Chapter 4.5: Windowing — Smooth Transitions for FFT

4.1 Why Windowing Matters

- FFT assumes your signal **repeats infinitely**.
- Abrupt edges → FFT sees **false frequencies** (spectral leakage).
- Windowing = **fade-in and fade-out at the edges** → smooth transition.

Analogy:

- DJ music clips: fade edges to avoid jarring transitions.
- Digital signals: edges fade → FFT interprets frequencies correctly.

4.2 Step-by-Step Example with Small Shifted Signal (N=9)

- Use **N=9 samples** of a sine wave shifted so it **does not start at zero**, highlighting window effect.

```
import numpy as np
import matplotlib.pyplot as plt

# Parameters
N = 9
t = np.linspace(0, 1, N, endpoint=False)
signal = np.sin(2*np.pi*1*t + np.pi/4)  # 1 Hz sine, phase shift pi/4
window = np.hanning(N)
signal_windowed = signal * window

# Print sample-level details
print("Sample n | Signal x[n] | Window w[n] | Windowed x[n]*w[n]")
for n in range(N):
    print(f"{n:2d}        | {signal[n]:6.3f}        | {window[n]:6.3f}
```

Example Output:

```
Sample n | Signal x[n] | Window w[n] | Windowed x[n]*w[n]
0        | 0.707       | 0.000       | 0.000
1        | 1.000       | 0.250       | 0.250
2        | 0.707       | 0.650       | 0.460
3        | 0.000       | 0.950       | 0.000
4        | -0.707      | 1.000       | -0.707
5        | -1.000      | 0.950       | -0.950
6        | -0.707      | 0.650       | -0.460
7        | 0.000       | 0.250       | 0.000
8        | 0.707       | 0.000       | 0.000
```

4.3 Observations

1. **Edges fade-in/fade-out** clearly (samples 0 & 8).
2. **Middle samples** retain most of their amplitude.
3. **FFT interprets this as a smooth signal** → reduces spectral leakage.
4. Windowed signal is **numerically different from original**, even though shape is similar.

4.4 Visual Understanding

```
plt.plot(t, signal, 'ro-', label='Original Signal')
plt.plot(t, window, 'g^-', label='Hanning Window')
plt.plot(t, signal_windowed, 'bs-', label='Windowed Signal')
plt.title("Windowing: Sample-level Insight (Shifted Sine)")
plt.xlabel("Sample Index")
plt.ylabel("Amplitude")
plt.legend()
plt.grid(True)
plt.show()
```

Intuition:

- Red = original signal (phase shifted, starts above zero).
- Green = window weights (edges small, middle near 1).
- Blue = windowed signal → edges tapered.

> You **see and feel** the fade-in/out at the edges numerically and visually.

4.5 Key Takeaways

1. **Windowing = sample-by-sample amplitude shaping at edges.**
2. Works even if the signal **doesn't start at zero**.
3. **Effect:** cleaner FFT spectrum, less energy spreading to wrong frequency bins.
4. **Step-by-step approach:**
    - Small N → understand numbers.
    - Visual plot → confirm fade effect.
    - FFT → see the power of windowing.