Announcements

• HW1 solutions posted -- grading due tonight
• HW2 due Friday
• SDR give after GSI Wednesday
• Finish Ch. 8, start Ch. 9

Satellite

• Saudisat 1c has an FM repeater

Last Time

• Discrete Fourier Transform
  – Similar to DFS
  – Sampling of the DTFT (subtitles....more later)
  – Properties of the DFT

• Today
  – Linear convolution with DFT
  – Fast Fourier Transform
Properties of DFT

• Inherited from DFS (EE120/20) so no need to be proved

• Linearity
\[ \alpha_1 x_1[n] + \alpha_2 x_2[n] \leftrightarrow \alpha_1 X_1[k] + \alpha_2 X_2[k] \]

• Circular Time Shift
\[ x[((n-m))N] \leftrightarrow X[k]e^{-j(2\pi/N)km} = X[k]W_N^{km} \]

Circular shift

Properties of DFT

• Circular frequency shift
\[ x[n]e^{j(2\pi/N)nl} = x[n]W_N^{-nl} \leftrightarrow X[((k-l))N] \]

• Complex Conjugation
\[ x^*[n] \leftrightarrow X^*[((-k))N] \]

• Conjugate Symmetry for Real Signals
\[ x[n] = x^*[n] \leftrightarrow X[k] = X^*[((-k))N] \]

Show....

Properties of DFT

• Parseval’s Identity
\[ \sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X[k]|^2 \]

• Proof (in matrix notation)
\[ x^*x = \left( \frac{1}{N} W_N^*x \right)^* \left( \frac{1}{N} W_N^*x \right) = \frac{1}{N^2} x^* W_N W_N^* x = \frac{1}{N} x^*x \]
Circular Convolution Sum

• Circular Convolution:

\[ x_1[n] \ast_c x_2[n] \triangleq \sum_{m=0}^{N-1} x_1[m]x_2[(n - m) \mod N] \]

for two signals of length N

• Note: Circular convolution is commutative

\[ x_2[n] \ast_c x_1[n] = x_1[n] \ast_c x_2[n] \]

Properties of DFT

• Circular Convolution: Let \( x_1[n], x_2[n] \) be length N

\[ x_1[n] \ast x_2[n] \leftrightarrow X_1[k] \cdot X_2[k] \]

Very useful!!! (for linear convolutions with DFT)

• Multiplication: Let \( x_1[n], x_2[n] \) be length N

\[ x_1[n] \cdot x_2[n] \leftrightarrow \frac{1}{N} X_1[k] \ast_c X_2[k] \]

Linear Convolution

• Next....
  - Using DFT, circular convolution is easy
  - But, linear convolution is useful, not circular
  - So, show how to perform linear convolution with circular convolution
  - Used DFT to do linear convolution

Circular Convolution Sum

• Circular Convolution:

\[ x_1[n] \ast_c x_2[n] \triangleq \sum_{m=0}^{N-1} x_1[m]x_2[(n - m) \mod N] \]

for two signals of length N

• Note: Circular convolution is commutative

\[ x_2[n] \ast_c x_1[n] = x_1[n] \ast_c x_2[n] \]
Compute Circular Convolution Sum

\[ y[n] = x_1[n] \ast x_2[n] = ? \]

Circular ‘flip’ multiply and add
Here: \( y[0] \)

Equivalent periodic convolution over a period

\[ y[n] = x_1[n] \ast x_2[n] = ? \]
Properties of DFT

- **Circular Convolution:** Let $x_1[n]$, $x_2[n]$ be length $N$

$$x_1[n] \otimes x_2[n] \leftrightarrow X_1[k] \cdot X_2[k]$$

Very useful!!! (for linear convolutions with DFT)

- **Multiplication:** Let $x_1[n]$, $x_2[n]$ be length $N$

$$x_1[n] \cdot x_2[n] \leftrightarrow \frac{1}{N} X_1[k] \otimes X_2[k]$$

Linear Convolution using the DFT

We start with two nonperiodic sequences:

$$x[n], 0 \leq n \leq L - 1$$
$$h[n], 0 \leq n \leq P - 1$$

We can think of $x[n]$ as a signal, and $h[n]$ as a filter impulse response.

We want to compute the linear convolution:

$$y[n] = x[n] * h[n] = \sum_{m=0}^{L-1} x[m] * h[n-m] = \sum_{m=0}^{P-1} x[n-m]h[m]$$

$y[n] = x[n] * h[n]$ is nonzero only for $0 \leq n \leq L + P - 2$, and is of length $L + P - 1 = M$.

Linear Convolution

- **Next....**
  - Using DFT, circular convolution is easy
  - But, **linear** convolution is useful, not circular
  - So, show how to perform linear convolution with circular convolution
  - Used DFT to do linear convolution

Linear Convolution using the DFT

We will look at two approaches for computing $y[n]$:

1. **Direct Convolution**
   - Evaluate the convolution sum directly.
     - This requires $L \cdot P$ multiplications

2. **Using Circular Convolution**
(2) Using Circular Convolution

- Zero-pad $x[n]$ by $P - 1$ zeros:
  \[
  x_{zp}[n] = \begin{cases} 
  x[n] & 0 \leq n \leq L - 1 \\
  0 & L \leq n \leq L + P - 2 
  \end{cases}
  \]

- Zero-pad $h[n]$ by $L - 1$ zeros:
  \[
  h_{zp}[n] = \begin{cases} 
  h[n] & 0 \leq n \leq P - 1 \\
  0 & P \leq n \leq L + P - 2 
  \end{cases}
  \]

