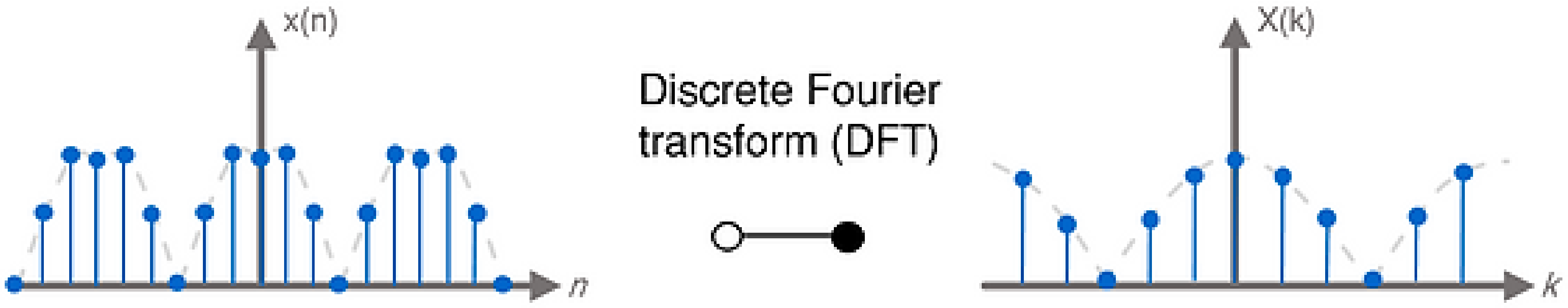
DISCRETE FOURIER TRANSFORM (DFT)

EEEN 462 – ANALOGUE COMMUNICATIONS

Friday, December 19, 2025

WHAT IS DISCRETE FOURIER TRANSFORM?

Discrete Fourier Transform (DFT) is a mathematical technique that transforms a **finite sequence of equally-spaced samples of a function** into **a same-length sequence of equally-spaced samples** of the Discrete-Time Fourier Transform (DTFT).



FROM CONTINUOUS TO DISCRETE

Fourier analysis originated with continuous functions, but digital systems require discrete implementations:

1. Continuous Time Continuous Frequency

3. Discrete Time Frequency

Continuous-Time Fourier Transform (CTFT)	Discrete-Time Fourier Transform (DTFT)	Discrete Fourier Transform (DFT)
$X(f) = \int_{-\infty}^{\infty} x(t)\, e^{-j2\pi ft}\, dt$	$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\omega n}$	$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j2\pi kn/N}$

2. Continuous Time Continuous Frequency

DISCRETE FOURIER TRANSDORM (DFT) DEFINITION

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

for k = 0, 1, 2, ..., N-1

Where

x[n] is Input sequence of length N (time domain

X[k] is DFT coefficients of length N (frequency domain)

N is Length of the sequence (must be finite)

INVERSE DISCRETE FOURIER TRANSFORM (IDFT) DEFINITION

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

for n = 0, 1, 2, ..., N-1

Where

x[n] is Input sequence of length N (time domain

X[k] is DFT coefficients of length N (frequency domain)

N is Length of the sequence (must be finite)

Inverse DFT (IDFT) reconstructs the original time-domain signal from its frequency-domain representation.

DFT MATRIX REPRESENTATION

DFT can be expressed as a matrix multiplication:

