

3F1 Signals and Systems: Handout 12

The Fast Fourier Transform (FFT)

Jossy Sayir

Michaelmas Term 2025

1 / 12

A bit of history...

- ▶ The 1965 paper by Cooley and Tukey on the Fast Fourier Transform (FFT) is considered by many to be one of the foremost technological advances of the 20th century
- ▶ The principle for the FFT had been discovered by Carl Friedrich Gauss in 1805 but had lain forgotten
- ▶ The FFT led to a huge advance in our signal processing abilities. The development of “spectrum analysers” that allowed one to view the spectrum of signals was the first hardware application of the FFT
- ▶ All of signal processing nowadays and many numerical algorithms depend on the FFT. For example, multiplication of long integers can be framed as a convolution and is best done via the FFT
- ▶ Convolutions (filtering, autocorrelations, etc.) are almost universally implemented via FFTs

2 / 12

The Fast Fourier Transform (FFT)

In a nutshell, FFT algorithms exploit properties of the DFT matrix \mathbf{F} to perform the DFT $\mathbf{F}\underline{x}$ in $O(N \log N)$ operations instead of the $O(N^2)$ normally needed for a matrix vector multiplication.

3 / 12

Factorising the FFT length N

- ▶ this is the basis for a large family of FFT algorithms, including Cooley Tukey (or “radix 2”).
- ▶ the method applies to all but prime FFT lengths N
- ▶ we assume that the FFT length N can be factorised into two integers M and L , i.e.,

$$N = ML$$

- ▶ M and L can possibly be factorised further but we will only consider one factorisation for now. If further factorisation is possible, the method can be applied recursively

⁰The presentation of FFT algorithms in my lecture is based on the book “Theory and Application of Digital Signal Processing” by Rabiner and Gold.

4 / 12

Arranging vectors into an $L \times M$ array

- ▶ the DFT maps an input vector \underline{x} to an output vector \underline{X} , both of length N and indexed from 0 to $N - 1$
- ▶ re-consider a vector (\underline{x} or \underline{X}) as an $L \times M$ array with L rows and M columns, i.e.,

$$\begin{array}{cccccc}
 & & & & & M \\
 & & & & & \longleftarrow \\
 & x_0 & x_1 & x_2 & \cdots & x_{M-1} \\
 & x_M & x_{M+1} & x_{M+2} & \cdots & x_{2M-1} \\
 & \vdots & \vdots & \vdots & \ddots & \vdots \\
 & x_{(L-1)M} & \cdots & & \cdots & x_{ML-1} \\
 & & & & & \downarrow L
 \end{array}$$

- ▶ we use the “row major form”: consecutive entries in a row are consecutive in \underline{x} while consecutive entries in a column are M apart in \underline{x}

5 / 12

Indexing into the $L \times M$ array

$$\begin{array}{cccccc}
 & & & & & M \\
 & & & & & \longleftarrow \\
 & x_0 & x_1 & x_2 & \cdots & x_{M-1} \\
 & x_M & x_{M+1} & x_{M+2} & \cdots & x_{2M-1} \\
 & \vdots & \vdots & \vdots & \ddots & \vdots \\
 & x_{(L-1)M} & \cdots & & \cdots & x_{ML-1} \\
 & & & & & \downarrow L
 \end{array}$$

- ▶ we've been indexing time domain vectors \underline{x} with the index k and frequency domain vectors with the index n in this lecture
- ▶ we now need a 2-dimensional indexing into the array

$$\begin{cases} k = \ell M + s \\ n = mL + r \end{cases}$$

- ▶ ℓ and r indicate the row number
- ▶ m and s indicate the column number

6 / 12

Re-arranging the DFT

Using

$$\begin{cases} k = \ell M + s \\ n = mL + r \end{cases}$$

we re-arrange the DFT expression

$$\begin{aligned} X_n = X_{mL+r} &= \sum_{k=0}^{N-1} x_k W_N^{kn} = \sum_{s=0}^{M-1} \sum_{\ell=0}^{L-1} x_{\ell M+s} W_N^{(\ell M+s)(mL+r)} \\ &= \sum_{s=0}^{M-1} \sum_{\ell=0}^{L-1} x_{\ell M+s} W_N^{\ell m L M} W_N^{s m L} W_N^{r \ell M} W_N^{s r} \\ &= \sum_{s=0}^{M-1} W_M^{s m} W_N^{s r} \sum_{\ell=0}^{L-1} x_{\ell M+s} W_L^{r \ell} \end{aligned}$$

where we note the following

$$\begin{cases} W_N^{\ell m L M} = W_N^{\ell m N} = 1 \\ W_N^{s m L} = W_M^{s m} \\ W_N^{r \ell M} = W_L^{r \ell} \end{cases}$$

7 / 12

The 2-stage Fast Fourier Transform

The expression we have obtained divides the N point DFT into 3 steps

$$X_{mL+r} = \underbrace{\sum_{s=0}^{M-1} W_M^{s m} \underbrace{W_N^{s r}}_{\text{twiddle factor}} \underbrace{\sum_{\ell=0}^{L-1} x_{\ell M+s} W_L^{r \ell}}_{L \text{ point DFT}}}_{M \text{ point DFT}}$$

1. Take the L point DFT of every column
2. Multiply every element by its twiddle factor W_N^{sr}
3. Take the M point DFT of every row

