Fast Fourier Transform (FFT)

Borbala Hunyadi

Delft University of Technology, The Netherlands

TU Delft

Recap: Discrete Fourier Transform

Definition

The Discrete Fourier Transform (DFT) of a sequence $x[n]$ is

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi \frac{kn}{N}}, \text{ for } 0 \leq k \leq N-1$$

Applications:

- Filtering
- Spectral analysis

Recap: Discrete Fourier Transform

Definition

The Discrete Fourier Transform (DFT) of a sequence $x[n]$ is

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi \frac{kn}{N}}, \text{ for } 0 \leq k \leq N-1$$

Applications:

- Filtering
- Spectral analysis

Is DFT efficient enough?

Computational complexity of the DFT

Let's define $W_N = e^{-j2\pi/N}$! Then the DFT can be expressed as:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \text{ for } 0 \leq k \leq N-1$$

Steps of the direct computation algorithm:

Stage 1:

Compute and store the values
$W_N^l = e^{-j2\pi l/N} = cos(2\pi l/N) - j \cdot sin(2\pi l/N)$

Stage 2:

for $k = 0 : N-1$
    $X[k] \leftarrow x[0]$
    for $n = 1 : N-1$
        $l = (kn)_N$
        $X[k] \leftarrow X[k] + x[n] W_N^l$
    end
end

Computational complexity of the DFT

Let's define $W_N = e^{-j2\pi/N}$! Then the DFT can be expressed as:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \text{ for } 0 \leq k \leq N-1$$

Steps of the direct computation algorithm:

Stage 1:

Compute and store the values
$W_N^l = e^{-j2\pi l/N} = cos(2\pi l/N) - j \cdot sin(2\pi l/N)$

N evaluations of sin and cos functions

Stage 2:

for $k = 0 : N-1$
  $X[k] \leftarrow x[0]$
  for $n = 1 : N-1$
    $l = (kn)_N$
    $X[k] \leftarrow X[k] + x[n] W_N^l$
  end
end

Computational complexity of the DFT

Let's define $W_N = e^{-j2\pi/N}$! Then the DFT can be expressed as:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \text{ for } 0 \le k \le N-1$$

Steps of the direct computation algorithm:

Stage 1:

Compute and store the values
$W_N^l = e^{-j2\pi l/N} = cos(2\pi l/N) - j \cdot sin(2\pi l/N)$

N evaluations of sin and cos functions

Stage 2:

for $k = 0 : N-1$
  $X[k] \leftarrow x[0]$
  for $n = 1 : N-1$
    $l = (kn)_N$
    $X[k] \leftarrow X[k] + x[n] W_N^l$
  end
end

N^2 complex multiplications and $N(N-1)$ complex additions

Computational complexity of the DFT

Let's define $W_N = e^{-j2\pi/N}$! Then the DFT can be expressed as:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \text{ for } 0 \leq k \leq N-1$$

Steps of the direct computation algorithm:

Stage 1:

Compute and store the values

$W_N^l = e^{-j2\pi l/N} = cos(2\pi l/N) - j \cdot sin(2\pi l/N)$

N evaluations of sin and cos functions

Stage 2:

for $k = 0 : N-1$
    $X[k] \leftarrow x[0]$
    for $n = 1 : N-1$
        $l = (kn)_N$
        $X[k] \leftarrow X[k] + x[n] W_N^l$
    end
end

N^2 complex multiplications and $N(N-1)$ complex additions

+ overhead: addressing, indexing...

Computational complexity of the DFT

Let's define $W_N = e^{-j2\pi/N}$! Then the DFT can be expressed as:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \text{ for } 0 \le k \le N-1$$

Steps of the direct computation algorithm:

$O(N^2)$ - **very costly**

Stage 1:

Compute and store the values

$W_N^l = e^{-j2\pi l/N} = cos(2\pi l/N) - j \cdot sin(2\pi l/N)$

N evaluations of sin and cos functions

Stage 2:

for $k = 0 : N-1$
  $X[k] \leftarrow x[0]$
  for $n = 1 : N-1$
    $l = (kn)_N$
    $X[k] \leftarrow X[k] + x[n] W_N^l$
  end
end

N^2 complex multiplications and $N(N-1)$ complex additions

+ overhead: addressing, indexing...

