3F1 Signals and Systems: Handout 13

Applications of the DFT

Jossy Sayir

Michaelmas Term 2025

An example where you want a circular convolution

- consider independent discrete random variables $X_1, X_2, \ldots, X_n$ defined over an alphabet $\mathcal{X} = \{0, 1, 2, \ldots, p-1\}$ where p is a positive integer
- consider the random variable

$$Z = X_1 \oplus X_2 \oplus \ldots \oplus X_n$$

where $\oplus$ is the "modulo p" addition, i.e.,

$$X \oplus Y = R_p(X + Y)$$

- what is the probability distribution $P_Z(k)$ for $k = 0, 1, \ldots, p-1$?

Cyclic convolution example (continued)

- consider for example modulo $p = 3$ addition and $n = 2$ random variables, i.e., $Z = X_1 \oplus X_2$ and both variables are defined over the alphabet $\{0, 1, 2\}$
- $Z = 0$ if $(X_1, X_2) = (0,0), (1,2)$ or $(2,1)$, i.e.,

$$\begin{cases} 0 \oplus 0 = 0, \\ 1 \oplus 2 = 0, \text{ and} \\ 2 \oplus 1 = 0 \end{cases}$$

- $Z = 1$ if $(X_1, X_2) = (0,1), (1,0)$ or $(2,2)$
- $Z = 2$ if $(X_1, X_2) = (0,2), (1,1)$ or $(2,0)$
- hence, if we consider the probability distributions for X_1, X_2 and Z as vectors $\vec{P}_{X_1}, \vec{P}_{X_2}$ and $\vec{P}_Z$, respectively, we observe that

$$\vec{P}_Z = \vec{P}_{X_1} \circledast \vec{P}_{X_2},$$

i.e., $\vec{P}_Z$ is the circular convolution of $\vec{P}_{X_1}$ and $\vec{P}_{X_2}$

Cyclic convolution example (continued)

- this is also the case for all n and p, i.e., $\vec{P}_Z$ will be the circular convolution of all probability vectors

$$\vec{P}_Z = \vec{P}_{X_1} \circledast \vec{P}_{X_2} \circledast \ldots \circledast \vec{P}_{X_n}$$

- this multiple circular convolution is best evaluated using an FFT $\rightarrow$ component-wise multiplication $\rightarrow$ inverse FFT
- for the special case $p = 2$, i.e., $X_1, X_2, \ldots, X_n$ are binary random variables with probability distributions $\vec{P}_{X_i} = \begin{bmatrix} 1 - p_i \\ p_i \end{bmatrix}$, we get the following steps:
  1. 2-point DFT of $\vec{P}_{X_i}$: $\vec{Q}_{X_i} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 - p_i \\ p_i \end{bmatrix} = \begin{bmatrix} 1 \\ 1 - 2p_i \end{bmatrix}$
  2. component-wise multiplication: $\vec{Q}_Z = \begin{bmatrix} 1 \\ \prod_{i=0}^{n} (1 - 2p_i) \end{bmatrix}$
  3. inverse 2-point DFT:

$$\vec{P}_Z = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \vec{Q}_z = \begin{bmatrix} \frac{1}{2} \left(1 + \prod_{i=0}^{n} (1 - 2p_i)\right) \\ \frac{1}{2} \left(1 - \prod_{i=0}^{n} (1 - 2p_i)\right) \end{bmatrix}$$

- those taking 3F7 may recognise this expression...

An example where you want a linear convolution

- for sequences $\{a_k\}_{k\geq 0}$ and $\{b_k\}_{k\geq 0}$, the discrete convolution $a \star b$ is called "linear" convolution in this context to distinguish it from the circular convolution
- the linear convolution of arbitrary sequences cannot in general be computed via an FFT
- however, consider the case of an FIR filter of length $L+1$