8 / 12

Complexity of the 2-stage FFT

- ▶ let's count multiplications (a more precise count would also include additions)
- ▶ assume for now that every DFT is taken by matrix multiplication. Multiplying a $k \times k$ matrix by a vector of length k requires k^2 multiplications
- ▶ The regular N point DFT requires: **N^2 multiplications**
- ▶ The FFT requires:
 1. M DFTs of columns of length L , i.e., **ML^2 multiplications**
 2. **$N = ML$ twiddle factor multiplications**
 3. L DFTs of rows of length M , i.e., **LM^2 multiplications**
- ▶ Total: **$ML + ML^2 + LM^2 = N(1 + M + L)$ multiplications**
- ▶ For example, if $N = 100$ and $M = L = 10$, the regular DFT requires $N^2 = 10,000$ multiplications whereas the 2-stage FFT requires $100 \times 21 = 2,100$ multiplications

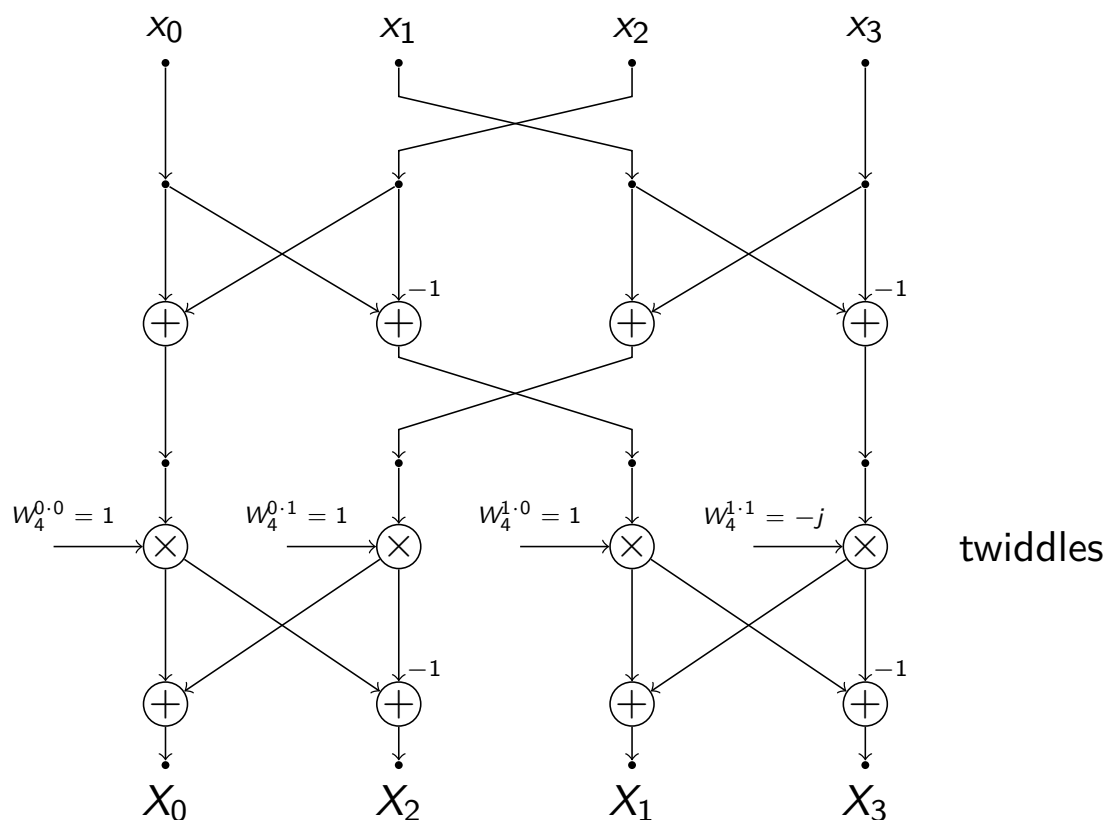
9/12

Multi-stage and radix-2 FFT

- ▶ the algorithm can be applied recursively, e.g., each 10 point DFT in our 10×10 FFT can be divided into a 2×5 FFT
- ▶ the most popular FFT is the $N = 2^n$ “radix-2” or Cooley Tukey FFT: it can be divided into a $L = 2$ point DFT and a $M = 2^{n-1}$ point DFT, and the M point DFT can in turn be divided, etc.
- ▶ the first sub-division results in 2^{n-1} two-point DFTs, 2^n twiddle-factor multiplications and two 2^{n-1} point DFTs
- ▶ the 2-point DFTs don't count in our “multiplication only” accounting as they only require addition and subtraction
- ▶ every further sub-division adds $N = 2^n$ twiddle factor multiplications and otherwise only additions and subtractions
- ▶ we can divide N a total of $\log_2 N = \log_2 2^n = n$ times.
- ▶ the complexity of the radix-2 FFT is hence $nN = N \log_2 N$
- ▶ for example, for $N = 2^{10} = 1024$, the regular DFT requires $N^2 = 2^{20} \approx 10^6$ multiplications and the radix-2 FFT requires $N \log_2 N = 2^{10} \times 10 \approx 10^4$ multiplications, 100 times less!!

10/12

Example: $2^2 = 4$ point FFT



11 / 12

Other FFTs

- ▶ there are countless other FFT algorithms that work for lengths N with specific properties
- ▶ for example, the Good Thomas or “prime factor” algorithm works when N can be decomposed into mutually prime numbers, i.e., numbers that don't share any prime factors. This algorithm is based on the “Chinese Remainder Theorem” decomposition of indices from 0 to $N - 1$ and requires no twiddle factors. It doesn't work for $N = 2^n$ as there is no way to decompose such an N into mutually prime factors
- ▶ the Good Thomas FFT is the transform of choice for FFTs over finite fields of order 2^m where the allowable FFT lengths are never a power of 2 (e.g. Reed Solomon decoding, taught in 4F5)

12 / 12