Robust Modeling and Equalization of High-Speed PAM4 Signals Using Adaptive Digital Signal Processing and Regression-Based Approaches

Problem A: High Speed Signal Simulation and Equalization

Team: Colo Colo
Coach: Valentina Vega
Authors: Vicente Opazo
Martin Andrighetti
Gerardo Rodriguez
Raul Opazo

Robust Modeling and Equalization of High-Speed PAM4 Signals Using Adaptive Digital Signal Processing and Regression-Based Approaches

1 Abstract

This article presents our solution to the *High-speed Signal Modeling* and *High-speed Signal Equalization* problems from the IMC Challenge 2025 sponsored by Huawei. These tasks simulate real-world challenges in high-speed communication systems, where digital PAM4 signals are transmitted through noisy and nonlinear channels, and must be accurately reconstructed or equalized, respectively.

For the *modeling task*, we developed a phase-aware regression framework with rich nonlinear features, combined with polyphase FIR smoothing, to approximate the analog channel response from digital training sequences and synthesize accurate analog outputs for unseen inputs. For the *equalization task*, we implemented a multi-phase regression-based equalizer with adaptive priors and hysteresis, enabling robust symbol recovery under diverse test conditions.

Our approach draws inspiration from state-of-the-art digital signal processing (DSP) techniques in communication systems, while remaining computationally efficient to meet the contest's complexity and timing constraints. Experimental results show that the proposed solution achieves low reconstruction error for the modeling task and competitive bit error rates (BER) for the equalization task across multiple hidden test cases.

Keywords: High-speed communication, PAM4, digital signal processing (DSP), signal modeling, signal equalization, regression methods, Cholesky decomposition, adaptive filtering, IMC Challenge 2025.

2 Introduction

High-speed interfaces are fundamental components in modern communication systems, enabling reliable data exchange between chips in computing, networking, wireless, and storage devices. With the continuous increase in transmission rates, physical links suffer from diverse impairments such as insertion loss, inter-symbol interference (ISI), channel nonlinearities, and additive noise. These effects significantly distort the transmitted waveforms, making accurate recovery of the underlying digital data a challenging task.

A common modulation used in such systems is four-level pulse amplitude modulation (PAM4), where symbols are represented by the levels $\{-3, -1, 1, 3\}$. Although PAM4 doubles the data throughput compared to binary modulation, it also increases susceptibility to noise and distortion, thus requiring sophisticated digital signal processing (DSP) methods at the receiver.

A typical DSP-assisted PAM4 communication system is illustrated in Figure 1.

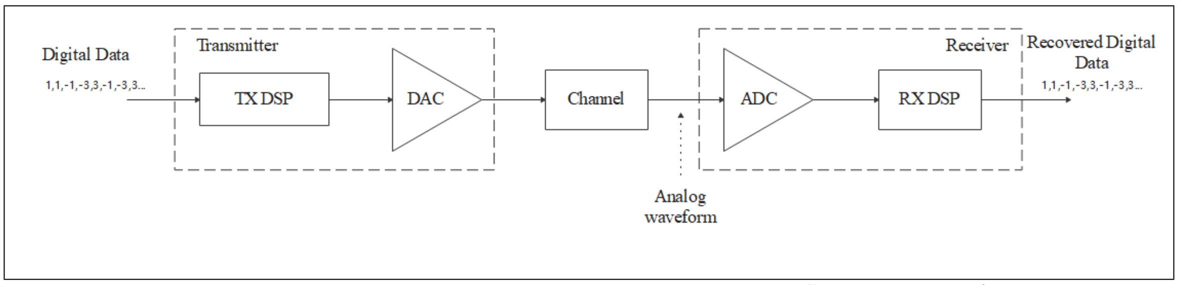

Figure 1: Modern high-speed communication system with DSP-assisted PAM4 transmission. The digital sequence is encoded into PAM4 symbols, converted to an analog waveform at the transmitter, and recovered at the receiver through equalization and slicing.

In the context of the IMC Challenge 2025 sponsored by Huawei, we addressed two complementary problems: *High-speed Signal Modeling* and *High-speed Signal Equalization*. The first task required learning a mapping from digital training sequences to analog waveforms, and then generating accurate predictions for unseen inputs. The second task required recovering the transmitted digital symbols from distorted and oversampled analog waveforms. Both problems were evaluated under strict constraints on runtime and computational complexity, reflecting the requirements of real-world hardware implementations.

To tackle these challenges, we designed a solution inspired by state-of-the-art DSP techniques but adapted to the constraints of the contest. For the modeling problem, we built a phase-aware regression framework with nonlinear feature expansion and polyphase filtering. For the equalization problem, we proposed a regression-based multi-phase equalizer with adaptive priors and hysteresis, enabling robust symbol detection across heterogeneous testcases.

The remainder of this article is structured as follows. Section 3 introduces the assumptions and symbols used. Section 4 presents the analysis of the problem, model building, and solution strategy. Section 5 discusses test results. Finally, conclusions and references are provided.

