### ESE 531: Digital Signal Processing

### Week 13 Lecture 23: April 11, 2021 Fast Fourier Transform





- Discrete Fourier Transform
  - Linear convolution through circular convolution
    - Overlap and add
    - Overlap and save
  - Circular convolution through DFT
- Today
  - The Fast Fourier Transform

### Fast Fourier Transform (FFT)



Penn ESE 531 Spring 2021 - Khanna

### Fast Fourier Transform Algorithms

We are interested in efficient computing methods for the DFT and inverse DFT:

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

$$W_N = e^{-j\left(\frac{2\pi}{N}\right)}.$$

Reminder: Inverse DFT via DFT

Recall that we can use the DFT to compute the inverse DFT:

$$\mathcal{DFT}^{-1}\{X[k]\} = \frac{1}{N} \left(\mathcal{DFT}\{X^*[k]\}\right)^*$$

Hence, we can just focus on efficient computation of the DFT.

 Straightforward computation of an N-point DFT (or inverse DFT) requires N<sup>2</sup> complex multiplications.



- Fast Fourier transform algorithms enable computation of an N-point DFT (or inverse DFT) with the order of just N · log<sub>2</sub> N complex multiplications.
  - This can represent a huge reduction in computational load, especially for large N.

Ν	<b>N</b> <sup>2</sup>	$N \cdot \log_2 N$	$\frac{N^2}{N \cdot \log_2 N}$
16	256	64	4.0
128	16,384	896	18.3
1,024	1,048,576	10,240	102.4
8,192	67,108,864	106,496	630.2



- Fast Fourier transform algorithms enable computation of an N-point DFT (or inverse DFT) with the order of just N · log<sub>2</sub> N complex multiplications.
  - This can represent a huge reduction in computational load, especially for large N.

Ν	<b>N</b> <sup>2</sup>	$N \cdot \log_2 N$	$\frac{N^2}{N \cdot \log_2 N}$
16	256	64	4.0
128	16,384	896	18.3
1,024	1,048,576	10,240	102.4
8,192	67,108,864	106,496	630.2
$6 imes 10^6$	$36  imes 10^{12}$	$135  imes 10^6$	$2.67  imes 10^{5}$

## Eigenfunction Properties

- Most FFT algorithms exploit the following properties of W<sub>N</sub><sup>kn</sup>:
  - Conjugate Symmetry

$$W_N^{k(N-n)} = W_N^{-kn} = (W_N^{kn})^*$$

Periodicity in n and k

$$W_N^{kn} = W_N^{k(n+N)} = W_N^{(k+N)n}$$

Power

$$W_N^2 = W_{N/2}$$

# FFT Algorithms via Decimation

- Most FFT algorithms decompose the computation of a DFT into successively smaller DFT computations.
  - Decimation-in-time algorithms decompose x[n] into successively smaller subsequences.
  - Decimation-in-frequency algorithms decompose X[k] into successively smaller subsequences.
- Note: Assume length of x[n] is power of 2 (N = 2<sup>v</sup>). If not, zero-pad to closest power of 2.



• We start with the DFT

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



• We start with the DFT

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

• Separate the sum into even and odd terms:

$$X[k] = \sum_{n \text{ even}} x[n] W_N^{kn} + \sum_{n \text{ odd}} x[n] W_N^{kn}$$



• We start with the DFT

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

• Separate the sum into even and odd terms:

$$X[k] = \sum_{n \text{ even}} x[n] W_N^{kn} + \sum_{n \text{ odd}} x[n] W_N^{kn}$$

These are two DFTs, each with half the number of samples (N/2)



$$X[k] = \sum_{n \text{ even}} x[n] W_N^{kn} + \sum_{n \text{ odd}} x[n] W_N^{kn}$$



$$X[k] = \sum_{n \text{ even}} x[n] W_N^{kn} + \sum_{n \text{ odd}} x[n] W_N^{kn}$$

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



$$X[k] = \sum_{n \text{ even}} x[n] W_N^{kn} + \sum_{n \text{ odd}} x[n] W_N^{kn}$$

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



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

Note that:

$$W_N^{2rk} = e^{-j\left(\frac{2\pi}{N}\right)(2rk)} = e^{-j\left(\frac{2\pi}{N/2}\right)rk} = W_{N/2}^{rk}$$

