EDISP (Win + FFT app)
(English) Digital Signal Processing
Windowing and FFT applications lecture

November 6, 2007

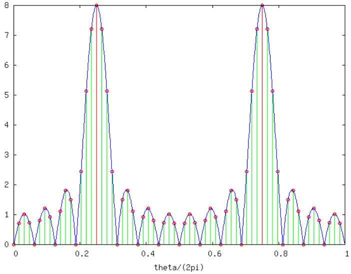
Limited observation time

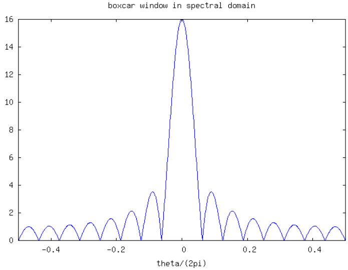
For DFT we used to cut a fragment of the signal

$$x_0[n] = x[n]g[n], \text{ where } g[n] = \begin{cases} 1 & \text{for} \quad n = 0, 1, \ldots, N-1 \\ 0 & \text{for} \quad \text{other } n \end{cases}$$

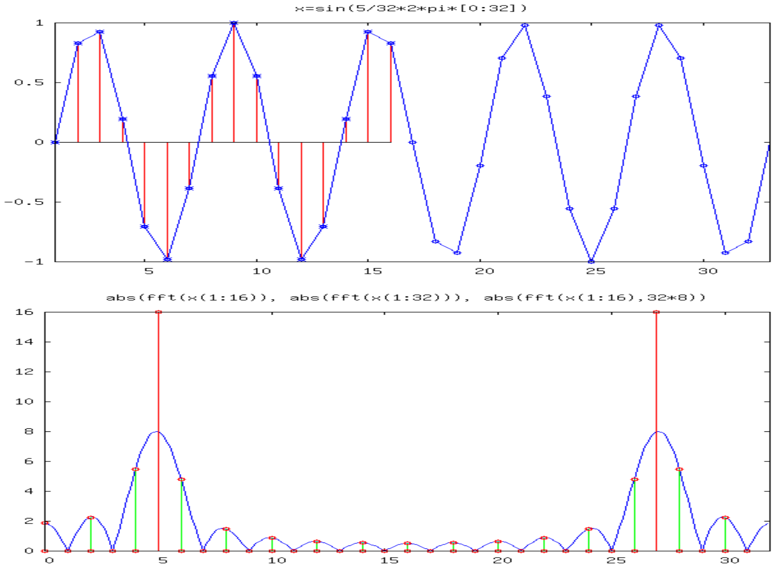
$g[n]$ is a window function. Here - a *boxcar window*
Window effect:

- selection of a signal fragment
- $x[n] \cdot g[n]$ in time $\longrightarrow$ $X(\theta) * G(\theta)$ in spectral domain $\longrightarrow$ *sidelobes* or *spectral leakage*

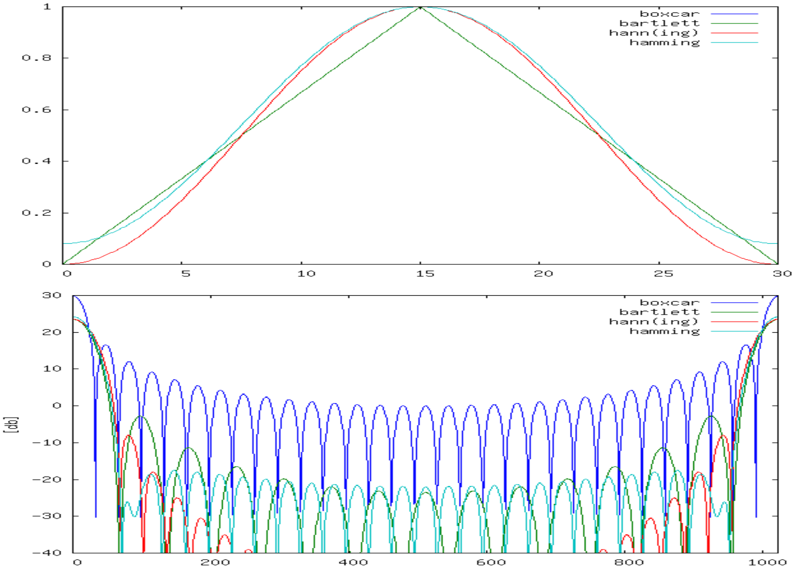




Leakage example



Window (apodization) functions



Raised cosine window family

- Hann window: Julius von Hann, 1839 – 1921, Austrian meteorologist; *hanning* is a verb form (*to hann*) $w(n) = 0.5\ \left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right)$
- Hamming window: Richard Hamming, 1915 – 1998, American mathematician; $w(n) = 0.53836 - 0.46164\ \cos\left(\frac{2\pi n}{N-1}\right)$
- Blackman window $w(n) = 0.42 - 0.5\cos\left(\frac{2\pi n}{N-1}\right) + 0.08\cos\left(\frac{4\pi n}{N-1}\right)$

Kaiser window

(D. Slepian, H.O. Pollak, H.J. Landau, around 1961, *Prolate spheroidal wave functions . . .*)

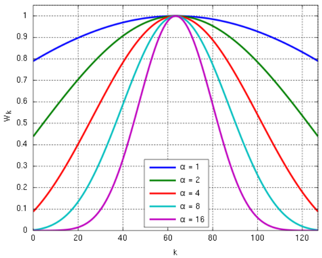
- time limited sequence with energy concentrated in finite frequency interval
- a family of windows with many degrees of freedom
- Kaiser (1974) – an approximation to optimal window: standard method to compute the optimal window was numerically ill-conditioned.