$$X = W \cdot x$$

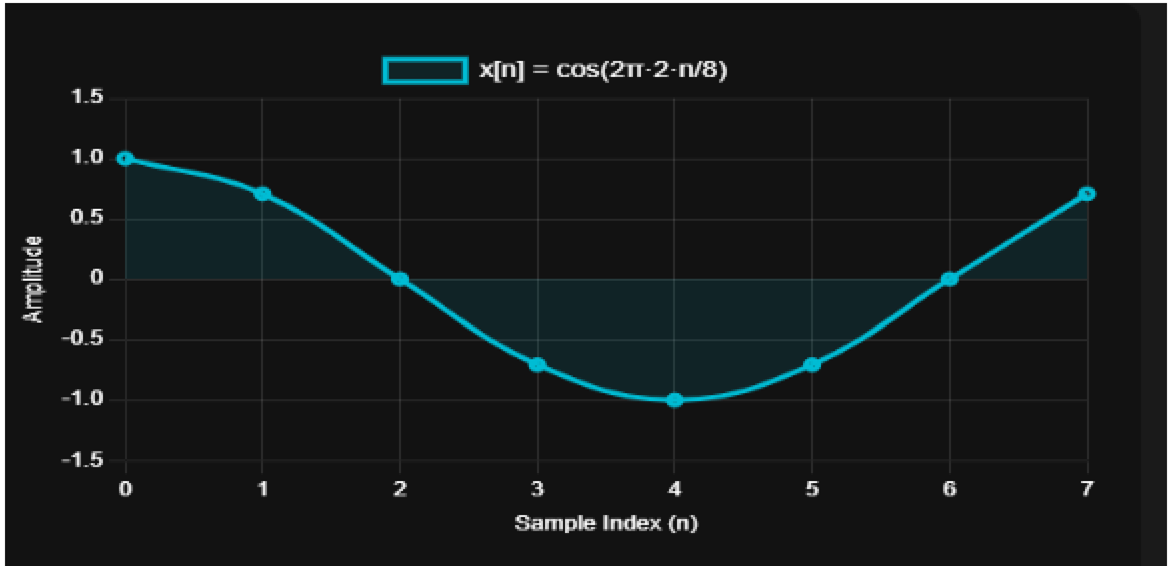
where

$$\begin{bmatrix} W_{00} & W_{01} & \cdots & W_{0(N-1)} \\ W_{10} & W_{11} & \cdots & W_{1(N-1)} \\ \cdots & \cdots & \cdots & \cdots \\ W_{(N-1)0} & W_{(N-1)1} & \cdots & W_{(N-1)(N-1} \end{bmatrix}$$

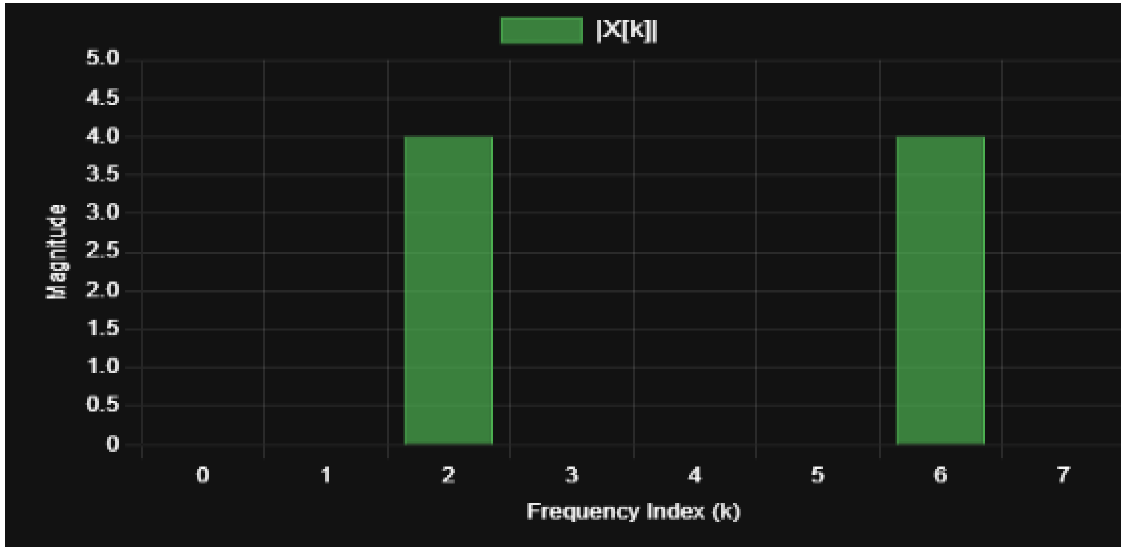
$$W_{kn} = e^{-j2\pi kn/N}$$

DFT COMPUTATION EXAMPLE

Let's compute the DFT of a simple cosine signal: x[n] = cos(2π·2·n/8) for n = 0, 1, ..., 7



Time Domain: x[n]



Frequency Domain: |X[k]|

This 8-point cosine at frequency index k=2 produces DFT coefficients with magnitude concentrated at k=2 and k=6 (due to symmetry for real-valued signals).

DFT PROPERTIES /01

1. Linearity

- $DFT(a \cdot x[n] + b \cdot y[n]) = a \cdot X[k] + b \cdot Y[k]$
- The DFT is a linear operator, essential for superposition analysis.

2. Time Shift

- $DFT(x[n-m]) = X[k] \cdot e^{-j2\pi km/N}$
- Shifting in time multiplies by a complex exponential in frequency.

3. Frequency Shift

- $DFT(x[n] \cdot e^{j2\pi mn/N}) = X[k-m]$
- Multiplication by complex exponential shifts frequency.

4. Time Reversal

- $DFT(x[-n]) = X[-k] = X[N-k]$
- Reversing time reverses frequency (with periodic extension).

DFT PROPERTIES /02

1. **For real-valued** input sequences x[n], the DFT exhibits conjugate symmetry:

$$X[k] = X^*[N-k] \quad \text{for } k = 1, 2, \ldots, N-1$$

2. **Implications for Real Signals**

- Magnitude is even symmetric: $|X[k]| = |X[N-k]|$
- Phase is odd symmetric: $\angle X[k] = -\angle X[N-k]$
- Only half the DFT coefficients are unique
- Reduces storage and computation requirements

CONVOLUTION PROPERTY OF DFT

- One of the most important properties of DFT is the **Convolution Theorem**, which states that convolution in the time domain corresponds to multiplication in the frequency domain:

$$x[n] * y[n] \quad \Leftrightarrow \quad X[k] \cdot Y[k]$$

PERFORMING LINEAR CONVOLUTION USING THE DFT

To perform linear convolution using DFT:

1. Zero-pad sequences to length ≥ M+N-1
2. Compute DFT of both padded sequences to create x[n] and y[n]
3. Multiply frequency domain results, X[k] · Y[k]
4. Compute inverse DFT

- This approach can be more efficient than direct convolution for longer sequences.

