

- Circular Convolution

- (Linear) Convolution (Filtering, linear interpolation)

- Correlation

1.) Circular Convolution (finite only)

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

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

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

↑
circular
convolution

a.) Definition by sums

$$t_n = \sum_{k=0}^{N-1} h_k s_{(n-k) \bmod N}$$

$$(t_n = \sum_{\substack{k, l \\ k+l \bmod N = n}} h_k s_l)$$

b.) Matrix form

$$\begin{pmatrix} t_0 \\ \vdots \\ t_{N-1} \end{pmatrix} = \begin{pmatrix} h_0 & h_{N-1} & \dots & h_1 \\ h_1 & h_0 & & \vdots \\ \vdots & & \ddots & h_{N-1} \\ h_{N-1} & & & h_0 \end{pmatrix} \begin{pmatrix} s_0 \\ \vdots \\ s_{N-1} \end{pmatrix}$$

circular
matrix $\text{Circ}(h_0, \dots, h_{N-1})$

Lecture 11:

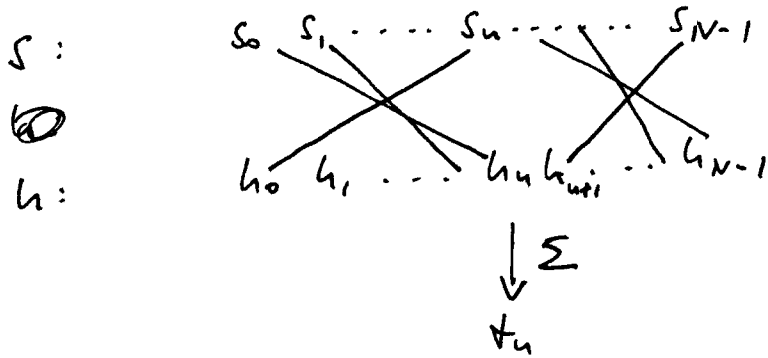
$$\text{Circ}(h_0, \dots, h_{N-1}) = \text{DFT}_N^{-1} \cdot D \cdot \text{DFT}_N$$

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

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 NVT is better (e.g. use blocking). It contrasts to generic NVT where there is reuse of the matrix entries.

c.) Visual



d.) Polynomials

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

$$T(x) = H(x) S(x) \pmod{(x^N - 1)}$$

} 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

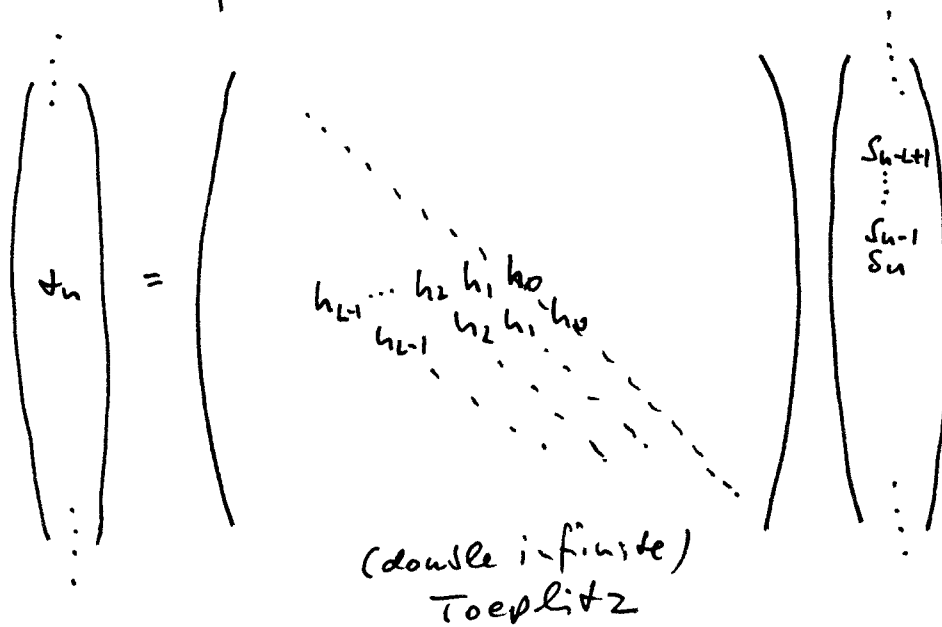
$$t = h * s = (\dots, t_{-1}, t_0, t_1, \dots) = (t_n)_{n \in \mathbb{Z}}$$

