

THE LOST ALGORITHMS

Volume Six — Sound

By

MARTIN CARR

INDEPENDENT PUBLISHER

Copyright © 2026 by Martin Carr.

All rights reserved.

No part of this book may be reproduced, digitalized, or transmitted in any form without written permission from the publisher, except in the case of brief quotations embodied in critical reviews.

Publisher: Independent Publisher

ISBN Placeholder

TABLE OF CONTENTS

Preface: Making Sound Small Enough to Send	4
How to Read This Book	7
A Note on Patents and the Public Domain	8
Part I — The Voice: Coding Speech	9
Predicting the Next Sample (Linear Predictive Coding)	10
Switching Predictors (Adaptive Switched-Predictive Quantization)	16
A Codebook for the Voice (Code-Excited Linear Prediction)	21
Part II — The Room and the Adaptive Ear	26
A Room in a Few Numbers (Sound-Field Encoding)	27
Learning to Cancel an Echo (The LMS Adaptive Filter)	33
Appendix — The Patents of This Volume	39
A Closing Note on the Companion Code	42
The Whole Shelf	43

PREFACE: MAKING SOUND SMALL ENOUGH TO SEND

Sound is a torrent of numbers. A single second of ordinary audio is tens of thousands of samples; a phone call, an hour of speech, a room full of voices — raw, it is far too much to send down a wire or hold on a chip. And yet you make crystal-clear calls over connections that carry a tiny fraction of that raw data, hear music streamed at a whisper of its original size, and talk hands-free in a car without hearing your own voice echo back. None of that is luck. It is a specific body of algorithms about how sound works — how a voice is produced, how a room carries it, how the ear perceives it — turned into ways of representing sound in far fewer numbers.

This final volume of the series is about those algorithms, and they are among the most ingenious in the whole shelf, because they do not merely compress data blindly — they model the source. The great insight of speech coding is that you are not sending a

waveform; you are sending a description of how to remake it. The human vocal tract is a tube with a few resonances, driven by the buzz of the vocal cords or the hiss of breath. Model that tube with a handful of numbers, describe its excitation with a codebook, and you can rebuild recognisable speech from a data rate hundreds of times smaller than the raw waveform. That idea — analyse the source, transmit the model, resynthesise at the far end — is why your voice fits down a wire, and it powers every mobile phone call ever made.

The volume has two halves. The first is the voice: linear prediction, which models the vocal tract; switched prediction, which adapts as the sound changes; and the code-excited quantisation that describes the excitation with a codebook — together, the anatomy of a speech codec. The second is the room and the ear: encoding a whole sound-field's spatial directions in a few channels, and the adaptive filter that learns, in real time, to cancel an echo it has never heard before. Each algorithm is anchored to a patent that has expired into the public domain, and each is handed to you as code you can run — watching a signal shrink to a few coefficients and rebuild, a source pan around a ring of speakers, an echo fade to silence.

The honesty this series keeps holds here too. Linear predictive coding was developed by Atal, Itakura and others in the 1970s; the least-mean-squares adaptive filter is Widrow and Hoff's from 1960; code-excited linear prediction is Atal and Schroeder's 1985 breakthrough. These are named academic inventions, and where a

chapter cites a specific patent, the Story says who first had the idea and what the filing added. History with receipts, especially where the history is this distinguished.

And every algorithm runs on nothing but Python's standard library — no audio libraries, no DSP toolkits, no numpy. The signals here are short and synthetic so the demonstrations are fast and the mathematics is visible, but the algorithms are the real ones, and you can run every one on a five-pound board and watch the machinery that carries the human voice around the world work in front of you.

HOW TO READ THIS BOOK

No mathematics beyond high-school algebra, and every idea — a filter, a predictor, a codebook, a decibel — is built from zero as it arrives. Install the companion library:

```
pip install patentml
```

Every chapter names the exact module it draws from, and every number printed in these pages — every prediction gain, every "reconstruction error", every decibel of echo suppression — came from a real run of that module's demo. Run it yourself and you will see the same.

Read in order or dip in anywhere. Part I builds a speech codec from its parts; Part II turns to the space sound lives in and the echoes it makes.

A NOTE ON PATENTS AND THE PUBLIC DOMAIN

A patent trades a twenty-year monopoly for the full disclosure of how an invention works; when it expires, the knowledge belongs to everyone. Speech and audio coding were fiercely patented — the codecs in mobile phones were licensed for fortunes — and one by one those foundational patents have lapsed. Every filing in this book has reached the end of its term. The implementations here are the author's own MIT-licensed code; the ideas are everyone's. This is a map, not legal advice: confirm the status of any specific patent before building a product on it.

