- Circular Convolution
- (Linear) Convolution (Filtering, linear interpolation)
- Correlation

## 1.) Circular Convolution (finite only)

$s = (s_0, \dots, s_{N-1})$

$h = (h_0, \dots, h_{N-1})$

$t = h \circledast s = (t_0, \dots, t_{N-1})$

↑ circular convolution

a.) Definition by sums

$$t_n = \sum_{k=0}^{N-1} h_k \, s_{(n-k) \bmod N} \qquad \left( t_n = \sum_{\substack{k,l \\ k+l \bmod N = n}} h_k s_l \right)$$

b.) Matrix form

$$\begin{pmatrix} t_0 \\ \vdots \\ t_{N-1} \end{pmatrix} = \underbrace{\begin{pmatrix} h_0 & h_{N-1} & \cdots & h_1 \\ h_1 & h_0 & & \vdots \\ \vdots & & \ddots & h_{N-1} \\ h_{N-1} & & h_1 & h_0 \end{pmatrix}}_{\text{circulant matrix } \mathrm{Circ}(h_0, \dots, h_{N-1})} \begin{pmatrix} s_0 \\ \\ \\ s_{N-1} \end{pmatrix}$$

Lecture II:

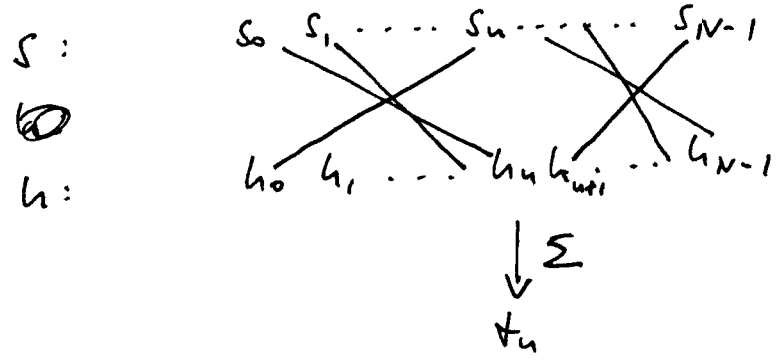
$$\mathrm{Circ}(h_0, \dots, h_{N-1}) = DFT_N^{-1} \cdot D \cdot DFT_N$$

$$D = \mathrm{diag}\left( DFT_N \begin{pmatrix} h_0 \\ \vdots \\ h_{N-1} \end{pmatrix} \right)$$

Remarks:

- circular convolution is $O(N \log N)$
- if $s$ and $h$ are both not known at compile time, 3 DFT's are needed, otherwise 2.
- For small sizes, direct MVM is better (e.g. use blocking). In contrast to generic MVM there is reuse of the matrix entries.

c.) Visual




d.) Polynomials

$s = (s_n) \longrightarrow \sum_{n=0}^{N-1} s_n x^n = S(x)$

$T(x) = H(x)\, S(x) \bmod (x^N - 1)$

} no further explanation but makes Karatsuba possible

2.) (Linear) Convolution

Also: filtering, linear interpolation

We consider only causal FIR (finite impulse response) filters.