3 Assumptions and Symbols

3.1 Assumptions

In order to design an effective solution under the competition setting, we adopted the following assumptions:

- **Realistic input data.** We assume that the provided waveforms correspond to realistic high-speed communication channels. Since the evaluation is based on hidden testcases without theoretical guarantees, we consider that the grader may behave adversarially by selecting challenging scenarios. Thus, the solution must be robust to diverse distortions rather than tuned to a specific synthetic distribution.
- **Shared channel structure.** We assume that the underlying physical effects behind both tasks — *signal modeling* and *signal equalization* — arise from similar communication channels. Therefore, insights and techniques developed in Problem A can be transferred and adapted to Problem B.
- **Oversampling rate (OSR).** In both tasks, the analog waveform is oversampled at a fixed rate (OSR = 16 for modeling and OSR = 4 for equalization). This assumption allows us to build phase-dependent models and to select optimal sampling offsets.
- **Additive noise and nonlinear distortion.** The received analog signal is assumed to be affected by additive noise, inter-symbol interference (ISI), and nonlinear distortion. These impairments motivate the use of adaptive filtering, regression-based modeling, and feature expansions.
- **PAM4 symbol alphabet.** The transmitted digital signal is restricted to the four-level alphabet $\{-3, -1, 1, 3\}$, which is consistent across both tasks. This fixed alphabet enables the use of threshold slicing and regression mapping.

3.2 Symbols

To formalize the proposed models, we define the following notation:

- s_k: transmitted PAM4 symbol at discrete time k, where $s_k \in \{-3, -1, 1, 3\}$.
- y_n: received analog sample at discrete index n, after oversampling. The total number of samples is M.
- OSR: oversampling rate, i.e., the number of analog samples per PAM4 symbol (OSR = 16 in Problem A, OSR = 4 in Problem B).
- r: phase index within the oversampling grid, $r \in \{0, \dots, \mathrm{OSR} - 1\}$.
- $\mathbf{x}_k$: predictor vector associated with symbol s_k, constructed from neighboring samples and nonlinear feature expansions.

- $\mathbf{h}_r$: regression coefficient vector (or filter) for phase r.
- $\hat{y}_n$: predicted analog sample at index n, computed as $\hat{y}_n = \mathbf{h}_r^\top \mathbf{x}_k$ for the corresponding phase r.
- $\hat{s}_k$: estimated digital symbol after equalization and slicing.

4 Main Text

4.1 Problem A: High-speed Signal Modeling

4.1.1 Analysis of the Problem

The objective of the *High-speed Signal Modeling* task was to learn the transformation from a digital PAM4 sequence to its corresponding analog waveform. The input consisted of a training pair $(\mathbf{s}^{(1)}, \mathbf{y}^{(1)})$, where $\mathbf{s}^{(1)} = \{s_k\}$ represents the transmitted digital symbols and $\mathbf{y}^{(1)} = \{y_n\}$ the oversampled analog waveform captured at the receiver. The goal was to estimate a function f such that $\mathbf{y}^{(1)} \approx f(\mathbf{s}^{(1)})$, and then apply this function to a second digital sequence $\mathbf{s}^{(2)}$ in order to predict its analog waveform $\hat{\mathbf{y}}^{(2)}$ with minimal root mean square error (RMSE).

Our understanding of the *High-speed Signal Modeling* task was built progressively through a process of trial and error. Instead of having a closed-form design from the beginning, we experimented with different ideas and refined the model based on the observed errors. The key insights emerged in the following order:

- **Simple linear regression.** We first attempted to directly map the digital PAM4 symbols to the analog waveform using basic linear regression.
- **Separation by phase.** Because the oversampling rate is $\mathrm{OSR} = 16$, we realized that each analog sample belongs to a specific phase within the symbol period. Training a separate regression model for each phase significantly improved accuracy compared to the single global model.
- **Nonlinear terms.** After improving with phase separation, we still observed systematic mismatches. Adding nonlinear feature expansions (e.g., squared terms, cross-products between taps) helped capture distortions introduced by the channel.
- **Nonlinear function over y.** We tested applying nonlinear transformations over the target samples y_n (such as inverse activation functions), which reduced bias in the regression.
- **Signal delay.** Through experimentation, we noticed that introducing a delay parameter D when aligning the predictors with the analog samples could better synchronize the digital-to-analog mapping, further reducing error.

- **FIR filter refinement.** Finally, we added a polyphase FIR filter stage applied to the regression outputs. This step can be seen as a second regression over the predicted waveform, serving to smooth residual errors and capture long-range dependencies.

These empirical observations formed the foundation of our final modeling approach. Each step represented a lesson learned from prior failures, leading to a progressively more accurate and robust solution.

4.1.2 Model Building

We construct a phase-aware regression model with nonlinear feature expansions and a second-stage polyphase FIR refinement, closely following the insights obtained during experimentation.

Preprocessing and alignment. Given a training pair $(\mathbf{s}^{(1)}, \mathbf{y}^{(1)})$ with oversampling $\mathrm{OSR} = 16$, we expand the symbol-rate sequence $\mathbf{s}^{(1)}$ to the sample rate:

$$x[n] \;=\; \alpha \cdot s^{(1)}_{\lfloor n/\mathrm{OSR} \rfloor},$$

which produces a normalized, piecewise-constant regressor at the ADC rate. The scaling factor α was chosen empirically: although $\frac{1}{3}$ would be a natural normalization (mapping PAM4 symbols into $\{-1, -\frac{1}{3}, \frac{1}{3}, 1\}$), in practice the value $\alpha \approx \frac{1}{39}$ provided consistently lower error across testcases.

