_n} = (2/N)\text{*F\_DFT} \qquad (3.7.36)$$

At times, rather than use the FFT, we will want to instead use the Fourier Series to estimate the true Fourier coefficients and these equations give us all that we need to solve problems using this approach.

2.) If one uses Matlab's built in FFT function, `fft.m`, (or similarly for the standard functions in Mathcad, Mathematica, Python, etc…), one obtains a vector of complex amplitudes that is the same length as the time vector. Specifically,

```
[F_Matlab]=fft(fvec);
% We observe that: length(F_Matlab)=length(fvec)
```

The book explains why this is the case in detail. (FYI, the computer uses the following formula to compute the DFT, which you can also see if you type "help fft" in Matlab.)

$$\boxed{G_k = \sum_{n=0}^{N-1} g_n \exp\left(-2\pi i\frac{kn}{N}\right)} \qquad (3.7.18)$$

Hence, the book proves that some of the N coefficients are complex conjugates of the others. Specifically, if we have N=8,

Element #	1	2	3	4	5	6	7	8
Matlab DFT or FFT Coefficient	G_0	G_1	G_2	G_3	G_4	G_5	G_6	G_7
Frequency	0	ω_1	$2\omega_1$	$3\omega_1$	$4\omega_1$	$-3\omega_1$	$-2\omega_1$	$-\omega_1$

Then we find that G_0 and G_4 are both real numbers, and $G_1= G_7^*$, $G_2= G_6^*$, and $G_3= G_5^*$.

The built-in FFT function is applied to all kinds of signals, which may or may not be functions of time, and so they leave it to the user to create a frequency vector and to do any further operations. The "`fftp.m`" function created by the instructor simply calls Matlab's "`fft.m`" function, discards the complex conjugates, and creates the frequency vector.

Two other important issues are discussed in the text:

1.) **Aliasing** is a phenomenon in which high frequency components of a signal are lost when the signal is sampled. That is, it has components with frequencies $\omega > \omega_{\max}$.

2.) **Leakage** occurs when a signal contains frequency components that are not contained within the set of frequencies whose complex amplitudes are estimated by the FFT, `ws`=[$0,\omega_1, 2\omega_1, 3\omega_1, \ldots , \omega_{\max}$]. When this occurs, the signal is not truly periodic, and often one observes that the initial and final conditions are not equal, i.e. $f(T) \neq f(0)$. As a result, the FFT does the best it can to approximate the non-periodic signal with a periodic one.