ECE 421
Introduction to Signal Processing

Dror Baron
Assistant Professor
Dept. of Electrical and Computer Engr.
North Carolina State University, NC, USA
Discrete Fourier Transform
Roadmap

- We have seen
  - Chapter 1 – from analog to digital and back
  - Chapter 2 – discrete time signals & systems; correlation
  - Chapter 3 – z-transforms; transfer functions; one-sided z
  - Chapter 4 – Fourier transforms
  - Chapter 5 – frequency domain analysis of LTI systems
  - Chapter 6 – sampling, reconstruction, end-to-end systems, ...

- About to discuss Chapter 7
  - Frequency sampling of discrete time signals $\rightarrow$ discrete Fourier transform (DFT)
  - Implementing linear filters with DFT
  - Application to frequency analysis
Frequency Sampling of Discrete Time Signals

[Reading material: Section 7.1]
Challenges for discrete time signals

- Fourier representations of aperiodic and periodic discrete time signals are helpful

- BUT...

- Aperiodic?
  - Impractical to know $X(\omega)$ for infinitely many $\omega \in [-\pi, \pi]$

- Periodic?
  - Most signals “in the wild” aren’t periodic

- *Need tools for finite duration signals*
What tools for finite duration signals?

- Finite duration signals motivate discrete Fourier transform (DFT)

- Another motivation/perspective is frequency sampling
Frequency Sampling
What are we sampling?

- Instead of aperiodic \( x(n) \), consider *periodic repetition*:
  \[
  x_p(n) = \sum_{l=-\infty}^{+\infty} x(n - lN)
  \]

- Notes:
  1) \( x_p(n) = \sum_{l=-\infty}^{+\infty} x(n + lN) \)
  2) It’s periodic-N
Fourier series for $x_p(n)$

- Periodic repetition $x_p(n)$ periodic $\Rightarrow$ can take its Fourier series

- $x_p(n) = \sum_{k=0}^{N-1} C_k e^{j2\pi kn/N}$

- $C_k = \frac{1}{N} \sum_{k=0}^{N-1} x_p(n) e^{-j2\pi kn/N}$

- Will soon see relation between $X(2\pi k/N)$ and $C_k$
Sampling $X(\omega)$

- Recall $X(\omega) = \sum_{n=-\infty}^{+\infty} x(n)e^{-j\omega n}$
- Let's take $N$ samples of $X(\omega)$
- It's periodic $\rightarrow$ focus on range $\omega \in [0, 2\pi) \rightarrow \omega = 2\pi k/N$

- $X \left(\frac{2\pi k}{N}\right) = \sum_{n=-\infty}^{+\infty} x(n)e^{-j\left(\frac{2\pi k}{N}\right)n}$

  \[= \cdots + \sum_{n=-N}^{-1} x(n)e^{-j\left(\frac{2\pi k}{N}\right)n} + \sum_{n=0}^{N-1} x(n)e^{-j\left(\frac{2\pi k}{N}\right)n} + \cdots\]

  \[= \sum_{l=-\infty}^{+\infty} \sum_{n=lN}^{lN+N-1} x(n)e^{-j\frac{2\pi k n}{N}}\]

  \[= \sum_{i=0}^{N-1} \sum_{l=-\infty}^{+\infty} x(lN + i)e^{-j\frac{2\pi k (lN+i)}{N}} \quad n=lN+i\]
A useful observation

- Observe that \( e^{-j\frac{2\pi k}{N}(lN+i)} = e^{-\frac{j2\pi k}{N}i} e^{-\frac{j2\pi k}{N}lN} \)

- \( e^{-\frac{j2\pi k}{N}lN} = 1 \rightarrow e^{-\frac{j2\pi k}{N}(lN+i)} = e^{-\frac{j2\pi k}{N}i} \)

- Back to derivation...

