Discrete Fourier Transform

Dror Baron
Associate Professor
Dept. of Electrical and Computer Engr.
North Carolina State University, NC, USA

Roadmap

- We have seen
  - Chapter 1 – from analog to digital and back
  - Chapter 2 – discrete time signals & systems
  - Chapter 4 – Fourier transforms
  - Chapter 5 – frequency domain analysis of LTI systems
- *About to discuss Chapters 7-8*
  - Frequency sampling of discrete time signals → discrete Fourier transform (DFT)
  - Application to spectral estimation
  - Fast Fourier transform (FFT) – fast computation of DFT

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 of Discrete Time Signals

[Reading material: Section 7.1]

What are we sampling?

- Instead of aperiodic x(n), consider *periodic repetition*:

$$x_p(n) = \sum_{l=-\infty}^{+\infty} x(n - lN)$$

- Notes:

1) $\mathrm{x_p(n)} = \sum_{l=-\infty}^{+\infty} x(n + lN)$
2) It's periodic-N

Fourier series for $x_p(n)$

- Periodic repetition $x_p(n)$ periodic → take its Fourier series

- $x_p(n) = \sum_{k=0}^{N-1} C_k e^{i2\pi kn/N}$
- $C_k = \frac{1}{N} \sum_{n=0}^{N-1} x_p(n)\, e^{-i2\pi kn/N}$

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

Sampling X(ω)

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

- $$X\left(\frac{2\pi k}{N}\right) = \sum_{n=-\infty}^{+\infty} x(n)e^{-\frac{i2\pi kn}{N}}$$

$$= \cdots + \sum_{n=-N}^{-1} x(n)e^{-\frac{i2\pi kn}{N}} + \sum_{n=0}^{N-1} x(n)e^{-\frac{i2\pi kn}{N}} + \cdots$$

$$= \sum_{l=-\infty}^{+\infty} \sum_{n=lN}^{lN+N-1} x(n)e^{-\frac{i2\pi kn}{N}}$$

$$= \sum_{m=0}^{N-1} \sum_{l=-\infty}^{+\infty} x(lN+m)e^{-\frac{i2\pi k}{N}(lN+m)}$$

n=lN+m

A useful observation

- Observe that $e^{-\frac{i2\pi k}{N}(lN+m)} = e^{-\frac{i2\pi}{N}m} e^{-\frac{i2\pi k}{N}lN} = e^{-\frac{i2\pi k}{N}m}$

- Back to derivation:

- $X\left(\frac{2\pi k}{N}\right) = \cdots = \sum_{m=0}^{N-1}\sum_{l=-\infty}^{+\infty} x(lN+m)e^{-\frac{i2\pi k}{N}(lN+m)}$

  $= \sum_{m=0}^{N-1}\sum_{l=-\infty}^{+\infty} x(lN+m)e^{-\frac{i2\pi km}{N}}$

  $= \sum_{m=0}^{N-1}[\sum_{l=-\infty}^{+\infty} x(lN+m)]e^{-\frac{i2\pi km}{N}}$

  $= \sum_{m=0}^{N-1} x_p(m)\, e^{-\frac{i2\pi km}{N}}$

  $= NC_k$ → *sampling X(ω) resembles Fourier series of $x_p(n)$*

Discussion

- Samples of $X(\omega)$ related to Fourier series of periodic extension $x_p(n)$

- Time-limited x(n) → one to one correspondence w/ $x_p(n)$
- Else x(n) will be "aliased" in $x_p(n)$

Discrete Fourier transform (DFT)

- Recall Fourier transform, $X(\omega) = \sum_{n=0}^{N-1} x(n) e^{-i\omega n}$
- Take N frequency samples at $\omega_k = \frac{2\pi k}{N}, k \in \{0, \ldots, N-1\}$
- We have $X(k) = X\left(\frac{2\pi k}{N}\right) = \sum_{n=0}^{N-1} x(n) e^{-i2\pi kn/N}$

- Inverse DFT (IDFT): $\mathrm{x(n)} = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{+i2\pi kn/N}$

- As before, $\frac{1}{N}$ because vectors of form $e^{i2\pi kn/N}$ are orthogonal

Spectral Estimation 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)