#### Remember this trick, it will turn up often.



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

$$X[k] = \sum_{r=0}^{(N/2)-1} x[2r] W_{N/2}^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} x[2r+1] W_{N/2}^{rk}$$

Decimation-in-Time FFT  
Let 
$$n = 2r$$
 ( $n$  even) and  $X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \quad k = 0, ..., N-1$   
 $X[k] = \sum_{r=0}^{(N/2)-1} x[2r] W_N^{2rk} + \sum_{r=0}^{(N/2)-1} x[2r+1] W_N^{(2r+1)k}$   
 $= \sum_{r=0}^{(N/2)-1} x[2r] W_N^{2rk} + W_N^k \sum_{r=0}^{(N/2)-1} x[2r+1] W_N^{2rk}$ 

$$X[k] = \sum_{r=0}^{(N/2)-1} x[2r] W_{N/2}^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} x[2r+1] W_{N/2}^{rk}$$



Hence:

$$X[k] = \sum_{r=0}^{(N/2)-1} x[2r] W_{N/2}^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} x[2r+1] W_{N/2}^{rk}$$
$$\triangleq G[k] + W_N^k H[k], \quad k = 0, \dots, N-1$$

where we have defined:

$$G[k] \triangleq \sum_{r=0}^{(N/2)-1} x[2r] W_{N/2}^{rk} \Rightarrow \text{DFT of even samples}$$
  
$$H[k] \triangleq \sum_{r=0}^{(N/2)-1} x[2r+1] W_{N/2}^{rk} \Rightarrow \text{DFT of odd samples}$$















Both G[k] and H[k] are periodic, with period N/2. For example

$$G[k] \triangleq \sum_{r=0}^{(N/2)-1} x[2r] W_{N/2}^{rk}$$



Both G[k] and H[k] are periodic, with period N/2. For example

$$G[k] \triangleq \sum_{r=0}^{(N/2)-1} x[2r] W_{N/2}^{rk}$$

$$G[k+N/2] = \sum_{r=0}^{(N/2)-1} x[2r] W_{N/2}^{r(k+N/2)}$$

Decimation-in-Time FFT

Both G[k] and H[k] are periodic, with period N/2. For example

$$G[k] \triangleq \sum_{r=0}^{(N/2)-1} x[2r] W_{N/2}^{rk}$$

$$G[k + N/2] = \sum_{r=0}^{(N/2)-1} x[2r] W_{N/2}^{r(k+N/2)}$$

$$= \sum_{r=0}^{(N/2)-1} x[2r] W_{N/2}^{rk} W_{N/2}^{r(N/2)}$$

Decimation-in-Time FFT

(

Both G[k] and H[k] are periodic, with period N/2. For example

$$G[k] \triangleq \sum_{r=0}^{(N/2)-1} x[2r] W_{N/2}^{rk}$$

$$G[k + N/2] = \sum_{r=0}^{(N/2)-1} x[2r] W_{N/2}^{r(k+N/2)}$$

$$= \sum_{r=0}^{(N/2)-1} x[2r] W_{N/2}^{rk} W_{N/2}^{r(N/2)} = 1$$

$$= \sum_{r=0}^{(N/2)-1} x[2r] W_{N/2}^{rk}$$

$$= G[k]$$



□ So,

$$G[k + (N/2)] = G[k]$$
  
 $H[k + (N/2)] = H[k]$ 

The periodicity of G[k] and H[k] allows us to further simplify. For the first N/2 points we calculate G[k] and W<sub>N</sub><sup>k</sup>H[k], and then compute the sum

$$X[k] = G[k] + W_N^k H[k] \qquad \forall \{k : 0 \le k < \frac{N}{2}\}.$$

How does periodicity help for  $\frac{N}{2} \leq k < N$ ?







$$X[k] = G[k] + W_N^k H[k] \qquad \forall \{k : 0 \le k < \frac{N}{2}\}.$$

for  $\frac{N}{2} \leq k < N$ :

$$W_N^{k+(N/2)} = ?$$

### X[k + (N/2)] = ?



$$X[k] = G[k] + W_N^k H[k] \qquad \forall \{k : 0 \le k < \frac{N}{2}\}.$$
  
for  $\frac{N}{2} \le k < N$ :  
$$W_N^{k+N/2} = \left(e^{-j\frac{2\pi}{N}}\right)^{k+N/2}$$
$$= \left(e^{-j\frac{2\pi}{N}k}\right) \left(e^{-j\frac{2\pi}{N} \times \frac{N}{2}}\right)$$
$$X[k + (N/2)] = ? \qquad = \left(e^{-j\frac{2\pi}{N}k}\right) \left(e^{-j\pi}\right) = -W_N^k$$

X[k + (N/2)] = ?