- \( X\left(\frac{2\pi k}{N}\right) = \cdots = \sum_{i=0}^{N-1} \sum_{l=-\infty}^{+\infty} x(lN + i) e^{-\frac{j2\pi k}{N}(lN+i)} \)

\[
= \sum_{i=0}^{N-1} \sum_{l=-\infty}^{+\infty} x(lN + i) e^{-\frac{j2\pi ki}{N}}
\]

\[
= \sum_{i=0}^{N-1} \left[ \sum_{l=-\infty}^{+\infty} x(lN + i) \right] e^{-\frac{j2\pi ki}{N}}
\]

\[
= \sum_{i=0}^{N-1} x_p(i) e^{-\frac{j2\pi ki}{N}}
\]

\[
= NC_k \rightarrow \text{sampling } X(\omega) \text{ resembles Fourier series of } x_p(n)
\]
Discussion

- **Now**: samples of $X(\omega)$ related to Fourier series of periodic extension $x_p(n)$

- **Before**: sampling $x_a(t)$ created copies of spectrum $X(F)$

- Could copies of spectrum be analogous to periodic extension?

- **Before**: aliasing when signal not band limited enough

- **Now**: Time-limited $x(n) \rightarrow$ one to one correspondence w/ $x_p(n)$
  - Else $x(n)$ will be “aliased” in $x_p(n)$
Example 7.1.1

- Consider $x(n)=a^nu(n)$, $|a|<1$

- Fourier transform, $X(\omega) = \frac{1}{1-ae^{-j\omega}}$

- Take $N$ samples, $X(2\pi k/N) = \frac{1}{1-ae^{-j2\pi k/N}}$

- Can compute $x_p(n)$ from $C_k=\frac{1}{N}X(2\pi k/N)$

- Is $x(n)$ identical to $x_p(n)$?
  - Will show $x_p(n) = \cdots = \sum_{l=-\infty}^{0} a^{n-lN} = \cdots = \frac{a^n}{1-a^N}$

- Large $N \rightarrow a^N<<1 \rightarrow$ small “aliasing”
Frequency Interpolation?
Motivation

- We’ve sampled $X(\omega)$, which was related to $C_k$
- Can we get back $X(\omega)$?

- *Can we interpolate between $X(2\pi k/N)$ to get back $X(\omega)$?*
Interpolation

- Suppose no “aliasing” → $x(n) = 0$ except for $n \in \{0, \ldots, N-1\}$
- Fourier series $x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X\left(\frac{2\pi k}{N}\right)e^{j2\pi kn/N}$

Rationale: $\frac{1}{N} X\left(\frac{2\pi k}{N}\right) = C_k$ and $C_k$ are coeffs of Fourier series for $x_p(n)$ → can compute $x(n)$ from $X\left(\frac{2\pi k}{N}\right)$
Interpolation

- Suppose no “aliasing” → $x(n)=0$ except for $n \in \{0,\ldots,N-1\}$
- Fourier series $x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X\left(\frac{2\pi k}{N}\right)e^{j2\pi kn/N}$

- Let’s substitute Fourier series into computation of $X(\omega)$
- $X(\omega) = \sum_{n=-\infty}^{+\infty} x(n)e^{-j\omega n}$
  $= \sum_{n=0}^{N-1} x(n)e^{-j\omega n}$
  $= \sum_{n=0}^{N-1} \left[ \frac{1}{N} \sum_{k=0}^{N-1} X\left(\frac{2\pi k}{N}\right)e^{j2\pi kn/N} \right] e^{-j\omega n}$
  $= \sum_{k=0}^{N-1} X\left(\frac{2\pi k}{N}\right) \left[ \frac{1}{N} \sum_{n=0}^{N-1} e^{j2\pi kn/N} - j\omega n \right]$}
  $= \sum_{k=0}^{N-1} X\left(\frac{2\pi k}{N}\right) \left[ \frac{1}{N} \sum_{n=0}^{N-1} e^{-j(\omega-2\pi k/N)n} \right]$
Let’s compute $P(\omega)$