To account for time alignment, we introduce an integer delay D and define the regression targets as

$$z[n] \;=\; \mathrm{invg}\big(y^{(1)}[n + D]\big),$$

where $\mathrm{invg}(\cdot)$ is the inverse of a fixed output nonlinearity. Depending on the dataset, either tanh or the identity function was more appropriate. For indices outside the valid range, the closest boundary value of $y^{(1)}$ is used to avoid invalid access.

Phase-aware linear regression. For each phase $r \in \{0, \dots, \mathrm{OSR}-1\}$, we build a design vector $\boldsymbol{\phi}_n$ for samples with $n \equiv r \pmod{\mathrm{OSR}}$. The feature blocks are:

1. **Bias term:** a constant 1.
2. **Linear multi-tap block:**

$$\{\, x[n + \mathrm{OSR} \cdot i] \,\}_{i \in \{-L_1, \dots, R_1\}},$$

   which captures inter-symbol interference (ISI).

3. **Quadratic per-tap block:**

$$\{\, \mathrm{sgn}(x[n + \mathrm{OSR} \cdot i]) \cdot \big(x[n + \mathrm{OSR} \cdot i]\big)^2 \,\}_{i \in \{-L_2, \dots, R_2\}},$$

   to represent odd-order nonlinearities.

4. **Current–tap interaction block:**

$$\{\,\mathrm{sgn}(x[n])\cdot x[n]^2\cdot x[n+\mathrm{OSR}\cdot i]\,\}_{i\in\{-L_3,\dots,R_3\}},$$

coupling the instantaneous symbol level with its neighbors.

5. **Pairwise products block:**

$$\{\,x[n+\mathrm{OSR}\cdot i]\cdot x[n+\mathrm{OSR}\cdot j]\,\}_{i\in\{-L_4,\dots,R_4\},\ j\in\{-L_5,\dots,R_5\}},$$

to capture second-order cross-terms between different neighbors.

6. **Transition descriptors:** features that detect symbol transitions between consecutive OSR-spaced samples, including indicators for transitions, direction of change, and interactions with the current level. These descriptors are computed over the window $[\,n-\mathrm{OSR}\cdot L_6,\ n+\mathrm{OSR}\cdot R_6\,]$.

7. **Triple products block:**

$$\{\,x[n+\mathrm{OSR}\cdot i]\cdot x[n+\mathrm{OSR}\cdot j]\cdot x[n]\,\}_{i\in\{-L_7,\dots,R_7\},\ j\in\{-L_8,\dots,R_8\}},$$

capturing cubic-like interactions with the current symbol.

8. **Threshold-crossing features:** for neighbor samples (s_1, s_2) spaced one OSR apart, we add (s_2-s_1) if a crossing occurs relative to reference thresholds (zero and normalized positive/negative levels). Otherwise, we append zero. These descriptors are computed over the window $[\,n-\mathrm{OSR}\cdot L_9,\ n+\mathrm{OSR}\cdot R_9\,]$.

Regression per phase. Each phase r is fitted with ridge-regularized least squares:

$$\min_{\mathbf{h}_r}\ \sum_{n\equiv r}\big(z[n]-\mathbf{h}_r^\top\boldsymbol{\phi}_n\big)^2\ +\ \lambda\|\mathbf{h}_r\|_2^2,$$

where λ is a small regularization term ensuring stability. The system is solved efficiently using a stabilized $\mathrm{LDL}^\top$ decomposition with diagonal boosting.

Polyphase FIR refinement. Once the first-stage predictions $\tilde{z}[n]=\mathbf{h}_{(n \bmod \mathrm{OSR})}^\top\boldsymbol{\phi}_n$ are obtained, we refine them with a polyphase FIR filter. For each phase r, a short symmetric filter $\mathbf{c}_r$ of length Q is trained to minimize residual error:

$$\hat{z}[n]\ =\ \sum_{k=0}^{Q-1} c_{(n \bmod \mathrm{OSR})}[k]\ \tilde{z}[n+k-R].$$

This can be interpreted as a second regression stage that smooths predictions and captures long-range dependencies not modeled by the first stage.

Final output. The reconstructed analog waveform is recovered by applying the forward nonlinearity:

$$\hat{y}[n] \;=\; g\big(\hat{z}[n]\big),$$

where g is chosen as either tanh or the identity depending on the dataset characteristics.