Now: how do you send a voice using a fraction of its samples? You stop sending the sound, and start sending a model of the mouth that made it.

PART I — THE VOICE: CODING SPEECH

The human voice is the most-compressed signal in the world — every mobile phone call sends it at a tiny fraction of its raw size, and it still sounds like you. The trick is not to compress the sound but to compress a model of the mouth that made it. A voice is a buzz or a hiss shaped by the tube of the vocal tract, and if you can describe that tube with a few numbers and its excitation with a few more, you can rebuild recognisable speech from almost nothing. This part assembles a speech codec from its pieces: the linear predictor that models the vocal tract, the switched predictor that keeps up as the sound changes, and the code-excited quantiser that describes what drives the tube. Three chapters that, together, are the anatomy of every phone call.

CHAPTER 1

PREDICTING THE NEXT SAMPLE (LINEAR PREDICTIVE CODING)

THE FILING

US Patent 5,414,796 — "Variable rate vocoder."

Filed 1993. Expired 2013; public domain.

The single most important idea in speech coding is that a speech sample is predictable from the samples just before it — because the vocal tract, the tube between your voice-box and your lips, changes shape only slowly and resonates in a way that makes each moment of sound a near-echo of the last. This patent concerns a vocoder: a coder that represents speech not as a waveform but as the coefficients of a predictor (the vocal-tract model) plus a low-rate excitation. Its engine is linear predictive coding, and it is why a voice fits down a wire.

THE IDEA

Sit with the waveform of a vowel and you notice it is almost periodic and very smooth — each sample is close to a weighted combination of the previous handful. Linear prediction turns that observation into a compression scheme: predict each sample $x[n]$ as a weighted sum of the previous p samples, and transmit only the small residual — the part the prediction got wrong.

$$\text{prediction}[n] = -(a_0 \cdot x[n-1] + a_1 \cdot x[n-2] + \dots + a_{p-1} \cdot x[n-p])$$

$$\text{residual}[n] = x[n] - \text{prediction}[n]$$

If the predictor is good, the residual is far smaller than the signal — its energy squeezed down — so it takes far fewer bits to send. The receiver, given the coefficients and the residual, runs the same recurrence in reverse to rebuild the exact signal. The whole game is finding the best coefficients $a_0 \dots a_{p-1}$, and there is a way to do it: compute the signal's autocorrelation (how similar it is to shifted copies of itself), then solve for the coefficients with the Levinson–Durbin recursion, an efficient method that falls straight out of the structure of the problem.

Here is the deep part, the part that makes this speech coding and not just signal compression. Those p coefficients are not arbitrary numbers — they are a model of your vocal tract. They describe the resonances (the "formants") of the tube your voice passed through at

that instant. So linear prediction does not merely compress the sound; it separates the source: the coefficients capture the filter (the shape of your mouth and throat), and the residual captures the excitation (the buzz of your vocal cords or the hiss of breath that drove it). Send a few coefficients for the tract and a cheap description of the excitation, and the far end can resynthesise your voice. You are not transmitting sound. You are transmitting a tiny, moment-by-moment blueprint of the mouth that made it.

THE STORY

The honesty this chapter owes: linear predictive coding of speech was developed in the early 1970s, principally by Bishnu Atal and Manfred Schroeder at Bell Labs and Fumitada Itakura in Japan — foundational, published, Nobel-of-the-field work that transformed telecommunications. The idea of analysing speech into a source and a filter is older still, reaching back to Homer Dudley's Vocoder at Bell Labs in the 1930s (and its show-stopping cousin the Voder, "spoken" by a human operator at the 1939 World's Fair). What patents like US5414796 industrialise, decades later, are specific vocoders built on LPC — here a variable-rate one that spends more bits on complex speech and fewer on silence, exactly the kind of coder that made digital mobile telephony practical. And practical it was: LPC is the foundation of essentially every mobile phone call. The GSM codec, the CDMA codecs, voice-over-IP, secure military voice (the LPC-10 standard), and the speech front-ends of countless

devices all rest on linear prediction. When you speak into a phone, an LPC analyser is deriving your vocal-tract coefficients dozens of times a second and sending them in place of your voice. A 1970s insight about the predictability of a vowel became the way the human voice travels the earth.

