Communications on Applied Nonlinear Analysis
ISSN: 1074-133X
Vol 33 No. 1s (2026)

Lagrange-Guided Bayesian Machine Learning Inversion (LGBMLI): A Mathematical Framework for Inverse Cubic Sensor Calibration under Uncertainty

Ganti Srikanth[1,2], **Gopinathan Sudheer**[3], **S Uma Devi**[4]

[1] Department of Mathematics, Jawaharlal Nehru Technological University, Kakinada, India.
[2] Department of Mathematics, Gayatri Vidya Parishad College of Engineering (A), Visakhapatnam, India.
[3] Department of Mathematics, Gayatri Vidya Parishad College of Engineering for Women, Visakhapatnam, India.
[4] Department of Engineering Mathematics, Andhra University, Visakhapatnam, India.

Article History:

Received: 04-03-2026

Revised: 20-04-2026

Accepted: 06-05-2026

Abstract:

Cubic polynomial transformations are commonly employed in sensor systems to represent nonlinear responses, achieving a balance between precision and computational efficiency. The inverse problem — retrieving physical input from measured output — is essential for sensor calibration but is complicated by multiple roots, noise sensitivity, and parameter uncertainty. This work introduces the Lagrange-Guided Bayesian Machine Learning Inversion (LGBMLI) framework, a mathematically grounded methodology that replaces Cardano's root generator with the numerically stable Lagrange analytical solver, integrates a corrected Bayesian posterior formulation, and employs ensemble learning for root selection. A formal theorem for root-region classification and a lemma for the monotonicity condition are established with complete proofs, providing well-founded justifications for the machine-learning features. The inverse problem is reformulated as posterior inference rather than exact root recovery: $p(x|y) \propto p(y|x)\, p(x)$. Results are reported as mean ± standard deviation over repeated runs on synthetic datasets with 80–85% multiple-root cases. The proposed methodology achieves 94.6% ± 1.8% root selection accuracy while reducing MSE reconstruction error by 34% compared to Cardano and by 25.45% compared to Newton–Raphson. The 94.2% empirical coverage of nominal 95% credible intervals (miscoverage ≈5.8%) indicates well-calibrated uncertainty quantification with a small residual approximation gap. Identifiability conditions, stability analysis, computational complexity, and extensions to outlier-contaminated and lacunary data are also discussed.

Keywords: cubic transformation, inverse problems, Bayesian inference, Lagrange solver, root discriminant, sensor calibration, ensemble learning.

Notation Convention Table

To ensure consistency throughout the paper, all principal symbols are defined in Table 1 below. The two functions f and g are distinguished by role: $f(x; \mathbf{b})$ always refers to the original sensor forward model with calibration coefficients $\mathbf{b}$, while $g(x)$ always refers to the monic normalised cubic obtained by dividing through by b_3 and shifting the constant term to

incorporate the observed output y_{obs}. Three discriminant quantities appear; each serves a distinct purpose and must not be conflated.

Table 1: Principal Notation

Symbol	Definition	Role
$f(x;\mathbf{b})$	$b_0 + b_1x + b_2x^2 + b_3x^3$	Original sensor forward model
$g(x)$	$x^3 + \alpha x^2 + \beta x + \gamma$	Monic normalised cubic for root analysis
$\mathbf{b}$	$(b_0, b_1, b_2, b_3)^\top$	Calibration coefficient vector
(α, β, γ)	$(b_2/b_3,\ b_1/b_3,\ (b_0 - y_{\text{obs}})/b_3)$	Monic cubic coefficients
x_{inf}	$-\alpha/3$	Inflection point of g
Δ	$18abcd - 4b^3d + b^2c^2 - 4ac^3 - 27a^2d^2$	Classical discriminant of general cubic
D	$Q^3 + R^2$	Reduced discriminant of monic cubic g
δ_2	$4\alpha^2 - 12\beta$	Derivative discriminant (monotonicity indicator)
Q	$(3\beta - \alpha^2)/9$	Auxiliary quantity for D
R	$(9\alpha\beta - 27\gamma - 2\alpha^3)/54$	Auxiliary quantity for D
p_1, p_2	See Eq. (8)–(9)	Lagrange structural invariants; note $p_1 = \Delta = -108D$
$\mathcal{D}$	$\{(x_i, y_i)\}_{i=1}^n$	Calibration dataset
$\mathcal{A}$	$[x_{\min}, x_{\max}]$	Admissible physical domain
$\hat{R}$	Gelman–Rubin statistic	MCMC convergence diagnostic
ESS	Effective sample size	MCMC adequacy diagnostic

The relations between discriminants are: $\Delta = -108D$ for the monic cubic, and $\delta_2 = -36Q$ (see Lemma 1 proof). These identities are used to translate between notations in the literature.

