📘 Chapter: FFT Twiddle Rotations and Frequency Bins — A Layman's Guide

🎯 Objective

To deeply understand **how FFT transforms a time-domain signal into frequency components**, and specifically:

- **Why FFT rotates signals** using complex exponentials (twiddle factors)
- How frequency bins are assigned and **what each one means**
- How **twiddle angles** grow across bins from DC to Nyquist
- How the FFT journey resembles **a tree**, or **divide-and-conquer**

🧭 FFT is a Journey — Time to Frequency

Imagine holding a signal in your hand — say, 8 points measured over time.

You want to find out **what frequencies are inside** this signal. You could use the DFT directly, but it's slow:

$$O(N^2) \quad \text{operations for } N \text{ points}$$

So we use the **Fast Fourier Transform (FFT)** to do the **same job** — much faster:

$$O(N \log N)$$

But how?

□ Step 1: Go Down — Divide the Signal (Time Domain)

FFT uses a **divide-and-conquer strategy**:

- Split signal into **even and odd** samples
- Recursively split until you get single samples (base case)

- At this stage, no frequency info — just raw time-domain values

This is like **factorial** or **mergesort**:

- Break problem down to the simplest pieces first

🔼 Step 2: Come Back Up — Merge with Meaning (Frequency Domain)

Here's where magic happens:

- As you start **merging** upward, you don't just add values.
- You **rotate** some of them using special complex values:

$$W_N^k = e^{-j\frac{2\pi k}{N}} \quad \text{(twiddle factor)}$$

You're essentially **spinning your input** at different rates to **"tune in"** to different frequencies.

> 🎧 Imagine listening for one musical note. You spin your ear in sync with that frequency — and only that note sounds loud to you.

🧠 What Are These Rotations?

Twiddle factors are complex exponentials that "match" different spinning frequencies:

- $W_N^0 = 1$ → No rotation → DC
- $W_N^1 = e^{-j2\pi/N}$ → Rotation per bin 1
- …
- $W_N^{N/2} = -1$ → Nyquist frequency

So, for each **bin k**:

$$\text{Rotation Angle } \theta_k = \frac{2\pi k}{N}$$

🔍 Example: 8-Point FFT, Sampling Rate $f_s = 800$ Hz

Bin k	Frequency $f_k = \frac{kf_s}{N}$	Twiddle Angle θ_k	Rotation Meaning
0	0 Hz (DC)	0	No rotation
1	100 Hz	$\pi/4$	45° clockwise
2	200 Hz	$\pi/2$	90° clockwise
3	300 Hz	$3\pi/4$	135° clockwise
4	400 Hz (Nyquist)	π	180° (flip)

Each bin isolates that frequency by rotating incoming values at that **specific rate** and summing them.

🦋 What's the Butterfly?

Each **butterfly operation** in the FFT combines two results:

```
a ──┐           ┌─► a + W * b
    ├─ FFT ──►
b ──×W┘          └─► a - W * b
```

Where:

- a, b: values to merge
- W: twiddle factor for bin k

It's fast, efficient, and preserves both **amplitude and phase**.

✅ FFT Builds Frequency Bins While Merging Up

> As you climb the FFT tree:

- The **first bin** computed is $X[0]$ — DC (0 Hz)
- Twiddle angle is 0 → no rotation

- As you go up:
  - You compute $X[1], X[2], \ldots, X[N/2]$
  - Twiddle angle increases up to π
- By the time you reach $X[N/2]$, you're at **Nyquist frequency**

That's why:

- **FFT is not just a fast algorithm** — it's a **guided transition from time to frequency**
- Each **bin maps to a physical frequency** via:

$$f_k = \frac{k}{N} f_s$$

🔁 Where Do the Rest of the Bins Go?

For **real-valued signals**:

- FFT produces N complex results
- First half: real frequencies (0 to Nyquist)
- Second half: mirror (negative frequencies), usually discarded

Output $X[k]$	Frequency
$X[0]$	0 Hz
$X[1]$	f_s/N
...	...
$X[N/2]$	$f_s/2$ (Nyquist)
$X[N/2+1]$ to $X[N-1]$	mirror of $X[1]$ to $X[N/2-1]$

🎓 Key Takeaways for Learners

Concept	Meaning
Twiddle factor	Complex exponential that rotates signal

Concept	Meaning
Rotation angle	$2\pi k/N$ — increases with frequency bin
FFT builds up	From DC to Nyquist
$X[k]$	Bin for frequency $f_k = \frac{k}{N} f_s$
Nyquist	Max frequency: $f_s/2$, occurs at $k = N/2$

🔬 Optional Exercise (Python, No Plot)

```
import numpy as np

N = 8
fs = 800  # Hz
for k in range(N):
    angle = 2 * np.pi * k / N
    freq = k * fs / N
    print(f"Bin {k}: Frequency = {freq} Hz, Rotation Angle = {np.round(angle, 2)} radians")
```

Try this with different `N` and `fs` values to see how bins and twiddle angles change.

📦 Final Word

- FFT is not just a formula — it's a **frequency lens**
- Twiddle factors guide the **rotation into frequency space**
- Every bin you compute is a **spotlight** on a specific frequency
- **Repetition is key** — revisit this until it becomes second nature