THE CODE

Autocorrelation, Levinson–Durbin for the coefficients, then analysis (`signal !' coeffs + residual`) and synthesis (`coeffs`). Here is the analysis from `patentml`'s `lpc`:

```
def analyse(x, order=8):
    r = autocorrelation(x, order)
    a, _ = levinson_durbin(r, order)      # solve for the vocal-tract
coefficients
    residual = []
    for n in range(len(x)):
        pred = -sum(a[k] * x[n - k] for k in range(1, order + 1) if n - k
>= 0)
        residual.append(x[n] - pred)      # transmit only what
prediction missed
    return a, residual
```

The demo builds a synthetic "voiced" sound — pitch pulses through a resonant vocal-tract filter — then analyses and resynthesises it. It

prints:

Linear predictive coding / vocoder core (from US5414796, expired 2013):

signal length : 400 samples

LPC order : 8 -> 8 coefficients model the vocal tract

prediction gain : 9.0 dB (residual carries far less energy than the signal)

analysis+synthesis reconstruction error: 0.00e+00 (lossless)

The two numbers that matter tell the whole story. Prediction gain: 9.0 dB — the residual carries far less energy than the original signal, because eight coefficients successfully predicted most of each sample from its predecessors; that energy reduction is exactly the compression, the reason you can send the small residual instead of the big waveform. And reconstruction error: 0.00 — given the coefficients and the residual, synthesis rebuilt the signal exactly, bit-for-bit, because analysis and synthesis run the same recurrence forwards and backwards. Four hundred samples were reduced to eight vocal-tract numbers plus a low-energy residual, and rebuilt perfectly. The module fuzz-tests two thousand signals, confirming that analyse-then-synthesise reconstructs exactly, and reports PASS.

THE INHERITANCE

Linear predictive coding is public domain — the foundational work long since, this vocoder patent since 2013 — and it is the tool for compressing or modelling any signal with short-term structure, speech above all but also any smooth, resonant, or auto-correlated data. Its gift is conceptual as much as practical: it teaches source–filter separation, the idea that the smartest way to compress a signal is often to model the thing that produced it and send the model. That idea reaches far beyond audio — it is how you compress anything you understand the physics of. But its home is the human voice, and there it is foundational: strip away the codecs, the standards, the acronyms of mobile telephony, and at the bottom is this — predict the next sample, send only the surprise, and describe the mouth in eight numbers. That engine of the world's voice is now yours, and the next two chapters build the rest of the codec on top of it.

CHAPTER 2

SWITCHING PREDICTORS (ADAPTIVE SWITCHED-PREDICTIVE QUANTIZATION)

THE FILING

US Patent 6,122,608 — "Method for switched-predictive quantization."

Filed 1997. Expired; public domain.

Chapter 1's predictor was fixed for the whole signal. But real sound changes — a voice moves from a vowel to a consonant, music from a held note to an attack, and a predictor tuned for one is wrong for the other. This patent's answer is to keep several predictors on hand and switch to whichever fits best at each moment. Its claims describe choosing, per block, the predictor that leaves the smallest residual, quantising that residual, and sending the choice along with it. It is adaptation by selection — the coder that keeps up with a changing signal.

THE IDEA

A single linear predictor is a compromise. Tuned to the smooth body of a vowel, it fumbles the sharp edge of a consonant; tuned to attacks, it wastes its accuracy on the smooth parts. Signals are non-stationary — their character shifts from moment to moment — and a fixed predictor is always slightly wrong somewhere.

Switched prediction refuses the compromise. Keep a small bank of predictors, each good at a different kind of signal behaviour — one for steady trends, one for sharp changes, one for near-constant stretches. Then process the signal in short blocks, and for each block, try every predictor and pick the one that leaves the smallest residual. Quantise that residual to a chosen step size, and transmit two things: which predictor won, and the quantised residual. The decoder, told which predictor was used, runs it and adds back the residual to reconstruct the block. Because the coder always uses the best-fitting predictor for each moment, the residuals stay small even as the signal's character changes — and small residuals mean few bits.

The quantisation is where the bits are actually saved, and it is a controlled trade: the residual is rounded to the nearest multiple of a step size, so it takes few bits, at the cost of a small, bounded error — never more than half a step. This makes the scheme gently lossy (the reconstruction is within half a quantiser step of the original, not exact), which is exactly the right trade for audio, where a tiny,

inaudible error buys a large saving. Switching predictors keeps the residual small; quantising it keeps the bits small; and the half-step error bound keeps the loss imperceptible.

THE STORY

Switched and adaptive prediction belongs to the family of differential and adaptive PCM techniques (DPCM, ADPCM) that were central to digital audio and telephony from the 1970s onward — the idea of transmitting differences rather than samples, and adapting the predictor or the quantiser to the signal. The international ADPCM standards (such as G.726, which halved the bit-rate of digital telephone speech) are built on exactly this principle of adapting to the signal moment by moment. US6122608 is a 1997 refinement of that lineage — switching among predictors per block to track a non-stationary signal — and its relatives run in the adaptive coders inside telephony, in audio codecs, and in any system that transmits a changing signal as a stream of small, predicted differences. The deep idea it carries forward is that a good coder is not a fixed machine but an adaptive one, constantly re-choosing how to model the signal in front of it. That principle — measure, select the best model, code the residual — echoes through all of modern signal compression.

