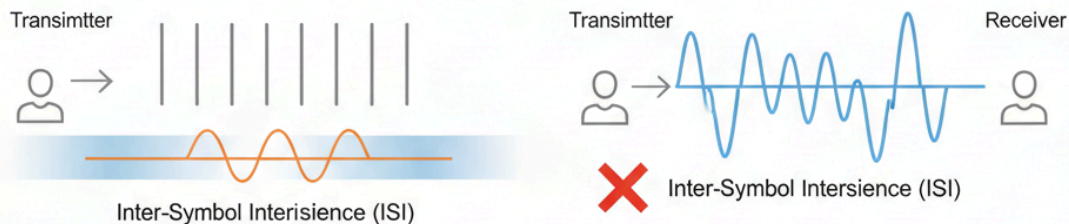


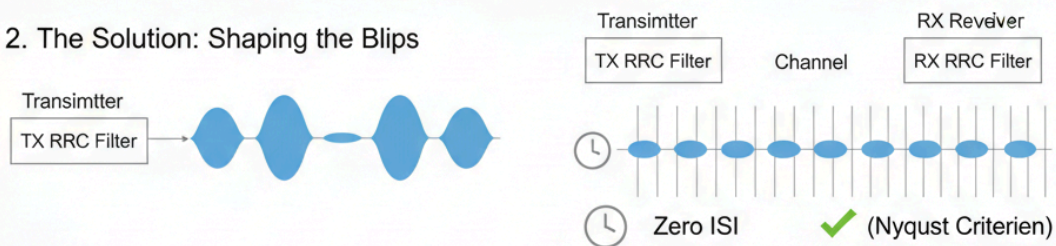
Chapter X: FFT & IFFT — The Secret Shortcut of Digital Filters

Root Raised Cosine (RRC) Explained

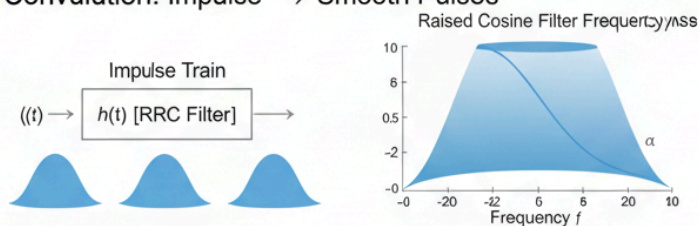
1. The Problem: Muddled Messages (ISI)



2. The Solution: Shaping the Blips



3. Convolution: Impulse → Smooth Pulses



4. Implementation Tricks

- Zero-Padding
- FFT/IFFT
- Oversampling

TX: Transmitter, RX: Inter-Symbol Interspace

Enter the Problem

So you've got a signal.

You want to filter it.

Normally, filtering = convolution in the time domain:

$$y[n] = (x * h)[n]$$

Translation?

Take every point of your signal, multiply by the filter's shape, and add it up.

For a long signal, this is like... trying to wash your whole car with a toothbrush.  

Slow. Painful. Too many multiplications.

✦ Enter the Hack

Math has a magical shortcut:

- Convolution in **time** = Multiplication in **frequency**.
- Multiplication in **time** = Convolution in **frequency**.

That's the duality.

So instead of toothbrushing your car...

💡 You roll your car into a car wash tunnel → *everything happens at once*.


💡 The Players

- **FFT (Fast Fourier Transform)**: Converts your signal from “time wiggles” into “frequency bins.” Like changing your view from the road to the map.
- **IFFT (Inverse FFT)**: Converts it back. You don't live in “map land,” you live in “road land.”

✂ How the Trick Works

1. Take your signal $x[n]$. Run FFT → get $X(f)$.
(Now you see the *ingredients* of the signal: which frequencies are in there, and how strong they are.)
2. Take your filter's impulse response $h[n]$. FFT it too → get $H(f)$.
(Now you know what the filter “lets through” and what it kills.)
3. Multiply them in the frequency domain:

$$Y(f) = X(f) \cdot H(f)$$

(One clean step instead of a million convolutions. Like using a colander to instantly separate pasta from water. )

4. Run IFFT → back to time domain, $y[n]$.
(Voilà: your filtered signal.)

🧠 Why It's a Big Deal

- **Speed:** Convolution in time = $O(N^2)$. With FFT = $O(N \log N)$. Huge savings when N is big.
- **Implementation:** Modern SDRs and DSP chips lean on FFT-based filtering because *time is money* (literally, in GHz).
- **Concept:** You don't filter by "smearing a brush" — you filter by *cutting frequencies like a DJ equalizer*. 🎵🎶

📦 Head First Analogy

Imagine your signal is a big soup 🍲 with lots of ingredients mixed in.

