

Digital Signal Processing

University Electrical Engineering

What this course is, and how to use it

This is the twelfth course in University Electrical Engineering: **digital signal processing** – the last of the EE strand. EE06 built the continuous-time theory; this course re-founds it in samples: the transforms that live on a computer, the filters you can actually ship, and the multirate machinery that moves between rates.

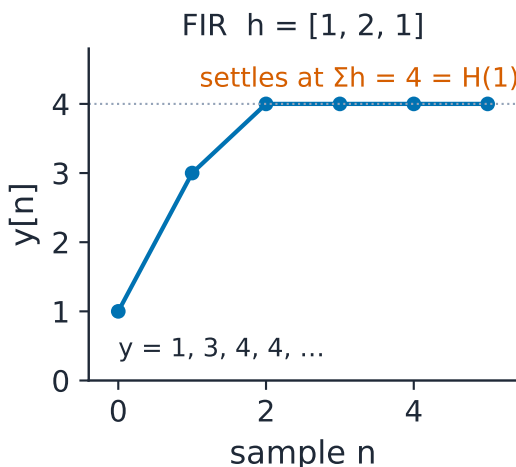
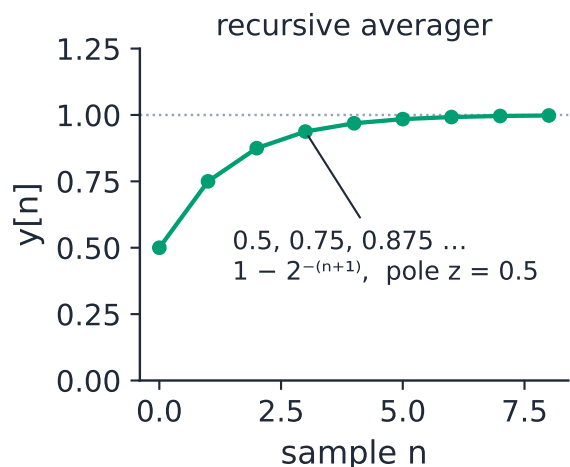
- Units 12.1–12.3 are the foundations: discrete systems, DTFT/DFT, FFT.
- Units 12.4–12.5 design filters: FIR by windows, IIR by bilinear transform.
- Units 12.6–12.7 change rates and build a small verified pipeline.

The checking habit: every filter design is verified against its specification (measured ripple, measured transition, measured stopband), every frequency-mapping claim is prewarped explicitly, and every speedup claim is an operation count, not folklore.

Discrete-time signals and systems review

Why it matters. Everything in this course – every filter, every spectrum, every speedup – is eventually a program operating on a list of numbers. The whole behaviour of that program is decided by one list: the impulse response.

The idea, with a picture. A discrete system is LTI when it is fully described by its **impulse response** $h[n]$: the output is **convolution** $y[n] = \sum_k h[k] \cdot x[n - k]$ – slide the taps across the input and add one diagonal at a time.



The FIR settles in three samples; the recursive averager never exactly settles – the tap sum is the destination in both

Causality means $h[n] = 0$ for $n < 0$ (no output before an input); **BIBO**

stability BIBO means $\sum |h[n]|$ converges. Every claim in this course reduces to one of those two sentences.

The derivation. Feed the complex exponential $x[n] = e^{j\omega n}$ through the convolution sum: each term picks up the factor $h[k]e^{j\omega(n-k)} = e^{j\omega n}h[k]e^{-j\omega k}$, so the output is $e^{j\omega n} \sum_k h[k]e^{-j\omega k}$ – the exponential is an eigenfunction, and the eigenvalue $H(e^{j\omega}) = \sum h[n]e^{-j\omega n}$ is the **frequency response**. The frequency response is just the DTFT of the taps: unit 12.2 builds on exactly this.

Worked example 1 – the sliding table. $h = [1, 2, 1]$ (a 3-tap smoother) on a step $x = [1, 1, 1, 1, \dots]$: slide and add – $y = 1, 3, 4, 4, \dots$. The steady value is $\sum h = 4$ ✓ (a step through any FIR settles at the tap sum – that is the DC gain $H(1)$). A finite pulse $[1, 1, 1]$ gives $1, 3, 4, 3, 1$, mirroring the taps: the output reads the taps forwards then backwards.