THE CODE

For each block, try every predictor, keep the one with the smallest residual, quantise, and record the choice. The demo codes a rising-then-levelling signal and reports the reconstruction. It prints:

Switched-predictive quantization (from US6122608, expired):

```
signal[:6] : [0.0, 3.27, 6.18, 8.41, 9.72, 9.95]  
reconstruct : [0.0, 3.5, 6.0, 8.5, 9.75, 9.875]  
max abs error: 0.239 (<= half a step)  
predictors used: [0, 2, 3]
```

The reconstruction tracks the original closely — and the two revealing lines are the last two. The maximum error is 0.239, comfortably within half of the quantiser step, exactly the bound the quantisation promises: the loss is real but tightly controlled, the small price for sending few bits. And predictors used: [0, 2, 3] — the coder did not stick with one predictor; across the signal it switched among three different ones, choosing at each block whichever left the smallest residual as the signal rose, curved, and levelled off. That switching is the whole idea, visible in a single line: adaptation by selection, keeping the residual small through a changing signal. The module fuzz-tests three thousand signals, confirming that reconstruction always stays within half a quantiser step of the original, and reports PASS.

THE INHERITANCE

Switched-predictive quantization is public domain, and it is the tool for coding any changing signal efficiently at a controlled quality: adaptive audio and speech coding, telemetry, sensor streams, any waveform whose character shifts and where a single fixed model would leave too much residual. Its lessons are two, and both are general. First, adapt: when a signal is non-stationary, do not force one model on all of it — keep several and switch to the best, block by block. Second, quantise deliberately: lossy compression is not sloppiness but a bounded trade, rounding to a step whose size you choose so the error stays below what anyone will notice. Together they turn Chapter 1's fixed predictor into a coder that keeps pace with real sound. That adaptive engine, and a working switched quantiser, are yours now that this filing has expired — the second piece of the speech codec, sitting between the vocal-tract model and the codebook that comes next.

CHAPTER 3

A CODEBOOK FOR THE VOICE (CODE-EXCITED LINEAR PREDICTION)

THE FILING

US Patent 5,371,853 — "Method and system for CELP speech coding and codebook for use therewith."

Filed 1991. Expired 2011; public domain.

Chapter 1 modelled the vocal tract and left a residual — the excitation that drove it. The final and cleverest step of speech coding is to describe that excitation not by transmitting it, but by choosing it from a shared codebook: sender and receiver both hold the same book of candidate excitation vectors, and the sender transmits only the index of the one that best rebuilds the sound. This patent is Code-Excited Linear Prediction — CELP — the technique that pushed speech to astonishingly low bit-rates and became the heart of nearly every modern voice codec.

THE IDEA

Recall the picture from Chapter 1: speech is a filter (the vocal tract, a few LPC coefficients) driven by an excitation (the residual — the buzz or hiss). The coefficients are cheap to send. The excitation is the expensive part, because it is a whole signal of its own. CELP's insight is to stop sending the excitation and start selecting it.

Both the encoder and the decoder hold the same codebook — a fixed dictionary of candidate excitation vectors. To code a frame of speech, the encoder does something almost audacious: it takes each candidate vector in the codebook, runs it through the LPC vocal-tract filter, and listens to the result — comparing the synthesised speech to the original. It picks the codebook entry whose synthesised output best matches the real speech, and transmits just that entry's index. This is "analysis by synthesis": the encoder chooses the excitation by trying candidates and keeping the one that sounds closest. The decoder, receiving the index, looks up the same vector, drives the same filter, and produces the same speech.

Two refinements make it powerful, and patentml's module implements both. Multi-stage vector quantisation uses several codebooks in succession, each correcting the residual error of the last, so a few small codebooks combine to represent an enormous variety of excitations cheaply. And prediction across frames — since consecutive speech frames are similar, predict the current frame's

parameters from the previous decoded frame and quantise only the difference. The result is a coder that describes a frame of the human voice in a handful of indices: a few for the vocal-tract filter, a few for the excitation codebook, and a pitch. Speech at a few thousand bits per second, still clearly your voice.