4.1.3 Model Solving

Phase-wise ridge regression (stage 1). For each phase $r \in \{0, \dots, \mathrm{OSR}-1\}$, we collect all indices $\mathcal{I}_r = \{\, n : \; n \equiv r \pmod{\mathrm{OSR}} \,\}$ and build a design matrix $X_r \in \mathbb{R}^{N_r \times P}$ row-by-row from the feature vector $\boldsymbol{\phi}_n$ (Section *Model Building*). The corresponding target vector is $\mathbf{z}_r = \{z[n]\}_{n \in \mathcal{I}_r}$. We estimate

$$\mathbf{h}_r \;=\; \arg\min_{\mathbf{h}} \, \big\| X_r \mathbf{h} - \mathbf{z}_r \big\|_2^2 \;+\; \lambda \, \|\mathbf{h}\|_2^2,$$

with a small ridge $\lambda > 0$ to improve conditioning.
Rather than inverting $(X_r^\top X_r + \lambda I)$ explicitly, we *accumulate* the normal equations efficiently:

$$\underbrace{X_r^\top X_r}_{\text{lower-triangular accumulated}} \qquad \text{and} \qquad \underbrace{X_r^\top \mathbf{z}_r}_{\text{right-hand side}} \quad ,$$

writing only the lower triangle and mirroring it to the upper triangle at the end (cache-friendly, reduced FLOPs). We then solve

$$\big(X_r^\top X_r + \lambda I\big)\, \mathbf{h}_r \;=\; X_r^\top \mathbf{z}_r$$

via a stabilized $\mathrm{LDL}^\top$ (Cholesky-like) factorization with:

- **Jacobi equilibration:** diagonal scaling to reduce dynamic range;
- **Diagonal boosting (fallback):** if a pivot is too small, we add a tiny positive value to the diagonal and retry;
- **No explicit inverses:** forward/back substitution only, which is both faster and more numerically stable.

This per-phase solve is deterministic. Complexity per phase is $O(N_r P^2)$ to build the normal equations and $O(P^3)$ for the $P \times P$ system.

Polyphase FIR refinement (stage 2). Let the first-stage prediction in the pre-nonlinearity domain be

$$\tilde{z}[n] \;=\; \mathbf{h}_{(n \bmod \mathrm{OSR})}^\top \, \boldsymbol{\phi}_n.$$

Residual errors show structured memory beyond the local feature window, so we learn a short, *per-phase* FIR $\mathbf{c}_r \in \mathbb{R}^Q$ (odd length, centered at $R = Q/2$). For each r we solve the ridge problem

$$\min_{\mathbf{c}_r} \sum_{\substack{n\in\mathcal{I}_r \\ n-R\geq 0,\; n+Q-R-1<N}} \Big(z[n] - \sum_{k=0}^{Q-1} c_r[k]\,\tilde{z}[n+k-R]\Big)^2 \;+\; \gamma\|\mathbf{c}_r\|_2^2,$$

again by accumulating the normal equations and applying the same stabilized LDL$^\top$ routine. This "FIR over predictions" is effectively a second regression that cleans residual ISI and smooths local irregularities.

Boundary handling and nonlinearity. If $n + D$ falls outside $[0,\, N-1]$, we substitute the closest valid $y^{(1)}$ sample when forming

$$z[n] \;=\; \mathrm{invg}\big(y^{(1)}[n+D]\big),$$

which provides a simple and robust boundary policy. At inference time, the final analog estimate is obtained by chaining both regression stages and applying the output nonlinearity:

$$\hat{z}[n] \;=\; \sum_{k=0}^{Q-1} c_{(n \bmod \mathrm{OSR})}[k]\,\tilde{z}[n+k-R], \qquad \hat{y}[n] \;=\; g\big(\hat{z}[n]\big).$$

Here (invg, g) act as a pre/post nonlinear pair: invg linearizes the training target, and g restores the final analog scale. This allows the regressions to remain nearly linear while still matching the nonlinear characteristics of the observed data.

Runtime and space profile. The implementation avoids heap allocations in hot loops (favoring stack or static buffers), accumulates only the lower triangle of the Gram matrix, and reuses structures across phases. Memory usage is $O(P^2)$ per phase for the Gram matrix and $O(P)$ for the right-hand side. The FIR training adds $O(Q^2)$ per phase, with modest Q, keeping both stages comfortably within contest time and memory limits.

Complete solution. By solving these two regression stages for every phase, we obtain the full set of coefficients $\{\mathbf{h}_r\}$ and $\{\mathbf{c}_r\}$. Together with the nonlinear mapping g, these parameters fully define the signal-modeling function: given a new digital input sequence, we can synthesize its analog waveform efficiently and with low reconstruction error.

4.1.4 Model Summarizing

Summary of the model. The final modeling solution can be described as a two-stage, phase-aware regression framework.

In the first stage, we train a ridge-regularized linear regression for each of the 16 oversampling phases, using multi-tap linear predictors enriched with nonlinear terms, interaction features, transition descriptors, and polynomial cross-products.

In the second stage, we refine the preliminary predictions with a short polyphase FIR filter trained per phase, which reduces residual inter-symbol interference and smooths long-range dependencies. Finally, the restored analog output is obtained by applying an output nonlinearity $g(\cdot)$ to the refined predictions.

What made the model work. Our empirical exploration (Section *Analysis*) led to three design choices that proved decisive:

1. **Phase separation:** Training one model per oversampling phase captures alignment-specific behavior that a single global model misses.
2. **Targeted nonlinearity:** Compact odd-order features (e.g., $\operatorname{sgn}(x)\, x^2$), pairwise and triple products, plus transition descriptors, were sufficient to capture dominant nonlinear distortions without exploding feature count.
3. **Second-stage FIR:** A short, per-phase FIR "on predictions" efficiently absorbs residual memory beyond the feature window and reduces structured errors, acting like a lightweight smoothing/post-equalization step.