Worked example 2 – the recursive cousin. The running average $y[n] = (y[n-1] + x[n])/2$ has an infinite impulse response $h[n] = 0.5^{n+1}$: its step response computes to $0.5, 0.75, 0.875, 0.9375, \dots$ – halving the distance to 1 each step, the exponential $1 - 2^{-(n+1)}$ ✓, because its single pole sits at $z = 0.5$. Compare the two smoothers: the FIR’s memory is exactly 3 samples; the recursive one never fully forgets. $|pole| < 1$ is stability – unit 12.5 makes a career of that.

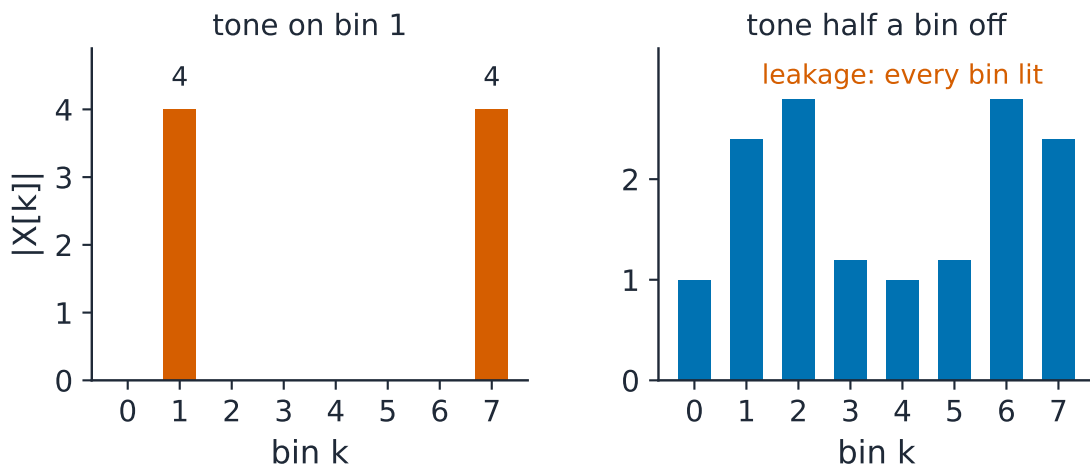
Common mistakes. Flipping the wrong signal in the table (convolution is commutative, but a hand table must stay consistent or the diagonals lie). Quoting $\sum h$ as “the gain” without saying it is the gain at DC only. And expecting a recursive smoother to settle – it approaches, never arrives.

Summary. LTI means convolution by $h[n]$; causality is $h[n] = 0$ for $n < 0$; stability is $\sum |h| < \infty$ (poles inside the unit circle); exponentials are eigenfunctions with eigenvalue $H(e^{j\omega})$.

DTFT and DFT

Why it matters. A computer holds N samples, not an integral – the spectrum must become a finite list of numbers. How that list lies to you is the single most practical topic in the course.

The idea, with a picture. The **DTFT** $X(e^{j\omega}) = \sum_n x[n]e^{-j\omega n}$ is periodic in ω with period 2π . Sampling it at N points gives the **DFT** – and sampling in frequency means the time signal is treated as periodic: that single fact explains **leakage**.



On-bin: two spikes. Half a bin off: the rectangular window's sidelobes paint every bin

Resolution is a time decision: bins are $\Delta f = f_s/N$ wide, so 1 Hz resolution at $f_s = 1$ kHz means $N = 1000$ – one second of data, no exceptions.

The derivation. Sampling the DTFT at $\omega_k = 2\pi k/N$ is multiplying it by a frequency comb; by duality the time signal becomes periodic with period N . If your tone sits exactly on a bin, the periodic copies join seamlessly – two clean spikes. If it sits between bins, the copies disagree and the disagreement smears energy into every bin. Windowing the data with $w[n]$ convolves the true spectrum with $W(e^{j\omega})$: the rectangle's sidelobes are only -13 dB down, so the smear is loud; the **Hamming window** suppresses its sidelobes to -43 dB, buying a clean spectrum at the price of a wider main lobe.

Worked example 1 – the smear, quantified. An 8-point DFT of a cosine at exactly bin 1 ($f = f_s/8$): $|X| = 4$ at bins 1 and 7, zero elsewhere ✓ (all 8 samples describe whole periods). Slide the tone half a bin ($f = 1.5 \cdot f_s/8$): the magnitudes compute to 2.40, 2.80, 1.19, 1.0, ... across every bin ✓ – a tone between bins is visible everywhere, and a small signal beside a large one vanishes into its sidelobes. That is leakage's real cost.