THE STORY

The honesty this chapter owes: Code-Excited Linear Prediction was invented in 1985 by Manfred Schroeder and Bishnu Atal — the same Atal from Chapter 1's linear prediction — a landmark that combined LPC with vector-quantised, analysis-by-synthesis excitation and defined the shape of speech coding for the next thirty years. Patents such as US5371853 industrialised specific CELP coders and codebooks through the early 1990s. CELP and its descendants utterly dominate voice communication: the AMR codec in GSM and 3G/4G phones, the codecs in nearly every mobile standard, the speech modes of Speex and Opus (the codec behind most modern voice-over-IP, video calls, and internet telephony), and secure-voice systems worldwide are all CELP-family coders. Very nearly every phone call and voice chat you have ever had was carried by a code-excited linear predictor choosing excitation vectors from a codebook, dozens of times a second, to rebuild your voice at the far end. It is one of the most successful algorithms in the history of communication, and its foundational patents have now expired.

THE CODE

Multi-stage codebooks quantise the excitation; prediction across frames sends only the change. The demo codes vectors with one and two stages and shows prediction shrinking a signal. It prints:

Predictive multi-stage VQ / CELP core (from US5371853, expired 2011):

reconstruction MSE 1 stage=0.3644 2 stages=0.0728

AR(1): signal energy=1.010 prediction residual energy=0.185

Both refinements are visible in two lines. Multi-stage quantisation: with one codebook the reconstruction error (MSE) is 0.3644; add a second codebook to correct the first stage's residual and the error drops fivefold to 0.0728 — successive small codebooks combining to represent the excitation far more accurately than either alone, the multi-stage idea that keeps CELP's codebooks small. And prediction across frames: a correlated signal with energy 1.010 leaves a prediction residual of energy just 0.185 — predicting each frame from the last shrank what needs to be coded to a fraction, so only that small difference is quantised. Together they are the machinery that describes a frame of voice in a few indices. The module verifies that adding stages monotonically reduces error, that prediction shrinks the coded energy, and that all codebook indices are valid, reporting PASS.

THE INHERITANCE

Code-excited linear prediction is public domain — the foundational technique long since, these codebook patents since the early 2010s — and while its home is speech, its two ideas are broadly powerful. Analysis by synthesis — choosing a code by trying candidates through the reconstruction and keeping the best-sounding — is a template for any coder that can afford to search: don't compute the answer, select it by testing. And multi-stage, predictive vector quantisation is a general high-fidelity compression tool, the same product-and-multi-stage quantisation now reused far from audio (as Volume 5's compression chapter noted, in the vector search behind modern AI). But it is worth ending Part I on what these three chapters have built together: a vocal-tract model (Chapter 1), an adaptive predictor for the residual (Chapter 2), and now a codebook for the excitation (Chapter 3) — which is, precisely, a speech codec, the thing inside every phone. You have assembled from public-domain parts the machine that carries the human voice around the world. Part II turns from the voice to the space it moves through.

PART II — THE ROOM AND THE ADAPTIVE EAR

Part I compressed the voice by modelling the mouth. This part turns outward to the space sound lives in and the way it reaches us. First: how do you capture not just a sound but its direction — a whole three-dimensional sound-field, a source that seems to come from a particular place — and store it in a handful of channels that any speaker layout can replay? Then: how does a device learn, in real time, to remove an echo or a noise it has never heard before, adapting to an unknown acoustic path on the fly? Two chapters on the geometry of hearing — the room, and the ear that cancels what it should not hear.

CHAPTER 4

A ROOM IN A FEW NUMBERS (SOUND-FIELD ENCODING)

THE FILING

US Patent 5,632,005 — "Encoder/decoder for multidimensional sound fields."

Filed 1995. Expired; public domain.

Stereo has two channels for two speakers; surround sound has a channel per speaker; and both are stuck with the layout they were mixed for. This patent describes something more elegant: encode the sound-field itself — the directions sound arrives from — into a small, layout-independent set of channels, and then decode that to whatever speakers you happen to have. Its claims describe an encoder/decoder for multidimensional sound fields, capturing spatial audio in a compact form that any speaker arrangement can reproduce. It is how you put a room in a few numbers.

THE IDEA

The problem with channel-based audio (stereo, 5.1) is that it bakes in the speakers. A stereo mix assumes two speakers in specific places; play it on a different layout and the spatial image is wrong. What if, instead of storing "what each speaker should play," you stored "what the sound-field is" — the sound arriving from each direction — and worked out the speaker signals only at playback?