Why this balance is robust. The stage-1 regressions explain most variance using local, interpretable features (good bias/variance trade-off), while the stage-2 FIR corrects longer-range effects without over-parameterizing the first stage. The stabilized $\mathrm{LDL}^\top$ solves (with diagonal scaling and boosting) eliminate numerical fragility that we observed with naive inverses, ensuring consistent performance across datasets and parameter presets.

Limitations. Very long-memory channels or highly nonstationary behavior cannot be fully captured by short windows and a single FIR length Q. We mitigate this by:

- allowing delay D tuning for coarse alignment,
- using phase-specific FIRs (instead of one global filter),
- keeping a small ridge in both stages to control variance under collinearity.

Takeaway for Problem B. Three ideas transfer directly to equalization: (i) operate per phase, (ii) keep regressions stable and lightweight, and (iii) use simple priors/smoothing to enforce temporal coherence. In Problem B we invert the perspective (recover symbols from waveform), but we reuse the same principles: per-phase linear models, careful regularization, and a light mechanism to propagate local decisions coherently across time.

4.2 Problem B: High-speed Signal Equalization

4.2.1 Analysis of the Problem

The goal of the *High-speed Signal Equalization* task is to recover a PAM4 sequence $\mathbf{s} = \{s_k\}$ from an oversampled analog waveform $\mathbf{y} = \{y_n\}$ with OSR = 4. Unlike Problem A, no paired (digital, analog) training data are provided: the symbols are hidden and must be inferred directly from the distorted waveform. In other words, the mapping is inverted and unsupervised: we must detect the digital sequence that best explains the observations.

Figure 2 shows an example testcase, overlaying the analog signal and an expanded, scaled digital pattern. A clear correlation is visible: naive slicing of the analog samples already produces a rough guess, but it is fragile under noise, nonlinear distortion, and inter-symbol interference (ISI).

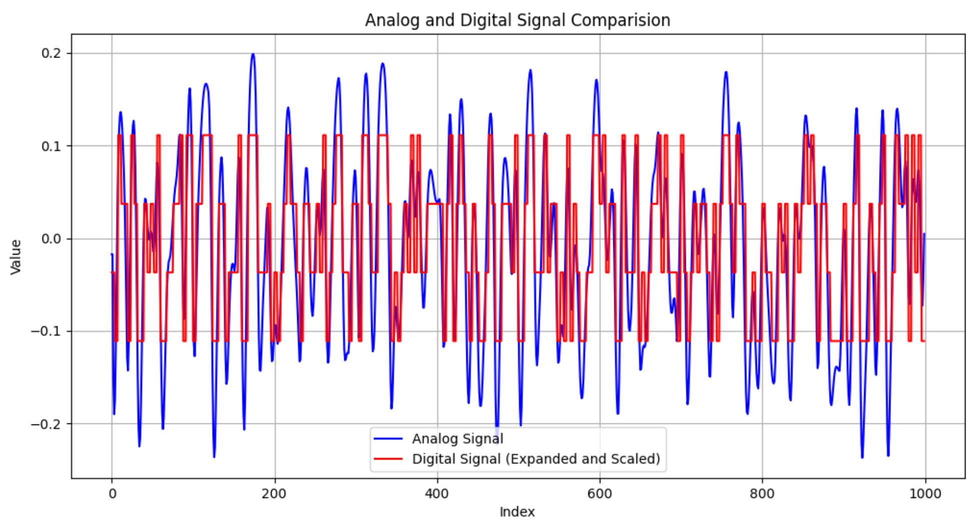

Figure 2: Analog (blue) and digital (red) signal comparison from an example testcase.

Our approach emerged through iterative experimentation, with the following insights discovered in order:

- **Start with what the data gives you (naive slicing).** Direct thresholding into four levels provides a fast baseline, but collapses on noisy or ISI-dominated segments.
- **Distribution-aware initialization (quartiles).** Using empirical quartiles of $\mathbf{y}$ for thresholds yields a noticeably better unsupervised initialization $\hat{\mathbf{s}}^{(0)}$ than ad-hoc fixed thresholds.

- **Phase awareness matters.** With OSR $= 4$, treating samples modulo 4 separately is essential. We therefore work per phase $r \in \{0, 1, 2, 3\}$, using $r = (k+D) \bmod \text{OSR}$ for symbol k (with a small integer delay D for coarse alignment), and index the nominal center as
$$n_0 \;=\; k \cdot \text{OSR} + r + D.$$
- **Local linear context helps more than hard slicing.** Replacing pure thresholds with a *per-phase linear predictor* that looks at a short window of past/future samples around n_0 improves robustness to ISI while remaining lightweight.
- **One pass is not enough: retrain and refine.** Because early symbol errors bias subsequent decisions, a single sweep is unstable. Re-estimating the per-phase linear models *at each iteration* using the current symbol estimates, and then re-decoding, steadily improves the sequence (EM-like refinement).
- **Stabilize the alignment before updating everything.** A two-sweep schedule per iteration worked best: first update only symbols in phase r=0 (conservative "anchor" sweep), then run a full multi-phase sweep over all symbols. Anchoring one phase first reduced drift and speed up convergence.
- **Favor local coherence with a soft neighbor prior.** When updating s_k, blend the regression score with a small average of the nearest decisions ($k \pm 1$) to discourage isolated flips. This is a *soft* prior (a light convex blend), not a hard override.
- **Prevent chattering with hysteresis.** Even with a neighbor prior, noisy regions can oscillate between adjacent levels. Requiring a small positive margin in the improvement before accepting a flip (hysteresis) eliminates most oscillations.
- **Keep the model small so iteration is viable.** Windows stay modest and predictors linear so that 20–30 refinement sweeps are computationally feasible. Heavier models slowed convergence without consistent gains on the hidden tests.