a.) Definition by sums

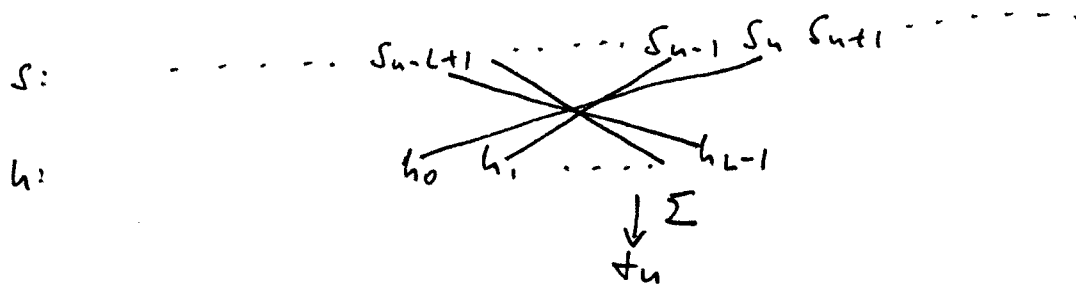
$$t_n = \sum_{k=0}^{L-1} h_k s_{n-k}$$

$$(t_n = \sum_{\substack{k, l \\ k+l=n}} h_k s_l)$$

b.) Matrix form



c.) Visual



d.) Polynomials

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

\uparrow series \uparrow polynomial series \uparrow series

$$\boxed{(h_0 + h_1 x + h_2 x^2) (s_0 + s_1 x + s_2 x^2)}$$

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

e.) Finite input

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

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

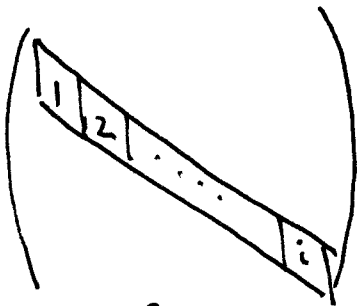
$$\begin{pmatrix} y_0 \\ \vdots \\ y_{N+L-1} \end{pmatrix} = \begin{pmatrix} h_0 & & & & & & \\ & \ddots & & & & & \\ & & h_{L-1} & & & & \\ & & & \ddots & & & \\ & & & & h_0 & & \\ & & & & & \ddots & \\ & & & & & & h_{L-1} \end{pmatrix} \begin{pmatrix} s_0 \\ \vdots \\ s_{N-1} \end{pmatrix} \Leftrightarrow T(x) = H(x) S(x)$$

$N+L-1 \times N$
Toeplitz

Algorithms:

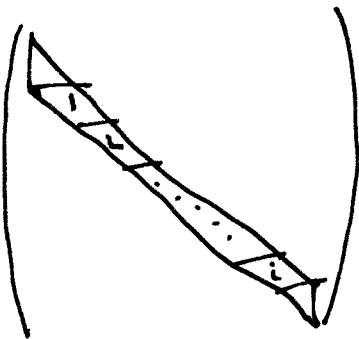
i.) Blocking (as in FFT, Sparsity)
- does not reduce cost, $O(LN)$