That is the idea behind sound-field encoding (the technique known as ambisonics). Instead of one channel per speaker, you store a few direction-independent channels that together describe the field: an omnidirectional channel *W* capturing the total sound pressure (everything, from everywhere), plus directional channels — *X* and *Y* for the front–back and left–right axes (and *Z* for up–down in the full 3-D case) — each a "figure-of-eight" capturing how much sound comes from along that axis. A source at a particular angle is encoded by distributing it across *W*, *X*, and *Y* according to its direction: its contribution to *X* is scaled by the cosine of its angle, to *Y* by the sine, so the few channels together record where every sound sits.

The payoff comes at playback. Given any set of speakers at any angles, you decode the field to them by projecting the encoded channels onto each speaker's direction — reconstructing, for each speaker, the sound that should emanate from its position to recreate the original field. Two speakers, five, a ring of eight, headphones —

the same few encoded channels drive any of them, because they describe the field, not the layout. You stored the room's spatial content once, in a handful of numbers, and let the decoder fit it to whatever you play it on.

THE STORY

The honesty this chapter owes: ambisonics — encoding a sound-field into direction-independent channels — was developed in the 1970s, principally by Michael Gerzon and colleagues at Oxford, a mathematically elegant approach to spatial audio decades ahead of its commercial time. Patents like US5632005 industrialised sound-field encoding/decoding through the 1990s. For years ambisonics was a connoisseur's technology, admired and little used — but it found its moment with virtual and augmented reality and 360° video, where the layout-independence that always made it elegant became essential: a VR headset must render a sound-field to headphones and rotate it as you turn your head, and a first-order ambisonic field (exactly the W, X, Y, Z channels this chapter describes) is the natural format for that. YouTube and Facebook adopted ambisonics for 360° video, VR audio engines use it, and spatial-audio formats build on it. A 1970s idea for storing a room's acoustics in four numbers became, forty years later, the way immersive audio is carried. The elegant technique waited for the application that needed it.

THE CODE

Encode sources into W, X, Y by their direction; decode by projecting onto each speaker's angle. Here is the pair from patentml's `sound_field_codec`:

```
def encode(sources):          # [(azimuth, gain)] -> B-format (W,
X, Y)
    W = X = Y = 0.0
    for az, g in sources:
        W += g * _SQRT1_2
        X += g * math.cos(az)    # how much comes from the
front-back axis
        Y += g * math.sin(az)    # ...and the left-right axis
    return W, X, Y

def decode(bformat, speaker_azimuths): # project the field onto
each speaker
    W, X, Y = bformat
    n = len(speaker_azimuths)
    return [(2.0 / n) (W * _SQRT1_2 + X * math.cos(az) + Y
math.sin(az))
            for az in speaker_azimuths]
```

The demo encodes a source and decodes it to a ring of four speakers. It prints:

Sound-field codec (from US5632005, expired):

source at 45 deg, 4 speakers (0/90/180/270):

['+0.604', '+0.604', '-0.104', '-0.104']

source at 0 deg -> loudest speaker index: 0 (the 0-deg speaker)

The spatial image is exactly right. A source placed at 45° — half-way between the 0° and 90° speakers — decodes to those two front speakers equally (both +0.604) and to the two rear speakers weakly and equally (both " 0.104): the sound sits speakers nearest its direction, precisely where 45° should be, with no speaker "owning" it. And a source at 0° decodes loudest to the speaker actually at 0°, as it must. The source was encoded into just three numbers (W, X, Y) and then placed correctly among a set of speakers the encoder never knew about — the layout-independence that is the whole point. The module fuzz-tests three thousand source angles across speaker rings of different sizes, confirming that the loudest speaker is always the one nearest the source's true direction, and reports PASS.

THE INHERITANCE

Sound-field encoding is public domain, and it is the tool for any spatial audio that must outlive its playback layout: VR and AR, 360° video, immersive installations, game audio that rotates with the listener, any system where you want to capture where sounds are and

decide how to play them later. Store the field in a few direction-carrying channels and you can render it to headphones, a soundbar, or a dome of speakers from the same source, and rotate it freely as the listener turns. Its lesson rhymes with Chapter 1's: don't store the output (the speaker signals), store the thing itself (the field) and derive the output on demand — the same source-and-render separation that let a voice fit down a wire lets a room fit in four numbers. That elegant, patient idea, and a working sound-field codec, are yours now that this filing has expired.

CHAPTER 5

LEARNING TO CANCEL AN ECHO (THE LMS ADAPTIVE FILTER)

THE FILING

US Patent 5,307,405 — "Network echo canceller."

Filed 1992. Expired 2012; public domain.