- Both zero-padded sequences $x_{zp}[n]$ and $h_{zp}[n]$ are of length $M = L + P - 1$

Both zero-padded sequences $x_{zp}[n]$ and $h_{zp}[n]$ are of length $M = L + P - 1$

We can compute the linear convolution $x[n] * h[n] = y[n]$ by computing circular convolution $x_{zp}[n] \otimes h_{zp}[n]$:

\[
y[n] = x[n] * h[n] = \begin{cases} 
  x_{zp}[n] \otimes h_{zp}[n] & 0 \leq n \leq M - 1 \\
  0 & \text{otherwise}
  \end{cases}
\]
In practice, the circular convolution is implemented using the DFT circular convolution property:

\[ x[n] * h[n] = x_{zp}[n] \otimes h_{zp}[n] = DFT^{-1}\{DFTx_{zp}[n] \cdot DFT\{h_{zp}[n]\}\} \]

for \(0 \leq n \leq M - 1, M = L + P - 1\).

- **Advantage:** This can be more efficient than direct linear convolution because the FFT and inverse FFT are \(O(M \cdot \log_2 M)\).
- **Drawback:** We must wait until we have all of the input data. This introduces a large delay which is incompatible with real-time applications like communications.

**Approach:** Break input into smaller blocks. Combine the results using 1. overlap and save or 2. overlap and add.
Overlap-Add Method

We decompose the input signal $x[n]$ into non-overlapping segments $x_r[n]$ of length $L$:

$$x_r[n] = \begin{cases} x[n] & rL \leq n \leq (r + 1)L - 1 \\ 0 & \text{otherwise} \end{cases}$$

The input signal is the sum of these input segments:

$$x[n] = \sum_{r=0}^{\infty} x_r[n]$$

The output signal is the sum of the output segments $x_r[n] * h[n]$:

$$y[n] = x[n] * h[n] = \sum_{r=0}^{\infty} x_r[n] * h[n]$$  \hspace{1cm} (1)

Each of the output segments $x_r[n] * h[n]$ is of length $N = L + P - 1$.

Example of overlap and add:

- $x_0[n]$ is of length $P=6$
- $x_1[n]$ is of length $P=6$
- $x_2[n]$ is of length $P=6$
- $y[n]$ is of length $P=38$

Finally, we just add up the output segments using (1) to obtain the output.

Impulse Response, Length 0

Overlap-Add, Sum of Output Segments

Overlap-Add, Sum of Input Segments

Overlap-Add, Output Segments, Length 0

Linear Convolution, Length 38

Input Signal, Length 33

Output Signal, Length 38

DFT-based circular convolution is usually more efficient:

- Zero-pad input segment $x_r[n]$ to obtain $x_{r,zp}[n]$, of length $N$.
- Zero-pad the impulse response $h[n]$ to obtain $h_{zp}[n]$, of length $N$ (this needs to be done only once).
- Compute each output segment using:

$$x_r[n] * h[n] = DFT^{-1}\{DFT\{x_{r,zp}[n]\} \cdot DFT\{h_{zp}[n]\}\}$$

Since output segment $x_r[n] * h[n]$ starts offset from its neighbor $x_{r-1}[n] * h[n]$ by $L$, neighboring output segments overlap at $P - 1$ points.

Finally, we just add up the output segments using (1) to obtain the output.
**Overlap-Save Method**

**Basic Idea**

We split the input signal $x[n]$ into overlapping segments $x_r[n]$ of length $L + P - 1$.

Perform a circular convolution of each input segment $x_r[n]$ with the impulse response $h[n]$, which is of length $P$ using the DFT.

Identify the $L$-sample portion of each circular convolution that corresponds to a linear convolution, and save it.

This is illustrated below where we have a block of $L$ samples circularly convolved with a $P$ sample filter.

---

**Recall:**

- **Valid linear convolution!**

---

**DFT vs DTFT (revisit)**

- **Back to moving average example:**

$$X(e^{j\omega}) = \sum_{n=0}^{4} e^{-j\omega n}$$

$$= e^{-j2\omega} \frac{\sin(\frac{5}{2} \omega)}{\sin(\frac{\omega}{2})}$$
DFT and Sampling the DTFT

\[ X(e^{j\omega}) = e^{-j\omega \sin^2(5\omega/2)/\sin^2(\omega/2)} \]

Circular Convolution as Matrix Operation

Circular convolution:

\[ h[n] \otimes x[n] = \begin{bmatrix} h[0] & h[N - 1] & \cdots & h[1] \\ h[1] & h[0] & \cdots & h[2] \\ \vdots & \vdots & \ddots & \vdots \\ h[N - 1] & h[N - 2] & \cdots & h[0] \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N - 1] \end{bmatrix} = H_c x \]

- \( H_c \) is a circulant matrix
- The columns of the DFT matrix are Eigen vectors of circulant matrices.
- Eigen vectors are DFT coefficients. How can you show?

Circular Convolution as Matrix Operation

- Diagonalize:
  \[ W_N H_c W_N^{-1} = \begin{bmatrix} H[0] & 0 & \cdots & 0 \\ 0 & H[1] & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & H[N - 1] \end{bmatrix} \]

- Right-multiply by \( W_N \)
  \[ W_N H_c = \begin{bmatrix} H[0] & 0 & \cdots & 0 \\ 0 & H[1] & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & H[N - 1] \end{bmatrix} W_N \]

- Multiply both sides by \( x \)
  \[ W_N H_c x = \begin{bmatrix} H[0] & 0 & \cdots & 0 \\ 0 & H[1] & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & H[N - 1] \end{bmatrix} W_N x \]