Introduction to DFT

Deployment of Telecommunication Infrastructures

Azadeh Faridi
DTIC – UPF, Spring 2009
Review of Fourier Transform

- Many signals can be represented by a fourier integral of the following form:

\[ x(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} X(jw) e^{jwt} dw \]  \hspace{1cm} (1)

- Where \( X(jw) \) is given by

\[ X(jw) = \int_{-\infty}^{+\infty} x(t) e^{-jwt} dt \]  \hspace{1cm} (2)

\( (2) \) is the inverse Fourier Transform equation
\( (1) \) is the Fourier Transform equation

- Note that \( X(jw) \) is called the spectrum or the frequency domain representation of \( x(t) \)

- Notation: \( \mathcal{F}\{x(t)\} = X(jw) \) OR \( x(t) \leftarrow \mathcal{F.T.} \rightarrow X(j\omega) \)
LTI, Continuous Time Systems

- For an LTI (Linear Time-Invariant) continuous-time system, with the impulse response $h(t)$ as shown below

\[ x(t) \rightarrow h(t) \rightarrow y(t) \]

- $y(t)$ is given by

\[ y(t) = x(t) \ast h(t) = \int_{-\infty}^{+\infty} x(\tau)h(t - \tau)\,d\tau \]

  which is called the convolution of $x$ and $h$

- because of F.T. properties, in frequency domain we will have:

\[ Y(jw) = X(jw)H(jw) \]

  analysis in frequency domain can be a lot easier
## Fourier transform properties

<table>
<thead>
<tr>
<th>Signal</th>
<th>Fourier transform</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ax(t) + by(t)$</td>
<td>$aX(jw) + bY(jw)$</td>
</tr>
<tr>
<td>$x(t-\tau)$</td>
<td>$e^{-j\omega \tau} X(jw)$</td>
</tr>
<tr>
<td>$e^{j\varphi t}x(t)$</td>
<td>$X(j(w-\varphi))$</td>
</tr>
<tr>
<td>$x(at)$</td>
<td>$\frac{1}{a} X(\frac{jw}{a})$</td>
</tr>
<tr>
<td>$x(t) * y(t)$</td>
<td>$X(jw)Y(jw)$</td>
</tr>
<tr>
<td>$x(t)y(t)$</td>
<td>$\frac{1}{2\pi} X(jw) * Y(jw)$</td>
</tr>
<tr>
<td>$x(t)$ Real</td>
<td>$X(jw) = X^*(-jw)$</td>
</tr>
</tbody>
</table>
Fourier transform properties

<table>
<thead>
<tr>
<th>Signal</th>
<th>Fourier transform</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ax(t) + by(t)$</td>
<td>$aX(jw) + bY(jw)$</td>
</tr>
<tr>
<td>$x(t - \tau)$</td>
<td>$e^{-j\omega\tau} X(jw)$</td>
</tr>
<tr>
<td>$e^{j\varphi t}x(t)$</td>
<td>$X(j(\omega - \varphi))$</td>
</tr>
<tr>
<td>$x(at)$</td>
<td>$\frac{1}{a} X\left(\frac{j\omega}{a}\right)$</td>
</tr>
<tr>
<td>$x(t)\ast y(t)$</td>
<td>$X(jw)Y(jw)$</td>
</tr>
<tr>
<td>$x(t)y(t)$</td>
<td>$\frac{1}{2\pi} X(jw) \ast Y(jw)$</td>
</tr>
<tr>
<td>$x(t)$ Real</td>
<td>$X(jw) = X^*(-jw)$</td>
</tr>
</tbody>
</table>