Worked example 2 – the resolution budget. At $f_s = 1$ kHz: $N = 100$ gives 10 Hz bins – two tones 6 Hz apart are ONE bump, however clean the plot looks. $N = 1000$ gives 1 Hz bins for one second of data. Zero-padding to 4000 points draws the same bump with $40\times$ more ink: padding interpolates the same spectrum, it resolves nothing new. Resolution was decided when you chose how long to measure.

Common mistakes. Reading bin numbers as hertz – the map is $f = k \cdot f_s/N$. Believing zero-padding buys resolution. And forgetting the DFT assumes your N samples repeat forever: the edge discontinuity between copy and copy is itself a broadband signal.

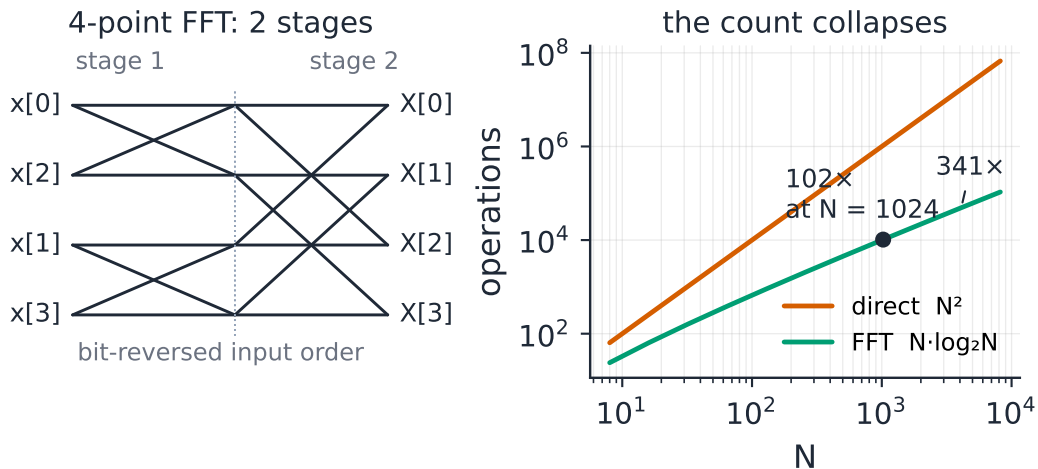
Summary. $X[k] = \text{DTFT sampled at } 2\pi k/N \Leftrightarrow \text{time periodic with period } N$; off-bin

tones leak through the window's sidelobes (rectangle -13 dB, Hamming -43 dB); $\Delta f = f_s/N$ is bought with measurement time, never with padding.

The FFT

Why it matters. The DFT costs N^2 complex operations – a 1024-point transform is a million multiplies. The **FFT** is the algorithm that made digital signal processing a subject at all.

The idea, with a picture. Factor an N -point DFT into two $N/2$ -point DFTs plus $N/2$ combining multiplies – and recurse. The **butterfly** is the combining unit: $\log_2 N$ stages of $N/2$ butterflies, each one multiply-and-add.



Same answer, 102× fewer operations at $N = 1024$ – and the gap widens with N

The derivation. Split $x[n]$ into even and odd samples: $X[k] = \sum_{\text{even}} x[n]W_N^{kn} + W_N^k \sum_{\text{odd}} x[n]W_N^{kn}$ with the **twiddle factor** $W_N = e^{-j2\pi/N}$. Each sum is an $N/2$ -point DFT (Danielson–Lanczos); the odd arm needs only the factor W_N^k to recombine. Recurse $\log_2 N$ deep. The even/odd splitting runs backwards through the address bits – hence **bit-reversed** input order for the decimation-in-time variant.

Worked example 1 – the count. Counting complex multiply–add operations: direct $N^2 = 1024^2 = 1\,048\,576$; FFT $N \log_2 N = 1024 \times 10 = 10\,240$ – **102× fewer** ✓. At $N = 4096$: 4096^2 against $4096 \times 12 = \mathbf{341\times}$ ✓. The gap widens with N : the FFT is $O(N \log N)$, and the whole order of the computation changes.