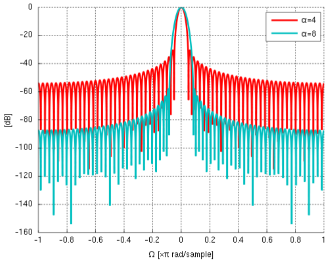
$$w_n = \begin{cases} \dfrac{I_0\left(\alpha\sqrt{1-\left(\frac{2n}{N}-1\right)^2}\right)}{I_0(\alpha)} & \text{if } 0 \leq n \leq N \\ 0 & \text{otherwise} \end{cases}$$

I_0 – zeroth order modified Bessel function of the first kind,

- α (real number) determines the shape of the window:
  - $\alpha = 0$ gives Boxcar,
  - $\alpha = 4$ gives -30 dB first sidelobe, -50 asymptotic,
  - $\alpha = 8$ gives -60 dB first sidelobe, -90 asymptotic,

Kaiser window





Calculating convolution by FFT

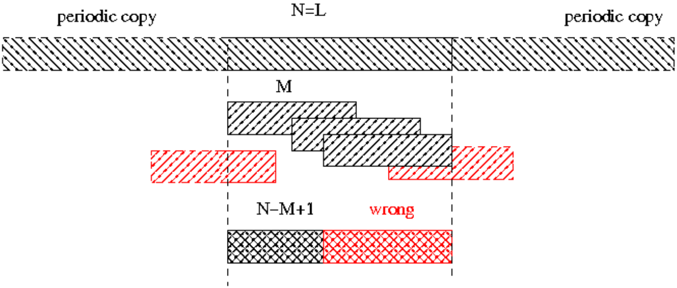
$$
\begin{array}{ccc}
X(\theta)\cdot Y(\theta) & \longrightarrow & Z(\theta) \\
\uparrow & & \downarrow \\
x(n) * y(n) & \longrightarrow & z(n)
\end{array}
$$

When one signal is loooooong. . .

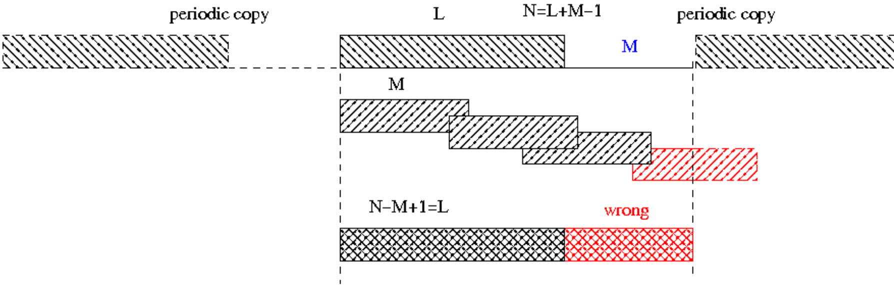
- Cut signal in pieces
- for each piece
  - calculate its FFT
  - multiply by FFT of the other signal
  - calculate the IFFT
- put pieces together (beware of *circular convolution*)
  - overlap-save method
  - overlap-add method

Never use windows with it! $<$ *joke* $>$Use Linux$<$ /*joke* $>$

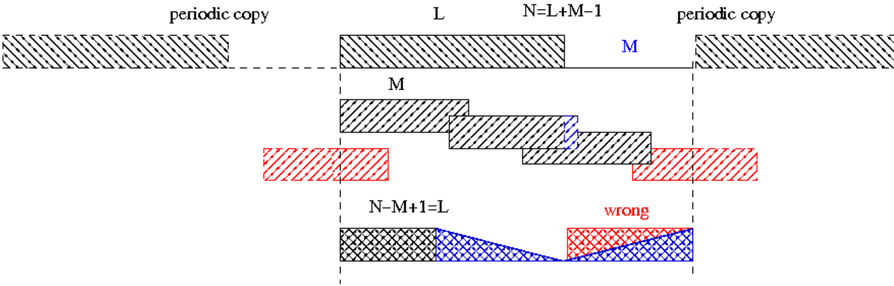
Circular convolution (problem)



Circular convolution (problem solved at some cost)



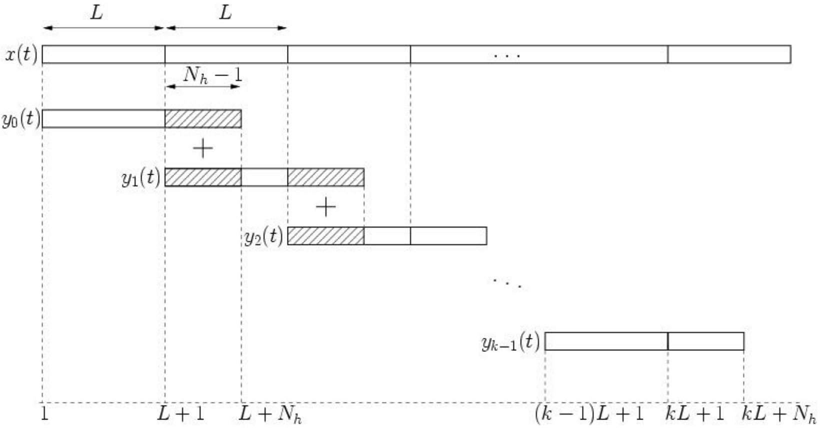
Linear convolution with help of circular



Overlap-save

see the blackboard (;-)

Overlap-add



from Wikipedia