- In time domain, you try to remove onions by fishing them out one by one with chopsticks. 💎 (painful = convolution).
- In frequency domain, FFT is like running the soup through a magical ingredient scanner → you see "Onions: 20%, Carrots: 40%, Chicken: 40%."
- Apply filter = multiply → "Set Onions = 0."
- IFFT = rebuild soup without onions. Done. ✓

🔑 Key Insight

👉 FFT + Multiply + IFFT is just **a faster way of doing convolution**.

It doesn't change the math — it just changes the battlefield.

Time domain = toothbrush.

Frequency domain = power washer. 💧

✓ Takeaway Box

- Convolution in time ↔ Multiplication in frequency
- FFT → Frequency domain
- Multiply by filter response
- IFFT → Back to time domain
- Faster, smarter, cleaner

Great — we'll take the seven steps **one at a time**, simply and tightly. No detours. I'll start with **Step 1 (Warmup)** and stop so you can confirm we move to Step 2.

Step 1 — Warmup: *What is convolution, really?*

1. The core idea (in one line)

Convolution = **every input sample sprays a scaled copy of the filter shape into the output**, and the output is the sum of all those sprayed copies.

Think: each input “turns on” the filter shape at its time location; the output is the overlap of all those shapes.

2. Visual/physical analogy

- Imagine the filter $h[n]$ is a **stamp** (a little shape).
- Each time an input sample $x[n]$ is nonzero, you press that stamp onto the output canvas at position n , scaled by the sample value.
- The final picture is all these stamps added together. That is $y[n] = x * h$.

3. The flip-and-slide rule (how to compute)

To compute $y[n]$ at time n :

1. Flip h around (mirror it).
2. Slide it so its origin lines up with n .
3. Multiply aligned samples by the input and sum.

That’s the classic “flip and slide” view — useful by hand.

4. Tiny numeric example (do one sample together)

Let

- $x = [1, -1, 1]$ (three input samples at times 0,1,2)
- $h = [1, 2, 1]$ (3-tap symmetric filter)

Convolution $y = x * h$ yields length 5 ($3+3-1$):

Compute step-by-step:

- $y[0] = 1 \cdot 1 = 1$
(only $x[0]$ aligns with $h[0]$)
- $y[1] = x[0] \cdot 2 + x[1] \cdot 1 = 1 \cdot 2 + (-1) \cdot 1 = 1$
- $y[2] = x[0] \cdot 1 + x[1] \cdot 2 + x[2] \cdot 1 = 1 \cdot 1 + (-1) \cdot 2 + 1 \cdot 1 = 0$

- $y[3] = x[1] \cdot 1 + x[2] \cdot 2 = (-1) \cdot 1 + 1 \cdot 2 = 1$
- $y[4] = x[2] \cdot 1 = 1$

So $y = [1, 1, 0, 1, 1]$.

Interpretation: each input sample placed a scaled $[1, 2, 1]$ under the output; where they overlapped their numbers added (some canceled at $y[2]$).

5. Why this matters to pulse shaping

- If x is a spike train (symbols at spaced times), convolution with the pulse h produces **smooth pulses** located where spikes were.
- Overlaps between shifted copies of h are exactly what we manage to get zero ISI (later steps).

Step 2: The duality idea — convolution in time \Leftrightarrow multiplication in frequency

1. The Core Rule

\Leftrightarrow **Convolution in time \leftrightarrow Multiplication in frequency**

That's the "shortest elevator pitch" for this whole topic.

2. What does it mean?

- If you filter by convolving your signal $x[n]$ with filter $h[n]$:

$$y[n] = (x * h)[n]$$

- Then, in the frequency domain:

$$Y(f) = X(f) \cdot H(f)$$

- So instead of messy adds in time, it's a **simple multiply** in frequency.

3. Analogy (Head First style)

Imagine cooking 🍳:

- **Time-domain way (convolution):** every time you add an ingredient, you stir the whole pot, one spoon at a time (slow).
- **Frequency-domain way:** you magically read the recipe's ingredient list (FFT gives you the spectrum). Want less salt? Just scale salt down (multiply). Done instantly.