Worked example 2 – the spectrum analyser. 1 Hz bins at $f_s = 1$ kHz needs $N \geq 1000$: choose $N = 1024$ (a power of two), giving $\Delta f = 1000/1024 = 0.977$ Hz ✓ just enough. At ten nanoseconds per operation the transform finishes in about $100 \mu\text{s}$ – real time against a 1.024 s record. Bin 100 of that DFT is frequency $100 \times 0.977 = 97.66$ Hz ✓ – the bin-to-hertz mapping is $k \cdot f_s/N$, and forgetting it is the classic lab error.

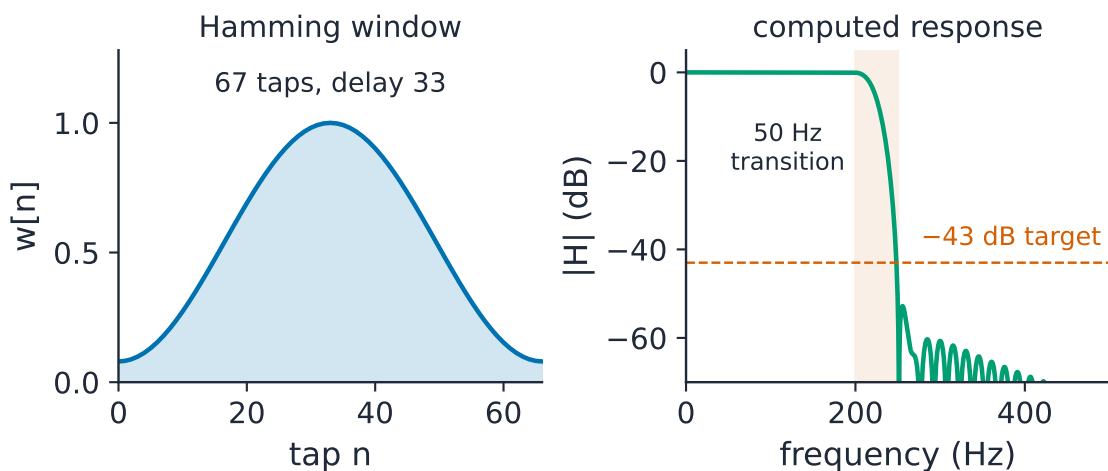
Common mistakes. Assuming N must be a power of two (mixed-radix and Bluestein handle any N , the power-of-two case is just the fastest). Expecting the FFT to change the answer – it computes the same DFT, only faster. And reading bins as hertz, again.

Summary. Even/odd split + twiddle factors, $\log_2 N$ stages, $N \log_2 N$ operations: a $102\times$ speedup at $N = 1024$ that grows with N ; the mathematics is the DFT, the engineering is the butterfly.

FIR filter design

Why it matters. The **FIR** filter is the workhorse you can prove things about: unconditionally stable, and – with symmetric taps – exactly linear phase. Frequencies in a flat passband keep their relative phases; amplitude filtering can still change a waveform.

The idea, with a picture. Start from the ideal low-pass's impulse response (a sinc), truncate it to N taps with a window $w[n]$, and the window's spectral shape IS the filter's fate: its main-lobe width becomes your transition band, its sidelobes become your stopband.



The window guides the design; compute the actual filter response to verify its passband and stopband

Definitions with numbers. Peak sidelobes of the windows themselves: rectangular -13 dB, Hann -31 dB, Hamming -43 dB, Blackman -58 dB. Transition width $\approx 3.3 \cdot f_s/N$ for Hamming. Real symmetric taps $h[n] = h[N - 1 - n]$ give **linear phase** with group delay $(N - 1)/2$ samples wherever the response is nonzero. Phase may jump by π when the real amplitude factor changes sign; phase and group delay are undefined at response zeros.

The derivation. Truncating $h[n]$ is multiplying by $w[n]$; in frequency the true response convolves with $W(e^{j\omega})$. So the implemented response is the ideal brick wall smeared by the window: passband ripple, a transition of one main-lobe width, and

a stopband whose attenuation depends on the convolution, cutoff and tap count. A window sidelobe value is not the resulting filter's stopband guarantee. Use the window table and transition estimate to choose a starting design, then compute its response. (The alternation theorem behind Parks–McClellan makes the equiripple design optimal where windows are merely convenient.)

