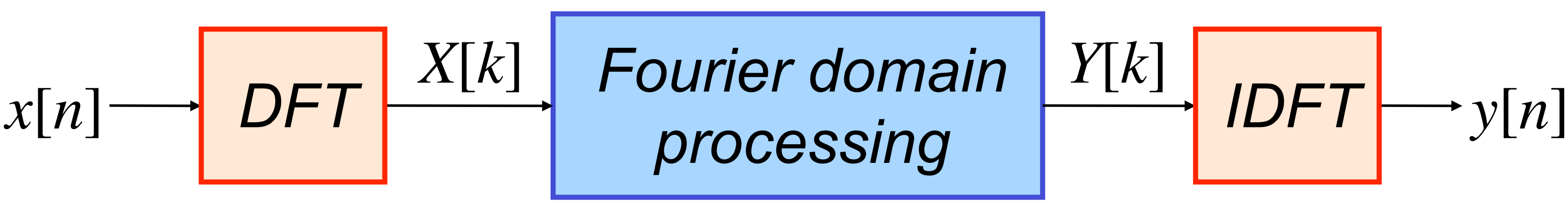
ELEN E4810: Digital Signal Processing
Topic 10: The Fast Fourier Transform

1. Calculation of the DFT
2. The Fast Fourier Transform algorithm
3. Short-Time Fourier Transform

1. Calculation of the DFT

- Filter design so far has been oriented to time-domain processing - cheaper!
- But: frequency-domain processing makes some problems very simple:

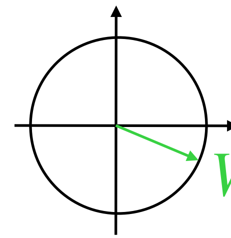


  - use all of $x[n]$, or use short-time windows
- Need an efficient way to calculate DFT

The DFT

- Recall the DFT:

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



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

W_N@$2\pi/N$
$\Rightarrow W_N{}^r$ has only
N distinct values

  - discrete transform of discrete sequence

- Matrix form:

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ \vdots \\ X[N-1] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N^1 & W_N^2 & \cdots & W_N^{(N-1)} \\ 1 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{(N-1)} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)^2} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \vdots \\ x[N-1] \end{bmatrix}$$

Structure $\Rightarrow$ opportunities for efficiency

Computational Complexity

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

- N complex multiplies
  + N-1 complex adds per point (k)
  × N points ($k = 0..\ N$-1)
  - cpx mult: $(a+jb)(c+jd) = ac - bd + j(ad + bc)$
    = 4 real mults + 2 real adds
  - cpx add = 2 real adds
- N points: $4N^2$ real mults, $4N^2$-$2N$ real adds

Goertzel's Algorithm

- Now: $X[k] = \sum_{\ell=0}^{N-1} x[\ell] W_N^{k\ell}$

$$= W_N^{kN} \sum_{\ell} x[\ell] W_N^{-k(N-\ell)}$$

looks like a convolution

- i.e. $X[k] = y_k[N]$

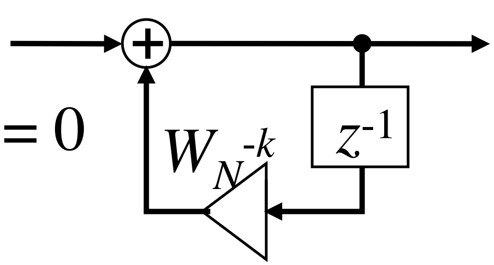
where $y_k[n] = x_e[n] \circledast h_k[n]$

$x_e[n] = \begin{cases} x[n] & 0 \le n < N \\ 0 & n = N \end{cases}$

$h_k[n] = \begin{cases} W_N^{-kn} & n \ge 0 \\ 0 & n < 0 \end{cases}$

$x_e[n]$

$x_e[N] = 0$



$y_k[n]$

$y_k[-1] = 0$

$y_k[N] = X[k]$

Goertzel's Algorithm

- Separate 'filters' for each $X[k]$
  - can calculate for just a few values of k
- No large buffer, no coefficient table
- Same complexity for full $X[k]$
  ($4N^2$ mults, $4N^2 - 2N$ adds)
  - but: can halve multiplies by making the denominator real:

$$H(z) = \frac{1}{1 - W_N^{-k} z^{-1}} = \frac{1 - W_N^{k} z^{-1}}{1 - 2\cos\frac{2\pi k}{N} z^{-1} + z^{-2}}$$

evaluate only for last step

2 real mults per step

2. Fast Fourier Transform FFT

- Reduce complexity of DFT from $O(N^2)$ to $O(N \cdot \log N)$
  - grows more slowly with larger N
- Works by decomposing large DFT into several stages of smaller DFTs
- Often provided as a highly optimized library

Decimation in Time (DIT) FFT

- Can rearrange DFT formula in 2 halves:

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

$k = 0..\ N\text{-}1$

Arrange terms in pairs...

$$= \sum_{m=0}^{\frac{N}{2}-1} \left(x[2m] \cdot W_N^{2mk} + x[2m+1] \cdot W_N^{(2m+1)k} \right)$$

Group terms from each pair

$$= \underbrace{\sum_{m=0}^{\frac{N}{2}-1} x[2m] \cdot W_{\frac{N}{2}}^{mk}}_{X_0[\langle k \rangle_{N/2}]} + W_N^k \underbrace{\sum_{m=0}^{\frac{N}{2}-1} x[2m+1] \cdot W_{\frac{N}{2}}^{mk}}_{X_1[\langle k \rangle_{N/2}]}$$

*$N/2$ pt DFT of x for **even** n*   *$N/2$ pt DFT of x for **odd** n*

Decimation in Time (DIT) FFT

$x[n]$ *for* ***even*** n — $x[n]$ *for* ***odd*** n

$$\mathrm{DFT}_N\{x[n]\} = \mathrm{DFT}_{\frac{N}{2}}\{x_0[n]\} + W_N^k \,\mathrm{DFT}_{\frac{N}{2}}\{x_1[n]\}$$

- We can evaluate an N-pt DFT as two $N/2$-pt DFTs (plus a few mults/adds)
- But if $\mathrm{DFT}_N\{\bullet\} \sim O(N^2)$

  then $\mathrm{DFT}_{N/2}\{\bullet\} \sim O((N/2)^2) = 1/4\ O(N^2)$

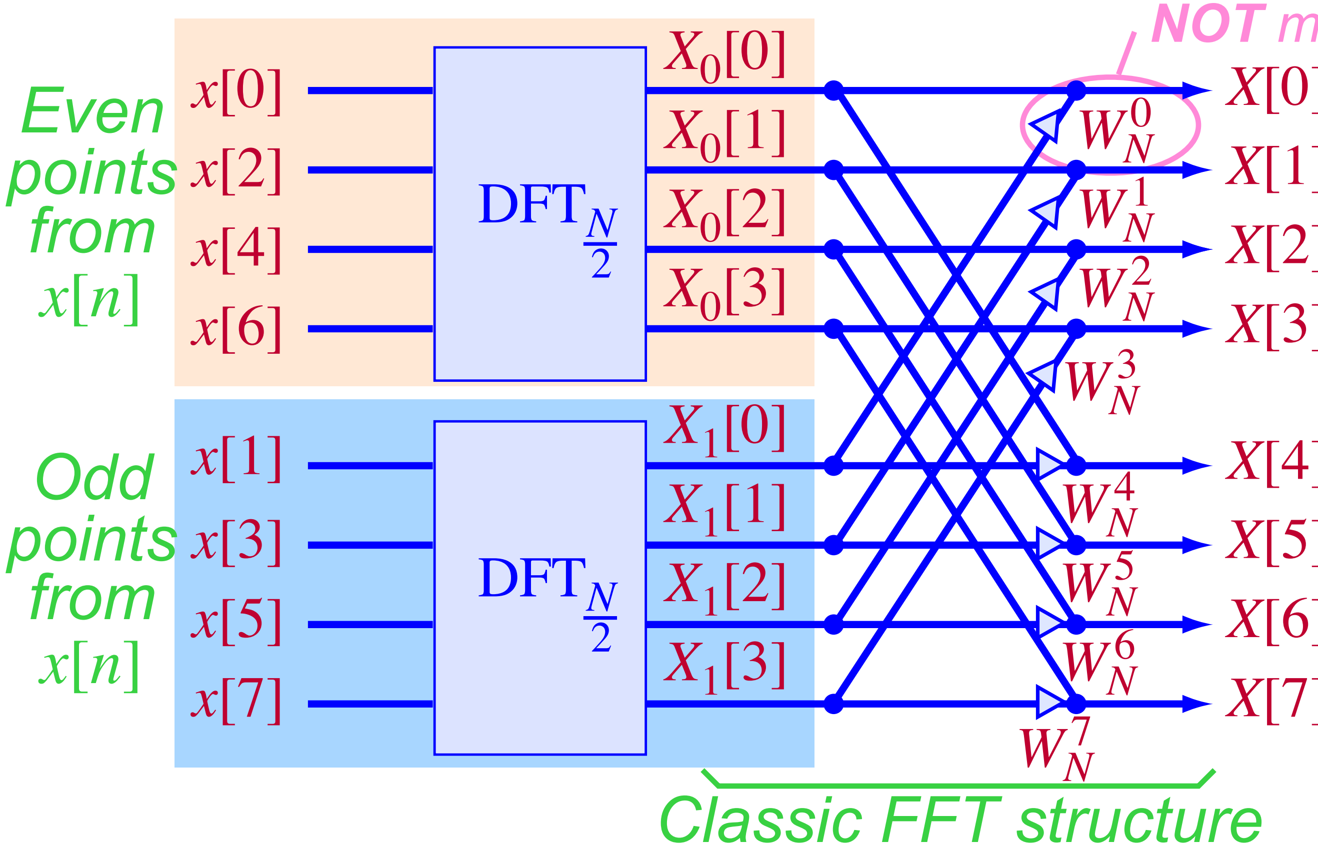
$\Rightarrow$ Total computation $\sim 2 \cdot 1/4\ O(N^2)$