Penn ESE 531 Spring 2021 – Khanna Adapted from M. Lustig, EECS Berkeley A /



### $X[k + (N/2)] = G[k] - W_N^k H[k]$

• We previously calculated G[k] and  $W_N^kH[k]$ .

 Now we only have to compute their difference to obtain the second half of the spectrum. No additional multiplies are required.







The *N*-point DFT has been reduced two N/2-point DFTs, plus N/2 complex multiplications. The 8 sample DFT is then:





Note that the inputs have been reordered so that the outputs come out in their proper sequence.

# Decimation-in-Time FFT

- Note that the inputs have been reordered so that the outputs come out in their proper sequence.
- We can define a *butterfly operation*, e.g., the computation of X[0] and X[4] from G[0] and H[0]:



# Decimation-in-Time FFT

- Note that the inputs have been reordered so that the outputs come out in their proper sequence.
- We can define a *butterfly operation*, e.g., the computation of X[k] and X[k+N/2] from G[k] and H[k]:




• Still  $O(N^2)$  operations.... What should we do?





We can use the same approach for each of the N/2 point DFT's. For the N = 8 case, the N/2 DFTs look like



\*Note that the inputs have been reordered again.



$$W_{N/4} = W_{8/4} = W_2 = e^{-j\pi} = -1$$



$$W_{N/4} = W_{8/4} = W_2 = e^{-j\pi} = -1$$
$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$$
$$= \sum_{n=0}^{1} x[n] W_2^{kn}$$
$$= x[0] W_2^{k\cdot 0} + x[1] W_2^{k\cdot 1}$$
$$= x[0] + x[1] W_2^k$$



$$W_{N/4} = W_{8/4} = W_2 = e^{-j\pi} = -1$$

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

$$= \sum_{n=0}^{1} x[n] W_2^{kn}$$

$$= x[0] W_2^{k \cdot 0} + x[1] W_2^{k \cdot 1}$$

$$= x[0] + x[1] W_2^k$$



$$W_{N/4} = W_{8/4} = W_2 = e^{-j\pi} = -1$$

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

$$= \sum_{n=0}^{1} x[n] W_2^{kn}$$

$$= x[0] W_2^{k\cdot 0} + x[1] W_2^{k\cdot 1}$$

$$= x[0] + x[1] W_2^k$$

Penn ESE 531 Spring 2021 – Khanna Adapted from M. Lustig, EECS Berkeley |− x[1]



$$W_{\mathsf{N}/4} = W_{\mathsf{8}/4} = W_2 = e^{-j\pi} = -1$$

The diagram of this stage is then





Replace N/2-point DFT with 2-point DFT and butterfly operations





Combining all these stages, the diagram for the 8 sample DFT is:





Combining all these stages, the diagram for the 8 sample DFT is:



•  $3 = \log_2(N) = \log_2(8)$  stages

- 4=N/2=8/2 multiplications in each stage
  - 1<sup>st</sup> stage has trivial multiplication

## Decimation-in-Time FFT

- □ In general, there are  $log_2N$  stages of decimation-in-time.
- Each stage requires N/2 complex multiplications, some of which are trivial.
- The total number of complex multiplications is (N/2) log<sub>2</sub>N, or is O(N log<sub>2</sub>N)

## Decimation-in-Time FFT

- □ In general, there are  $log_2N$  stages of decimation-in-time.
- Each stage requires N/2 complex multiplications, some of which are trivial.
- The total number of complex multiplications is (N/2) log<sub>2</sub>N, or is O(N log<sub>2</sub>N)
- The order of the input to the decimation-in-time FFT algorithm must be permuted.
  - Net effect is reversing the bit order of indexes



This is illustrated in the following table for N = 8.