Fast Fourier Transform

- A family of computationally efficient algorithms to compute DFT
- Not a new transform!

Different working principles:

1. Divide and conquer approach
2. DFT as convolution: linear filtering approach

Fast Fourier Transform

- A family of computationally efficient algorithms to compute DFT
- Not a new transform!

Different working principles:

1. **Divide and conquer approach**
2. DFT as convolution: linear filtering approach

Divide and conquer FFT

Essential ingredients:

Divide and conquer FFT

Essential ingredients:

- Break down the N-point DFT to a cascade of smaller-size DFTs

Divide and conquer FFT

Essential ingredients:

- Break down the N-point DFT to a cascade of smaller-size DFTs

  Guessing game: I am thinking of a random number between 1 and 16. Can you guess which number is it?

Divide and conquer FFT

Essential ingredients:

- Break down the N-point DFT to a cascade of smaller-size DFTs

  Guessing game: I am thinking of a random number between 1 and 16. Can you guess which number is it?

- Exploit symmetries

Divide and conquer FFT

Essential ingredients:

- Break down the N-point DFT to a cascade of smaller-size DFTs

  Guessing game: I am thinking of a random number between 1 and 16. Can you guess which number is it?

- Exploit symmetries

$$W_N^k = W_N^{k+N}$$
$$W_N^{Lk} = W_{N/L}^k$$

Radix-2 FFT

Radix-2 FFT is the most important divide and conquer type FFT algorithm. It can be used if $N = 2^r$. This can always be achieved using zero-padding the sequence.

Decimation in time (DIT) solution:

- Divide the N long sequence $x[n]$ to 2 N/2 long sequences
- The N-point DFT of $x[n]$ can be computed by properly combining the 2 N/2-point DFTs
- Repeat the subdivision until the sequences are 2 samples long (2-point DFT)

Radix-2 FFT

N-point DFT ($N = 2^r$) solved by a cascade of r stages:

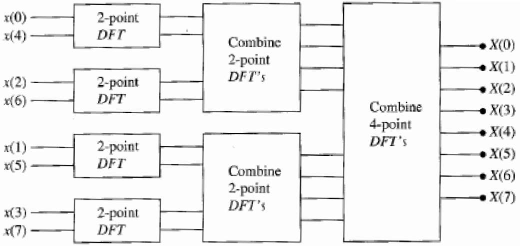


Figure 8.1.5 Three stages in the computation of an $N = 8$-point DFT.

The following slides show the exact operations performed during these stages.

2-point DFT: How to compute in a simple way?

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \text{ for } 0 \leq k \leq N-1$$

Let us write out the expression for both DFT coefficients:

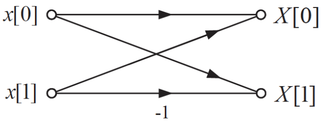
$$X[0] = x[0] + x[1] W_2^0 = x[0] + x[1]$$
$$X[1] = x[0] + x[1] W_2^1 = x[0] - x[1]$$

2-point DFT: How to compute in a simple way?

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \text{ for } 0 \leq k \leq N-1$$

Let us write out the expression for both DFT coefficients:

$$X[0] = x[0] + x[1] W_2^0 = x[0] + x[1]$$
$$X[1] = x[0] + x[1] W_2^1 = x[0] - x[1]$$

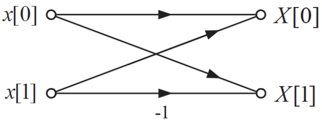


2-point DFT: How to compute in a simple way?

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \text{ for } 0 \leq k \leq N-1$$

Let us write out the expression for both DFT coefficients:

$$X[0] = x[0] + x[1]W_2^0 = x[0] + x[1]$$
$$X[1] = x[0] + x[1]W_2^1 = x[0] - x[1]$$



The 2-point DFT coefficients are given by taking the sum and the difference of the samples. This simple operation is represented by the so-called butterfly diagram.