- $P(\omega) = \frac{1}{N} \sum_{n=0}^{N-1} e^{-j\omega n}$

\[
= \frac{1}{N} \frac{1-e^{-j\omega N}}{1-e^{-j\omega}}
= \frac{1}{N} \frac{e^{-j\omega N/2}(e^{j\omega N/2}-e^{-j\omega N/2})}{e^{-j\omega/2}(e^{j\omega/2}-e^{-j\omega/2})}
= \frac{1}{N} \frac{\sin(\omega N/2)}{\sin(\omega/2)} e^{-j\omega(N-1)/2}
\]

- Substituting into $X(\omega)$ expression:

\[
X(\omega) = \sum_{k=0}^{N-1} X \left(\frac{2\pi k}{N}\right) P(\omega - 2\pi k/N)
\]

- Resembles convolution with LPF but it’s Dirichlet kernel (resembles sinc + artifacts due to periodic $X(\omega)$)
The Discrete Fourier Transform
Finite duration perspective

- Recall two perspectives
  - Finite duration signals motivate discrete Fourier transform (DFT)
  - Frequency sampling also motivates DFT

- We’ve sampled the Fourier transform, let’s consider $x(n)$ with finite duration $L \leq N$

- $x_p(n) = \begin{cases} x(n), & 0 \leq n \leq L - 1 \\ 0, & L \leq n \leq N - 1 \end{cases}$

- For $L < N$ must perform zero padding (add $N - L$ zeros)
  - Allows to analyze length-$L$ signal with as many freqs as we want
  - For example, $N \gg L$ helps make smooth-looking plot
The DFT

- Recall the Fourier transform,
  \[ X(\omega) = \sum_{n=0}^{L-1} x(n) e^{-j\omega n} \]

- Take N frequency samples at \( \omega_k = \frac{2\pi k}{N}, k \in \{0, \ldots, N - 1\} \)

- We have \( X(k) = X\left(\frac{2\pi k}{N}\right) = \sum_{n=0}^{L-1} x(n) e^{-j2\pi kn/N} \)

- Because \( x(n)=0 \) for \( n \in \{L, L+1, \ldots, N-1\} \), can simplify:
  \[ X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N} \]

- And the inverse DFT (IDFT):
  \[ x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{+j2\pi kn/N} \]
Active learning (example 7.1.2)

- Consider finite duration signal, $x(n) = \begin{cases} 1, & 0 \leq n \leq L - 1 \\ 0, & L \leq n \leq N - 1 \end{cases}$

- Fill in gaps while computing Fourier transform
  
  $$X(\omega) = \sum_{n=-\infty}^{+\infty} x(n)e^{-j\omega n}$$

  $$= \sum_{n=??} ??? x(n)e^{-j\omega n}$$

  $$= \sum_{n=??} ??? 1e^{-j\omega n}$$

  $$= \frac{1-??}{1-??}$$

  what are the ranges?

  geometric series

  Can be expressed $X(\omega) = \frac{\sin(\omega L/2)}{\sin(\omega/2)} e^{-j\omega(L-1)/2}$
DFT as a Transform

[Reading material: Sections 7.1.3, 7.1.4, & 7.2.2]
Recall
- DFT: \( X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N} \)
- IDFT: \( x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{+j2\pi kn/N} \)

Define auxiliary variable, \( w_N = e^{-j2\pi/N} \)
- DFT: \( X(k) = \sum_{n=0}^{N-1} x(n) w_N^{+kn} \)
- IDFT: \( x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) w_N^{-kn} \)

DFT matrix form:
\[
X_N = \begin{bmatrix}
X(0) \\
\vdots \\
X(N-1)
\end{bmatrix}
= \begin{bmatrix}
w_N^{0\cdot0} & \cdots & w_N^{0\cdot(N-1)} \\
\vdots & \ddots & \vdots \\
w_N^{(N-1)\cdot0} & \cdots & w_N^{(N-1)\cdot(N-1)}
\end{bmatrix}
\begin{bmatrix}
x(0) \\
\vdots \\
x(N-1)
\end{bmatrix} = W_N x_N
\]
IDFT

- Linear algebra: $x_N = (W_N)^{-1}X_N$
- What is the inverse matrix?