Worked example 1 – size the filter. A Hamming FIR at $f_s = 1$ kHz must pass up to 200 Hz and attenuate from 250 Hz. Choose a 225 Hz ideal cutoff in the 50 Hz transition: $N = 3.3 \cdot 1000/50 = 66 \checkmark$ – take 67, odd, for a Type-I symmetric filter with delay $(67 - 1)/2 = 33$ samples (33 ms). At $N = 31$ the same window gives $3.3 \cdot 1000/31 = 106$ Hz transition: the two designs bracket the specification; the delay is a cost to record, not a footnote.

Worked example 2 – verify the design. Form the 67 taps from $h[n] = 0.45 \text{sinc}(0.45(n - 33))w[n]$, where $\text{sinc}(x) = \sin(\pi x)/(\pi x)$ and its value at zero is 1. Normalise the tap sum to 1. Evaluate $H(f) = \sum_{n=0}^{66} h[n]e^{-j2\pi f n/1000}$. On a 0.025 Hz grid, the computed passband (0–200 Hz) ranges from -0.034 to $+0.0045$ dB, and the largest stopband level (250–500 Hz) is about -52.3 dB. These sampled checks meet the targets of ± 0.05 dB ripple and at most -43 dB stopband level; a finer grid or extremum search checks between the samples. If a design instead gives -37 dB, it fails that target: inspect its taps, cutoff, window, tap count and measurement band rather than guessing a unique cause.

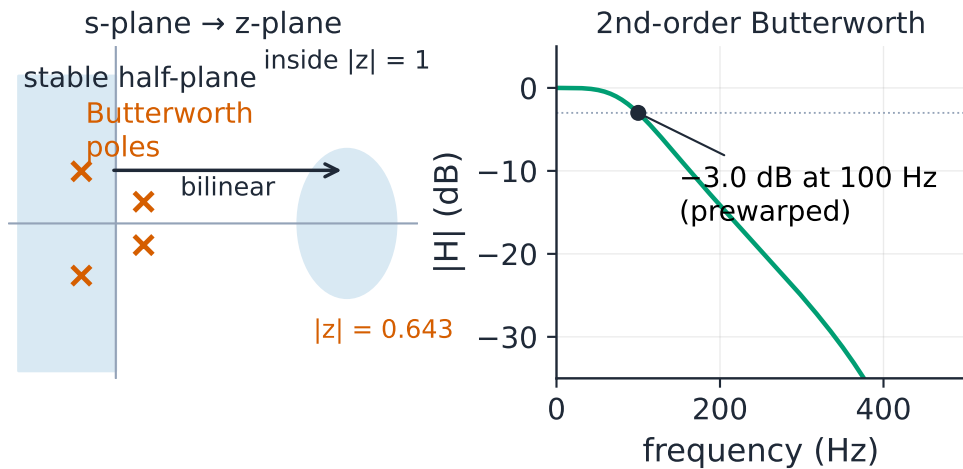
Common mistakes. Designing N from the cutoff and never measuring. Choosing an even N when a Type-I (odd, symmetric) design was intended. Forgetting the $(N - 1)/2$ delay in a control loop. And asking windows for optimality – they are convenient, not best.

Summary. Window the ideal sinc: the sidelobe table guides window choice, $3.3f_s/N$ fixes the transition, symmetry fixes linear phase at delay $(N - 1)/2$; then measure the response against every promised number.

IIR filter design

Why it matters. The same 100 Hz cutoff that costs an FIR 67 taps costs a biquad IIR 5 coefficients – because it inherits forty years of analog filter lore through one substitution.

The idea, with a picture. The bilinear transform $s = \frac{2}{T} \cdot \frac{1-z^{-1}}{1+z^{-1}}$ maps the whole analog left half-plane inside the unit circle: every stable analog filter becomes a stable digital one, automatically.



Stability comes free; the price is frequency warp, repaid by prewarping

The price is frequency **warp**: the analog axis $-\infty \dots \infty$ wraps once around the unit circle, compressively, so every critical frequency must be **prewarped** $\omega_a = \frac{2}{T} \tan(\omega_d T/2)$ before use.

The derivation. Set $s = j\omega_a$ in the transform: it solves to $z = \frac{1+sT/2}{1-sT/2}$, whose magnitude is 1 exactly when $\text{Re } s = 0$ – the analog imaginary axis maps onto the unit circle, left half-plane strictly inside. The tangent relation is the warp: near dc $\tan \theta \approx \theta$ (negligible distortion), but a digital frequency approaching Nyquist compresses toward the analog infinity. Butterworth, Chebyshev and elliptic tables transplant intact because only the frequency axis bent – the polynomial geometry survived.