Putting these observations together led to our final design: a *multi-phase linear regression equalizer* that (i) initializes by quartile slicing, (ii) *retrains* per-phase linear predictors every iteration, (iii) updates symbols using a two-sweep schedule (phase-0 anchor $\rightarrow$ all phases), (iv) applies a soft neighbor prior to enforce local consistency, and (v) uses a small hysteresis margin to prevent oscillations. The model is intentionally lightweight because equalization requires many refinement passes, nevertheless, the core principle from Problem A persists—*phase-specific processing is crucial* for accuracy in oversampled systems.

4.2.2 Model Building

Problem formulation. We observe an oversampled waveform $\mathbf{y} = \{y_n\}_{n=0}^{N-1}$ with oversampling factor OSR $= 4$, and we seek the underlying PAM4 sequence

$$s_k \in \{-3, -1, +1, +3\}, \qquad k = 0, \ldots, K-1.$$

Each symbol s_k spans OSR consecutive samples, but due to an unknown delay D, the effective center is shifted.

Initialization. We compute empirical quartiles (q_1, q_2, q_3) of the sequence $\mathbf{y}$. Each y_n is mapped to the nearest PAM4 level according to these thresholds, giving an initial decision sequence

$$\hat{s}_k^{(0)} = Q(y_{k \cdot \mathrm{OSR}}),$$

where $Q(\cdot)$ is a 4-level quantizer based on quartiles. Let $x_k^{(0)} = v(\hat{s}_k^{(0)})$ denote the corresponding numeric values (scaled to regression targets).

Phase-specific linear regressors. For each phase $r \in \{0, 1, 2, 3\}$ we define a regression model

$$\hat{u}_{k,r} = \mathbf{w}_r^\top \boldsymbol{\psi}_{k,r},$$

with feature vector

$$\boldsymbol{\psi}_{k,r} = \big[1,\ y[n_0 + \ell]\ \text{ for } \ell = -L, \dots, R\big]^\top, \qquad n_0 = k \cdot \mathrm{OSR} + r + D.$$

The weights $\mathbf{w}_r$ are estimated by ridge regression at each iteration:

$$\mathbf{w}_r = \arg\min_{\mathbf{w}} \sum_{k \in \mathcal{K}_r} \big(x_k - \mathbf{w}^\top \boldsymbol{\psi}_{k,r}\big)^2 + \lambda \|\mathbf{w}\|^2,$$

where $\mathcal{K}_r$ is the set of indices with phase r and $\lambda > 0$ is a small ridge.

Iterative refinement. At iteration $t = 1, 2, \dots,$ we update the decision sequence as follows:
1. Re-estimate $\{\mathbf{w}_r\}$ for all phases using $\{x_k^{(t-1)}\}$ as regression targets. 2. Perform two sweeps over the symbols:

- **Phase-0 sweep.** Update only symbols k such that $(k + D) \bmod \mathrm{OSR} = 0$.
- **Multi-phase sweep.** Update all symbols across $r = 0, 1, 2, 3$.

In each sweep, for symbol k at phase r:

$$u_{k,r} = \mathbf{w}_r^\top \boldsymbol{\psi}_{k,r},$$

$$u_{k,r}^{\text{blend}} = (1 - \alpha)\, u_{k,r} + \alpha \cdot \tfrac{1}{2}(x_{k-1}^{(t-1)} + x_{k+1}^{(t-1)}),$$

where α is a small weight for the neighbor prior (different in the two sweeps). The candidate decision is

$$\tilde{s}_k = Q(u_{k,r}^{\text{blend}}).$$

Hysteresis rule. Let $J(\cdot)$ be the local regression cost proxy. We accept $\tilde{s}_k$ only if

$$J(\tilde{s}_k) + \delta < J(\hat{s}_k^{(t-1)}),$$

where $\delta > 0$ is a small margin (different in the two sweeps). Otherwise we keep $\hat{s}_k^{(t)} = \hat{s}_k^{(t-1)}$.

Termination. The iteration stops if either:

$$\#\{k : \hat{s}_k^{(t)} \neq \hat{s}_k^{(t-1)}\} < \tau,$$

for a small threshold τ, or t reaches the maximum (about 20–30 iterations). The final output is

$$\hat{\mathbf{s}} = \{\hat{s}_k^{(t)}\}.$$

Summary. Formally, the model is a *multi-phase ridge regression equalizer with iterative retraining, neighbor-blended updates, a two-sweep schedule (phase-0 then all phases), and hysteresis margining.* Its design keeps the regressors lightweight but allows stable convergence within the contest runtime.