- IDFT matrix form:

\[
x_N = \begin{bmatrix}
x(0) \\
\vdots \\
x(N-1)
\end{bmatrix} = \frac{1}{N} \begin{bmatrix}
w_N^{-0,0} & \cdots & w_N^{-0,(N-1)} \\
\vdots & \ddots & \vdots \\
w_N^{-(N-1),0} & \cdots & w_N^{-(N-1),(N-1)}
\end{bmatrix} \begin{bmatrix}
X(0) \\
\vdots \\
X(N-1)
\end{bmatrix}
\]

- Can see $(W_N)^{-1} = \frac{1}{N}(W_N)^*$
Active learning (example 7.1.3)

- Signal $x=[0 \ 1 \ 2 \ 3]^T$
  - $T$ for transpose (make vertical by flipping)

1) What is $w_4$? (Recall $w_N=e^{-j2\pi/N}$)

2) Fill in powers of $w_4$

$$W_4 = \begin{bmatrix}
w_4 & w_4 & w_4 & w_4 \\
w_4 & w_4 & w_4 & w_4 \\
w_4 & w_4 & w_4 & w_4 \\
w_4 & w_4 & w_4 & w_4
\end{bmatrix}$$
Active learning continued

- For $N=2$ can show $w_2=-1$ and $W_2 = \begin{bmatrix} w_2^0 & w_2^0 \\ w_2^0 & w_2^1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

3) What is $W_4$?

4) Compute $X = W_4 \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}$; will verify result w/Matlab
Relation to Fourier of periodic

- DFT closely related to some Fourier transforms we’ve seen

- Recall that periodic sequence $x_p(n)$ can be expressed as Fourier series, $x_p(n) = \sum_{k=0}^{N-1} C_k e^{j2\pi kn/N}$

- Saw before $X(k) = NC_k$
Relation to Fourier of *aperiodic*

- For aperiodic $x(n)$, can take samples of $X(\omega)$ at $\omega = 2\pi k/N$
- These are DFT coeffs of periodic extension

\[ x_p(n) = \sum_{l=-\infty}^{+\infty} x(n - lN) \]
More relations

- Can relate DFT to z transform using $z = e^{j2\pi k/N}$
  - Details in book

- Relation to periodic continuous time signal
  - Suppose signal is band limited
  - $N$ samples capture all information about $x_a(t)$
  - $N$ series coeffs capture all information
  - These coeffs related to DFT coeffs
Circular Convolution
DFT properties

- DFT has many useful properties
  - Similar to what we’ve seen for Fourier & z
  - Details in book

- Does convolution also work similarly?

- What’s $x_3(n)$ that satisfies $X_3(k) = X_1(k) \times X_2(k)$?
  - Need to re-work convolution
Circular convolution derivation

- Let’s derive $x_3$ directly

$$x_3(m) = \frac{1}{N} \sum_{k=0}^{N-1} X_3(k)e^{j2\pi km/N}$$

$$= \frac{1}{N} \sum_{k=0}^{N-1} X_1(k)X_2(k)e^{j2\pi km/N}$$

$$= \frac{1}{N} \sum_{k=0}^{N-1} \left[ \sum_{n=0}^{N-1} x_1(n) e^{-j2\pi kn/N} \right] \left[ \sum_{l=0}^{N-1} x_2(l) e^{-j2\pi kl/N} \right] e^{j2\pi km/N}$$

$$= \frac{1}{N} \sum_{n=0}^{N-1} x_1(n) \sum_{l=0}^{N-1} x_2(l) \left[ \sum_{k=0}^{N-1} e^{j2\pi k(-n-l+m)/N} \right]$$