1. Introduction

1.1 Importance of Inverse Cubic Sensor Calibration

Sensors are ubiquitous in modern technology — from industrial process monitoring and medical diagnostics to environmental sensing and autonomous systems. They convert physical quantities (temperature, pressure, strain, concentration) into measurable electrical or digital outputs through mathematical *transfer functions*. When the sensor response is mildly nonlinear, a cubic polynomial transfer function $f(x;\mathbf{b}) = b_0 + b_1x + b_2x^2 + b_3x^3$ provides an effective compromise between accuracy and computational complexity [1, 2]. Cubic models are widely adopted in temperature sensors (thermistors, RTDs) where resistance varies nonlinearly with temperature; pressure transducers exhibiting polynomial nonlinearity

over their operating range; chemical concentration sensors (pH, dissolved oxygen) with Nernst-equation nonlinearities; strain gauges and load cells under large-deformation regimes; and magnetometers and inertial sensors in navigation systems.

The *forward problem* — predicting sensor output y from physical input x — is straightforward. However, the *inverse problem* — recovering x from measured y — is essential for every practical calibration and measurement system. It is precisely this inverse step that converts raw sensor readings into engineering-meaningful quantities. Without accurate inversion, even a well-designed sensor yields unreliable measurements.

Inverse cubic calibration presents unique mathematical difficulties. First, a cubic may have one or three real roots; selecting the physically admissible root among up to three candidates is non-trivial, especially in the 80–85% multiple-root regime encountered in practice. Second, measurement noise ε in y is amplified in x by the factor $1/|f'(x)|$, causing instability near critical points of f. Third, calibration coefficients $\mathbf{b}$ are themselves uncertain due to limited calibration data, manufacturing variability, and sensor drift. Fourth, where $f'(x) \approx 0$, small errors in y produce large errors in x, making classical root-finding unreliable. Addressing all four challenges simultaneously requires a framework that is analytically grounded, probabilistically sound, and practically efficient.

1.2 Background and Motivation

Classical analytical solutions based on Cardano's method [3, 4] provide closed-form roots but suffer from numerical instability arising from branch-selection ambiguity, cancellation errors at near-zero discriminants, and incorrect complex-root handling in floating-point software [5, 6]. Iterative methods such as Newton–Raphson [7] and Halley's method are faster but depend on accurate initial guesses and provide no uncertainty quantification [8, 9]. Deiters and Macias-Salinas [10] demonstrated that numerical methods are roughly ten times more efficient than analytical methods for multiple roots, but they still lack uncertainty quantification.

Bayesian frameworks offer probabilistic approaches to inverse problems, integrating prior knowledge and quantifying uncertainty through posterior distributions [11, 12]. Tikhonov regularisation [13] stabilises ill-posed problems by incorporating a penalty term. Markov Chain Monte Carlo (MCMC) methods, including the Metropolis–Hastings algorithm, enable posterior sampling when analytical posteriors are intractable [14, 15]. Bilionis and Zabaras [16] used Bayesian inference for inverse problems with limited forward evaluations. Mohammed-Djafar [17] integrated regularisation, Bayesian inference, and machine learning for linear inverse systems. The present work extends these ideas to nonlinear cubic transformations with multiple roots.

Ensemble methods such as Random Forest [18, 19] and XGBoost [20, 21] are highly effective in classification tasks, but their application to cubic root selection has lacked formal theoretical justification. This gap is addressed here through a formal theorem and lemma connecting ML features to discriminant theory and derivative constraints.