On a phone call, some of the far-end voice leaks back to the speaker as an echo, delayed and distorted by an acoustic and electrical path nobody measured in advance. You cannot subtract an echo you have not characterised — so the device must learn the echo path, live, from the signal itself, and keep relearning as it drifts. This patent's claims describe a network echo canceller that does exactly that with an adaptive filter. It is a filter that teaches itself, and it is one of the most beautiful ideas in all of signal processing.

THE IDEA

Here is the situation. A far-end signal (the other person's voice) arrives, is played out, and a portion of it returns as an echo — filtered by an unknown path: the room's acoustics, the coupling between speaker and microphone, the electronics. You want to remove the echo from the microphone signal before sending it back, so the far end does not hear itself. But you do not know the echo path, and it changes (someone moves, a door opens). How do you subtract something you cannot describe?

The adaptive filter learns it. You have the one thing you need: the far-end signal before it became an echo. Feed that into a filter whose job is to reproduce the echo — to output the same thing the unknown path produces. Subtract the filter's output from the real microphone signal, and what remains is the error: the near-end voice you want to keep, plus whatever echo the filter failed to cancel. Now the magic step: use that error to nudge the filter's weights in the direction that would have made the error smaller, a tiny bit, every single sample. This is the least-mean-squares (LMS) update — adjust each weight in proportion to the error and the input that drove it. Do it continuously, and the filter descends toward the true echo path: with each sample it cancels a little better, the error shrinks, and within a fraction of a second the filter's weights have become a copy of the unknown echo path, subtracting the echo almost perfectly. And because it never stops adapting, when the path changes the error grows again and the filter re-learns. It is a filter that discovers, and keeps rediscovering, a system it was never told.

THE STORY

The honesty this chapter owes: the least-mean-squares adaptive filter was invented in 1960 by Bernard Widrow and Marcian "Ted" Hoff at Stanford — a foundational result of adaptive signal processing, and one of the most cited and consequential algorithms of the field. Widrow's adaptive filters solved adaptive noise cancellation and echo cancellation and much else, and the mathematics (gradient descent on the mean-squared error) turned out to be a direct ancestor of the learning rules in neural networks. Patents like US5307405 industrialised specific echo cancellers on telephone networks through the early 1990s. The technique is everywhere sound is cleaned in real time: the echo cancellers that make speakerphones, conference systems, and every hands-free and voice-over-IP call usable (without them, the far end hears their own voice bounced back — intolerable); noise-cancelling headphones, which adaptively filter the anti-noise to match the ambient sound; hearing aids, which suppress feedback the same way; and adaptive equalisers in every high-speed modem and disk drive. When you take a call on speakerphone and the other person does not hear themselves echo, an LMS adaptive filter learned your room's echo path in the first fraction of a second and has been quietly cancelling it ever since.

THE CODE

Filter the far-end to estimate the echo, subtract to get the error, and

nudge the weights by the LMS rule each sample. Here is the step from patentml's `adaptive_filter`:

```
def step(self, x, desired):
    self._push(x)
    y = self.predict()          # the filter's guess at the echo
    e = desired - y             # what's left after cancellation
    norm = sum(v * v for v in self._x) + 1e-9
    for i in range(self.n):
        self.w[i] += self.mu * e * self._x[i] / norm # LMS: nudge toward
less error
    return e
```

The demo sends far-end noise through an unknown echo path and lets the filter learn to cancel it. It prints:

LMS adaptive filter / echo canceller (from US5307405, expired 2012):

```
unknown echo path      : [0.6, -0.3, 0.15, 0.0, 0.05]
learned filter taps    : [0.6, -0.3, 0.15, 0.0, 0.05]
residual echo, first 200 samples : 0.0019
residual echo, last 200 samples : 0.000000 (converged -> echo
cancelled)
echo suppression      : 93 dB
```

The filter taught itself the echo path exactly. It was told nothing about the path $[0.6, -0.3, 0.15, 0.0, 0.05]$ — yet its learned taps converged to precisely those values, discovered purely from the error signal, one small nudge per sample. And the consequence is the last two lines: the residual echo, still audible in the first 200 samples (0.0019) while the filter was learning, falls to essentially zero once it converges — 93 dB of echo suppression, an echo made a million times quieter, cancelled by a filter that reverse-engineered an unknown acoustic path in real time. The module fuzz-tests three hundred random echo paths, confirming that the filter learns each path and suppresses the echo far below its starting level, and reports PASS.