- Can utilize structure of inner summation
Inner summation

\[ \sum_{k=0}^{N-1} e^{j2\pi k (-n-l+m)/N} = \begin{cases} N, & \text{mod}(m-n-l, N) = 0 \\ 0, & \text{else} \end{cases} \]

Modulo examples: \( \text{mod}(3,2)=1 \), \( \text{mod}(-1,2)=1 \)

Let’s return to derivation

\[ x_3(m) = \frac{1}{N} \sum_{n=0}^{N-1} x_1(n) \sum_{l=0}^{N-1} x_2(l) \left[ \sum_{k=0}^{N-1} e^{j2\pi k (-n-l+m)/N} \right] \]

\[ = \frac{1}{N} \sum_{n=0}^{N-1} x_1(n) \sum_{l=0}^{N-1} x_2(l) \begin{cases} N, & \text{mod}(m-n-l, N) = 0 \\ 0, & \text{else} \end{cases} \]

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

Note: \( l=\text{mod}(m-n,N) \) implies \( \text{mod}(m-n-l,N)=0 \)
Example 7.2.1

- $x_1(n) = \{2, 1, 2, 1\}$, $x_2(n) = \{1, 2, 3, 4\}$
- Want to compute circular convolution

- $x_1$:
  - 2 1 2 1
- $\text{flip}(x_2)$:
  - 1 4 3 2
- $x_1 \text{flip}(x_2)$:
  - 2 4 6 2
- $x_3(0) = \sum(x_1 \text{flip}(x_2)) = 2+4+6+2 = 14$
- $\text{shift}(\text{flip}(x_2))$:
  - 2 1 4 3
- $x_1 \text{shift}(\text{flip}(x_2))$:
  - 4 1 8 3
- $x_3(1) = \sum(x_1 \text{shift}(\text{flip}(x_2))) = 4+1+8+3 = 16$

- And so on...
Example 7.2.1 continued

- Can be performed in Matlab:
  - x1=[2 1 2 1];
  - x2=[1 2 3 4];
  - x1f=fft(x1);
  - x2f=fft(x2);
  - x3f=x1f.*x2f;
  - x3=ifft(x3f);
More Properties of DFT

[Reading material: Section 7.2]
Time reversal property

- Time reversal: \( x((-n))_N = x(N-n) \leftrightarrow X((-k))_N = X(N-k) \)
  - Subscript denotes modulo-N

- Example:
  - \( x(n) = \{0, 1, 2, 3\} \)
  - \( x((-n))_N = \{0, 3, 2, 1\} \)
  - \( X(k) = \{6, -2+2j, -2, -2-2j\} \)
  - DFT of flipped version?

Matlab:

- \( x = [0 \ 1 \ 2 \ 3]; \)
- \( x_r = [0 \ 3 \ 2 \ 1]; \)
- \( \text{fft}(x) \)
- \( \text{fft}(x_r) \)
Circular time shift property

- Circular time shift: \( x(n-l)_N \leftrightarrow X(k)e^{-j2\pi kl/N} \)
- \( x_1 = \{0, 1, 2, 3\} \)
- \( x_2 = \{3, 0, 1, 2\} \)
- \( \text{DFT}\{x_2\} = \{6, 2+2j, 2, 2-2j\} \)
- Can verify in Matlab: \( \text{fft}(x_1) .* \exp(-j*2*pi/N*[0:3]) \)
More

- Circular frequency shift: $x(n)e^{j2\pi ln/N} \leftrightarrow X((k-l))_N$

- Complex conjugate: $x^*(n) \leftrightarrow X^*((-k))_N = X^*(N-k)$

- Time domain product: $x_1(n)x_2(n) \leftrightarrow \frac{1}{N} X_1(k) \odot X_2(k)$
  - Circular convolution denoted by $\odot$

- Parseval: $\sum_{n=0}^{N-1} x(n)y^*(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)Y^*(k)$
Linear Filters Using DFT

[Reading material: Section 7.3]
Linear vs. circular convolution

- We’ve seen how to compute circular convolution using DFT
  - Fast Fourier transform (FFT) – fast algorithm for computing DFT
  - Circular convolution using FFT is quite efficient

- Linear convolution appears in many applications
  \[
y(n) = \sum_{k=-\infty}^{+\infty} h(n - k)x(k)
\]

- Applications include digital filters in communication / control systems (e.g., audio equalization)

- Will compute linear convolution using circular convolution
**Why is this useful?**

- The convolution equation for linear filtering seems simple
  - It’s probably simple to implement and fast
  - Why care?