1.3 Contributions

The specific contributions of this work are:

1. Lagrange solver integration: Replacement of Cardano's root generator with the numerically stable Lagrange analytical solver of Ganti and Sudheer [22], providing new structurally informative ML features p_1, p_2.
2. Probabilistic inverse formulation: The inverse problem is stated as posterior inference $p(x \mid y_{\text{obs}}, \mathrm{b}, \mathcal{D})$ rather than exact root recovery.
3. Rigorous theorem and lemma with complete proofs: A Root-Region Classification Theorem and a Monotonicity Lemma — both with monotonicity-on-intervals arguments and explicit sign-change reasoning — provide well-founded mathematical justification for ML features.
4. Corrected Bayesian formulation: Resolution of the conflict between Gaussian and uniform prior specifications; Tikhonov regularisation interpreted as a Gaussian prior.
5. Identifiability and stability analysis: Explicit conditions under which calibration coefficients are identifiable and a formal error-propagation formula.
6. Rigorous posterior root probabilities: Formal definition $P_k = \Pr(r_k \in \mathcal{A} \mid y_{\text{obs}}, \mathcal{D})$ with numerical approximation.
7. Statistical validation: Results reported as mean $\pm$ standard deviation over repeated runs with 95% credible intervals on accuracy.
8. Computational complexity analysis: Explicit per-phase cost comparison of Lagrange, Cardano, MCMC, and ML inference.

Real-world sensor networks also face outlier-contaminated measurements and lacunary measurements (data-transmission losses, packet dropouts). Extensions to handle these conditions are discussed in Section 8, building on recent frameworks in fuzzy paranormed spaces [23, 24] and ideal-convergence theory [25].

2. Mathematical Formulation

2.1 Sensor Forward Model

The sensor forward model is:

$$y = f(x; \mathbf{b}) = b_0 + b_1 x + b_2 x^2 + b_3 x^3, \tag{1}$$

where $x \in \mathbb{R}$ is the physical input, $y \in \mathbb{R}$ is the measured output, and $\mathbf{b} = (b_0, b_1, b_2, b_3)^{\top}$ are calibration coefficients. In practice we observe $y_{\text{obs}} = f(x; \mathbf{b}) + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$.

2.2 The Inverse Problem as Posterior Inference

The inverse problem must not be formulated as exact root recovery under noisy conditions. With measurement noise, the correct formulation is posterior inference:

$$p(x \mid y_{\text{obs}}, \mathbf{b}) \;\propto\; p(y_{\text{obs}} \mid x, \mathbf{b})\; p(x). \tag{2}$$

For Gaussian noise $\varepsilon \sim \mathcal{N}(0, \sigma^2)$:

$$p(x \mid y_{\text{obs}}, \mathbf{b}) \propto \exp\left[-\frac{(y_{\text{obs}} - f(x;\mathbf{b}))^2}{2\sigma^2}\right] p(x). \quad (3)$$

The roots of $g(x) = x^3 + \alpha x^2 + \beta x + \gamma = 0$ (obtained by normalising $b_3 x^3 + b_2 x^2 + b_1 x + (b_0 - y_{\text{obs}}) = 0$ by dividing by b_3, with coefficients $\alpha = b_2/b_3$, $\beta = b_1/b_3$, $\gamma = (b_0 - y_{\text{obs}})/b_3$) identify the *modes* of this posterior, not exact solutions. This formulation is more rigorous than solving the noisy cubic exactly.

2.3 Cubic Discriminant Theory

Three discriminant quantities are used throughout; they must not be conflated.

For the general cubic $ax^3 + bx^2 + cx + d = 0$, the classical discriminant is:

$$\Delta = 18abcd - 4b^3 d + b^2 c^2 - 4ac^3 - 27a^2 d^2. \quad (4)$$

For the monic normalised cubic $g(x) = x^3 + \alpha x^2 + \beta x + \gamma$, define auxiliary quantities:

$$Q = \frac{3\beta - \alpha^2}{9}, \qquad R = \frac{9\alpha\beta - 27\gamma - 2\alpha^3}{54}. \quad (5)$$

The reduced discriminant of g is:

$$D = Q^3 + R^2. \quad (6)$$

The two discriminants satisfy $\Delta = -108D$ for the monic cubic, so their sign conventions are opposite: $\Delta > 0 \Leftrightarrow D < 0$ (three distinct real roots); $\Delta < 0 \Leftrightarrow D > 0$ (one real root, two complex conjugates); $\Delta = 0 \Leftrightarrow D = 0$ (repeated root). This paper uses D consistently for all root analysis.