$= 1/2$ the computation $(+\varepsilon)$ of direct DFT

One-Stage DIT Flowgraph

$$X[k] = X_0\left[\langle k \rangle_{\frac{N}{2}}\right] + W_N^k X_1\left[\langle k \rangle_{\frac{N}{2}}\right]$$

*"**twiddle factors**": always apply to odd-terms output **NOT** mirror-image*



Same as $X[0..3]$ except for factors on $X_1[\bullet]$ terms

Multiple DIT Stages

- If decomposing one DFT_N into two smaller $\mathrm{DFT}_{N/2}$'s speeds things up ... Why not further divide into $\mathrm{DFT}_{N/4}$'s ?
- i.e. $X[k] = X_0\left[\langle k \rangle_{\frac{N}{2}}\right] + W_N^k X_1\left[\langle k \rangle_{\frac{N}{2}}\right]$

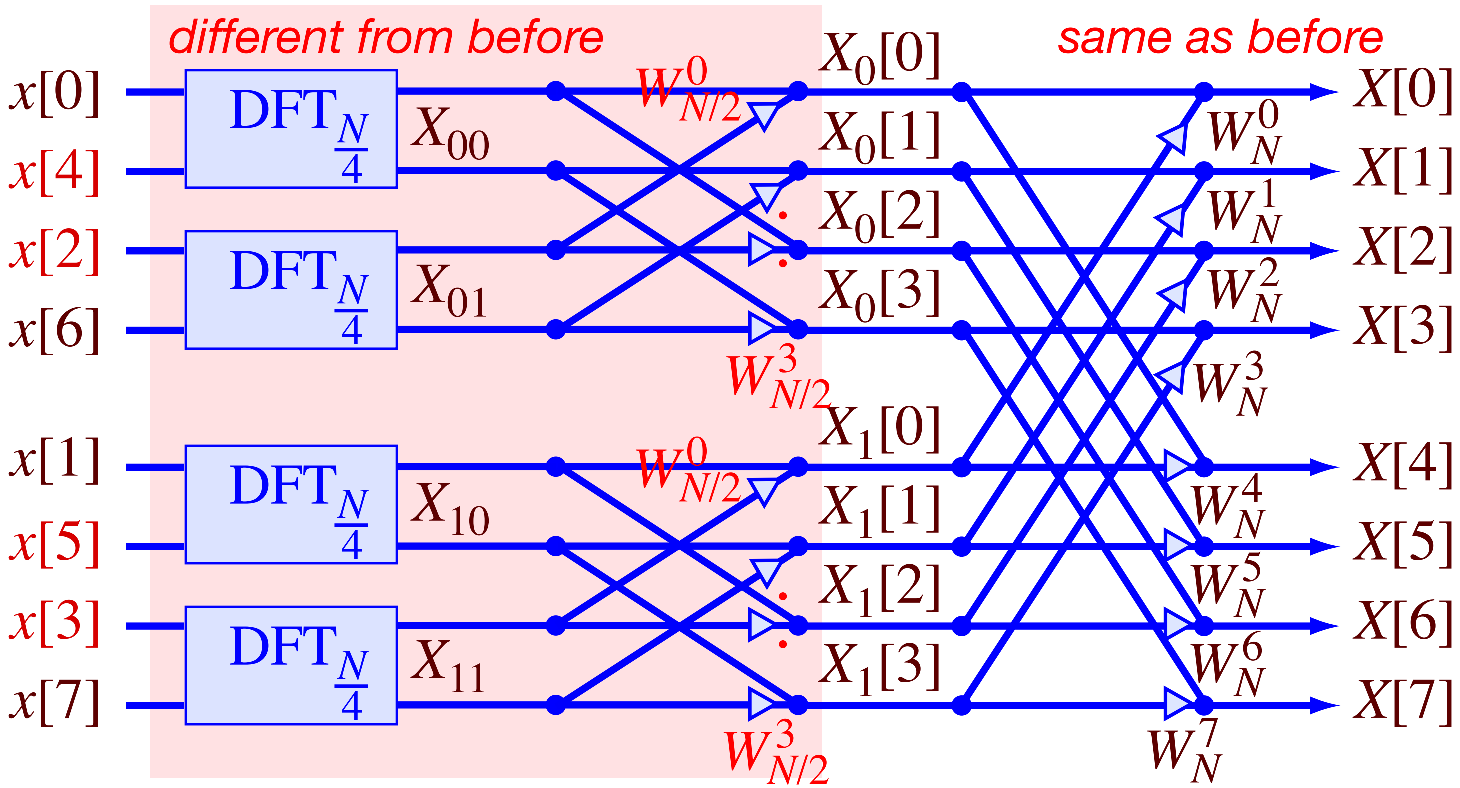
  $0 \le k < N$
- make: $X_0[k] = X_{00}\left[\langle k \rangle_{\frac{N}{4}}\right] + W_{\frac{N}{2}}^k X_{01}\left[\langle k \rangle_{\frac{N}{4}}\right]$

  $0 \le k < N/2$

  *N/4-pt DFT of **even** points in **even** subset of $x[n]$*

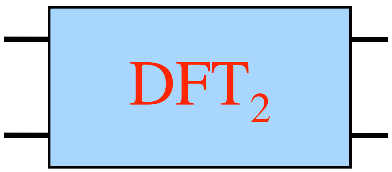
  *N/4-pt DFT of **odd** points from **even** subset*
- Similarly, $X_1[k] = X_{10}\left[\langle k \rangle_{\frac{N}{4}}\right] + W_{\frac{N}{2}}^k X_{11}\left[\langle k \rangle_{\frac{N}{4}}\right]$

Two-Stage DIT Flowgraph



Multi-stage DIT FFT

- Can keep doing this until we get down to 2-pt DFTs:

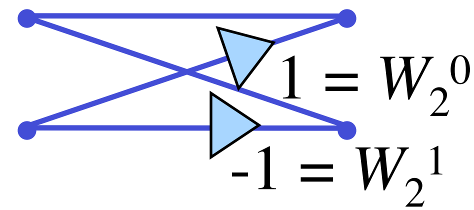


$X[0] = x[0] + x[1]$

$X[1] = x[0] - x[1]$

$\equiv$

"butterfly" element



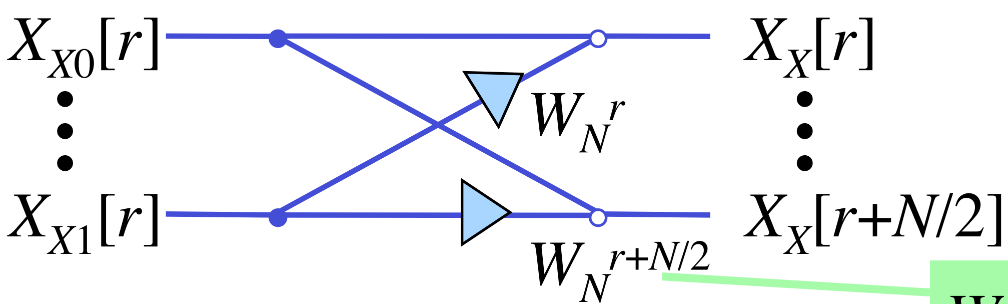
→ $N = 2^M$-pt DFT reduces to M stages of twiddle factors & summation ($O(N^2)$ part vanishes)

→ real mults $< M\cdot 4N$, real adds $< 2\cdot M\cdot 2N$

→ complexity $\sim O(N\cdot M) = O(N\cdot\log_2 N)$

FFT Implementation Details

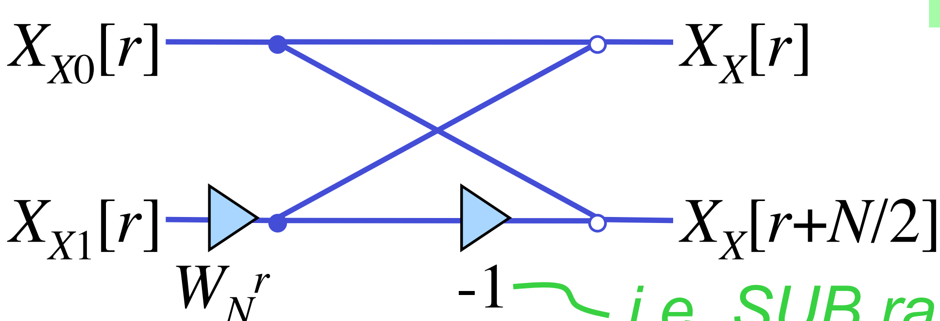
- Basic butterfly (at any stage):



2 cpx mults

$$
\begin{aligned}
W_N^{r+\frac{N}{2}} &= e^{-j\frac{2\pi\left(r+\frac{N}{2}\right)}{N}} \\
&= e^{-j\frac{2\pi r}{N}} \cdot e^{-j\frac{2\pi N/2}{N}} \\
&= -W_N^r
\end{aligned}
$$

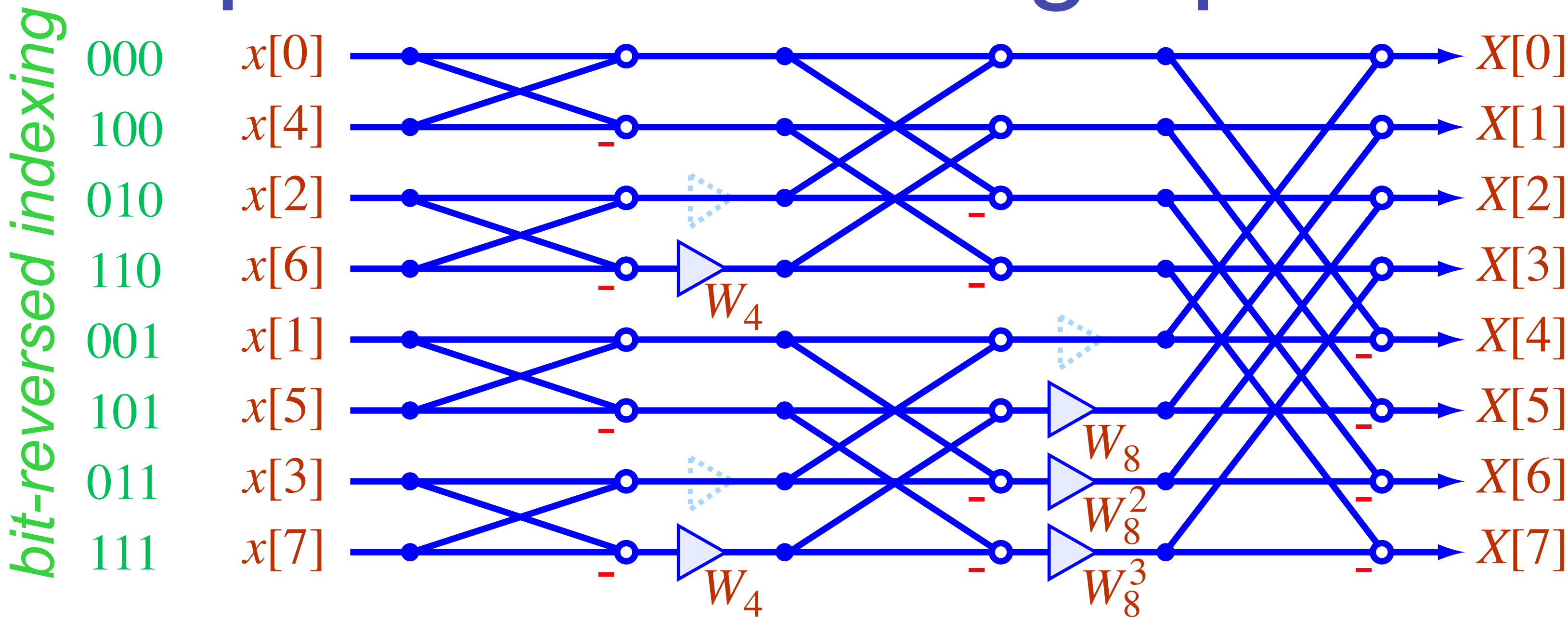
- Can simplify:



just one cpx mult!

i.e. SUB rather than ADD

8-pt DIT FFT Flowgraph



- -1's absorbed into summation nodes
- $W_N{}^0$ disappears
- 'in-place' algorithm: sequential stages

FFT for Other Values of N

- Having $N = 2^M$ meant we could divide each stage into 2 halves = "radix-2 FFT"
- Same approach works for:
  - $N = 3^M$ radix-3
  - $N = 4^M$ radix-4 - more optimized radix-2
  - etc...
- Composite $N = a \cdot b \cdot c \cdot d \rightarrow$ mixed radix (different N/r point FFTs at each stage)
  - .. or just zero-pad to make $N = 2^M$

Inverse FFT

only differences from forward DFT

- Recall IDFT: $$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k] W_N^{-nk}$$

- Thus:

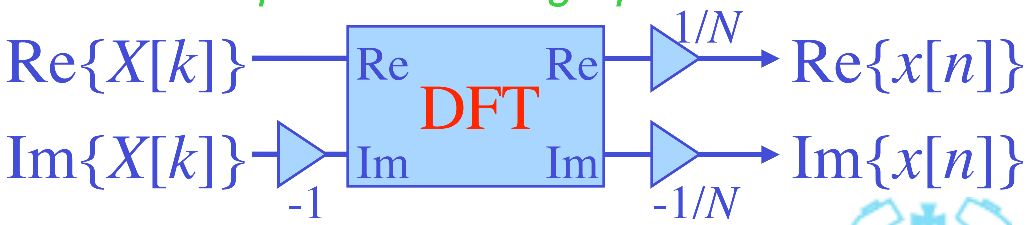
Forward *DFT of* $x'[n] = X^*[k]|_{k=n}$
i.e. time sequence made from spectrum

$$Nx^*[n] = \sum_{k=0}^{N-1}\left(X[k]W_N^{-nk}\right)^* = \sum_{k=0}^{N-1} X^*[k]W_N^{nk}$$

- Hence, use FFT to calculate IFFT:

$$x[n] = \frac{1}{N}\left[\sum_{k=0}^{N-1} X^*[k]W_N^{nk}\right]^*$$

pure real flowgraph



DFT of Real Sequences

- If $x[n]$ is pure-real, DFT wastes mult's
- Real $x[n]$ → Conj. symm. $X[k] = X^*[-k]$
- Given two real sequences, $x[n]$ and $w[n]$ call $y[n] = j \cdot w[n]$, $v[n] = x[n] + y[n]$
- N-pt DFT $V[k] = X[k] + Y[k]$
  but: $V[k]+V^*[-k] = X[k]+\underbrace{X^*[-k]}_{X[k]}+Y[k]+\underbrace{Y^*[-k]}_{-Y[k]}$