- Consider multipath
  - Suppose we sample at 100M samples/sec
  - Path lengths differ by 1 kM
  - Speed of light 300,000 kM/s
  - Time difference $1/300,000$ sec $\rightarrow$ 3.33 usec $\rightarrow$ 333 samples
  - May need filter w/1,000 taps $\rightarrow$ thousands of computations per sample

- Using FFT $(N\log(N))$ operations
- $N=1,000$ $\rightarrow$ $\log(N) \ll N$ $\rightarrow$ much more efficient
Example

- Length-L input $x(n)$, length-M filter $h(n)$
- Output of length $L+M-1$

- Example: $x=\{1,2,3\}$, $h=\{1,1\}$
  - $y=\{1,2,3\}+\{0,1,2,3\}=\{1,3,5,3\}$ (length 4)
  - Have verified $4=3+2-1$

- Can we compute this convolution using DFT?
Example part 2

- Let’s zero pad x & h to length L+M-1, compute DFT’s, multiply, convert back

- x={1,2,3}, L=3
  - x=[1 2 3]; L=length(x);
- h={1,1}, M=2
  - h=[1 1]; M=length(h);
- N=L+M-1=4
  - N=L+M-1;
- x={1,2,3,0}
  - xtilde=[x zeros(1,N-L)];
- X(k)={6,-2-2j,2,-2+2j}
  - xtildef=fft(xtilde);
- h={1,1,0,0}
  - htilde=[h zeros(1,N-M)];
- h(k)={2,1-j,0,1+j}
  - htildef=fft(htilde);
- Y(k)=X(k)H(k)={12,-4,0,-4}
  - yf=htildef.*xtildef;
- y(n)=IFFT{Y(k)}={1,3,5,3}
  - y=ifft(yf);

- Script on course webpage
Example done incorrectly

- Without zero padding, things won’t work
- \( x=[1\ 2\ 3]; \)
- \( h=[1\ 1\ 0]; \% \text{same length} \)
- \( \text{xf}=	ext{fft}(x); \% \{6,-1.5+\sqrt{3}/2j,-1.5-\sqrt{3}/2j\} \)
- \( \text{hf}=	ext{fft}(h); \% \{2,0.5-\sqrt{3}/2j,0.5+\sqrt{3}/2j\} \)
- \( y=	ext{ifft}(\text{xf}.*\text{hf}); \% \{4,3,5\} \)

\( \text{How did } \{1,3,5,3\} \text{ become } \{4,3,5\}? \)

- The last “3” in \{1,3,5,3\} got folded into “1” at time n=0
  - \( \text{mod}(3,3)=0 \)
  - This resembles aliasing
Filtering Long Sequences
Challenge

- Reconsider multipath problem
- Input x contains 100M samples per sec
- Impractical to do this with 100M
  - Need to sample x
  - Store x
  - Compute FFT
  - Multiply with H(k)
  - IFFT...
  - And 1 second delay(!)

**Plan B**: partition x into blocks, compute stuff block by block, then put together (maybe patch between blocks)
Overlap save method

- Finite impulse response (FIR) filter $h$ of length $M$
- Partition input $x$ into blocks of length $L>>M$
  - Data block $X_m =$ last $M-1$ samples from previous block & next $L$ samples
  - Entire data block length $= L + M - 1 = N$
- Compute DFT of length-$N$ data block
- $Y_m(k) = H(k)X_m(k)$
- IDFT gives us $\{y_m(0), y_m(1), ..., y_m(N-1)\}$
- First $M-1$ points of $y_m$ corrupted by “aliasing”
- Last $L$ points not corrupted $\rightarrow$ we output them

*Matlab script on course webpage*
Overlap add method

- Prevent “aliasing” in each block by zero padding
- End of current output block must be *added* to beginning of next output block
- Details in book
Frequency Analysis Using DFT

[Reading material: Sections 7.4-7.5]
Why frequency analysis?