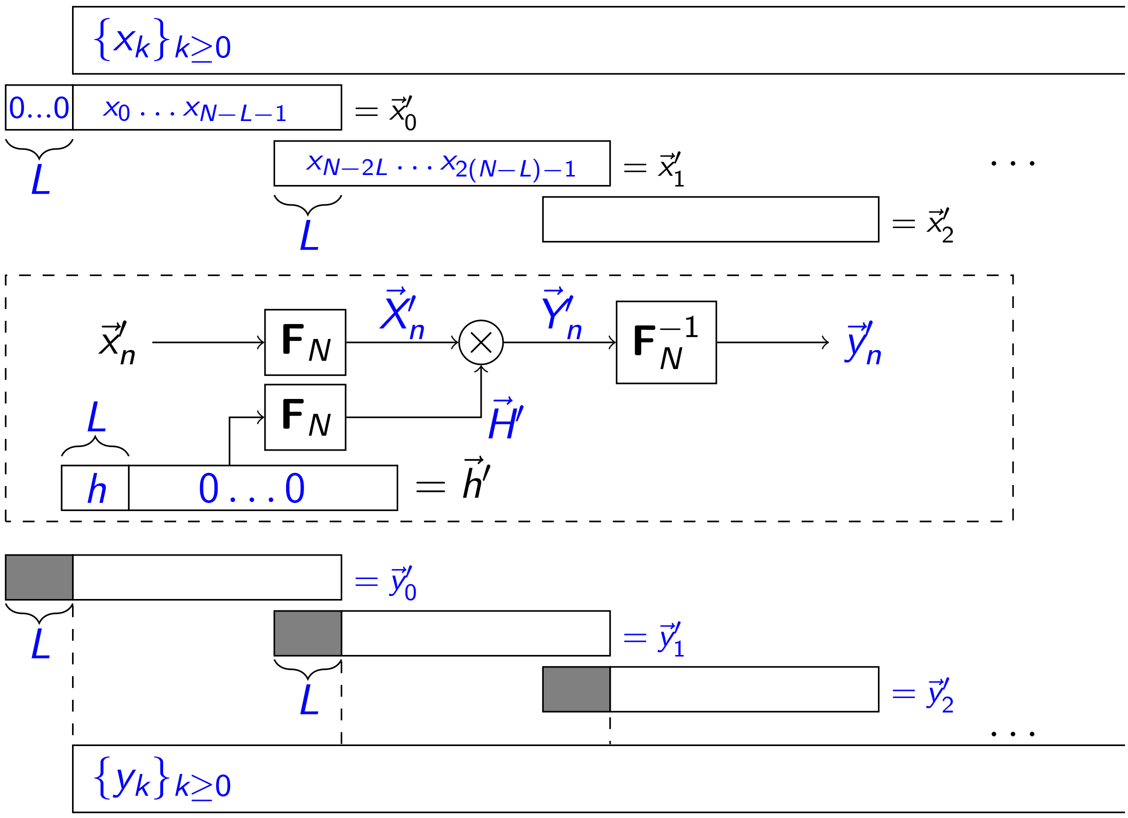
$$\{y_k\}_{k\geq 0} = \{h_k\}_{0\leq k\leq L} \star \{x_k\}_{k\geq 0} \text{ , i.e., } y_k = \sum_{\ell=0}^{L} h_\ell x_{k-\ell}$$

- if you take a block $\vec{x} = x_0, \ldots, x_{N-1}$ and evaluate the circular convolution $\vec{y}_c = \vec{h} \circledast \vec{x}$,
  - the first L outputs differ from the linear convolution because they depend on the last L entries of the input block, whereas in the linear convolution they depend on the entries $x_{-L}, \ldots, x_{-1}$ which are assumed to be zero
  - assuming that $N \gg L$, many outputs beyond the first L will be identical to the linear convolution!!

Overlap and save method: using length N FFT to do linear convolutions with an FIR of length $L+1 \ll N$

1. take FFT $\vec{H}'$ of $\vec{h}' = [\vec{h}, 0, 0, \ldots, 0]$, i.e., $\vec{h}$ and N–L zeros
2. take FFT $\vec{X}'_0$ of block $\vec{x}'_0 = [0, \ldots, 0, x_0, x_1, \ldots, x_{N-L-1}]$, i.e., L zeros followed by the first $N-L$ terms of $\{x_k\}_{k\geq 0}$
3. multiply $\vec{X}'_0$ componentwise by $\vec{H}'$ to obtain $\vec{Y}'_0$
4. take the inverse FFT $\vec{y}'_0$ of $\vec{Y}'_0$ and discard the first L terms of $\vec{y}'_0$ to obtain $\vec{y}_0$, giving the first $N-L$ terms of the output sequence $\{y\}_{k\geq 0}$. Set $n=1$.
5. take FFT $\vec{X}'_n$ of $\vec{x}'_n = [x_{n(N-L)-L}, \ldots, x_{n(N-L)+N-L-1}]^T$, i.e., the last L symbols of $\vec{x}'_{n-1}$ followed by the next $N-L$ symbols of $\{x_k\}_{k\geq 0}$
6. multiply $\vec{X}'_n$ componentwise by $\vec{H}'$ to obtain $\vec{Y}'_n$
7. take the inverse FFT $\vec{y}'_n$ of $\vec{Y}'_n$ and discard the first L terms of $\vec{y}'_n$ to obtain $\vec{y}_n$, giving the next $N-L$ terms of the output sequence $\{y\}_{k\geq 0}$, then increment n and go back to step 5