Worked example 1 – the biquad. A 2nd-order Butterworth low-pass at 100 Hz, $f_s = 1$ kHz: prewarp $\omega_a = 2000 \tan(\pi \cdot 100/1000) = 649.8$ rad/s ✓. With s normalised by ω_a , the substitution constant is $K = 1/\tan(0.1\pi) = 3.078$ and the prototype $H(s) = 1/(s^2 + 1.4142s + 1)$ transforms to

$$H(z) = \frac{1 + 2z^{-1} + z^{-2}}{14.82 - 16.94z^{-1} + 6.12z^{-2}}$$

Worked example 2 – verify, then pay. DC gain: the denominator sums to $14.82 - 16.94 + 6.12 = 4.00$, so $H(1) = 4/4 = 1.000$ ✓ (Butterworth is flat at dc). The response at exactly 100 Hz evaluates to -3.0 dB ✓ – the prewarp put the -3 dB point precisely on the promised frequency, which is the entire purpose of prewarping. The poles sit at $|z| = 0.643$, inside the unit circle because the mapping guarantees it ✓. Skip the prewarp – design the analog filter at $2\pi \times 100 = 628.3$ rad/s – and the -3 dB point lands at 96.9 Hz; set $T = 1$ as well ($K = 1$) and it cuts at 250 Hz, reading a complacent -0.05 dB at 100 Hz while missing the specification entirely – the numbers always show the difference. The bill: 5 coefficients against the FIR’s 67 taps, but the phase is nonlinear near the cutoff and, in fixed point, the denominator coefficients are the fragile ones – a rounding of 6.12 moves the cutoff measurably. Efficiency, phase and sensitivity trade in one line.

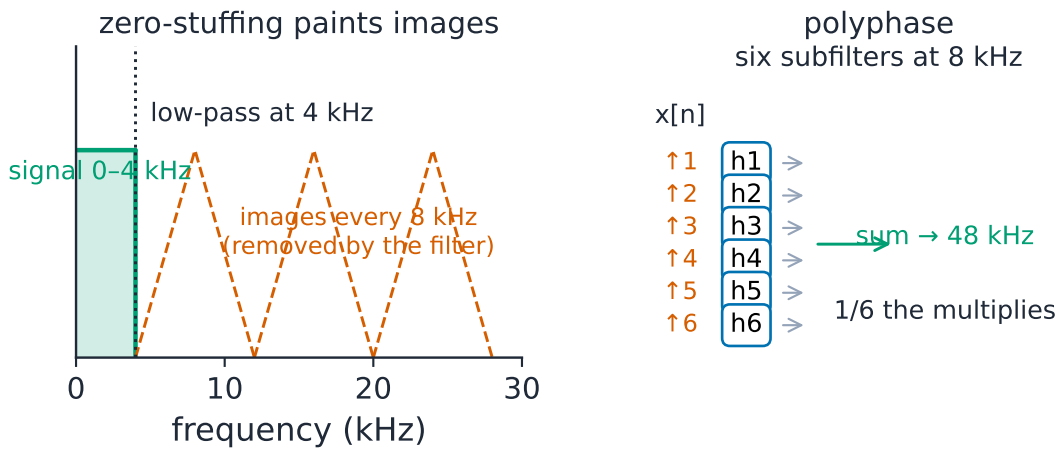
Common mistakes. Skipping the prewarp (the cutoff lands low – 96.9 Hz for 100 Hz at 1 kHz is small, 2.86 kHz for 4 kHz at 10 kHz is fatal). Setting $T = 1$ while still quoting analog hertz. Testing stability by luck instead of trusting the mapping. And expecting linear phase – IIR phase near its poles is the opposite of flat.

Summary. Bilinear: LHP $\rightarrow$ unit disk (stability free), frequencies warped by $\frac{2}{T} \tan(\cdot)$ (prewarp every critical frequency); the Butterworth biquad lands as $H(z) = \frac{1+2z^{-1}+z^{-2}}{14.82-16.94z^{-1}+6.12z^{-2}}$ (dc 1.000, -3.0 dB at 100 Hz, poles at 0.643).

Multirate processing

Why it matters. Sample rate is compute budget: a filter running at 48 kHz costs six times the same filter at 8 kHz. Multirate is the art of doing the work where it is cheap.