- Many signals are spectrally sparse

- Applications include:
  - Radio signals (AM/FM) - want to identify carrier frequency
  - Some include pure sinusoid to aide synchronization

- We don’t know the spectral occupancy → will estimate it
Simple model

- Analog signal $x_a(t)$ sampled
- Have finite number of samples $x(n)$

- To keep simple: *two sinusoids*
  - $x(n) = a_1 \cos(\omega_1 n + \Phi_1) + a_2 \cos(\omega_2 n + \Phi_2)$

- $X(\omega)$ contains two deltas

- More realistic signals contain multiple sinusoids, measurement noise, ... → tougher to estimate spectral content
Spectrum of finite duration samples

- We have finite duration $x(n)$
- Can model it as $x(n) = x(n)b(n)$
- $b(n) = \begin{cases} 
1, & 0 \leq n \leq N - 1 \\
0, & \text{else} \end{cases}$
- $X(\omega) = X(\omega) * B(\omega)$
  - $X(\omega)$ contains two deltas
  - $B(\omega)$ Dirichlet kernel (resembles sinc)
Qualitative comments about spectrum

- *Width* of main lobe proportional to $1/N$
- Big $N \rightarrow$ small $1/N \rightarrow$ narrow main lobe $\rightarrow$ two frequencies can be nearby

- *Height* of secondary lobes – if one sinusoid has much greater amplitude, other could be hidden in side lobes
Windowing

- **Bad news** – even for big N have limits on how well we can separate nearby freqs

- **Good news** – can do better than convolution w/Dirichlet kernel
  - Will convolve deltas w/narrow main & low side lobes

- **Windowing** – \( x(n) = x(n)w(n) \)
  - \( w(n) \) is the window
  - Window has finite duration N
  - Different trade-offs between main lobe width & side lobe height
Example 7.4.1

- Consider \( x_a(t) = \begin{cases} e^{-t}, & t \geq 0 \\ 0, & t < 0 \end{cases} \)
- We sample it – what’s its spectrum?

- Recall \( X(F) = \frac{1}{1+j2\pi F} \)
- \( x(n) = e^{-nT}u(n) = (e^{-T})^nu(n) \)

- Let’s compute DFT in Matlab:
  - \( T=0.05 \); % sampling period
  - \( N=20 \); % # samples
  - \( n=0:N-1 \); % time range
  - \( x=\exp(-T*n) \); % signal
  - \( \text{plot(fftshift(abs(fft(x))))} \); % fftshift aligns DFT freqs in “natural” order
Example 7.4.1 with bad windowing

- Suppose instead we plot $N_2 \gg N = 20$ frequency samples
  - Zero-pad $x$ and recompute DFT

$N_2 = 200$; % # frequency samples
$x_2 = [x \text{ zeros}(1,N_2-N)]; \% \text{zero-padded signal}$
$\text{plot(fftshift(abs(fft(x_2))))}$

- Bad window $\rightarrow$ interpolated freqs
Discrete Cosine Transform (DCT)
From complex to real valued transforms

- DFT requires computation with complex arithmetic → slower

- But most signals are real valued

- Symmetric (even) real signal has symmetric real transform
  - Copy signal from time \( n=1,\ldots,N-1 \) to negative \( n \)
  - Signal became symmetric w/symmetric transform coeffs
  - Can compute w/cosines → real-valued arithmetic

- Advantages of DCT
  - Algorithms for fast computation (resemble FFT)
  - Energy compaction properties (DCT coeffs are sparse)