Combine two 2-point DFTs into a 4-point DFT

$$X[k] = \sum_{n=0}^{3} x[n] W_4^{kn}, \text{ for } 0 \leq k \leq 3$$

$$X[k] = x[0] + x[1] W_4^{k} + x[2] W_4^{2k} + x[3] W_4^{3k}$$

Combine two 2-point DFTs into a 4-point DFT

$$X[k] = \sum_{n=0}^{3} x[n] W_4^{kn}, \text{ for } 0 \leq k \leq 3$$

$$\begin{aligned} X[k] &= x[0] + x[1] W_4^{k} + x[2] W_4^{2k} + x[3] W_4^{3k} \\ &= (x[0] + x[2] W_4^{2k}) + (x[1] W_4^{k} + x[3] W_4^{3k}) \end{aligned}$$

Decimation in time: divide the sum to a sum of even and a sum of odd samples

Combine two 2-point DFTs into a 4-point DFT

$$X[k] = \sum_{n=0}^{3} x[n] W_4^{kn}, \text{ for } 0 \le k \le 3$$

$$\begin{aligned} X[k] &= x[0] + x[1] W_4^{k} + x[2] W_4^{2k} + x[3] W_4^{3k} \\ &= (x[0] + x[2] W_4^{2k}) + (x[1] W_4^{k} + x[3] W_4^{3k}) \\ &= (x[0] + x[2] W_4^{2k}) + W_4^{k} (x[1] + x[3] W_4^{2k}) \end{aligned}$$

Combine two 2-point DFTs into a 4-point DFT

$$X[k] = \sum_{n=0}^{3} x[n] W_4^{kn}, \text{ for } 0 \le k \le 3$$

$$\begin{aligned} X[k] &= x[0] + x[1]W_4^k + x[2]W_4^{2k} + x[3]W_4^{3k} \\ &= (x[0] + x[2]W_4^{2k}) + (x[1]W_4^k + x[3]W_4^{3k}) \\ &= (x[0] + x[2]W_4^{2k}) + W_4^k(x[1] + x[3]W_4^{2k}) \\ &= (x[0] + x[2]W_2^k) + W_4^k(x[1] + x[3]W_2^k) \end{aligned}$$

using the property $W_N^{LK} = W_{N/L}^k$ $N = 4$ and $L = 2$

Combine two 2-point DFTs into a 4-point DFT

$$X[k] = \sum_{n=0}^{3} x[n] W_4^{kn}, \text{ for } 0 \leq k \leq 3$$

$$\begin{aligned}
X[k] &= x[0] + x[1]W_4^{k} + x[2]W_4^{2k} + x[3]W_4^{3k} \\
&= (x[0] + x[2]W_4^{2k}) + (x[1]W_4^{k} + x[3]W_4^{3k}) \\
&= (x[0] + x[2]W_4^{2k}) + W_4^{k}(x[1] + x[3]W_4^{2k}) \\
&= (x[0] + x[2]W_2^{k}) + W_4^{k}(x[1] + x[3]W_2^{k}) = G[k] + W_4^{k}H[k]
\end{aligned}$$

$G[k] \equiv x[0] + x[2]W_2^k$ is the 2-point DFT of even samples

$H[k] \equiv x[1] + x[3]W_2^k$ is the 2-point DFT of odd samples

4-point DFT from 2-point DFTs

$$X[k] = G[k] + W_4^k H[k]$$

$$\begin{aligned}
X[0] &= G[0] + H[0] \\
X[1] &= G[1] + W_4 H[1] \\
X[2] &= G[2] + W_4^2 H[2] \\
X[3] &= G[3] + W_4^3 H[3]
\end{aligned}$$

