📃 Chapter 7.1 : Filter Theory & Concepts - Part I

1. Why Filters?

In real-world signals (say audio, sensor data, or drone IQ samples), not all frequencies are useful.

- You may want to keep **slow variations** but remove high-frequency noise (low-pass).
- Or remove hum at 50 Hz but keep the rest (notch filter).
- Or focus only on a narrow band, like 2.4 GHz Wi-Fi (band-pass).

☞ A **filter** is simply a rule that tells:
"Which frequencies should pass, which should be suppressed?"

---

2. Signals in Frequency

A digital signal (sequence of samples) can be seen as a **mixture of sinusoids**.
Filtering = modifying the amplitudes of those sinusoids.

Example:
Say signal = 10 Hz + 200 Hz components.

- If you use a low-pass at 50 Hz → only 10 Hz survives.
- If you use high-pass at 50 Hz → only 200 Hz survives.

---

3. Cutoff Frequency & Sampling

You cannot design filters without knowing:

- **Fs** = Sampling frequency
- **f_c** = Desired cutoff (Hz)

Since everything in DSP is "digital time," we normalize cutoff:

$$W_n = \frac{f_c}{F_s/2}$$

- $F_s/2$ = Nyquist (max representable frequency).
- W_n = normalized cutoff between 0–1.

☞ If $F_s = 1000$ Hz, Nyquist = 500 Hz.

- A cutoff at 100 Hz → $W_n = 100/500 = 0.2$.

---

4. FIR vs IIR: The Recipe Metaphor

Filtering is like cooking:

- You have **ingredients (samples)**.
- You have a **recipe (coefficients)**.
- The recipe decides how much of past inputs and past outputs you mix.
- **FIR (Finite Impulse Response):** Only past **inputs** matter.

$$y[n] = b_0 x[n] + b_1 x[n-1] + \ldots + b_M x[n-M]$$

- **IIR (Infinite Impulse Response):** Uses past inputs **and outputs** (feedback).

$$y[n] = b_0 x[n] + b_1 x[n-1] + \ldots - a_1 y[n-1] - \ldots$$

☞ FIR = “only current & past ingredients.”
☞ IIR = “taste last dish and adjust today’s spice” (feedback).

---

5. Order of a Filter

- **Order = number of past samples remembered.**
- A higher order means a **steeper transition** (sharper cutoff).
- Example:
  - 1st order low-pass → slope = –20 dB/decade
  - 4th order low-pass → slope = –80 dB/decade

---

6. Example: Butterworth Low-Pass

```
from scipy.signal import butter, lfilter
import numpy as np
import matplotlib.pyplot as plt

fs = 1000       # Sampling rate
fc = 100        # Cutoff
order = 4       # Filter order

# Design
Wn = fc / (fs/2)             # Normalize
b, a = butter(order, Wn, btype='low')

# Apply to a mixed signal
t = np.linspace(0, 1, fs, endpoint=False)
x = np.sin(2*np.pi*10*t) + np.sin(2*np.pi*200*t)  # 10 Hz + 200 Hz
y = lfilter(b, a, x)

plt.subplot(2,1,1); plt.plot(t, x); plt.title("Input: 10Hz + 200Hz")
plt.subplot(2,1,2); plt.plot(t, y); plt.title("Output: Only 10Hz remains")
plt.show()
```

☞ In plain English:

- `b` = feedforward coefficients (how inputs are mixed).
- `a` = feedback coefficients (how past outputs adjust).

---

7. Intuitive Example with Numbers

Say input samples = [3, 6, 7].

- If FIR with `b = [1/2, 1/3]`:

$$y[n] = \frac{1}{2}x[n] + \frac{1}{3}x[n-1]$$

- Output = weighted average of current + previous input.

☞ This **smooths out variations** = reduces high frequency noise.

---

8. Frequency vs Time View

- In **time domain**, filtering = weighted averaging.
- In **frequency domain**, filtering = attenuation of unwanted frequencies.

Both are two faces of the same coin.

---

9. References (must-reads)

- **Book:** “Understanding Digital Signal Processing” – Richard Lyons (very headfirst-style).
- **SciPy Docs:** scipy.signal.butter
- **MIT OpenCourseWare DSP lectures** (free videos).

📒 Part 2: FIR, IIR & Adaptive Filters

1. FIR Filters (Finite Impulse Response)

- Output depends only on current and **finite past inputs**.
- Always stable.
- Easier to design with linear phase.
- Example: moving average.

☞ Think: “fixed recipe, no feedback.”

---

2. IIR Filters (Infinite Impulse Response)

- Output depends on inputs and **past outputs** (feedback).
- More efficient (lower order achieves sharp cutoff).

- But risk of instability if coefficients chosen poorly.

☞ Think: “cook, taste, adjust.”

---

3. Adaptive Filters

- Filters that **change their coefficients automatically** as signal conditions change.
- Used in:
  - Noise cancellation (ANC headphones).
  - Channel equalization in communication.
  - Echo suppression.
- Algorithms: LMS (Least Mean Squares), RLS (Recursive LS).

☞ Think: “the chef keeps adjusting spices based on feedback from the eater.”

---

4. Big Picture Roadmap

- **Step 1:** Understand FIR & IIR basics (done).
- **Step 2:** Play with butter, chebyshev, elliptic filters.
- **Step 3:** Move to adaptive filters (LMS, RLS).
- **Step 4:** Apply to **real drone IQ data** (noise removal, spectrum isolation).
- **Step 5:** Research paper direction: “Adaptive Filtering for Counter-Drone IQ Data Analysis.”

✅ **stitched one-stop read** that covers:

- What filters do,
- FIR vs IIR math,
- Cutoff & normalization,
- Practical code,
- And where we’re headed (adaptive).