Fast Fourier Transform (FFT)

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:

$$
\begin{aligned}
&\textbf{for } k = 0 : N-1 \\
&\quad X[k] \leftarrow x[0] \\
&\quad \textbf{for } n = 1 : N-1 \\
&\quad\quad l = (kn)_N \\
&\quad\quad X[k] \leftarrow X[k] + x[n] W_N^l \\
&\quad \textbf{end} \\
&\textbf{end}
\end{aligned}
$$

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:

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

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

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:

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

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

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

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)

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:

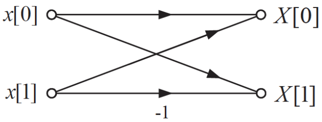
$$X[0] = x[0] + x[1] W_2^0 = x[0] + x[1]$$
$$X[1] = x[0] + x[1] W_2^1 = x[0] - x[1]$$

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:

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

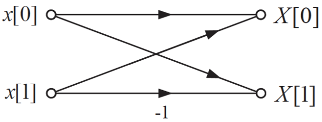


2-point DFT: How to compute in a simple way?

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

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

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

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

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

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

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

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

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

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

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

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

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

$G[k] \equiv x[0] + x[2]W_2^{k}$ 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

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

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

4-point DFT from 2-point DFTs

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

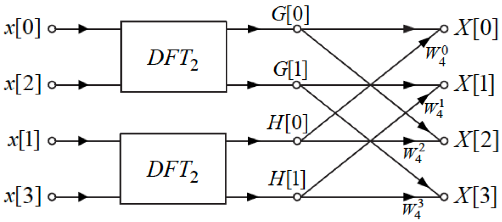
$$\begin{aligned}
X[0] &= G[0] + H[0] \\
X[1] &= G[1] + W_4 H[1] \\
X[2] &= G[2] + W_4^2 H[2] = G[0] + W_4^2 H[0] \\
X[3] &= G[3] + W_4^3 H[3] = G[1] + W_4^3 H[1]
\end{aligned}$$

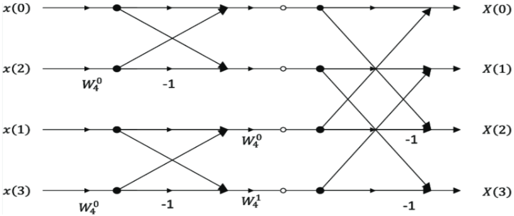
$G[k]$ and $H[k]$ are 2-point DFTs, hence, 2-periodic

4-point DFT from 2-point DFTs

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

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




x(0)
x(2)
x(1)
x(3)
W40
-1
W40
W41
-1
X(0)
X(1)
X(2)
X(3)

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

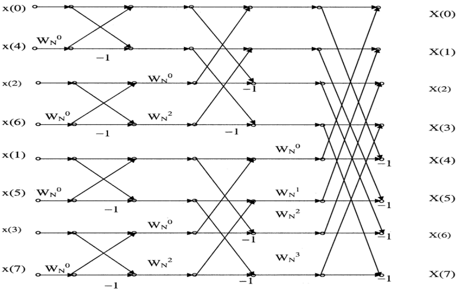
$H[k]$ is the N/2-point DFT of odd samples, hence N/2-periodic

Pseudo Code for FFT

Algorithm 1 Fast Fourier Transform (FFT)

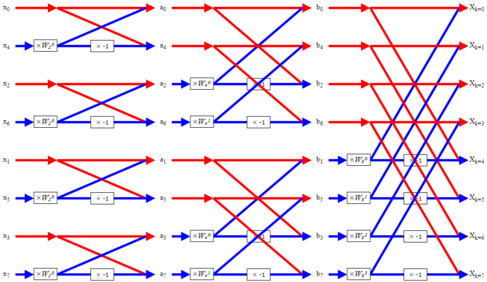
Input: x (Array of complex numbers of size N)
Output: FFT result (Array of complex numbers of size N)
Step 1: Zero Padding (if necessary)
if N is not a power of 2 **then**
    Let next_power_of_2 = smallest power of 2 greater than or equal to N
    Zero pad the input array x by appending (next_power_of_2 − N) zeros to it
    Let x_{padded} be the zero-padded array of size next_power_of_2
end if
Step 2: Base Case
if size of x_{padded} is 1 **then**
    return x_{padded}
end if
Step 3: Divide the input array into even and odd indexed parts
Let even $= [x_{\text{padded}}[0], x_{\text{padded}}[2], x_{\text{padded}}[4], \ldots, x_{\text{padded}}[N-2]]$
Let odd $= [x_{\text{padded}}[1], x_{\text{padded}}[3], x_{\text{padded}}[5], \ldots, x_{\text{padded}}[N-1]]$
Step 4: Recursively apply FFT to the even and odd parts
Let even_fft = FFT(even)
Let odd_fft = FFT(odd)
Step 5: Prepare the result array to combine the even and odd parts
Initialize result array of size N
Step 6: Calculate the twiddle factors and combine the results
for $k = 0$ to $\frac{N}{2} - 1$ **do**
    Let twiddle_factor $= e^{-\frac{2\pi i k}{N}}$
    result[k] = even_fft[k] + twiddle_factor × odd_fft[k]
    result[$k + \frac{N}{2}$] = even_fft[k] − twiddle_factor × odd_fft[k]
end for
Step 7: Return the combined result
return result

Example: 8-point FFT



$W_N^0 = 1$, $W_N^1 = (1-j)/\sqrt{2}$, $W_N^2 = -j$, $W_N^3 = -(1+j)/\sqrt{2}$

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 $W_N^{k+N/2} = -W_N^k$


x0
x4
x2
x6
x1
x5
x3
x7
×W2^0
× -1
a0
a4
a2
a6
a1
a5
a3
a7
×W4^0
×W4^1
b0
b4
b2
b6
b1
b5
b3
b7
×W8^0
×W8^1
×W8^2
×W8^3
Xk=0
Xk=1
Xk=2
Xk=3
Xk=4
Xk=5
Xk=6
Xk=7

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. $O(N log_2 N)$.