EDISP (Filters 3)
(English) Digital Signal Processing
Digital (Discrete Time) advanced filters - tips & tricks lecture

November 25, 2015

Filters 3

- IIR - impulse/step response invariance
- IIR - optimization methods
- Tips, tricks, examples

Impulse/step response invariance

$$h(n) = T_s h_c(nT_s)$$

$\longrightarrow$ aliasing in frequency domain!

$$H_c(s) = \sum_{k=1}^{N} \frac{A_k}{s - s_k} \text{CT filter in partial fraction exp}$$

$$h_c(t) = u(t) \sum_{k=1}^{N} A_k e^{s_k t}$$

$$h_n = \sum_{k=1}^{N} T_s A_k e^{s_k n T_s} \cdot u(n)$$

$$= \sum_{k=1}^{N} T_s A_k (e^{s_k T_s}) n \cdot u(n)$$

$$H(z) = \sum_{k=1}^{N} \frac{T_s A_k}{1 - (e^{s_k T_s}) z^{-1}}$$

Step invariance - similar way, slightly different results

IIR - CAD (optimization) methods

$\longrightarrow$ Approximate an ideal $A_0(\theta)$

- minimize error on discrete set of frequencies θ_i

$$\varepsilon_{mx} = max_{i\in[1,L]}|A(\theta_i) - A_0(\theta_i)|$$

- easier:

$$\varepsilon_{2p} = \sum_{i=1}^{L} [A(\theta_i) - A_0(\theta_i)]^{2p}$$

with $p >> 1$ ($p = 1$ - mean square; $p \longrightarrow \infty$ - $\varepsilon_{2p} \longrightarrow \varepsilon_{mx}$)

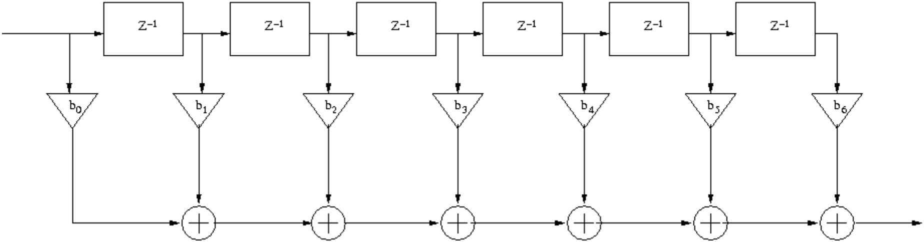
- use well-known gradient optimization method

$$H(z) = H \prod_{n=1}^{n} \frac{1 + a_n z^{-1} + b_n z^{-2}}{1 + c_n z^{-1} + d_n z^{-2}} \quad \text{(biquad sections)}$$

iterative solution of $\frac{\delta\varepsilon_{2p}(\Phi_n)}{\delta\Phi_n} = 0, \;\; \Phi = [a_1,\ b_1,\ c_1,\ d_1,\ a_2,\ \ldots]$ (nonlinear!)

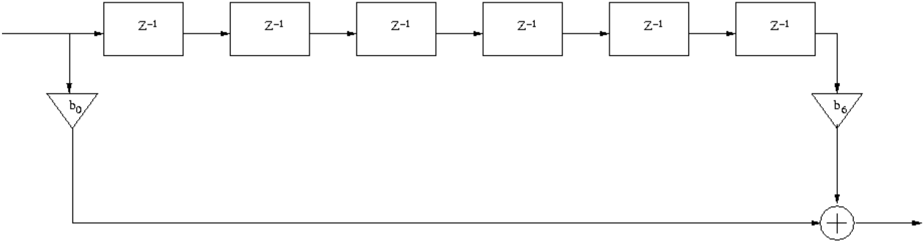
Filtering tricks

Example - comb filter



$a_0 = 1,\ a_K = -1,\ a_{1...K-1} = 0$

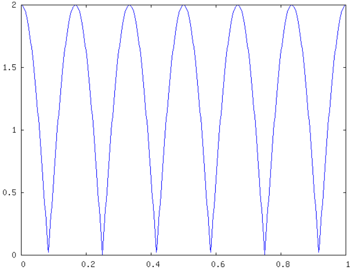
Example - comb filter

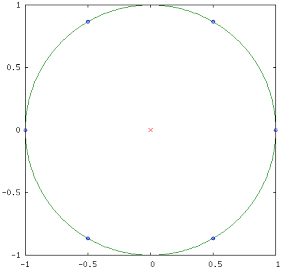


$a_0 = 1,\ a_K = -1,\ a_{1\ldots K-1} = 0$

$H(z) = 1 - z^{-K} \longrightarrow K$ zeros on the unit circle ($K - th$ roots of unity)

$H(\theta) = 1 - e^{-jK\theta} = e^{-jK\theta/2}(e^{+jK\theta/2} - e^{-jK\theta/2}) = e^{-jK\theta/2}(2j\sin K\theta/2)$





Comb filter practical tricks

We want to make a simple LP filter $h(n) = \sum_{k=0}^{K} \delta(n-k)$
(rectangular impulse response, $A(\theta) = \frac{sin(K/2\theta)}{sin(\theta)}$).
We need it for decimating the signal **after** filtering...