$|X(jw)| = |X(-jw)|$

$\angle X(jw) = -\angle X(-jw)$
### Basic Fourier Transform Pairs

<table>
<thead>
<tr>
<th>Signal: ( x(t) )</th>
<th>Fourier transform: ( X(jw) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( e^{jw_0 t} )</td>
<td>( 2\pi \delta(w - w_0) )</td>
</tr>
<tr>
<td>( \cos w_0 t )</td>
<td>( \pi \left[ \delta(w - w_0) + \delta(w + w_0) \right] )</td>
</tr>
<tr>
<td>( \sin w_0 t )</td>
<td>( \frac{\pi}{j} \left[ \delta(w - w_0) - \delta(w + w_0) \right] )</td>
</tr>
<tr>
<td>( 1 )</td>
<td>( 2\pi \delta(w) )</td>
</tr>
<tr>
<td>( I(</td>
<td>t</td>
</tr>
<tr>
<td>( \frac{\sin Wt}{\pi t} )</td>
<td>( I(</td>
</tr>
<tr>
<td>( \delta(t - t_o) )</td>
<td>( e^{-jw t_o} )</td>
</tr>
</tbody>
</table>
Basic Fourier Transform Pairs

**Signal: \( x(t) \)**

- \( e^{j \omega_0 t} \)
- \( \cos \omega_0 t \)
- \( \sin \omega_0 t \)
- \( 1 \)
- \( I(|t| < T) \)
- \( \sin Wt \)
- \( \frac{\pi t}{\pi} \)
- \( \delta(t - t_0) \)

**Fourier transform: \( X(j\omega) \)**

- \( 2\pi \delta(\omega - \omega_0) \)
- \( \pi [\delta(\omega - \omega_0) + \delta(\omega + \omega_0)] \)
- \( \frac{\pi}{j} [\delta(\omega - \omega_0) - \delta(\omega + \omega_0)] \)
- \( 2\pi \delta(\omega) \)
- \( \frac{2 \sin \omega T}{\omega} \)
- \( I(|\omega| < W) \)
- \( e^{-j\omega t_0} \)
Calculating FT on computers

- We can't numerically represent c.t. signal $x(t)$ on computer (need infinite points)
  - ⇒ use sampling and convert to d.t. signal $x[n]$
  - ⇒ use Discrete Time FT (DTFT) on $x[n]$

- **DTFT:**

$$X(e^{jw}) = \sum_{n=-\infty}^{+\infty} x[n]e^{-jwn}$$

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{jw})e^{jwn} dw$$

- **NOTE:** $X(e^{jw})$ is periodic in frequency, with period $2\pi$
  - Low frequencies around $w = 0$
  - High frequencies around $w = \pi$
For an LTI (Linear Time-Invariant) discrete-time system, with the **impulse response** $h[n]$ as shown below

\[ y[n] = x[n] \ast [n] = \sum_{k=-\infty}^{+\infty} x[k] h[n - k] \]

which is called the **convolution** of $x$ and $h$

Similarly to c.t. case, in **frequency domain** we have:

\[ Y(e^{j\omega}) = X(e^{j\omega})H(e^{j\omega}) \]
Convolution Properties

- $x(t) \ast \delta(t - t_o) = x(t - t_o)$  (Similarly for DT systems)
  - What is $x(t) \ast \delta(t)$?

- **Continuous-time, finite duration signals:**
  - If
    $$\text{Duration}(x(t)) = T_x \quad \text{and} \quad \text{Duration}(y(t)) = T_y$$
  - then
    $$\Rightarrow \text{Duration}(x(t) \ast y(t)) = T_x + T_y$$

- **Discrete-time, finite duration signals:**
  - If
    $$\text{Duration}(x[n]) = N_x \quad \text{and} \quad \text{Duration}(y[n]) = N_y$$
  - then
    $$\Rightarrow \text{Duration}(x[n] \ast y[n]) = N_x + N_y - 1$$
DTFT on computer?