The derivative discriminant (monotonicity indicator) is:

$$\delta_2 = 4\alpha^2 - 12\beta. \quad (7)$$

Its relation to Q is $\delta_2 = -36Q$ (since $Q = (3\beta - \alpha^2)/9$ implies $\alpha^2 - 3\beta = -9Q$, so $4\alpha^2 - 12\beta = 4(\alpha^2 - 3\beta) = -36Q$). Hence $\delta_2 \leq 0 \Leftrightarrow Q \geq 0$, providing a direct link between the monotonicity condition and the reduced discriminant.

2.4 Lagrange Analytical Solver for Cubic Candidate Roots

To improve numerical reliability, the Lagrange analytical solver of Ganti and Sudheer [22] is used in place of Cardano's formula. Unlike Cardano's formula, which may suffer from branch-selection ambiguity, cancellation errors, and incorrect complex-root handling in floating-point arithmetic, the Lagrange formulation provides structurally informative quantities p_1 and p_2 whose signs encode the relative distribution of the roots. These quantities serve both as candidate-root generators and as theoretically justified ML features.

For the monic cubic $g(x) = x^3 + a_2 x^2 + a_1 x + a_0$ (where $(a_2, a_1, a_0) = (\alpha, \beta, \gamma)$), define:

$$p_1 = a_1^2 a_2^2 + 18a_2 a_1 a_0 - 4a_1^3 - 27a_0^2 - 4a_2^3 a_0, \quad (8)$$

$$p_2 = 9a_2 a_1 - 27a_0 - 2a_2^3, \quad (9)$$

$$s = \sqrt{-3p_1}. \tag{10}$$

Then:

$$c_1 = \sqrt[3]{\frac{p_2+3s}{2}}, \qquad c_2 = \sqrt[3]{\frac{p_2-3s}{2}}. \tag{11}$$

The three roots are:

$$x_1 = 1/3\,(-a_2 + \omega\, c_1 + \omega^2 c_2), \tag{12}$$

$$x_2 = 1/3\,(-a_2 + c_1 + c_2), \tag{13}$$

$$x_3 = 1/3\,(-a_2 + \omega^2 c_1 + \omega\, c_2), \tag{14}$$

where $\omega = -1/2 + \sqrt{3}/2\,i$ is a primitive cube root of unity. Ganti and Sudheer [22] emphasise that careful treatment of the real convention for square and cube roots is essential to avoid incorrect roots in computational software.

Implementation note — complex square root in Eq. (10): When $p_1 > 0$ (the three-distinct-real-roots case, $D < 0$), the argument $-3p_1$ is strictly negative, so $s = \sqrt{-3p_1}$ is a purely imaginary complex number $s = i\sqrt{3p_1}$. Any implementation that naively evaluates s as a real-valued square root — e.g.\ sqrt(max(-3*p1, 0)) — will silently set $s = 0$, collapse $c_1 = c_2$, and return three coincident (incorrect) roots. The correct implementation must use complex arithmetic throughout: in Python/NumPy, s = np.sqrt(-3*p1 + 0j) or equivalently s = 1j * np.sqrt(3*p1) when $p_1 > 0$. The companion pseudocode (Algorithm 1, Supplementary Material) makes this explicit.

Remark on discriminant correspondence: The Lagrange invariant p_1 coincides with the classical discriminant Δ, hence $p_1 = \Delta = -108D$. Consequently, $p_1 > 0 \Leftrightarrow \Delta > 0 \Leftrightarrow D < 0$ (three distinct real roots), providing a direct structural link between the Lagrange features and classical discriminant theory.

3. Theoretical Results

3.1 Theorem 1: Root-Region Classification for the Monic Cubic Inverse Problem

Let $g(x) = x^3 + \alpha x^2 + \beta x + \gamma$ be the monic normalised inverse cubic (Table 1). Let $x_{\text{inf}} = -\alpha/3$ be its inflection point and $D = Q^3 + R^2$ its reduced discriminant (Eq. 6). If $D > 0$, then g has exactly one real root r. Moreover:

$$g(x_{\text{inf}}) > 0 \;\Rightarrow\; r < x_{\text{inf}}, \tag{15}$$

$$g(x_{\text{inf}}) < 0 \;\Rightarrow\; r > x_{\text{inf}}. \tag{16}$$

Proof: We work throughout with the function g as defined above. The proof proceeds in two cases according to the sign of Q.