Decimal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111



## This is illustrated in the following table for N = 8.

Decimal	Binary	Bit-Reversed Binary
0	000	000
1	001	100
2	010	010
3	011	110
4	100	001
5	101	101
6	110	011
7	111	111



## This is illustrated in the following table for N = 8.

Decimal	Binary	Bit-Reversed Binary	Bit-Reversed Decimal
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7



Combining all these stages, the diagram for the 8 sample DFT is:





The DFT is

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



The DFT is

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

If we only look at the even samples of X[k], we can write k = 2r,

$$X[2r] = \sum_{n=0}^{N-1} x[n] W_N^{n(2r)}$$



The DFT is

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

If we only look at the even samples of X[k], we can write k = 2r,

$$X[2r] = \sum_{n=0}^{N-1} x[n] W_N^{n(2r)}$$

We split this into two sums, one over the first N/2 samples, and the second of the last N/2 samples.

$$X[2r] = \sum_{n=0}^{(N/2)-1} x[n] W_N^{2rn} + \sum_{n=0}^{(N/2)-1} x[n+N/2] W_N^{2r(n+N/2)}$$

Decimation-in-Frequency FFT

But 
$$W_N^{2r(n+N/2)} = W_N^{2rn} W_N^{rN} = W_N^{2rn} = W_{N/2}^{rn}$$
.  
We can then write

$$X[2r] = \sum_{n=0}^{(N/2)-1} x[n] W_N^{2rn} + \sum_{n=0}^{(N/2)-1} x[n+N/2] W_N^{2r(n+N/2)}$$
$$= \sum_{n=0}^{(N/2)-1} x[n] W_N^{2rn} + \sum_{n=0}^{(N/2)-1} x[n+N/2] W_N^{2rn}$$

Penn ESE 531 Spring 2021 – Khanna Adapted from M. Lustig, EECS Berkeley

Decimation-in-Frequency FFT

But 
$$W_N^{2r(n+N/2)} = W_N^{2rn} W_N^{rN} = W_N^{2rn} = W_{N/2}^{rn}$$
.  
We can then write

$$X[2r] = \sum_{n=0}^{(N/2)-1} x[n] W_N^{2rn} + \sum_{n=0}^{(N/2)-1} x[n+N/2] W_N^{2r(n+N/2)}$$
  
= 
$$\sum_{n=0}^{(N/2)-1} x[n] W_N^{2rn} + \sum_{n=0}^{(N/2)-1} x[n+N/2] W_N^{2rn}$$
  
= 
$$\sum_{n=0}^{(N/2)-1} (x[n] + x[n+N/2]) W_{N/2}^{rn}$$

This is the N/2-length DFT of first and second half of x[n] summed.

Penn ESE 531 Spring 2021 – Khanna Adapted from M. Lustig, EECS Berkeley



$$X[2r] = \mathsf{DFT}_{\frac{N}{2}} \{ (x[n] + x[n + N/2]) \}$$
  
$$X[2r+1] = \mathsf{DFT}_{\frac{N}{2}} \{ (x[n] - x[n + N/2]) W_N^n \}$$

(By a similar argument that gives the odd samples)



$$X[2r] = \mathsf{DFT}_{\frac{N}{2}} \{ (x[n] + x[n + N/2]) \}$$
  
$$X[2r+1] = \mathsf{DFT}_{\frac{N}{2}} \{ (x[n] - x[n + N/2]) W_N^n \}$$

(By a similar argument that gives the odd samples)

 Continue the same approach on the N/2 DFTs, and N/4 DFTs until we reach the 2-point DFT, which is a simple butterfly operation



The diagram for and 8-point decimation-in-frequency DFT is as follows



This is just the decimation-in-time algorithm reversed! The inputs are in normal order, and the outputs are bit reversed.



Combining all these stages, the diagram for the 8 sample DFT is:



•  $3 = \log_2(N) = \log_2(8)$  stages

- 4=N/2=8/2 multiplications in each stage
  - 1<sup>st</sup> stage has trivial multiplication



- A similar argument applies for any length DFT, where the length N is a composite number
- For example, if N=6, with decimation-in-frequency you could compute three 2-point DFTs followed by two 3-point DFTs









$$X[2r] = \mathsf{DFT}_{\frac{N}{2}} \{ (x[n] + x[n + N/2]) \}$$
  
$$X[2r+1] = \mathsf{DFT}_{\frac{N}{2}} \{ (x[n] - x[n + N/2]) W_N^n \}$$

(By a similar argument that gives the odd samples)







$$X[2r] = \mathsf{DFT}_{\frac{N}{2}} \{ (x[n] + x[n + N/2]) \}$$
  
$$X[2r+1] = \mathsf{DFT}_{\frac{N}{2}} \{ (x[n] - x[n + N/2]) W_N^n \}$$

(By a similar argument that gives the odd samples)