The idea, with a picture. **Decimation** by M (keep every M th sample) must first **low-pass** below $f_s/(2M)$ – or the folded spectrum aliases, irreversibly. **Interpolation** by L (stuff $L - 1$ zeros between samples) must then low-pass below the old Nyquist to kill the images.



Upsampling paints images every 8 kHz; one low-pass erases them; polyphase does the same arithmetic for one-sixth the multiplies

The derivation. Zero-stuffing by L in time makes $X_{up}(e^{j\omega}) = X(e^{j\omega L})$ – the same spectrum replicated L times around the circle: those are the images, and they are exactly as real as the signal. Keeping 1 sample in M wraps M spectral copies onto each other: whatever lives above the new Nyquist folds back into the band. The two low-passes are not bureaucracy; each erases a mathematical certainty. **Polyphase**

is then pure algebra: the interpolation filter's L phase-sets each see only every L th input, so one filter at $f_s L$ becomes L subfilters at f_s – identical arithmetic, $1/L$ the multiplies.

Worked example 1 – 8 kHz speech to 48 kHz. Interpolate by 6, then a low-pass at 4 kHz (the old Nyquist) removes the images at 8k, 16k, 40k... Run naively, that filter executes at 48 kHz – six times the work. Polyphase reorganises it into 6 subfilters each running at 8 kHz: the same arithmetic, $1/6$ the multiplies ✓ – and this is not an approximation, it is the same computation scheduled differently.

Worked example 2 – one converter, not two. Resampling 48 kHz audio up to 72 kHz needs $L = 3$, $M = 2$ ($48 \times 3/2 = 72$ ✓). Two passes would need an interpolator, a filter, a decimator and another filter. One rational converter does it: upsample by 3 to 144 kHz, apply a single low-pass at 24 kHz (the input's own Nyquist – it stops the ↑ 3 images and the ↓ 2 aliases at once), keep every second sample. One filter, one pass, the same result.

Common mistakes. Decimating before filtering (the alias is then not noise, it is permanent). Forgetting the $1/L$ amplitude loss of zero-stuffing (compensate with a gain of L in the filter). And treating polyphase as an approximation – it is exact bookkeeping.

Summary. ↓ M needs a low-pass at $f_s/2M$ first; ↑ L needs a low-pass at the old Nyquist after; polyphase delivers the same arithmetic for $1/L$ the multiplies; rational

L/M converters merge into one filter at the intermediate rate.

Signal-processing project

Why it matters. The course closes the way practice opens: with a specification, an implementation, and a measurement – and the honest gap between the three.

The idea, with a picture. Build a tone detector as a pipeline, each stage verified alone before the next exists: anti-alias filter $\rightarrow$ window $\rightarrow$ FFT $\rightarrow$ threshold. A defect found in a stage costs an hour; found in the integrated system, a week of bisection.

verify each stage alone — then measure the whole



spec	measured	margin
1 Hz	0.977 Hz	ok
<1 s	1.024 s	✗
-10 dB	-10 + 27 dB	ok

Specify first, verify each stage alone, then measure the whole against every spec line

The method. Write the specification first – a tone detector: 1 Hz bins, under 1 s latency, detection at -10 dB SNR – implement in verified stages, and report **measured performance** against every line, failures included.

Worked example 1 – the spec bites back. 1 Hz bins at $f_s = 1$ kHz needs $N \geq 1000$ samples – that is already 1.0 s of data, and the spec says under 1 s latency. The conflict surfaces at design time, which is when you want it: $N = 1024$ gives 0.977 Hz ✓ but a 1.024 s record ✗; $N = 1000$ gives 1.000 Hz – exactly on the line. The margin table records the trade; nothing is silent.

Worked example 2 – can -10 dB even be detected? A sinusoid's coherent energy lands in one bin while white noise spreads over all $N/2$: the bin **processing gain** is $10 \log_{10}(N/2)$ – at $N = 1024$ that is 27 dB ✓. An input SNR of -10 dB becomes about +17 dB in the decision bin: detectable with margin – the window costs a few dB of that gain and it still clears – and the number came from the transform, not from hope.

Common mistakes. Measuring only the happy path. Silent goals – spec lines that exist in the code but not in the report. Tuning the detector to the test tone (overfitting is lying with extra steps). And skipping per-stage checks: integration hides what verification would show.

Summary. Specify first, verify stages alone, trust measured margins; processing gain

$10 \log_{10}(N/2)$ sets feasibility.