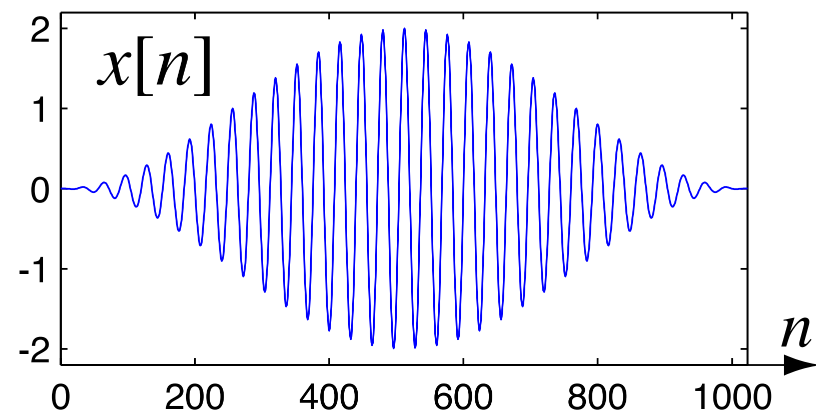
$\Rightarrow X[k]={}^1/_2(V[k]+V^*[-k])$, $W[k]={}^{-j}/_2(V[k]-V^*[-k])$

- i.e. compute DFTs of **two** N-pt real sequences with a single N-pt DFT

3. Short-Time Fourier Transform (STFT)

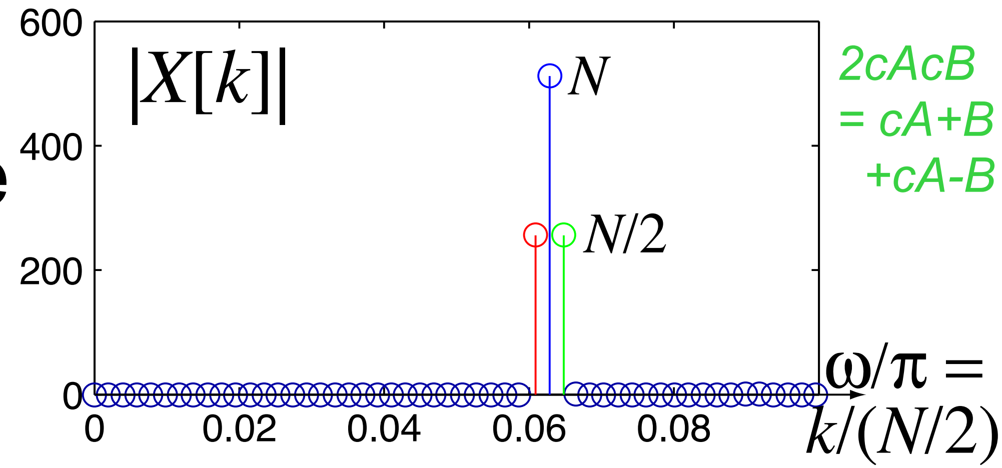
- Fourier Transform (e.g. DTFT) gives spectrum of an entire sequence:
- How to see a time-varying spectrum?
- e.g. slow AM of a sinusoid carrier:

$$x[n] = \left(1 - \cos\frac{2\pi n}{N}\right)\cos\omega_0 n$$



Fourier Transform of AM Sine

- Spectrum of whole sequence indicates modulation indirectly...
- ... as cancellation between closely-tuned sines

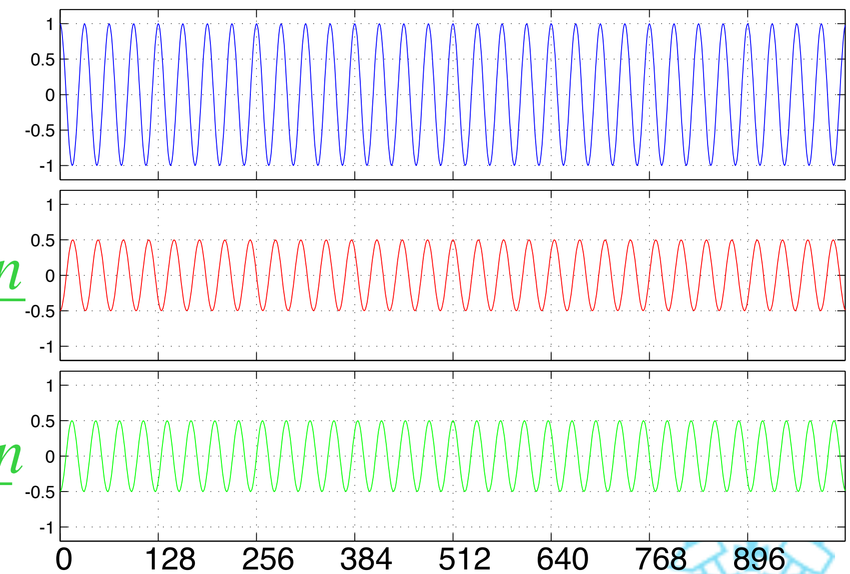


2cAcB
= cA+B
+cA-B

$$N \sin \frac{2\pi k n}{N}$$

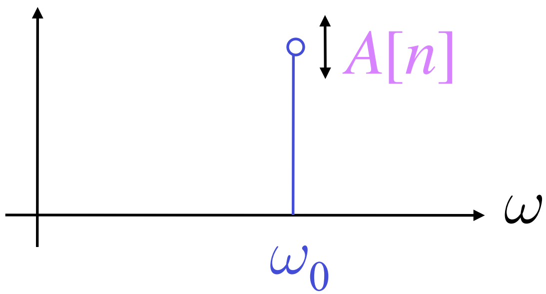
$$-\frac{N}{2} \sin \frac{2\pi (k-1) n}{N}$$

$$-\frac{N}{2} \sin \frac{2\pi (k+1) n}{N}$$



Fourier Transform of AM Sine

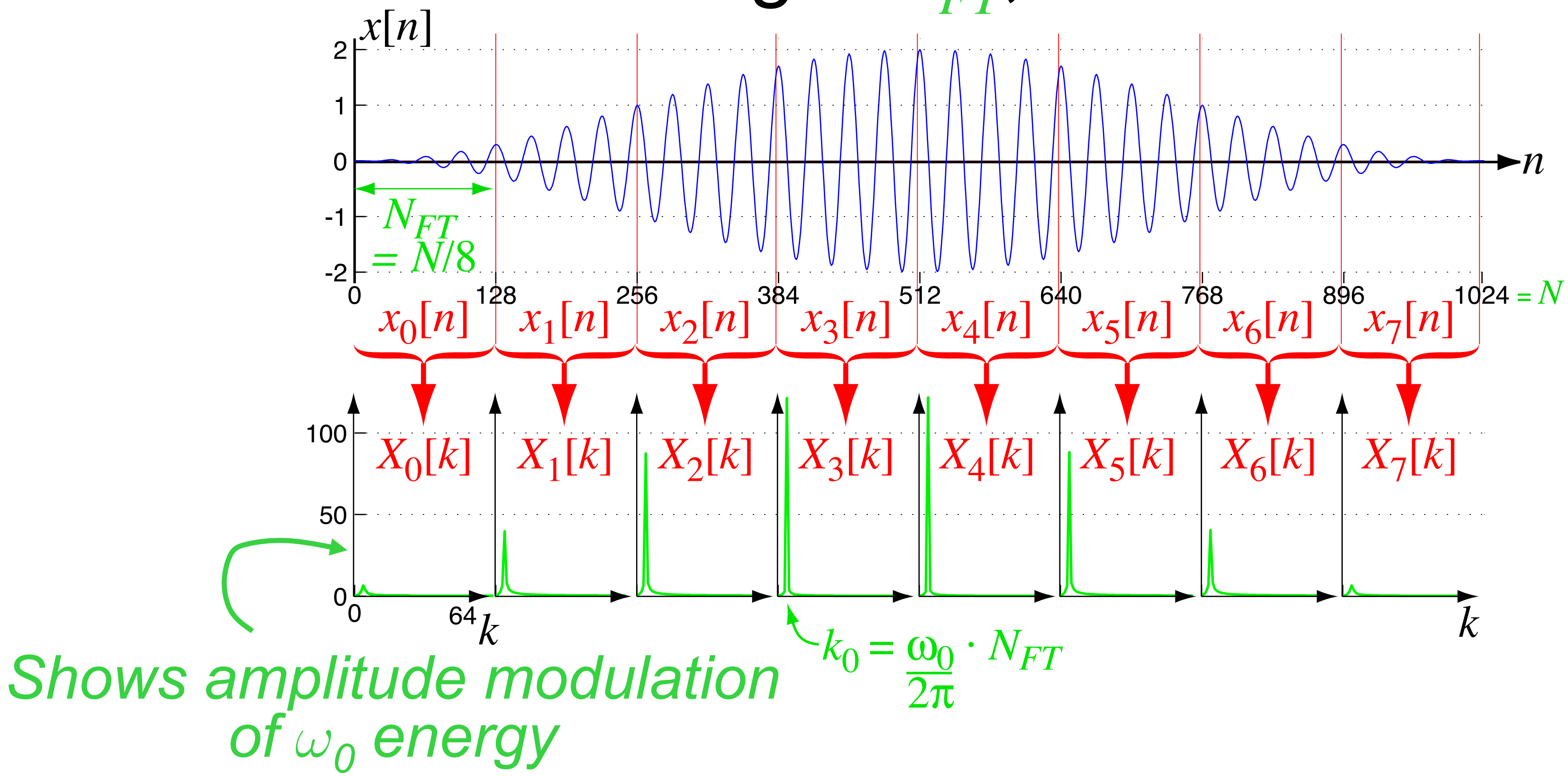
- Sometimes we'd rather separate modulation and carrier:
  $x[n] = A[n] \cos \omega_0 n$
  - $A[n]$ varies on a different (slower) timescale