4-point DFT from 2-point DFTs

$$X[k] = G[k] + W_4^k H[k]$$

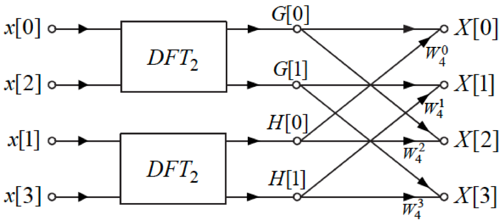
$$\begin{aligned}
X[0] &= G[0] + H[0] \\
X[1] &= G[1] + W_4 H[1] \\
X[2] &= G[2] + W_4^2 H[2] = G[0] + W_4^2 H[0] \\
X[3] &= G[3] + W_4^3 H[3] = G[1] + W_4^3 H[1]
\end{aligned}$$

$G[k]$ and $H[k]$ are 2-point DFTs, hence, 2-periodic

4-point DFT from 2-point DFTs

$$X[k] = G[k] + W_4^k H[k]$$

$$X[0] = G[0] + H[0]$$
$$X[1] = G[1] + W_4 H[1]$$
$$X[2] = G[2] + W_4^2 H[2] = G[0] + W_4^2 H[0]$$
$$X[3] = G[3] + W_4^3 H[3] = G[1] + W_4^3 H[1]$$



General case: N-point DFT from N/2-point DFTs

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$$

General case: N-point DFT from N/2-point DFTs

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$$

$$= \sum_{r=0}^{N/2-1} x[2r] W_N^{2kr} + W_N^k \sum_{r=0}^{N/2-1} x[2r+1] W_N^{2kr}$$

Decimation in time: divide the sum to a sum of even and a sum of odd samples

General case: N-point DFT from N/2-point DFTs

$$
\begin{aligned}
X[k] &= \sum_{n=0}^{N-1} x[n] W_N^{kn} \\
&= \sum_{r=0}^{N/2-1} x[2r] W_N^{2kr} + W_N^k \sum_{r=0}^{N/2-1} x[2r+1] W_N^{2kr} \\
&= \sum_{r=0}^{N/2-1} x[2r] W_{N/2}^{kr} + W_N^k \sum_{r=0}^{N/2-1} x[2r+1] W_{N/2}^{kr}
\end{aligned}
$$

using the property
$W_N^{LK} = W_{N/L}^k$

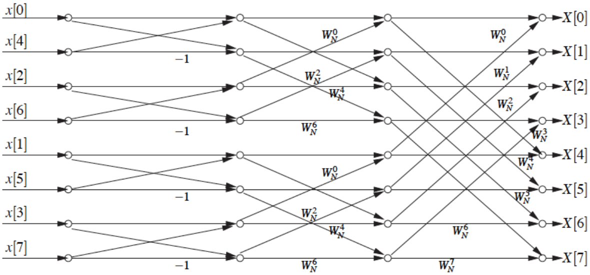
General case: N-point DFT from N/2-point DFTs

$$
\begin{aligned}
X[k] &= \sum_{n=0}^{N-1} x[n] W_N^{kn} \\
&= \sum_{r=0}^{N/2-1} x[2r] W_N^{2kr} + W_N^k \sum_{r=0}^{N/2-1} x[2r+1] W_N^{2kr} \\
&= \sum_{r=0}^{N/2-1} x[2r] W_{N/2}^{kr} + W_N^k \sum_{r=0}^{N/2-1} x[2r+1] W_{N/2}^{kr} = G[k] + W_N^k H[k]
\end{aligned}
$$

$G[k]$ is the N/2-point DFT of even samples, hence N/2-periodic

$H[k]$ is the N/2-point DFT of odd samples, hence N/2-periodic

Example: 8-point FFT



Start with 2-point DFTs of samples arranged in bit-reversed order and combine the results in each stage! Note that the butterflies can be further simplfied with $W_N^{k+N/2} = -W_N^k$

Computational complexity of Radix-2 FFT

- $v = log_2 N$ stages
- per stage, there are N/2 butterflies
- per butterly, 1 complex multiplication and 2 complex additions

Total: $log_2 N \cdot N/2$ complex multiplications and $log_2 N \cdot N$ complex additions, i.e. $O(N log_2 N)$.