$s = (\dots s_{-1}, s_0, s_1, \dots) = (s_n)_{n \in \mathbb{Z}}$

$h = (h_0, \dots, h_{L-1})$

causal: $h$ has no negative indices

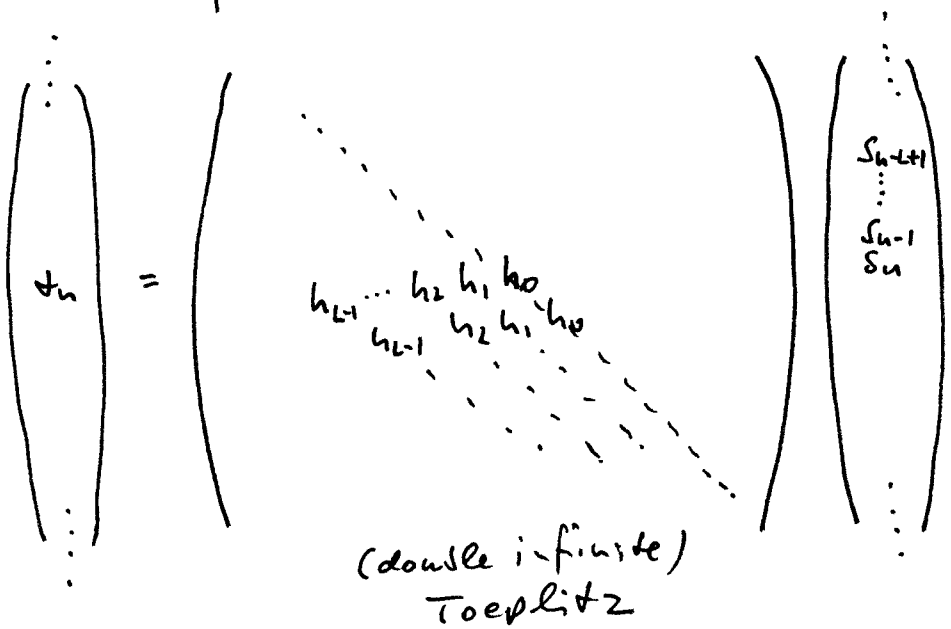
FIR: $h$ has finite length $L$

$t = h * s = (\dots t_{-1}, t_0, t_1, \dots) = (t_n)_{n \in \mathbb{Z}}$

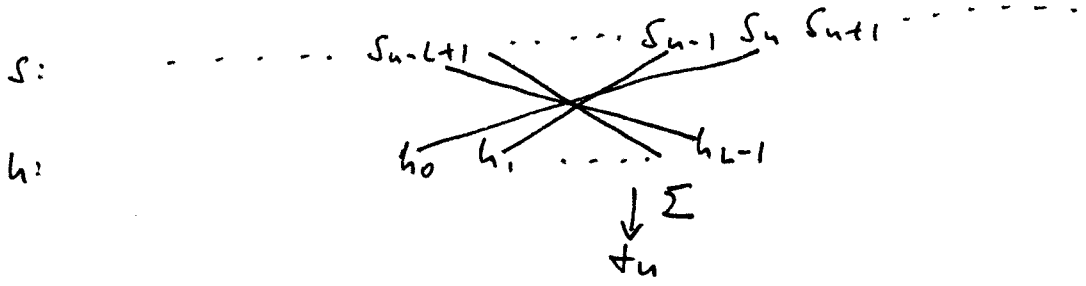
a.) Definition by sums

$t_n = \sum_{k=0}^{L-1} h_k s_{n-k}$ $\qquad \left( t_n = \sum_{\substack{k,\ell \\ k+\ell = n}} h_k s_\ell \right)$

b.) Matrix form




c.) Visual




d.) Polynomials

$$T(x) = H(x)\,S(x)$$

$T(x)$: series; $H(x)$: polynomial; $S(x)$: series

$$(h_0 + h_1 x + h_2 x^2)(s_0 + s_1 x + s_2 x^2)$$
$$= h_0 s_0 + x(s_0 h_1 + h_0 s_1) + x^2(s_0 h_2 + s_1 h_1 + s_2 h_0) + x^3(s_1 h_2 + s_2 h_1) + x^4 s_2 h_2$$

e.) Finite input

$s = (s_0, \ldots, s_{N-1})$

e.g.: zero extension, and compute all non-zero outputs

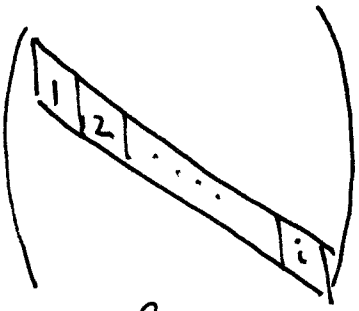
$$\begin{pmatrix} t_0 \\ \vdots \\ t_{N+L-2} \end{pmatrix} = \begin{pmatrix} h_0 & & & \\ \vdots & \ddots & & \\ h_{L-1} & & \ddots & \\ & \ddots & & h_0 \\ & & & \vdots \\ & & & h_{L-1} \end{pmatrix} \begin{pmatrix} s_0 \\ \vdots \\ s_{N-1} \end{pmatrix} \quad \longleftrightarrow \quad T(x) = H(x)\, S(x)$$

$N+L-1 \times N$ Toeplitz

Algorithms:

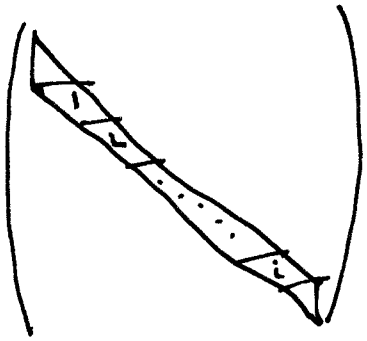
i.) Blocking (as in FFTW, Sparsity)
- does not reduce cost, $O(LN)$

ii.) Overlap-Add




compute in shown order and add results
- does not reduce cost
- loopable
- smaller filters have higher ratio $L/N$