- One approach:
  - chop $x[n]$ into short sub-sequences ..
  - .. where slow modulator is ~ constant
  - DFT spectrum of pieces → show variation

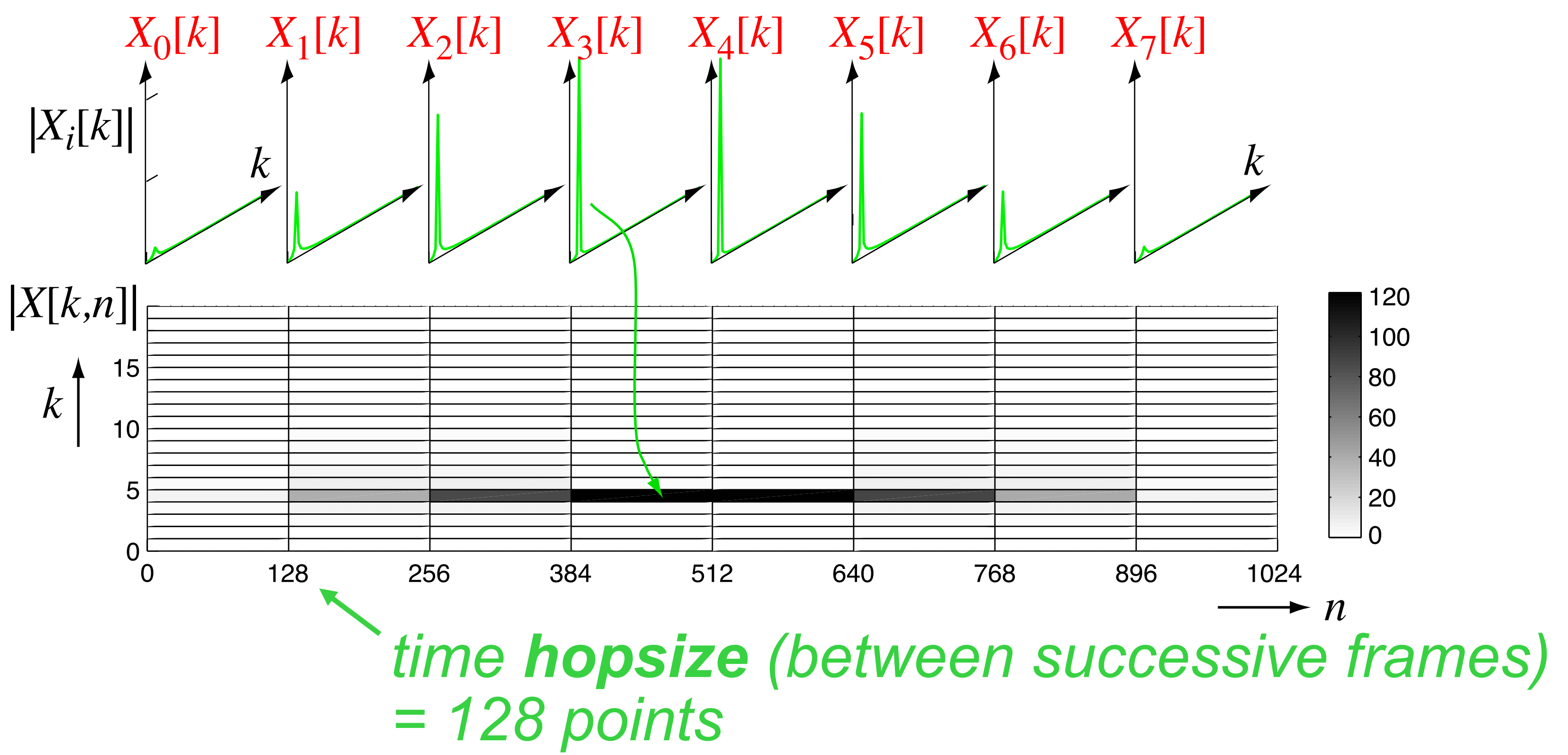
FT of Short Segments

- Break up x[n] into successive, shorter chunks of length N_{FT}, then DFT each:



The Spectrogram

- Plot successive DFTs in time-frequency:



- This image is called the Spectrogram

Short-Time Fourier Transform

- Spectrogram = STFT magnitude plotted on time-frequency plane
- STFT is (DFT form):

$$X[k,n_0] = \sum_{n=0}^{N_{FT}-1} x[n_0+n]\cdot w[n]\cdot e^{-j\frac{2\pi kn}{N_{FT}}}$$

frequency index

time index

N_{FT} points of x starting at n_0

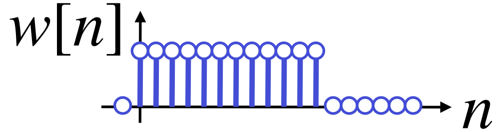
window

DFT kernel

  - intensity as a function of time & frequency

STFT Window Shape

- $w[n]$ provides 'time localization' of STFT
  - e.g. rectangular



    selects $x[n]$, $n_0 \le n < n_0 + N_W$

- But: resulting spectrum has same problems as windowing for FIR design:

DTFT form of STFT

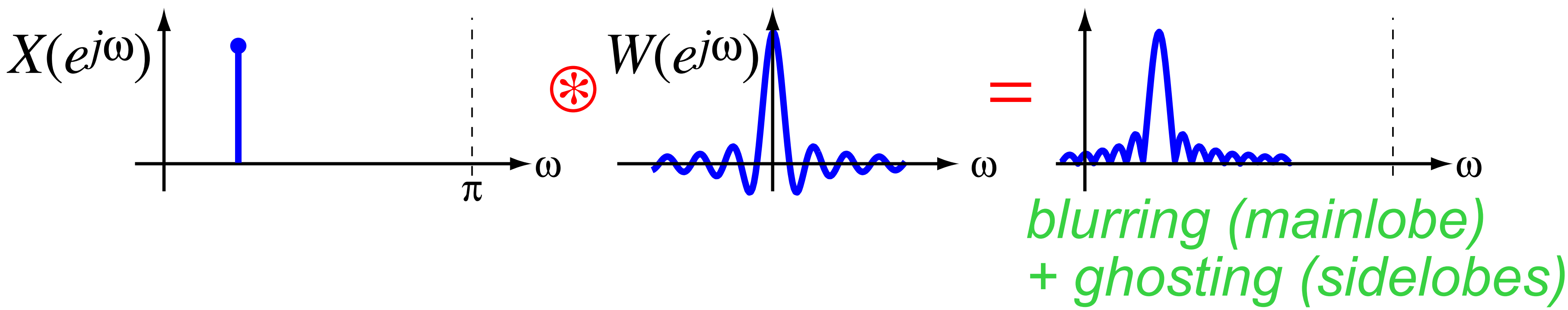
$$X(e^{j\omega}, n_0) = \mathrm{DTFT}\{x[n_0 + n] \cdot w[n]\}$$