THE INHERITANCE

The LMS adaptive filter is public domain — Widrow and Hoff's 1960 invention long since, these echo-canceller patents since 2012 — and it is the tool for any problem where you must match, cancel, or track an unknown, changing system from the signal alone: echo and feedback cancellation, adaptive noise removal, channel equalisation, system identification, active noise control. Its idea is one of the most quietly profound in this whole series: a system can learn what it needs to know by repeatedly measuring its own error and stepping to reduce it — no model given in advance, just gradient descent on a mistake. That is the same principle that, scaled up, trains neural networks; Widrow's adaptive filter and the perceptron

are cousins, and gradient-descent-on-error runs from this echo canceller straight into modern machine learning. It is a fitting note to end the series on — the machine that improves itself by attending to how wrong it is. That self-teaching filter, and a working echo canceller, are yours now that this filing has expired.

And there the series ends. From the balanced trees and bitmaps of the database, through the ciphers and signatures of cryptography, the coders that shrink every file, and now the algorithms that carry the human voice and fill a room with sound — you have walked six volumes of machinery that the companies of the late twentieth century patented, monopolised for twenty years, and then, by the terms of the bargain, set free. Every algorithm in every volume is public domain, disclosed in full, and handed to you here as code you can run on a five-pound board. The vault is open — all of it. Go and build.

APPENDIX — THE PATENTS OF THIS VOLUME

Sound and speech coding rest on famous, named academic inventions, so this appendix separates the specific patents each chapter's code is anchored to from the foundational work the Story sections credited. Linear prediction, code-excited linear prediction, the LMS adaptive filter, and ambisonics are landmark inventions with distinguished authors; the patents cited below are industrial applications of them, and the book says so throughout. All filings have reached the end of their term.

PATENTS ANCHORING THE CHAPTERS

Ch	Patent	Title	Filed
----	--------	-------	-------

----	-----	-----	-----
------	-------	-------	-------

1	US5414796	Variable rate vocoder	1993
---	-----------	-----------------------	------

2	US6122608	Method for switched-predictive quantization	1997
---	-----------	---	------

|
| 3 | US5371853 | Method and system for CELP speech coding and
codebook for use therewith | 1991 |
| 4 | US5632005 | Encoder/decoder for multidimensional sound fields
| 1995 |
| 5 | US5307405 | Network echo canceller | 1992 |

FOUNDATIONS CREDITED IN THE STORY SECTIONS

Idea	Origin	Credited to
Linear predictive coding of speech	1970s	Bishnu Atal, Manfred
Schroeder (Bell Labs), Fumitada Itakura — after Homer Dudley's		
1930s Vocoder		
Code-Excited Linear Prediction (CELP)	1985	Manfred Schroeder
& Bishnu Atal		
LMS adaptive filter	1960	Bernard Widrow & Ted Hoff (Stanford)
Ambisonics / sound-field encoding	1970s	Michael Gerzon
(Oxford) |

Two of these carry a lesson worth keeping. The LMS adaptive filter (Chapter 5) learns by descending on its own error — the same principle that, scaled up, trains neural networks, tying this volume's last algorithm back to the machine learning of Volumes One and Two. And ambisonics (Chapter 4) waited forty years for VR to make

its layout-independence essential — a reminder that an idea's moment can arrive long after its invention, and long after its patent has expired.

CHAPTER 6

A CLOSING NOTE ON THE COMPANION CODE

Every algorithm in this book ships in the patentml library:

```
pip install patentml
```

Each module names its patent, carries a runnable demo that builds its own test data, and is verified against a reference — the same PASS quoted at the end of each chapter's Code section, with nothing outside Python's standard library. Watch a voice shrink to a few coefficients and rebuild, a source pan across a ring of speakers, an echo fade 93 dB into silence. The book tells you why and how; the library lets you run it.

CHAPTER 7

THE WHOLE SHELF

The Lost Algorithms of Sound completes the series' current arc. Six volumes, each drawn from patents that expired into the public domain and handed over as working, dependency-free code:

- Volumes One & Two — Machine Learning · the optimisers, predictors, and pattern-finders.
- Computer Vision · the story of the pixel.
- Volume Three — Databases · how the world's data got fast.
- Volume Four — Cryptography & Data Hiding · secrets, signatures, and things hidden in plain sight.
- Volume Five — Compression · how to make a file smaller.
- Volume Six — Sound · how space and speech fit down a wire.

A patent is a bargain: a twenty-year monopoly in exchange for publishing exactly how an invention works. When it expires, the

second half of the bargain comes due — the knowledge belongs to everyone. Across six volumes, this series has done what almost no one does: gone to the expired patent record, taken the ideas the world's companies and labs disclosed and then lost their monopoly over, and handed them to the next generation of builders as code they can run, change, and ship — on a laptop, a Raspberry Pi, or a five-pound board.

That was the whole idea. Knowledge given now, instead of sitting idle for another twenty years. The vault is open — every volume of it.

Go and build.