- $H(z) = \sum_{k=0}^{K} z^{-k} = \frac{1-z^{-K}}{1-z^{-1}}$ (geometrical series...)
- Cascade integrator $H_1(z) = \frac{1}{1-z^{-1}}$ with a comb filter $H_2(z) = 1 - z^{-K}$
- put decimator by K **between** integrator and comb
  $\longrightarrow$ comb becomes $1 - z^{-1}$ (differentiator)
- warnings (integrator):
  - integrator itself is unstable
  - DC component will always overflow the integrator
  - some tricks with integrator/comb arithmetic (2's complement) could help
- Some correction of characteristics is needed afterwards (LP was simple, not ideal)

Calculating convolution (=filtering) by FFT

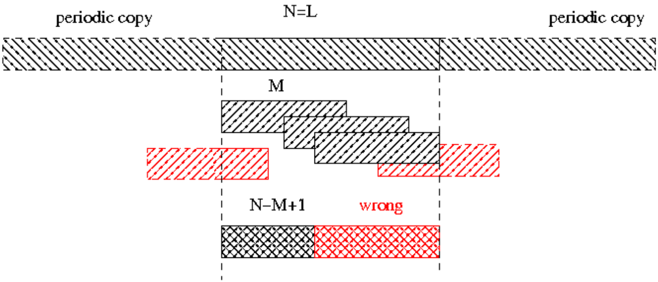
$$\begin{array}{ccc} X(\theta)\cdot Y(\theta) & \longrightarrow & Z(\theta) \\ \uparrow & & \downarrow \\ x(n) * y(n) & \longrightarrow & z(n) \end{array}$$

When one signal is loooooong...

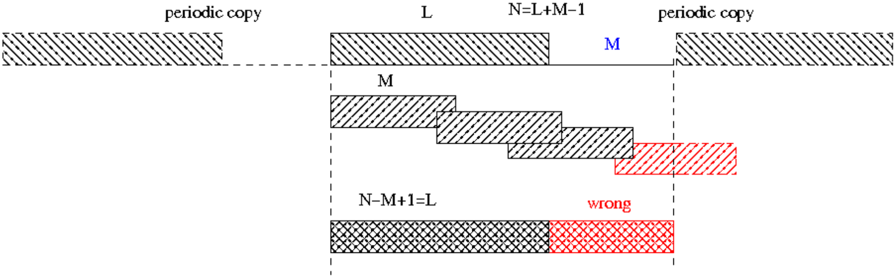
- Cut signal in pieces
- for each piece
  - calculate its FFT
  - multiply by FFT of the other signal
  - calculate the IFFT
- put pieces together (beware of *circular convolution*)
  - overlap-save method
  - overlap-add method

Never use windows with it! $< \textit{joke} >$Use Linux$< /\textit{joke} >$

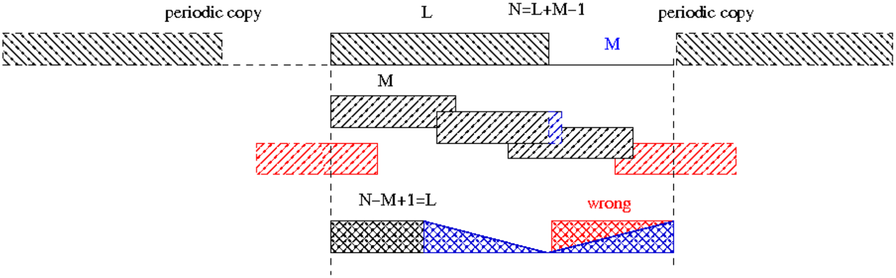
Circular convolution (problem: we want LINEAR conv.!)



Circular convolution (problem solved at some cost)



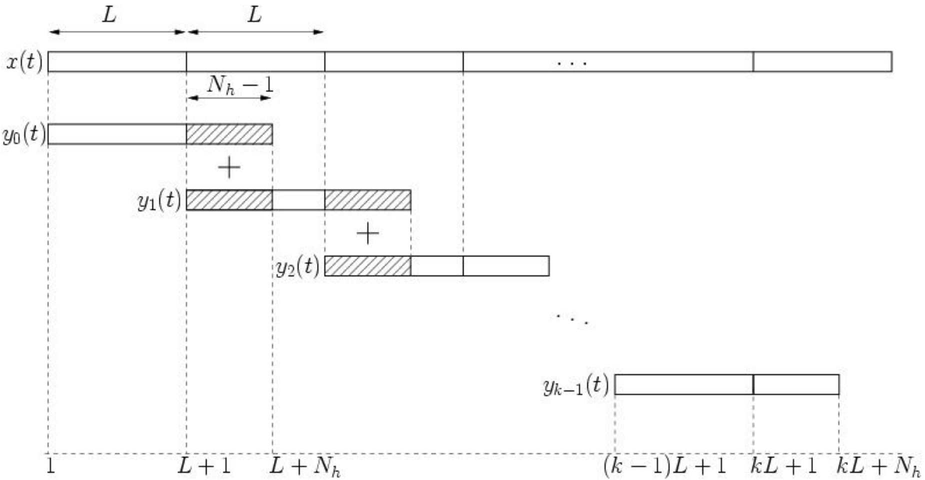
Linear convolution with help of circular



Overlap-save

see the blackboard (;-)
(overlapping blocks on input, bad “tails” of result discarded)

Overlap-add



from Wikipedia