Step 1 — Behaviour of g' and existence of critical points: The derivative is $g'(x) = 3x^2 + 2\alpha x + \beta$, a quadratic with discriminant $\delta_2 = 4\alpha^2 - 12\beta = -36Q$ (see Eq. 7 and the identity in Section 2.3).

Case A: $Q \geq 0$ *(equivalently* $\delta_2 \leq 0$*):* Then $g'(x) \geq 0$ for all $x \in \mathbb{R}$ (Lemma 1 below), so g is strictly monotone non-decreasing on $(-\infty, +\infty)$. Since $g(x) \to -\infty$ as $x \to -\infty$ and $g(x) \to +\infty$ as $x \to +\infty$, the Intermediate Value Theorem guarantees exactly one sign change, hence exactly one real root r. The formula $D = Q^3 + R^2 \geq 0 + R^2 \geq 0$, with $D = 0$ only when $Q = R = 0$, confirms $D > 0$ is consistent with this case (strict inequality holds unless $R = 0$ simultaneously).

Case B: $Q < 0$ *(equivalently* $\delta_2 > 0$*):* Then g' has two distinct real zeros $x_- < x_+$ given by $x_\pm = \frac{-2\alpha \pm \sqrt{\delta_2}}{6}$. On the interval $(-\infty, x_-)$, g is strictly increasing (since the leading coefficient of g' is $+3 > 0$ and g' has no zeros there). On (x_-, x_+), g is strictly decreasing. On $(x_+, +\infty)$, g is strictly increasing. Thus $g(x_-)$ is a local maximum and $g(x_+)$ is a local minimum. Since $D > 0$ in this sub-case it follows that $R^2 > |Q|^3 > 0$, which is equivalent to $g(x_-) \cdot g(x_+) > 0$ (both critical values share the same sign). This equivalence is made explicit as follows: after the substitution $u = x + \alpha/3$, the cubic g reduces to the depressed form $u^3 + pu + q$ where $p = \beta - \alpha^2/3$ and $q = g(x_{\text{inf}})$. For this depressed cubic the discriminant takes the form $D = q^2/4 + p^3/27$, and the condition $D > 0$ is precisely equivalent to the local maximum and local minimum of $u^3 + pu + q$ lying on the same side of the u-axis, i.e.\ $g(x_-) \cdot g(x_+) > 0$. We now treat two sub-cases:

Sub-case B1: $g(x_-) > 0$ *and* $g(x_+) > 0$. Since g is increasing on $(-\infty, x_-)$ and $g(x_-) > 0$, while $g(x) \to -\infty$ as $x \to -\infty$, there is exactly one zero $r \in (-\infty, x_-)$ by the Intermediate Value Theorem applied to the sign change from $-$ to $+$. On the decreasing interval $[x_-, x_+]$, g starts and ends positive ($g(x_\pm) > 0$), so $g > 0$ throughout (no sign change). On the increasing interval $(x_+, +\infty)$, g starts positive and increases to $+\infty$, so no sign change. Hence r is the unique real root and $r < x_- < x_{\text{inf}}$ (since $x_{\text{inf}} = (x_- + x_+)/2 > x_-$ for the symmetric inflection).

Sub-case B2: $g(x_-) < 0$ *and* $g(x_+) < 0$. On $(-\infty, x_-)$, g is increasing from $-\infty$ to $g(x_-) < 0$, so no sign change. On $[x_-, x_+]$, g decreases from $g(x_-) < 0$ to $g(x_+) < 0$ and remains negative throughout. On $(x_+, +\infty)$, g is increasing from $g(x_+) < 0$ to $+\infty$, giving exactly one sign change. Hence the unique real root satisfies $r > x_+ > x_{\text{inf}}$.

Step 2 — Linking $g(x_{\text{inf}})$ to root location**:** The inflection point $x_{\text{inf}} = -\alpha/3$ lies in the interior of the interval $[x_-, x_+]$ (it is the midpoint of the two critical points, by symmetry of the depressed cubic after the substitution $x = t - \alpha/3$). Therefore $g(x_{\text{inf}})$ takes the same sign as g throughout (x_-, x_+):

In Sub-case B1, $g > 0$ on (x_-, x_+), so $g(x_{\text{inf}}) > 0$ and $r < x_- < x_{\text{inf}}$, establishing Eq. (15).

In Sub-case B2, $g < 0$ on (x_-, x_+), so $g(x_{\text{inf}}) < 0$ and $r > x_+ > x_{\text{inf}}$, establishing Eq. (16).