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



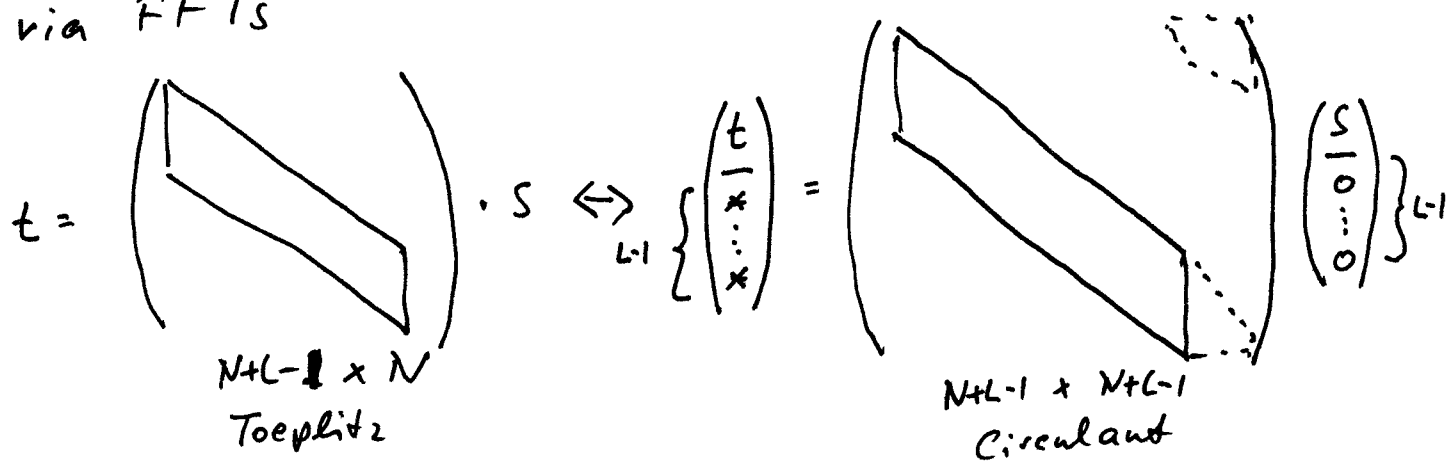
compute in shown order
- does not reduce cost
- requires clean-up code (for this signal extension)

iv.) Karatsuba

- can be written as matrix factorization
- recursively applied: $O(L^{\log_2 3} N)$
 $\approx O(L^{0.585} N)$

(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

$$\begin{pmatrix} \vdots \\ t_n \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ \vdots & h_0 h_1 \dots h_{L-1} & \vdots & \vdots \\ \vdots & h_0 h_1 \dots h_{L-1} & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} \vdots \\ s_n \\ s_{n+1} \\ \vdots \\ s_{n+L-1} \\ \vdots \end{pmatrix}$$

(double i-finite)

c.) Visual

$$\begin{array}{ccccccc} s: & & \dots & s_n & s_{n+1} & \dots & s_{n+L-1} \\ & & & | & | & & | \\ h: & & & h_0 & h_1 & & h_{L-1} \\ & & & & & \downarrow \Sigma & \\ t: & & & & & & t_n \end{array}$$

d.) Polynomial

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

$$\hat{H}(x) = \cancel{h_{L-1}} + h_{L-2}x + \dots + h_0 x^{L-1}$$

e.) Finite input

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

$$\begin{pmatrix} t_{-L+1} \\ \vdots \\ t_0 \\ \vdots \\ t_{N-1} \end{pmatrix} = \begin{pmatrix} h_{L-1} & & & & \\ & \ddots & & & \\ & & h_0 & & \\ & & & \ddots & \\ & & & & h_{L-1} \\ & & & & & \ddots \\ & & & & & & h_0 \end{pmatrix} \begin{pmatrix} s_0 \\ \vdots \\ s_{N-1} \end{pmatrix}$$

$$\Leftrightarrow T(x) = x^{-L} \hat{H}(x) S(x)$$

$(N+L-1) \times N$
Toeplitz

- Same algorithms as for convolution