$$= \int_{-\pi}^{\pi} e^{j\theta n_0} X(e^{j\theta}) W(e^{j(\omega-\theta)}) d\theta$$

spectrum of short-time window is convolved with (twisted) parent spectrum

STFT Window Shape

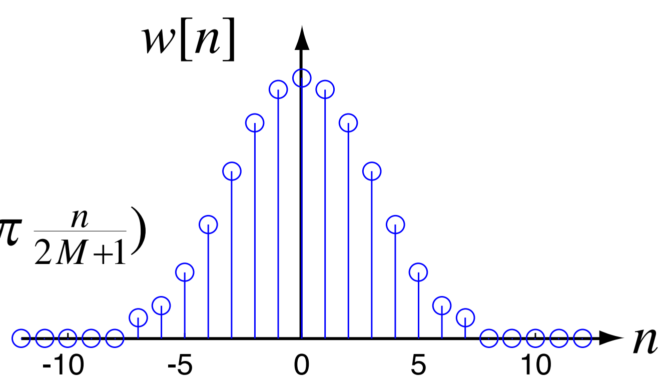
- e.g. if $x[n]$ is a pure sinusoid,

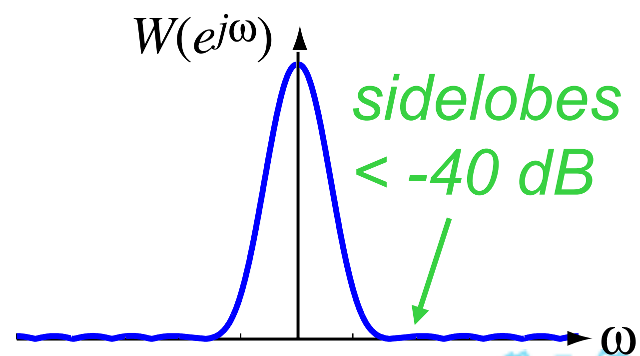


- Hence, use tapered window for $w[n]$

e.g. Hamming

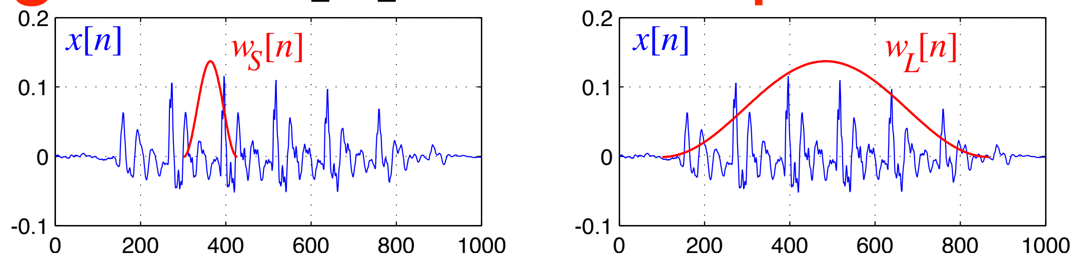
$$w[n] = 0.54 + 0.46\cos(2\pi \tfrac{n}{2M+1})$$





STFT Window Length

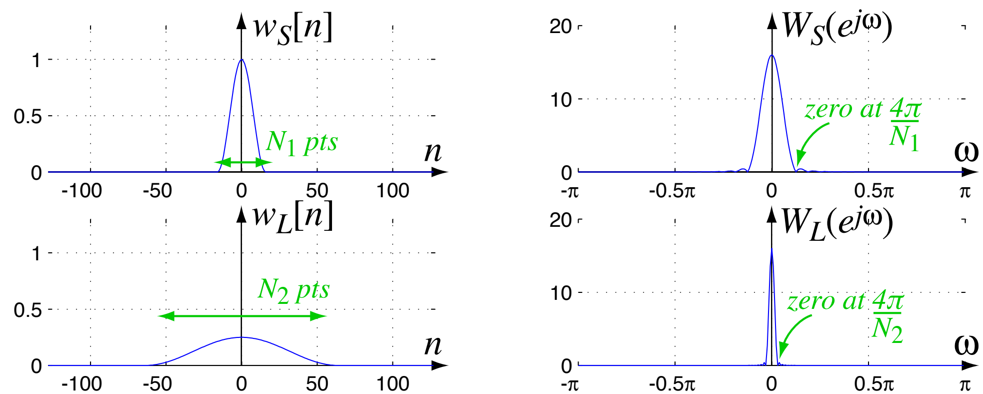
- Length of $w[n]$ sets temporal resolution



short window measures only local properties

longer window averages spectral character

- Window length ∝ 1/(Mainlobe width)

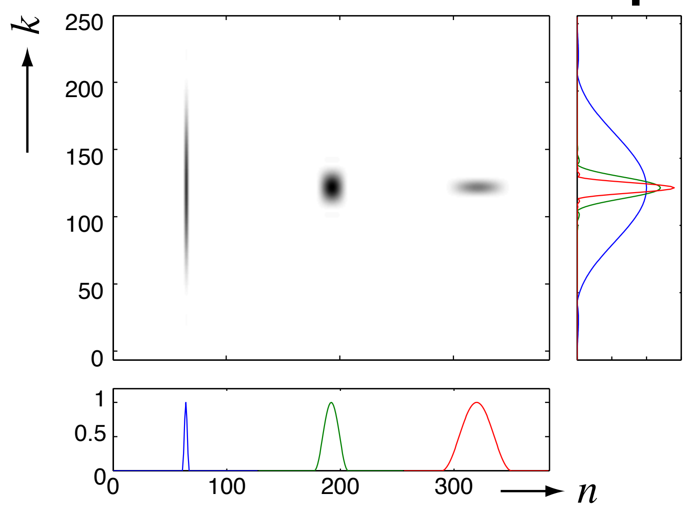


shorter window → more blurred spectrum

- *more* time detail ↔ *less* frequency detail

STFT Window Length

- Can illustrate time-frequency tradeoff on the time-frequency plane:

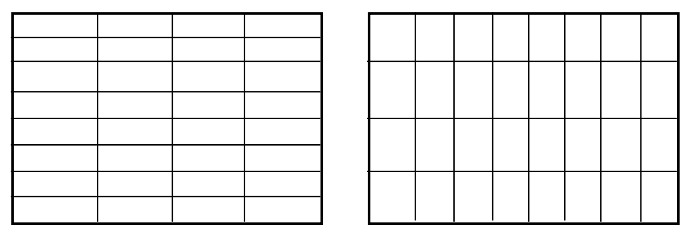


disks show 'blurring' due to window length; ***area*** *of disk is constant*

→ ***Uncertainty principle:***

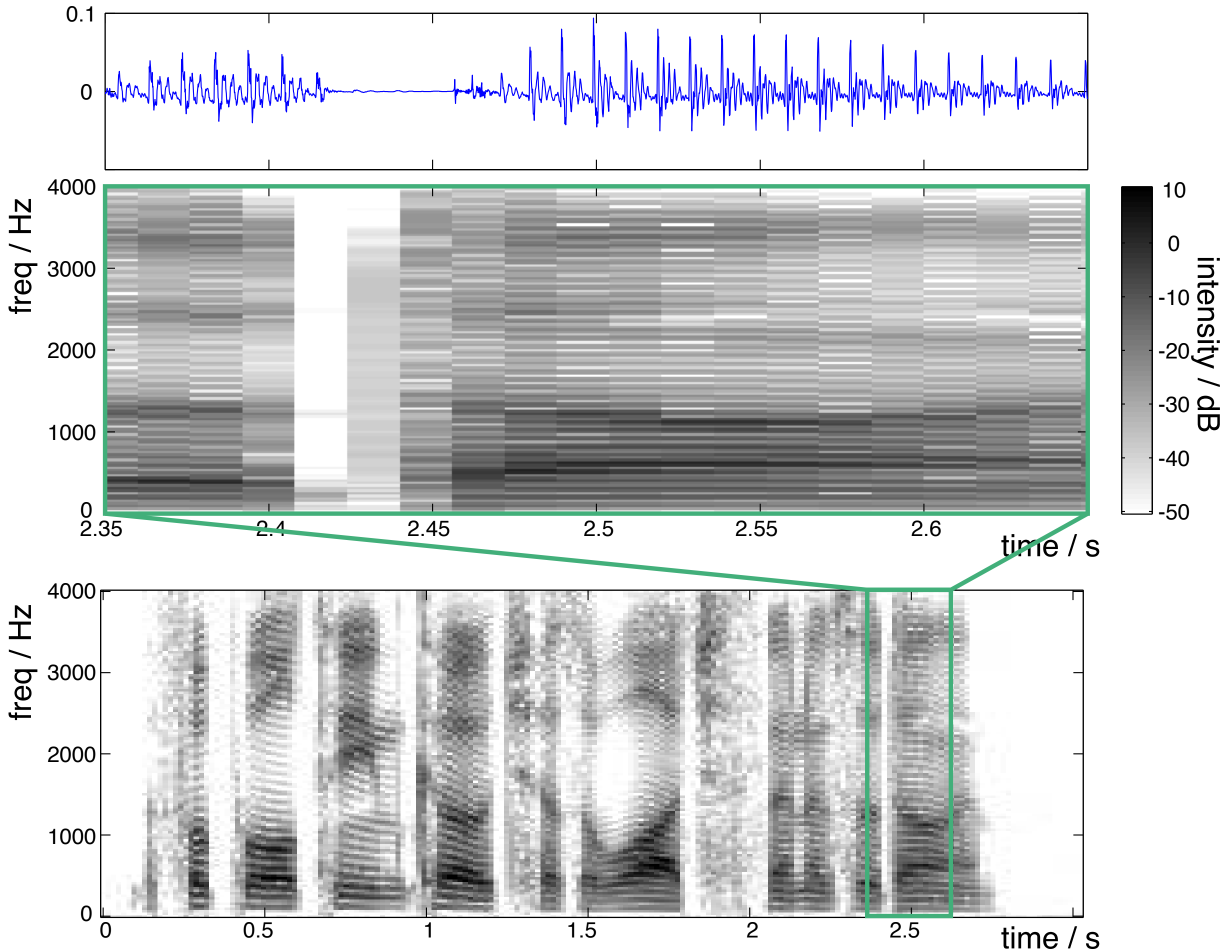
$$\delta f \cdot \delta t \geq k$$

- Alternate tilings of time-freq:



half-length window → half as many DFT samples

Spectrograms of Real Sounds



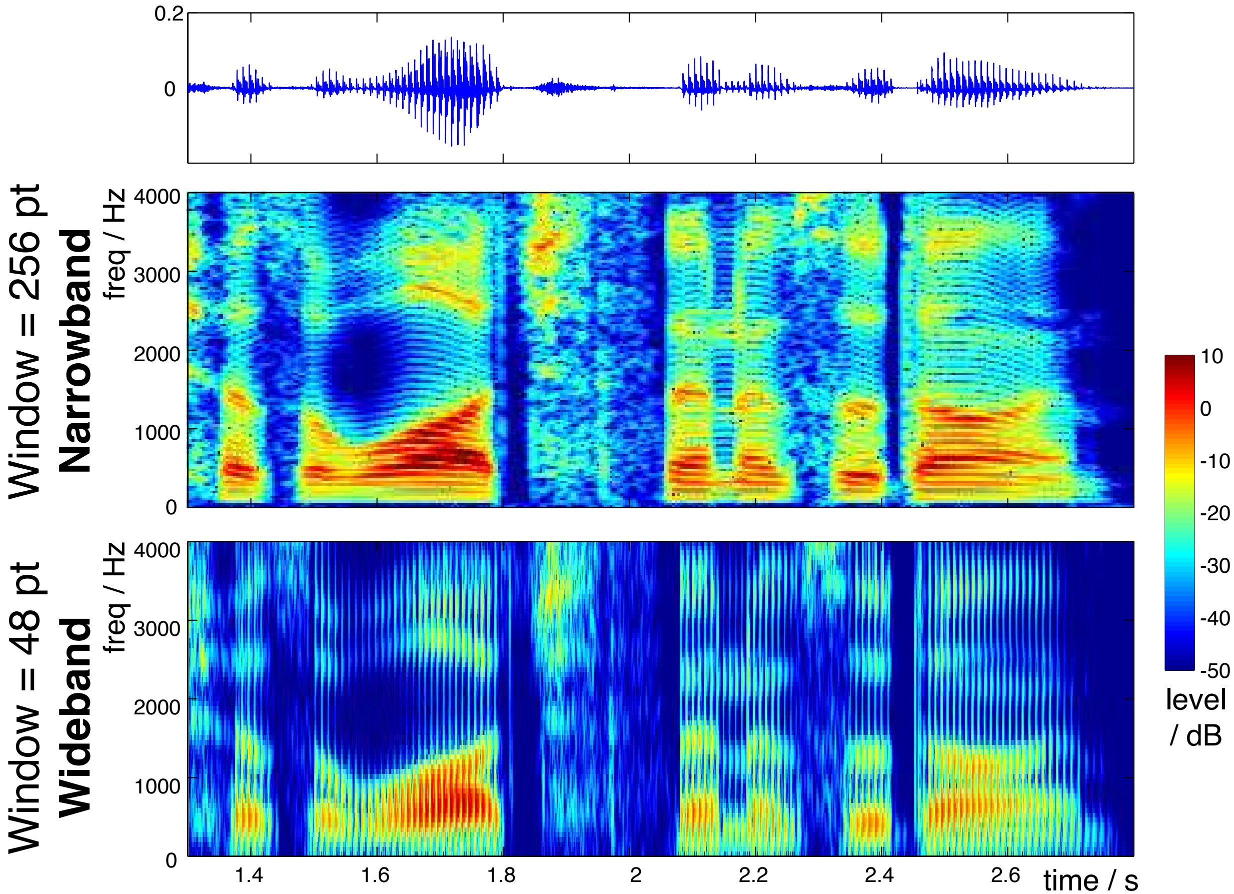
time-domain

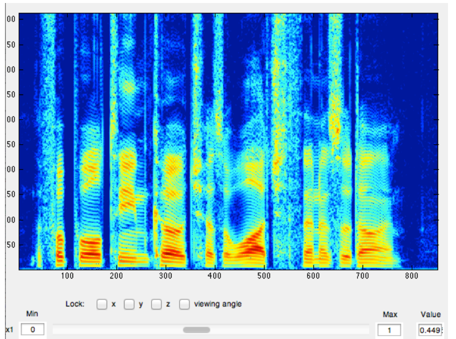
successive short DFTs

individual t-f cells merge into continuous image

Narrowband vs. Wideband

- Effect of varying window length:



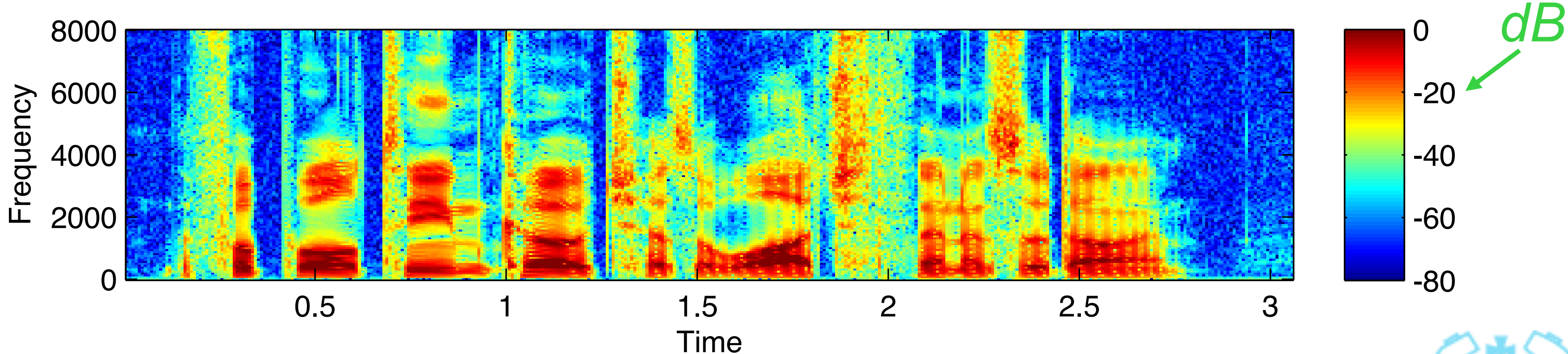


Spectrogram in Matlab

```
>> [d,sr]=wavread('mpgr1_sx419.wav');
>> Nw=256;
>> specgram(d,Nw,sr)
>> caxis([-80 0])
>> colorbar
```