Multiplication is way easier than endless stirring.

4. Proof sketch (just to trust the math)

We saw in Step 1:

$$y[n] = \sum_k x[k] \cdot h[n - k]$$

Take its Fourier transform:

$$Y(e^{j\omega}) = X(e^{j\omega}) \cdot H(e^{j\omega})$$

That's the convolution theorem. ✓

5. Why it matters for us (link to pulse shaping)

- In practice, filtering = convolution.
- But if you hop to frequency domain, convolution turns into multiplication → **much faster**.
- This is why FFT/IFFT comes into play later — it's not “new math,” it's the **same math in a faster outfit**.

☞ So far we've:

- Step 1: Convolution = spraying filter shapes + adding.
- Step 2: That same operation = just multiply spectra.

Step 3 (RRC Context)

We want to **shape digital pulses** (like +1, −1 symbols) so that:

- They don't spill into neighbors (avoid ISI).
- They don't hog spectrum (smooth edges).

Time-domain approach:

- You take the sharp spikes and spray the **RRC pulse shape** at each symbol location.
- Then you add them up.
- This is convolution with the RRC filter in time domain.
- It works, but slow and messy for long symbol trains.

Frequency-domain shortcut:

- Every filter, including RRC, has a **frequency fingerprint** (its spectrum).

- Instead of spraying pulses, you just **multiply** the signal spectrum with the RRC spectrum.
- This instantly enforces “no ISI” (zero crossings at symbol points) and spectrum-limiting (smooth edges).

Head First Analogy (RRC Edition)

Imagine you’re piping **water pulses** ☑ into a cleanroom:

- **Without RRC:** Each valve opens and shuts abruptly → water splashes everywhere (ISI + wide frequency mess).
- **RRC in time domain:** You add a shaped nozzle that smooths the water jet. Each valve opening is convolved with the nozzle shape. Works, but slow to simulate.
- **RRC in frequency domain:** Instead of messing with every nozzle, you adjust the **master water filter** that decides which splash frequencies are allowed. That’s just multiplying spectra.

So:

- **Time domain** → spray + add nozzle shapes.
- **Frequency domain** → tweak the spectral knobs once and you’re done.

☞ Step 4 — Why FFT is Efficient (RRC Case)

Time-domain convolution (slow way):

- If you convolve a long train of symbols with the RRC filter (which itself may be 50–100 taps long), you must do **$N \times M$ multiplications** (N = signal length, M = filter length).
- That grows big very fast when transmitting millions of symbols.

Frequency-domain trick (FFT way):

- Convolution in time \leftrightarrow Multiplication in frequency.
- So, instead of sliding the RRC pulse over the signal again and again, we:
 - Take FFT of the symbol train.
 - Take FFT of the RRC filter.
 - Multiply spectra point-by-point.
 - Do IFFT to get the shaped waveform back in time domain.

That’s much faster: **FFT reduces the cost from $N \times M$ to $\sim N \log N$.**

Head First Analogy (RRC Edition)

Imagine you’re running a **factory machine** 🏭 shaping chocolate bars (symbols):

- **Time-domain method:** Each bar gets **individually passed through a mold** (RRC pulse shape). This is slow if you have millions of bars.
- **FFT method:** Instead of molding bars one by one, you **transform the whole batch into chocolate powder (frequency domain)**, mix in the flavor packet (RRC filter spectrum), and then solidify it back (IFFT). One operation flavors all bars at once.

Result = same shaped chocolate, but mass-produced efficiently.

💡 For RRC: This means we can shape entire blocks of data quickly, making real-time transmission (like in SDRs and 5G) practical.

👉 **Step 5: How IFFT brings you back home (time domain)?**

Awesome 🐉 let's dive into **Step 5: How IFFT brings you back home (time domain)** — still framed around RRC.

Step 5 — IFFT: Coming Back to Time Domain

After Step 4, we've done our magic:

- $\text{FFT}(\text{symbols}) \times \text{FFT}(\text{RRC filter}) \rightarrow \text{spectrum of shaped signal}$.

But the airwaves (or DAC \rightarrow RF hardware) only understand **time-domain waveforms** 🕒.

So we must bring this shaped signal **back to the time axis**.