Illustration of the overlap and save method

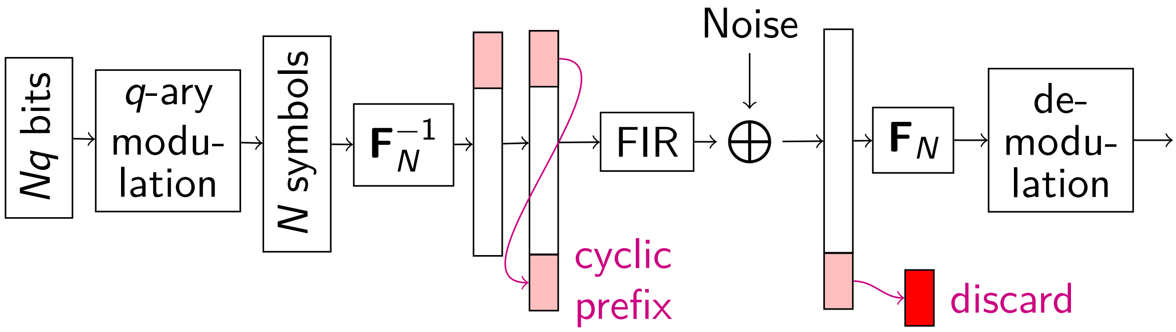


Tricking the "real world" into doing circular convolutions

- in communications, the Additive White Gaussian Noise (AWGN) channel we studied in IB Paper 6 is a simplification
- in transmission over cable or wireless, "echo" leads to a convolution of the input signal with the channel response
- communicating digital data over a convolving channel is hard: every transmitted digital symbol is "smeared" over several outputs. This is known as "inter-symbol interference" (ISI)
- we could use the FFT to communicate in the frequency domain, as follows
  1. prepare the digital signal (as learned in IB Paper 6) in the frequency domain, by mapping bits to N complex constellations
  2. take an inverse FFT, and transmit the resulting time-domain vector
  3. at the receiver, taken an FFT and process the frequency-domain vector
- time domain convolution (ISI) becomes frequency domain multiplication $\implies$ no ISI

Orthogonal Frequency-Domain Modulation (OFDM)

- the idea on the previous slide would work if the channel did cyclic convolutions, but the channel does linear convolutions
- we can "trick" a channel of length L to do cyclic convolutions by copying L or more symbols from the end of each block to precede the block. This is known as a "cyclic prefix"
- When the channel reaches the actual beginning of a block, its FIR will be loaded with the same values as it would if it were doing a cyclic convolution (the last L symbols of the block)



- more about this in 3F4

Interpolation and decimation

- signal processing systems often require a change of sampling frequency f_s so processing can be done at a higher f_s (e.g. for better accuracy) or lower f_s (e.g. to reduce computations)
- upping to a higher frequency is called "interpolation":
    1. zero-stuffing: insert k zeros between every two samples
    2. filtering: low-pass and scale the resulting signal
- The zero-stuffing step generates a signal with a repeated spectrum as it is the digital equivalent to the"train of impulses" we analysed when discussing the sampling theorem.
- Hence, the filtering step (which those less versed in signal processing call "interpolation" as they see it as interpolating the signal between the non-zero samples) is a low-pass filter!
- reducing to a lower frequency is called "decimation". Essentially, you just drop samples and keep only one in every k samples. Beware of aliasing! This can only be done when the bandwidth is less than the $f_s^{\text{new}}/2$.

Interpolation and decimation via the FFT

- gives a "graceful" transition between frequencies
- f_s^{new} does not need to be a multiple or divisor of f_s^{old}
- Pick N and N' such that $N'/N = f_s^{new}/f_s^{old}$
  1. take the N point FFT $\vec{X}$ of a block $\vec{x}$ of inputs
  2. next,
     - if upping f_s, insert zeros on either side of $X_{N/2}$ (if N is even) or between $X_{\lceil (N-1)/2 \rceil}$ and $X_{\lfloor (N+1)/2 \rfloor}$ (if N is odd)
     - if downing f_s, delete components symmetrically around $N/2$
  3. take the N' point inverse FFT
- this method is better than interpolation/decimation but
  - it has the ususal caveat of a "cut" in the FFT domain being a multiplication with a rectangular window (there are window-based approaches to compensate for this)
  - larger N will in general give better results
  - numerical imprecisions in the FFT may result in a complex signal: check that the imaginary parts are negligible, then retain only the real part

Time/Frequency analysis

- say you recorded 10 minutes of music sampled at 48 kHz
- you obtained 29 million samples
- it is possible to take the FFT of such a vector ($N \log N$ complexity, approx. 700 million operations or about 1-2 seconds on a modern computer)
- such an FFT would say nothing about the music signal
- the "interesting" frequency features in music are on the order of 1 second (for adagio or jazz balads) to 0.1 seconds (for prestissimo or jazz bebop)
- a time-frequency spectrogram displays the evolution of the spectrum against time
- overlapping windowed FFTs are taken at regular time intervals and the resulting magnitude is displayed in colour scale