FIR Filter Design — Manual, Intuitive Walkthrough (No Math First)

🎯 GOAL:

> Design a simple **FIR Low-Pass Filter** — lets only keep **low frequencies** (e.g. < 1 kHz), and suppress higher ones.

Let's **pretend** we are designing for a system with:

- Sampling rate $f_s = 8000$ Hz
- Cutoff frequency $f_c = 1000$ Hz

❖ STEP 1: Decide on Filter Type and Purpose

Q: What kind of filter?

→ FIR Low-Pass

Q: Why?

→ We want to **smooth** the signal and remove higher-frequency noise (like in audio or baseband I/Q data)

❖ STEP 2: Decide on Filter Shape (Impulse Response $h[n]$)

Let's design a basic **5-tap FIR filter** (keep it short to understand):

Let's **manually choose the coefficients** for averaging:

❖ Trial FIR Filter:

$$h[n] = \left[\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}\right]$$

(Also called a **moving average filter**.)

This means:

- Each output is the **average of the last 5 inputs**
- It **smooths** the signal
- It acts as a **low-pass filter** — high-frequency variation gets canceled out

---

🛠 STEP 3: See It in Action (Convolution)

Say input signal:

$$x[n] = [2, 4, 6, 8, 10, 12, 14]$$

Let's **convolve** it with our $h[n]$:

We'll slide the filter window over the input and compute output one step at a time.

▸ Output $y[0]$

Need 5 values: $x[0]$ to $x[4]$ → [2, 4, 6, 8, 10]

$$y[0] = (2 + 4 + 6 + 8 + 10)/5 = 30/5 = 6$$

▸ Output $y[1]$

Now next window: [4, 6, 8, 10, 12]

$$y[1] = (4 + 6 + 8 + 10 + 12)/5 = 40/5 = 8$$

▸ Output $y[2]$: [6, 8, 10, 12, 14] $\rightarrow$

$$(6+8+10+12+14)/5 = 50/5 = 10$$

So, Output $y[n] = [6, 8, 10]$

📈 Interpretation:

- Input was rising: 2, 4, ..., 14
- Output is **slower**, smoother: 6, 8, 10
- Sharp jumps got **smoothed** (high frequencies filtered out)

Congratulations — you just **designed a working FIR filter manually**! ✅

🔬 STEP 4: Think in Frequency Terms (Intuitively)

- Averaging 5 values reduces fast changes → kills high frequencies
- So this moving average **acts as a low-pass filter**

That's what we meant when we said:

> "Filtering in time domain (via convolution) equals sculpting in frequency domain."

No need for FFT or freq plots — just **observe output smoothness**.

📚 What You Just Learned (Without Formulas)

Concept	What It Means
FIR = finite-length filter	Works on a fixed number of past inputs

Concept	What It Means
$h[n]$ = filter shape	You choose it — defines the behavior
Convolution	Apply the shape over time via dot product
Moving average = low-pass	Smooths sharp changes (removes noise)
Longer filters = smoother	More taps = sharper frequency cut, more delay

---

🔄 NEXT: Redesign a Different Filter Manually?

You can now try:

◈ 1. High-pass filter manually

Let $h[n] = [1, -1]$
This computes difference between samples — so it emphasizes changes (edges or noise)

◈ 2. Edge detector

Let $h[n] = [-1, 0, +1]$
This picks up rising or falling transitions

⍰ Recap Summary (No-Math FIR Design)

Step	What You Did
Chose filter type	FIR, Low-pass
Guessed shape $h[n]$	Used moving average
Applied convolution	Manually computed output
Saw effect	Smoother output, high-freq suppression

🔜 Ready to Do the Same in Python?

Now that you've seen the entire logic manually, we can:

- Replicate this 5-tap average filter in Python
- Plot output vs input
- Then scale up to real filter design tools like `firwin()`

Shall we do the **Python match of this same moving average filter** next?