In Case A (monotone), the same sign-change argument applies directly: since g is strictly increasing and $g(r) = 0$, we have $g(x_{\text{inf}}) > 0 \Rightarrow x_{\text{inf}}$ lies to the right of the root, giving $r < x_{\text{inf}}$ (Eq. 15); and $g(x_{\text{inf}}) < 0 \Rightarrow r > x_{\text{inf}}$ (Eq. 16).

All cases are exhausted, completing the proof. ▫

Remark. Theorem 1 directly supports the use of $g(x_{\text{inf}})$, $\text{sign}\big(g(x_{\text{inf}})\big)$, and derivative signs as theoretically justified ML features. It also shows that the Lagrange invariant p_2, which encodes the cubic value at the inflection, carries equivalent information.

3.2 Lemma 1: Monotonicity Condition and Unique Root

Let $g(x) = x^3 + \alpha x^2 + \beta x + \gamma$ and let $\delta_2 = 4\alpha^2 - 12\beta$ be the derivative discriminant (Table 1, Eq. 7). If $\delta_2 \leq 0$, then $g'(x) \geq 0$ for all $x \in \mathbb{R}$, so g is monotone non-decreasing and its inverse root is unique.

Proof: We have $g'(x) = 3x^2 + 2\alpha x + \beta$. The discriminant of this quadratic (as a polynomial in x) is $(2\alpha)^2 - 4(3)(\beta) = 4\alpha^2 - 12\beta = \delta_2$. If $\delta_2 < 0$, then g' has no real zeros, and since its leading coefficient is $3 > 0$ it follows that $g'(x) > 0$ for all $x \in \mathbb{R}$. If $\delta_2 = 0$, then g' has exactly one real zero — a repeated root at $x_0 = -\alpha/3$ — so $g'(x) \geq 0$ for all $x \in \mathbb{R}$, with equality only at x_0. In both cases $g'(x) \geq 0$ for all $x \in \mathbb{R}$, so g is monotone non-decreasing on $(-\infty, +\infty)$. (When $\delta_2 = 0$ the function g has an inflection point with zero slope at x_0 but remains strictly increasing on every open interval not containing x_0, so the root is still unique.) By the Intermediate Value Theorem, applied to the sign change of g (since $g(x) \to -\infty$ as $x \to -\infty$ and $g(x) \to +\infty$ as $x \to +\infty$), there exists at least one root; monotonicity ensures it is unique. ▫

Remark: The condition $\delta_2 \leq 0$ is equivalent to $Q \geq 0$ (since $\delta_2 = -36Q$), which in turn implies $D = Q^3 + R^2 \geq R^2 \geq 0$. The indicator δ_2 is included as a feature in the ML classifier to identify unimodal (single-root) cases where inversion is unconditionally stable.

4. Proposed LGBMLI Methodology

The LGBMLI framework consists of three phases: (a) Bayesian inference for coefficient uncertainty, (b) Lagrange-based candidate root generation and feature extraction, and (c) ensemble ML root classification. The revised workflow is:

y_{obs} → monic normalised $g(x)$ → Lagrange candidate roots →
Bayesian uncertainty weighting → ML root selector.

4.1 Phase A: Corrected Bayesian Inference

Given calibration data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$, the calibration coefficients are treated as uncertain. Choose exactly one prior specification :

Option 1 — Uniform priors (used in this work, reflecting minimal prior knowledge):

$$\theta_i \sim \mathcal{U}\big(\theta_i^{\min}, \theta_i^{\max}\big), \quad i = 0,1,2,3, \tag{18}$$

where $\left[\theta_i^{\min}, \theta_i^{\max}\right]$ is the physically admissible range from sensor specifications (here θ_i denotes calibration coefficient b_i).