(hann) window length

actual sampling rate (to label time axis)



STFT as a Filterbank

- Consider one 'row' of STFT:

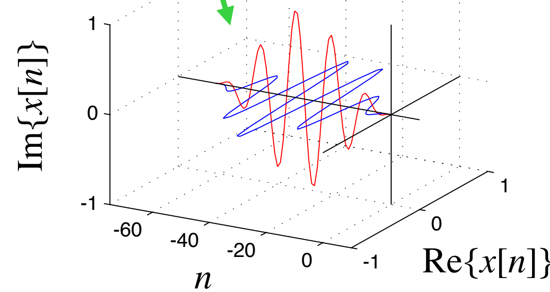
$$X_k[n_0] = \sum_{n=0}^{N-1} x[n_0+n] \cdot w[n] \cdot e^{-j\frac{2\pi kn}{N}}$$

just one freq.

$$= \sum_{m=0}^{-(N-1)} h_k[m] x[n_0 - m]$$

convolution *with* **complex** *IR*

where $h_k[n] = w[-n] \cdot e^{j\frac{2\pi kn}{N}}$



- Each STFT row is output of a filter (subsampled by the STFT hop size)

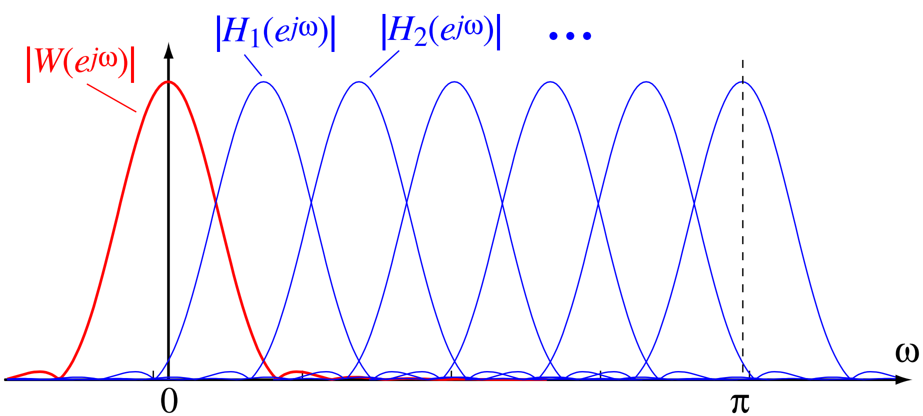
STFT as a Filterbank

- If $h_k[n] = w[(-)n] \cdot e^{j\frac{2\pi kn}{N}}$

  then $H_k(e^{j\omega}) = W\left(e^{(-)j\left(\omega - \frac{2\pi k}{N}\right)}\right)$ *shift-in-ω*

- Each STFT row is the same bandpass response defined by $W(e^{j\omega})$,

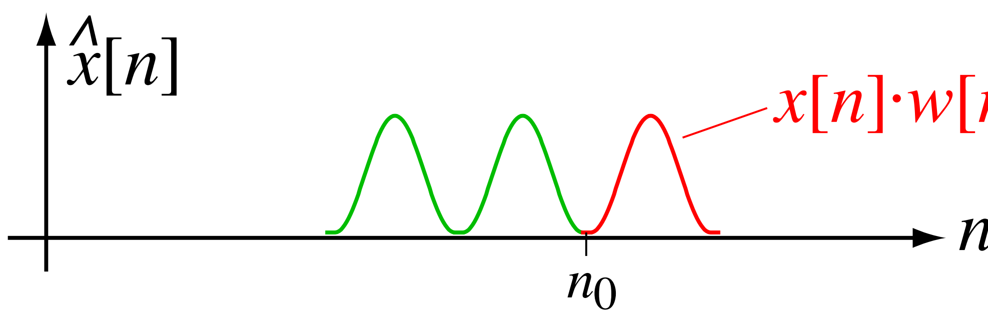
  frequency-shifted to a given DFT bin:



A bank of identical, frequency-shifted bandpass filters: ***"filterbank"***

STFT Analysis-Synthesis

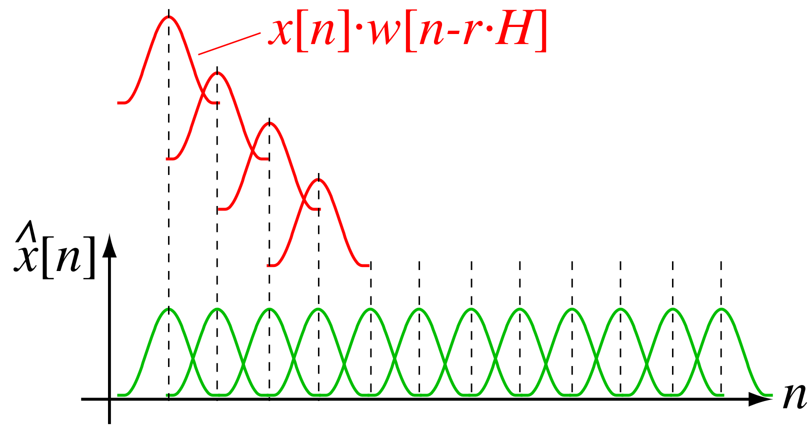
- IDFT of STFT frames can reconstruct (part of) original waveform
- e.g. if $X[k, n_0] = \mathrm{DFT}\{x[n_0 + n] \cdot w[n]\}$ then $\mathrm{IDFT}\{X[k, n_0]\} = x[n_0 + n] \cdot w[n]$
- Can shift by n_0, combine, to get $\hat{x}[n]$:



- Could divide by $w[n-n_0]$ to recover $x[n]$...

STFT Analysis-Synthesis

- Dividing by small values of $w[n]$ is bad
- Prefer to overlap windows:



i.e. sample $X[k,n_0]$

at $n_0 = r\cdot H$ where $H = N/2$ (for example)

hopsize *window length*

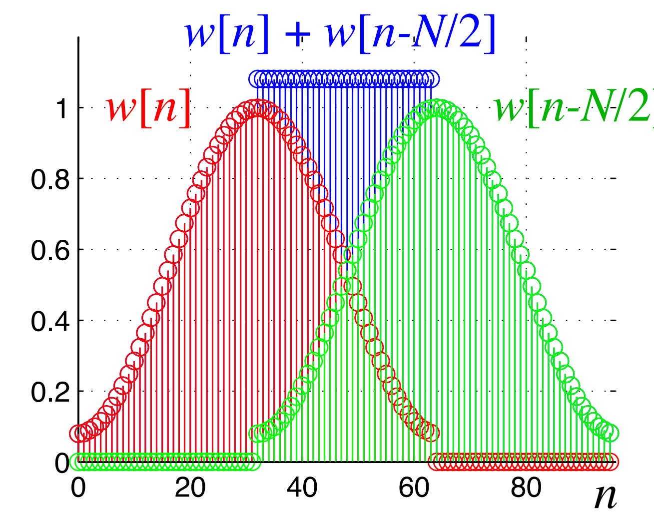
- Then $\hat{x}[n] = \sum_r x[n]w[n - rH]$

$= x[n]$ if $\sum_{\forall r} w[n - rH] = 1$

STFT Analysis-Synthesis

- Hann or Hamming windows with 50% overlap sum to constant

$$\left(0.54 + 0.46\cos(2\pi\tfrac{n}{N})\right) + \left(0.54 + 0.46\cos(2\pi\tfrac{n-\frac{N}{2}}{N})\right) = 1.08$$



- Can modify individual frames of $X[k,n]$ and then reconstruct
  - complex, time-varying modifications
  - tapered overlap makes things OK

STFT Analysis-Synthesis

- e.g. Noise reduction:

STFT of original speech

Speech corrupted by white noise

Energy threshold mask