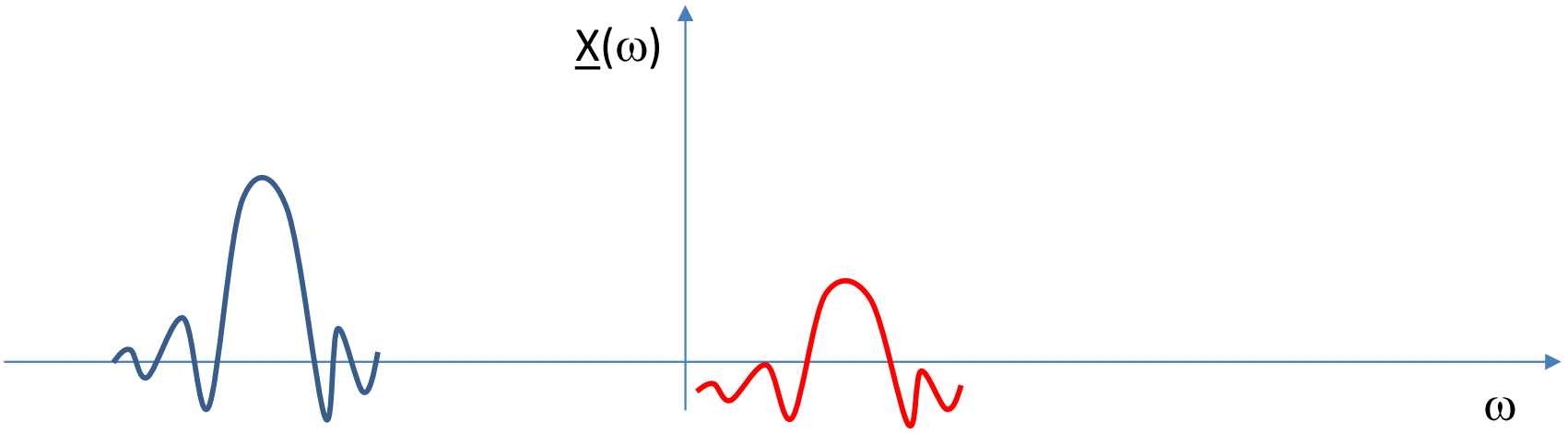
- To keep simple: *two sinusoids*
- $x(n)=a_1\cos(\omega_1 n+\Phi_1)+a_2\cos(\omega_2 n+\Phi_2)$

- $X(\omega)$ contains 2 conjugate pairs of deltas (4 deltas in total)

- More realistic signals contain multiple sinusoids, measurement noise, … → tougher to estimate spectral content

Spectrum of finite duration samples

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



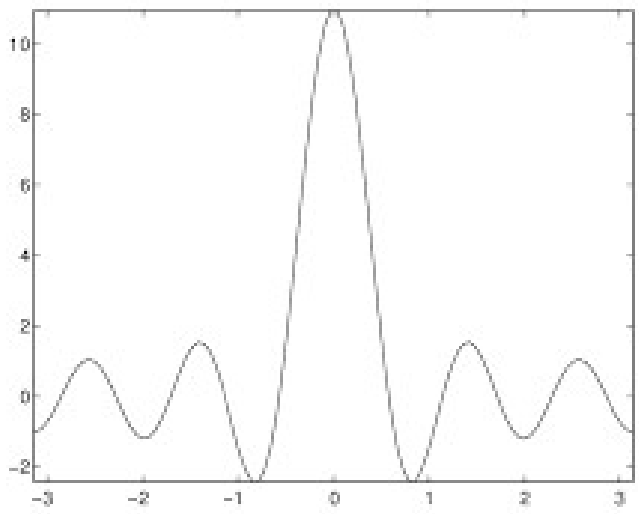
Dirichlet kernel

- Dirichlet kernel P(ω) is Fourier transform of b(n)
- $B(\omega) = \sum_{n=0}^{N-1} e^{-i\omega n}$

$$= \frac{1-e^{-i\omega N}}{1-e^{-i\omega}}$$

$$= \frac{e^{-i\omega N/2}\left(e^{+i\omega N/2}-e^{-i\omega N/2}\right)}{e^{-i\omega/2}\left(e^{+i\omega/2}-e^{-i\omega/2}\right)}$$