4.2.3 Model Solving

Training of regressors. At the beginning of each iteration t, the per-phase weights $\mathbf{w}_r$ are trained by ridge regression. Because the feature dimension $P = L + R + 1$ is small (typically $P \approx 12$), the normal equations

$$(\Psi_r^\top \Psi_r + \lambda I)\, \mathbf{w}_r = \Psi_r^\top \mathbf{x}_r$$

can be solved efficiently with an LDL$^\top$ factorization, where Ψ_r is the feature matrix for phase r and $\mathbf{x}_r$ are the current targets. This keeps training per iteration $O(P^2 \cdot M)$, which is feasible since P is modest and the ridge term λ ensures numerical stability.

Update sweeps. After training, symbol updates are carried out in two sweeps:

1. **Phase-0 sweep.** Only update symbols aligned to $r = 0$, with stronger neighbor blending ($\alpha \approx 0.08$) and stricter hysteresis margin ($\delta \approx 10^{-4}$).
2. **Multi-phase sweep.** Update all symbols ($r = 0, 1, 2, 3$) with slightly weaker parameters ($\alpha \approx 0.05$, $\delta \approx 8 \cdot 10^{-5}$).

This ordering stabilizes alignment: phase-0 acts as a reliable reference before broader refinement.

Update rule. For each symbol k in the current sweep:

$$u_{k,r} = \mathbf{w}_r^\top \boldsymbol{\psi}_{k,r}, \qquad u_{k,r}^{\text{blend}} = (1-\alpha)u_{k,r} + \alpha \cdot \tfrac{1}{2}(x_{k-1}^{(t-1)} + x_{k+1}^{(t-1)}).$$

The candidate $\tilde{s}_k = Q(u_{k,r}^{\text{blend}})$ is accepted only if

$$J(\tilde{s}_k) + \delta < J(\hat{s}_k^{(t-1)}),$$

otherwise $\hat{s}_k^{(t)} = \hat{s}_k^{(t-1)}$. Here $J(\cdot)$ is a lightweight proxy for the local regression cost, and δ is the hysteresis margin.

Convergence. Iterations are repeated until either:

- the number of changes per sweep falls below a threshold τ (e.g., $\max(10, 0.0005K)$), or
- the iteration count reaches $T_{\max} \approx 20$–30.

In practice, convergence was usually reached in fewer than 25 iterations.

Runtime considerations. With parameters $(L, R) = (8, 4)$, OSR $= 4$, and short regressors, each iteration runs in linear time with respect to N (up to small P^2 factors). The memory footprint is modest ($O(P^2)$ per phase), and the LDL$^\top$ solver is both fast and numerically stable. This balance of simplicity and stability made the iterative loop viable within contest limits.

Outcome. The procedure produces a final decoded sequence $\hat{\mathbf{s}}$ that is consistent across phases, resistant to noise, and stable against oscillations. The retraining loop enables regressors to adapt progressively to the improving decisions, while the two-sweep schedule, neighbor blending, and hysteresis ensure convergence to a coherent sequence.

4.2.4 Model Summarizing

What made the model work. Our final equalizer combines several simple but complementary ideas:

1. **Phase separation:** Training one linear regressor per oversampling phase captures alignment-specific distortions that a global model cannot.
2. **Iterative retraining:** Updating regressors at each iteration allowed the models to adapt to progressively cleaner symbol estimates, similar in spirit to EM refinement.
3. **Two-sweep schedule:** Stabilizing phase-0 symbols first provided a reliable reference before updating all phases, reducing drift.

4. **Neighbor prior:** A light blend with immediate neighbors discouraged isolated errors while preserving local detail.
5. **Hysteresis:** Requiring a positive margin before flips prevented oscillations in noisy regions.

Why this balance is robust. The model is deliberately lightweight: each regressor is linear, windows are short, and neighbor priors are weak. This ensures convergence within 20–30 iterations, while still correcting for inter-symbol interference and noise. The two-sweep schedule and hysteresis rules prevent unstable behavior, so the algorithm converges reliably across different testcases.

Limitations. Despite its stability, the method has important limitations:

- **Long-memory channels:** Our short windows cannot capture channels with very extended ISI.
- **Local minima:** The refinement process may converge to suboptimal symbol sequences; once stuck, the algorithm cannot escape.
- **Noise sensitivity:** If the quartile-based initialization starts too far from the truth (e.g., under very high noise), convergence is slowed or may stall.

Takeaway. The equalizer can be summarized as a *multi-phase ridge regression decoder with iterative retraining, two-sweep refinement, neighbor priors, and hysteresis.* Its strength lies in combining very simple modules into a coherent pipeline that balances accuracy with runtime constraints. This approach is robust and could be scaled further with richer features or parallel computation.

5 Test Result Description

5.0.1 Test Results: Problem A

The evaluation of the *High-speed Signal Modeling* task was based on hidden testcases provided by the contest organizers. Our final scores were as follows:

Test Case	Score
Test Case 1	22.32/25
Test Case 2	14.63/25
Test Case 3	14.19/25
Test Case 4	14.31/25

Observations. The model performed very strongly on Test Case 1, achieving over 22 points out of 25, while the remaining test cases stabilized around 14 points. Since the test-cases are hidden, it is not possible to directly diagnose the reasons for this discrepancy.

A plausible explanation is that the latter groups involved channels with much longer memory or higher complexity, which would require significantly larger feature windows and more sophisticated models than the compact regressors we employed.