PARSSEVAL'S THEOREM

1. Parseval's theorem states that the total energy in a signal is conserved between time and frequency domains:

$$\sum_{n=0}^{N-1} |x[n]|^2 = 1/N \sum_{k=0}^{N-1} |X[k]|^2$$

2. Interpretation and Applications:

- **Energy Conservation:** DFT is a unitary transform (up to scaling)
- **Power Spectral Density:** $|X[k]|^2/N$ represents power at frequency bin k
- **Signal-to-Noise Ratio:** Can be computed in either domain
- **Filter Design:** Ensures filter implementations preserve signal energy

3. This property is fundamental to many signal processing applications, including compression, filtering, and spectral analysis.

MATLAB FUNCTION TO COMPUTE DFT – DIRECT IMPLEMENTATION

```
function X = myDFT(x)
N = length(x);         % Length of input sequence
X = zeros(1, N);       % Initialize output array
W = exp(-1j * 2 * pi / N);  % Twiddle factor
for k = 0:N-1
    sum_val = 0;
    for n = 0:N-1
        sum_val = sum_val + x(n+1) * (W^(k*n));  % x(n+1) because MATLAB
                                                  %  indexing starts at 1
    end
    X(k+1) = sum_val;  % Store result
end
end
```

MATLAB FUNCTION TO COMPUTE DFT – VECTORISED IMPLEMENTATION

```
function X = myDFT(x)
% Compute DFT using vectorized operations (faster)
N = length(x);
n = 0:N-1;      % Time indices
k = n';         % Frequency indices (column vector)
% Create DFT matrix using vectorized operations
% X[k] = Σ_{n=0}^{N-1} x[n] * exp(-j*2π*k*n/N)
X = x * exp(-1j * 2 * pi * (k * n) / N);
end
```

COMPUTATIONAL COMPLEXITY

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j2\pi kn/N}$$
$$\text{for } k = 0, 1, 2, \ldots, N-1$$

1. Direct computation of the DFT from its definition requires
   - N complex multiplications per output
   - N outputs → N^2 complex multiplications, **$O(N^2)$**
   - N(N-1) complex additions
   - Impractical for large N
2. This complexity motivated the development of the **Fast Fourier Transform (FFT)**, which reduces the complexity to **$O(N \log N)$** by exploiting symmetries in the DFT calculation.

COMPARISON OF DFT & FFT

ASPECT	DFT (DIRECT COMPUTATION)	FFT (FAST COMPUTATION)
Complexity	$O(N^2)$	$O(N \log N)$
Multiplications	N^2	$(N/2) \log_2 N$
Additions	$N(N-1)$	$N \log_2 N$
Speed (N=1024)	1× (baseline)	~200× faster
N requirement	Any N	Power of 2 (radix-2)
Implementation	Simple, direct formula	Algorithmic, recursive/iterative

APPLICATIONS OF DFT

1. Spectral Analysis

Identifying frequency components in signals (audio, vibration, EEG)

2. Filtering

Frequency domain filtering via multiplication (convolution theorem)

3. Communications

OFDM modulation, channel equalization, spectrum sensing

4. Image Processing

2D DFT for image filtering, compression, pattern recognition

5. Audio Processing

Equalizers, compression (MP3), pitch detection, effects

6. Radar/Sonar

Range and velocity estimation via Doppler analysis

PRACTICAL IMPLEMENTATIONS

1. Most real-world applications use FFT implementations of DFT for computational efficiency.
2. Libraries like **FFTW (C), NumPy (Python), and MATLAB's fft()** provide optimized implementations

LIMITATIONS OF DFT

1. Finite Length

- DFT assumes signal is periodic with period N, which may not match reality

2. Spectral Leakage

- Non-integer period signals cause energy to "leak" into adjacent bins

3. Frequency Resolution

- $\Delta f = f_s/N$, limited by observation window length

4. Picket Fence Effect

- DFT samples the continuous spectrum, possibly missing peaks

MITIGATING TECHNIQUES

1. **Windowing:** Apply window functions (e.g. Hamming) to reduce leakage
2. **Zero Padding:** Increases frequency bin density (interpolation)
3. **Increased N:** Longer observation improves frequency resolution
4. **Advanced Techniques:** Parametric methods, time-frequency analysis

WINDOW FUNCTIONS

- Window Functions e.g. Hamming Hanning → Blackman
- **Window functions taper** the signal at edges to make it appear more periodic, reducing spectral leakage at the cost of frequency resolution.