$$= \frac{\sin(\omega N/2)}{\sin(\omega/2)} e^{-i\omega(N-1)/2}$$

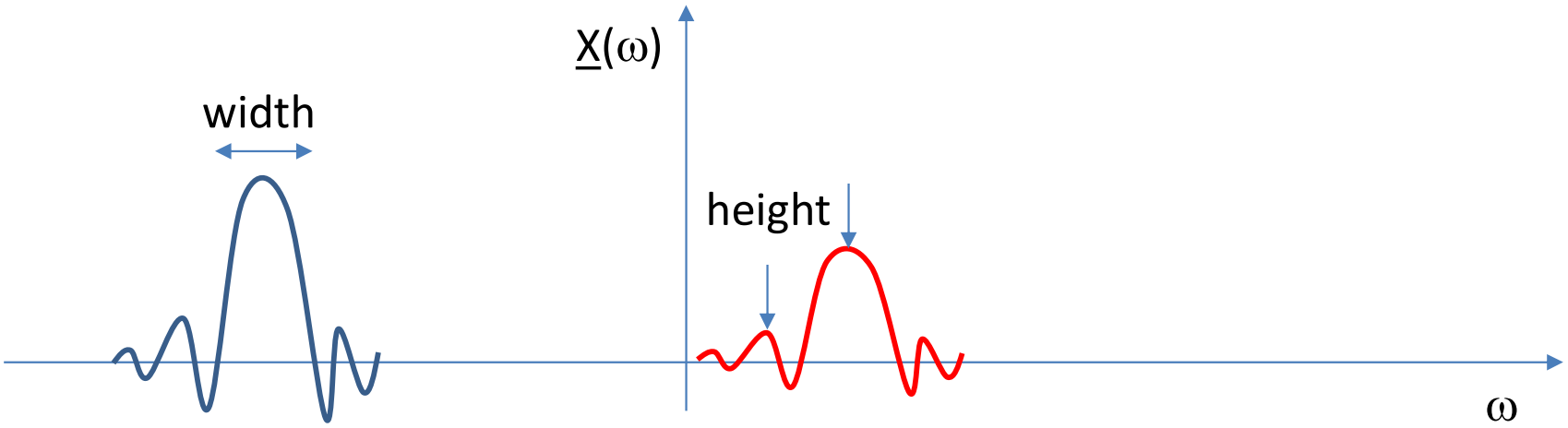


- Oscillates within envelope that decays as $1/\omega$
- Resembles sinc (low pass in cont. time) + artifacts due to periodic X(ω)
- Numerator zero for $\omega = \frac{2\pi k}{N}$ (except for DC frequency)
- DFT samples $\underline{X}(\omega)$ at $\omega = \frac{2\pi k}{N}$

Qualitative comments about spectrum

- *Width* of main lobe proportional to 1/N
- Big N → small 1/N → narrow main lobe → 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 – $\underline{x}(n)=x(n)w(n)$
  - w(n) is window
  - Window has finite duration N
  - Different trade-offs between main lobe width & side lobe height

Fast Fourier Transform

[Reading material: Chapter 8]

Naïve computation of DFT

- Recall $X(k) = \sum_{n=0}^{N-1} x(n) e^{-i2\pi kn/N}$

- Matlab:

```
N=2000; % signal length
x=randn(N,1); % random input
xf=zeros(N,1); % initialize DFT coeffs
tic % start clock
for k=0:N-1 % loop over k
  xf(k+1)=exp(-i*2*pi*k*(0:N-1)/N)*x; % computes everything one line
end;
toc % ends clock
```

- Complicated line: inner product between row exp(…) column x

Why is it slow?

- Main line runs N times
  - N complex multiplications
  - N-1 complex additions
  - N complex exponentials
  - More real-valued arithmetic (set up exponents)

- *Quadratic* computation

What's wrong with quadratic runtime?

- Let's compare N^2 (DFT runtime) with $N \times \log_2(N)$ for FFT

N	N^2	$N \log_2(N)$
10^3	10^6 (feasible 1960s)	~10^4
10^6	10^{12} (feasible 2020s)	~2×10^7
10^9	10^{18} **infeasible**	~3×10^{10}

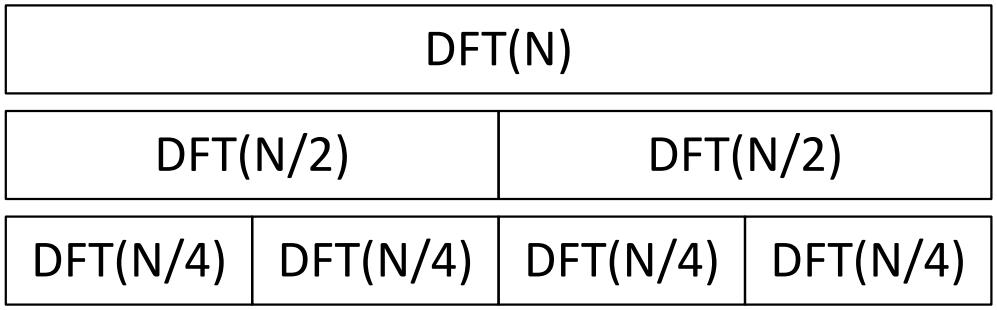
- Reminder about computer speeds:
  - 1960s: dozens to thousands operations/second
  - 1980s: millions
  - 2000s: billions
  - 2020s: 100X using general purpose graphics processing units (GPUs)

Divide and Conquer

How can we improve DFT speed?

- *Divide* into blocks
- Compute DFT of each block (*conquer*)
- Merge - “book keeping” to patch together DFT’s of blocks
  - Merging step must be fast (linear in N)

- Trick: use divide & conquer hierarchically within blocks



Example: Sorting

- Let's see simpler $N \times \log_2(N)$ algorithm
- *Sort* set of numbers, x={1,9, -2, 3, 6, -1, 7, 4}
  - Step1: partition into x_1={1,9, -2, 3} and x_2={6, -1, 7, 4}
  - Step2: sort each component
    - Sort(x_1)={-2, 1, 3, 9}, Sort(x_2)={-1,4,6,7}
    - Recursive calls might be needed
  - Step3: merge sub-components
    - -2 < -1 → Sort(x)←{-2}, Sort(x_1)←{1, 3, 9}
    - -1 < 1 → Sort(x)←{-2, -1}, Sort(x_2)←{4, 6, 7}
    - 1 < 4 → Sort(x)←{-2, -1, 1}, Sort(x_1)←{3, 9}
    - …
  - Merging requires runtime linear in N

- Can show that mergesort requires $N \times \log_2(N)$ runtime
- DFT also combines sub-solutions in linear time → $N \times \log_2(N)$