- $x[n]$ is representable on computer
- but $X(e^{jw})$ is still a continuous function of $w$
  - can't be represented on computer

⇒ Use DFT (Discrete Fourier Transform)
  - based on Fourier series
  - works on finite duration signals. Basically,
    - take a finite duration sequence (of length N),
    - make it periodic,
    - calculate the Fourier series → N values
  - generates a discrete frequency response
  - for N points need N Fourier samples
DFT

\[ x[n] \leftarrow \text{DFT} \rightarrow X[k] \]

- For a signal \( x[n] \) of duration \( N \)

\[
X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}
\]

\[
x[n] = \frac{1}{N} \sum_{n=0}^{N-1} X[k] W_N^{-kn}
\]

- Where \( W_N = e^{-j \frac{2\pi}{N}} \)

- DFT is basically a sampling of DTFT in frequency

\[
X[k] = \begin{cases} 
X(e^{jw}) |_{w=\frac{2\pi k}{N}}, & 0 \leq k \leq N - 1 \\
0, & \text{otherwise}
\end{cases}
\]

- It is derived using the Fourier series of the periodic version of \( x[n] \)
DFT in Octave/Matlab

- DFT can be calculated in Octave using the `fft` function
  - `fft(x,N0)`
  - `x` is the signal
  - `N0` is the number of points in the DFT
    - It works in the same way as appending zeros to your signal
    - larger `N0` will give nicer graphs
    - but `N0 = N` is enough to get correct inverse `fft`
    - **Be careful when choosing N0!!!**
  - `ifft` is the Octave function for calculating the inverse DFT
  - by default `fft` generates the spectrum in `[0,2*pi]`, but we are usually interested in `[-pi,pi]`. use `fftshift` for this!

- **Exercise:**
  - `x[n] = [1 1 1 1]`
  - calculate the `fft` of `x` with different values of `N0` and plot the `abs()` of the frequency responses on the same plot. Compare the results.
DFT Properties

- **Linearity**
  
  \[ x_3[n] = ax_1[n] + bx_2[n] \quad \longleftrightarrow \quad X_3[k] = aX_1[k] + bX_2[k] \]
  
  - N3 = max [N1, N2]

- **Circular shift**
  
  \[ x_1[n] = x[((n - m))_N] \quad \longleftrightarrow \quad X_1[k] = e^{-j\frac{2\pi km}{N}} X[k] \]
  
  - This means that \( |X_1[k]| = |X[k]| \)

- **Duality**
  
  \[ x[n] \quad \longleftrightarrow \quad X[k] \]
  
  \[ X[n] \quad \longleftrightarrow \quad Nx[(-k)_N] \]

Notation: \( ((k))_N = (k \mod N) \)
Circular convolution

\[ x_3[n] = x_1[n] \bigcirc N x_2[n] \iff X_3[k] = X_1[k]X_2[k] \]

- where

\[ x_1[n] \bigcirc N x_2[n] = \sum_{m=0}^{N-1} x_1[((m))_N]x_2[((n - m))_N] \]

- for \( n \in [0, N - 1] \)
- This is because a shift in the periodic version of \( x[n] \) is equivalent to a circular shift in \( x[n] \)

**Note:** \( \text{DFT}^{-1} \{ X[k]H[k] \} \neq x[n] * h[n] \) (linear convolution)
- to calculate linear convolution using fft and ifft you need to use a trick! (see next page)
Let

- \( x_1[n] \) be of length \( N_1 \)
- \( x_2[n] \) be of length \( N_2 \)

We can calculate the linear convolution of two signals by multiplying their frequency responses and then taking the inverse FT. To do this properly using fft function, you need to choose \( N_0 \) carefully, as follows:

- \( X_1 = \text{fft}(x_1, N_1+N_2-1) \)
- \( X_2 = \text{fft}(x_2, N_1+N_2-1) \)
- \( x_{1\_\text{conv}\_x2} = \text{ifft}(X_1.*X_2) \)

**Exercise:** let \( x_1 = x_2 = [1 1 1 1] \). calculate \( \text{conv}(x_1, x_2) \) directly and using the method described above. You should get the same result using either method.