Possible improvements. A natural next step would be to increase the effective window size or incorporate more structured long-memory terms.

The main challenge is computational: larger systems require faster solvers. We experimented with conjugate gradient methods to accelerate the linear system solves, but our implementation did not work.

Nevertheless, further optimization of iterative solvers remains a promising avenue for scaling the approach. In addition, exploiting parallelization or GPU acceleration could allow the model to handle much larger regressors within the same runtime budget, potentially improving performance across the more demanding testcases.

Summary. Overall, the results show that the proposed model is competitive on at least part of the hidden test set, but its performance is sensitive to the channel conditions and window size. Future work should focus on balancing richer feature sets with efficient solvers in order to close the gap across all test groups.

5.0.2 Test Results: Problem B

The evaluation of the *High-speed Signal Equalization* task was based on hidden testcases provided by the contest organizers. Our final scores were as follows:

Test Case	Score
Test Case 1	20.00 / 20
Test Case 2	14.17 / 20
Test Case 3	4.10 / 20
Test Case 4	1.00 / 20
Test Case 5	1.00 / 20
Test Case 6	1.00 / 20
Test Case 7	2.00 / 40
Test Case 8	2.00 / 40

Observations. The equalizer achieved perfect performance on Test Case 1 and strong results on Test Case 2, but performance degraded sharply from Test Case 3 onwards. In the last five test cases the model failed to reach the minimum bit-error rate (BER) required to start earning points. Since the test cases are hidden, it is unclear how far our BER was from the required threshold, but the sharp drop suggests that these scenarios involved much more challenging channels or noise conditions than our compact model could accommodate.

Possible improvements. The current approach was deliberately kept lightweight in order to converge within 20–30 iterations under contest runtime limits. To handle the more demanding cases, a natural extension would be to enlarge the regression windows or introduce richer feature sets. This would, however, require significantly more computational power. Exploiting parallelization and GPU acceleration could allow us to scale to larger models while still remaining efficient, bringing the equalizer closer to the competitive BER levels achieved by state-of-the-art equalization methods in high-speed communications.

Summary. Overall, the results demonstrate that the proposed equalizer is effective on easier channels but struggles with the most challenging testcases. The limitations appear to be more computational than conceptual: with larger models and accelerated solvers, the same design principles could potentially extend to modern equalization tasks at the research frontier.

6 References and Appendices

6.1 References

[1] Wikipedia, *Digital-to-analog converter*, Wikimedia Foundation, 2025. Available at: https://en.wikipedia.org/wiki/Digital-to-analog_converter.

[2] Wikipedia, *Ordinary least squares*, Wikimedia Foundation, 2025. Available at: https://en.wikipedia.org/wiki/Ordinary_least_squares. This page provides basic background on regression methods.

[3] Wikipedia, *Cholesky decomposition*, Wikimedia Foundation, 2025. Available at: https://en.wikipedia.org/wiki/Cholesky_decomposition. Reference for the $LDL^\top$ solver used in our implementation.

[4] DSP StackExchange, *What is an equalizer and how does it work?*, StackExchange Network, 2016. Available at: dsp.stackexchange.com/q/31540. A practical discussion of equalization in digital signal processing.

6.2 Appendix

Iterative refinement pseudocode. For clarity, we summarize the refinement loop of Problem B in ”Python pseudo-code”

```
# Inputs: y (analog waveform) and constants (in capital letters)

# 0) Initialization (quartile slicing)
s_hat = quartile_slicing(y) # initial symbols
x = levels_to_values(s_hat) # numeric levels for regression targets

for t in range(MAX_IT):
    # 1) Retrain per-phase linear regressors using current targets
    for r in range(OSR):
        X_r = build_features_phase(y, x, r)
        z_r = current_targets(x, r)
        w[r] = fit_ridge(X_r, z_r)

    changes = 0

    # 2) Two refinement sweeps each iteration
    # First stabilize r=0 symbols, then refine all phases
    for mode in ["r0_only", "all_phases"]:
        if mode == "r0_only":
            phases = [0] # update only r=0 symbols
        else:
            phases = [0,1,2,3] # update all symbols

        for k in sweep_order(phases, D, len(s_hat)):
            r = (k + D) % OSR
            if r not in phases:
                continue  # skip symbol if not in current sweep

            psi = features_at(k, r, y, x)
            u = dot(w[r], psi)

            # Soft neighbor prior
            nb = x[k]
            if 0 < k < len(x)-1:
                nb = 0.5*x[k] + 0.25*(x[k-1] + x[k+1])
            u_blend = (1 - LAMBDA_NB)*u + LAMBDA_NB*nb

            # Hysteresis decision
            old = s_hat[k]
            cand = nearest_PAM4(u_blend)
            improv = confidence_gain(u_blend, old, cand)

            if cand != old and improv > MARGIN:
                s_hat[k]  = cand
                x[k]      = level_value(cand)
                changes += 1
            else:
                x[k]      = level_value(old)

    # 3) Early stopping
    if t >= 2 and changes < MIN_CHANGES:
        break

# Output: final detected symbols s_hat
```
Listing 1: Simplified equalization loop: retraining + neighbor prior + hysteresis