That's exactly what the **Inverse FFT (IFFT)** does:

- It converts the frequency-domain multiplication result back into a time-domain waveform.
- This time-domain waveform is now the **smooth RRC-shaped signal** ready for DAC and transmission.

Head First Analogy (RRC Edition)

Think of the FFT/IFFT process like **Google Translate** 🌐:

- You took your symbols (time words) \rightarrow translated into frequency language (FFT).
- There you made simple edits (just multiplied with the RRC spectrum).
- Now you need to give the output back to humans (antennas, DAC).
- IFFT = **translating back into the native time-domain language** so the real world can understand.

Without IFFT, your transmitter is stuck in frequency space — and antennas don't speak "frequency-domain math."

💡 For RRC:

- The IFFT gives you the smooth **time-domain pulse train** where each symbol is shaped just right (zero ISI at sampling instants, limited bandwidth).

👉 Next is **Step 6: Practical win — block processing with FFT/IFFT**.

Perfect 🎯 Let's do **Step 6: Practical win — block processing with FFT/IFFT**, keeping it tied to RRC.

Step 6 — Block Processing with FFT/IFFT

Why we bother with FFT/IFFT at all (big picture):

- RRC shaping needs convolution with long filters.
- Doing this symbol-by-symbol in time domain is inefficient.
- But if we use FFT, we can **process whole blocks of data in one go**.

How it works in practice:

1. Collect a block of symbols (say 1024).
2. Zero-pad both the block and the RRC filter (so convolution works properly).
3. FFT both.
4. Multiply spectra.
5. IFFT to get shaped time-domain block.
6. Move to the next block (with overlap handling if needed).

This way, you trade **many slow convolutions** for **a few FFTs/IFFTs**, which computers (and SDR hardware) handle super fast.

Head First Analogy (RRC Edition)

Think of baking 🍪:

- **Time-domain method:** Bake cookies **one by one** in a small oven — they'll come out RRC-shaped, but it takes forever.
- **FFT block method:** Put **trays of 100 cookies** into a big oven at once. Same shaping, way faster.

The oven = FFT/IFFT.

The tray size = block length.

The recipe shaping = RRC filter.

So instead of babysitting cookies, you mass-produce shaped pulses.

💡 For RRC:

- Block FFT/IFFT makes real-time systems (like SDRs, LTE/5G modems) possible.

- You're not really changing the theory — just making the math run **orders of magnitude faster**.

👉 **Step 7: Big picture — why comms engineers love this trick (RRC + FFT/IFFT).**

Alright 🚀 let's finish strong with **Step 7: The Big Picture — Why Comms Engineers Love FFT/IFFT for RRC Filtering**.

Step 7 — Big Picture Payoff

So far:

- We learned that **RRC filtering = convolution** to smooth symbols, avoid ISI, and limit bandwidth.
- Time-domain convolution works but is **computationally heavy**.
- Frequency-domain (FFT → multiply → IFFT) makes it **fast and practical**.
- Block processing lets entire systems handle streams of data efficiently.

Now, why engineers swear by this method:

1. Efficiency

- FFT reduces cost from $N \times M$ to $N \log N$.
- That's the difference between “toy demo” and “real-world LTE/5G modem.”

2. Flexibility

- The same FFT/IFFT machinery can handle **RRC filtering, channel equalization, OFDM modulation**, etc.
- One hammer, many nails 🔨.

3. ISI Control

- RRC's magic: the pulse has **zero crossings at symbol instants**.
- Matched filter at the receiver locks onto those zero-ISI points.
- FFT/IFFT ensures this shaping is applied without wasting CPU cycles.

4. Bandwidth Control

- Roll-off factor α in RRC directly shapes the spectrum.
- With FFT/IFFT, you can literally see and tweak this in frequency domain.

Head First Analogy (RRC Edition)

Think of running a **busy airport** ➔:

- Without RRC shaping: planes take off whenever → chaos, near misses (ISI), wasted airspace (spectrum).
- With RRC shaping: each plane follows a **smooth takeoff path** that guarantees

no overlap.

- With FFT/IFFT: instead of guiding each plane individually, you build a **control tower system** that handles hundreds of flights at once with efficiency and safety.

That's why FFT/IFFT + RRC is the **workhorse combo** behind modern comms — from SDR demos to 5G towers.

✓ That completes the **7-step journey**.

We now have the “forest view” of *why convolution \rightarrow FFT/IFFT \rightarrow RRC shaping* is such a cornerstone.