Penn ESE 531 Spring 2021 - Khanna













Non-Power-of-2 FFTs

- Good component DFTs are available for lengths up to 20(ish). Many of these exploit the structure for that specific length
  - For example, a factor of

$$W_N^{N/4} = e^{-j\frac{2\pi}{N}(N/4)} = e^{-j\frac{\pi}{2}} = -j$$

Just swaps the real and imaginary components of a complex number. Hence a DFT of length 4 doesn't require any complex multiples.

- Half of the multiples of an 8-point DFT also don't require multiplication
- Composite length FFTs can be very efficient for any length that factors into terms of this order



• For example N = 693 factors into

• N = (7)(9)(11)

- each of which can be implemented efficiently. We would perform
  - 9 x 11 DFTs of length 7
  - 7 x 11 DFTs of length 9, and
  - 7 x 9 DFTs of length 11
## Non-Power-of-2 FFTs

- Historically, the power-of-two FFTs were much faster (better written and implemented).
- For non-power-of-two length, it was faster to zero pad to power of two.
- Recently this has changed. The free FFTW package implements very efficient algorithms for almost any filter length. Matlab has used FFTW since version 6

FFT Computation FLOPS



## FFT Computation Time







•  $W_N$  is fully populated  $\rightarrow N^2$  entries



$$\begin{pmatrix} X[0] \\ \vdots \\ X[k] \\ \vdots \\ X[N-1] \end{pmatrix} = \begin{pmatrix} W_N^{00} & \cdots & W_N^{0n} & \cdots & W_N^{0(N-1)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ W_N^{k0} & \cdots & W_N^{kn} & \cdots & W_N^{k(N-1)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ W_N^{(N-1)0} & \cdots & W_N^{(N-1)n} & \cdots & W_N^{(N-1)(N-1)} \end{pmatrix} \begin{pmatrix} x[0] \\ \vdots \\ x[n] \\ \vdots \\ x[N-1] \end{pmatrix}$$

- $W_N$  is fully populated  $\rightarrow N^2$  entries
- FFT is a decomposition of  $W_N$  into a more sparse form:

$$F_{N} = \begin{bmatrix} I_{N/2} & D_{N/2} \\ I_{N/2} & -D_{N/2} \end{bmatrix} \begin{bmatrix} W_{N/2} & 0 \\ 0 & W_{N/2} \end{bmatrix} \begin{bmatrix} \text{Even-Odd Perm.} \\ \text{Matrix} \end{bmatrix}$$

□  $I_{N/2}$  is an identity matrix.  $D_{N/2}$  is a diagonal matrix with entries 1,  $W_N$ , ...,  $W_N^{N/2-1}$ 



$$F_{N} = \begin{bmatrix} I_{N/2} & D_{N/2} \\ I_{N/2} & -D_{N/2} \end{bmatrix} \begin{bmatrix} W_{N/2} & 0 \\ 0 & W_{N/2} \end{bmatrix} \begin{bmatrix} \text{Even-Odd Perm.} \\ \text{Matrix} \end{bmatrix}$$

Example: N = 4

$$F_{4} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & W_{4} \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -W_{4} \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



- □ What if the signal x[n] has a k sparse frequency
  - A. Gilbert et. al, "Near-optimal sparse Fourier representations via sampling
  - H. Hassanieh et. al, "Nearly Optimal Sparse Fourier Transform"
  - Others...
  - O(K Log N) instead of O(N Log N)



From: http://groups.csail.mit.edu/netmit/sFFT/paper.html



- Fast Fourier Transform
  - Enable computation of an N-point DFT (or DFT<sup>-1</sup>) with the order of just N · log<sub>2</sub> N complex multiplications.
  - Most FFT algorithms decompose the computation of a DFT into successively smaller DFT computations.
    - Decimation-in-time algorithms
    - Decimation-in-frequency
  - Historically, power-of-2 DFTs had highest efficiency
  - Modern computing has led to non-power-of-2 FFTs with high efficiency
  - Sparsity leads to reduced computation on order K · logN



## HW 7 due 4/12 HW 8 due 4/19