iii.) Overlap-Save



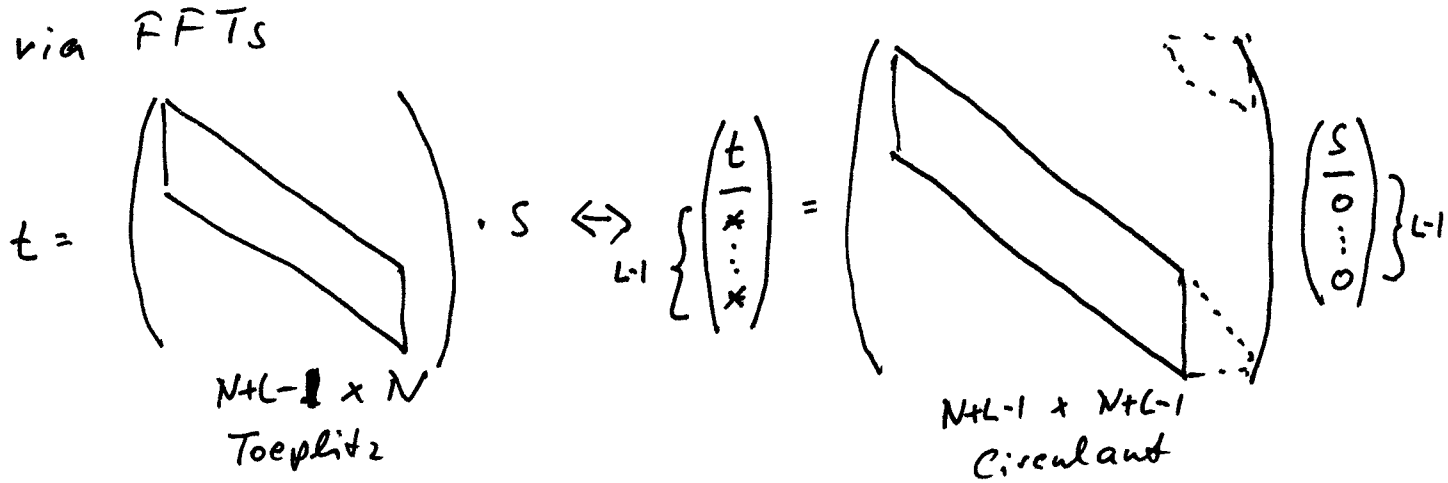

compute in shown order
- does not reduce cost
- requires clean-up code (for this signal extension)

iv.) Karatsuba
- can be written as matrix factorization
- recursively applied: $O(L^{\log_2 3 - 1} N)$

$\approx O(L^{0.585} \cdot N)$

(1. Assignment)

v.) via FFTs




- cost $O(N \log N)$
- matrix can be extended further, e.g., to get a 2-power size circulant
- Circulant at best half dense
- Again: 3 FFTs if $h$ is not known at compile-time

Summary:

small $L/N$: blocking, or overlap add/save to increase $L/N$

medium $L/N$: Karatsuba

large $L/N$: FFT

3.) Correlation

$s = (s_n)_{n \in \mathbb{Z}_L}$

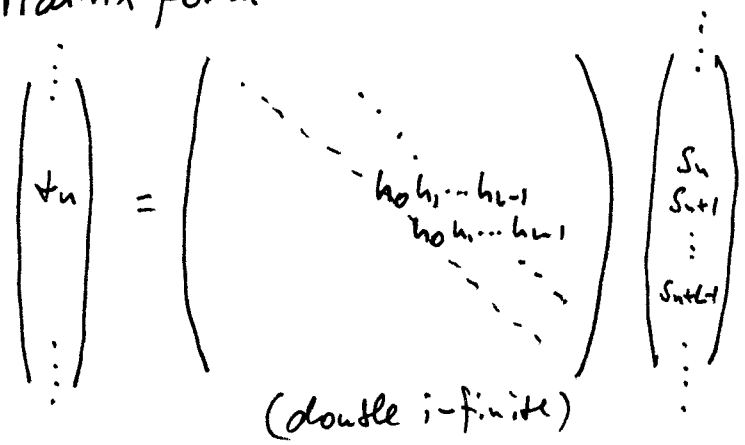
$h = (h_0, \dots, h_{L-1})$

$t = h \square s = (t_n)_{n \in \mathbb{Z}_L}$

a.) Definition by sums

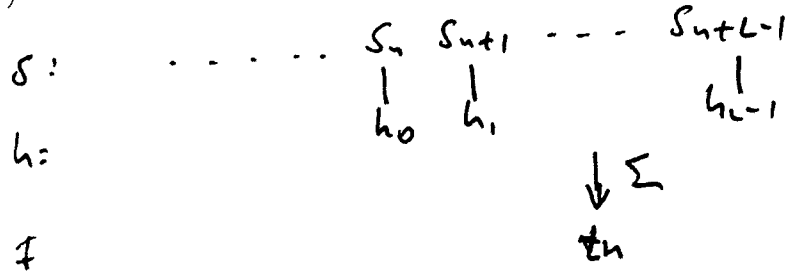
$$t_n = \sum_{k=0}^{L-1} h_k s_{k+n}$$

b.) Matrix form




(double i-finite)

c.) Visual




d.) Polynomial

$$T(x) = x^{-L} \hat{h}(x)\, S(x)$$

$$\hat{h}(x) = h_{L-1} + h_{L-2}x + \ldots + h_0 x^{L-1}$$

e.) Finite Input

$s = (s_0, \ldots, s_{N-1})$, zero extension


$\longleftrightarrow T(x) = x^{-L} \hat{h}(x) S(x)$

$N+L-1 \times N$

Toeplitz

- Same algorithms as for convolution