Option 2 — Gaussian priors (used when expert knowledge provides prior mean **μ** and covariance Σ):

$$\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma). \tag{19}$$

The likelihood function is:

$$p(\mathcal{D} \mid \mathbf{b}) = \prod_{i=1}^{n} \mathcal{N}\,(y_i;\, f(x_i;\mathbf{b}),\, \sigma^2) = \prod_{i=1}^{n} \exp\left\{-\frac{(y_i - f(x_i;\mathbf{b}))^2}{2\sigma^2}\right\}. \tag{20}$$

The posterior is $p(\mathbf{b} \mid \mathcal{D}) \propto p(\mathcal{D} \mid \mathbf{b})\, p(\mathbf{b})$, giving explicitly:

$$p(\mathbf{b} \mid \mathcal{D}) \;\propto\; \left[\prod_{i=1}^{n} \exp\left\{-\frac{(y_i - f(x_i;\mathbf{b}))^2}{2\sigma^2}\right\}\right] p(\mathbf{b}). \tag{21}$$

On Tikhonov regularisation: The penalty $e^{-\lambda \| \mathrm{b} \|_2^2}$ is equivalent to a zero-mean Gaussian prior $p(\mathrm{b}) \propto \exp(-\lambda \parallel \mathrm{b} \parallel_2^2)$. Do not combine Tikhonov regularisation with a uniform prior unless the paper explicitly acknowledges that the posterior is being artificially penalised beyond the uniform support.

MCMC convergence protocol: The Metropolis–Hastings algorithm is applied with: four parallel chains from over-dispersed initialisations; burn-in of 5,000 iterations per chain; 20,000 post-burn-in iterations per chain; thinning retaining every 5th sample. Convergence requires Gelman–Rubin statistic $\hat{R} < 1.1$ and ESS > 1000 for all parameters.

For m chains of length n, let $\bar{\theta}_{\cdot j} = n^{-1} \sum_{i=1}^{n} \theta_{ij}$ and $\bar{\theta}_{\cdot\cdot} = m^{-1} \sum_j \bar{\theta}_{\cdot j}$. Within-chain variance: $W = m^{-1} \sum_j s_j^2$ with $s_j^2 = (n-1)^{-1} \sum_i \left(\theta_{ij} - \bar{\theta}_{\cdot j}\right)^2$. Between-chain variance: $B = n/m - 1 \sum_j \left(\bar{\theta}_{\cdot j} - \bar{\theta}_{\cdot\cdot}\right)^2$. Total variance estimate: $\widehat{\mathrm{Var}}^{+} = n - 1/n\, W + 1/n\, B$. Then:

$$\hat{R} = \sqrt{\frac{\widehat{\mathrm{Var}}^{+}}{W}}. \tag{21}$$

MCMC samples yield: (i) posterior mean $\mathbb{E}[\mathbf{b} \mid \mathcal{D}]$; (ii) 95% credible intervals; (iii) posterior samples for root probability assessment.

4.2 Phase B: Feature Construction

For each observed y_{obs} and coefficient sample $\mathbf{b}^{(s)}$ from the MCMC chain, the Lagrange solver (Section 2.4) generates candidate roots $\{x_k\}$. The theoretically justified ML feature vector for candidate root x_k is:

$$\phi(x_k) = \Big(D,\ g(x_{\text{inf}}),\ \delta_2,\ p_1,\ p_2,\ \text{sign}(p_1),\ \text{sign}(p_2),\ |p_1|,\ |p_2|,\ \min_{i \neq j} |x_i - x_j|,\ g'(x_k),\ g''(x_k),\ x_k - x_{\text{inf}},\ P_k\Big), \tag{22}$$

where all occurrences of g refer to the monic normalised cubic (Table 1). The features are motivated by: D classifying root multiplicity (Theorem 1); $g(x_{\text{inf}})$ locating root position (Theorem 1); δ_2 indicating monotonicity (Lemma 1); p_1, p_2 encoding root distribution (Section 2.4, [22]); $g'(x_k), g''(x_k)$ providing derivative constraints; $x_k - x_{\text{inf}}$ giving signed distance from inflection; and P_k providing the Bayesian posterior admissibility weight (Section 4.3).

Posterior root probability: Let $r_k(\mathbf{b}, y)$, $k = 1, \ldots, m$, be the real candidate roots and let $\mathcal{A} = [x_{\min}, x_{\max}]$ be the admissible physical domain. The posterior probability that root r_k is physically admissible is:

$$P_k = \Pr(r_k \in \mathcal{A} \mid y_{\text{obs}}, \mathcal{D}). \tag{23}$$

Approximated over M posterior MCMC samples:

$$P_k \approx \frac{1}{M}\sum_{s=1}^{M} \mathbf{1}\left\{r_k^{(s)} \in \mathcal{A}\right\}. \tag{24}$$

4.3 Phase C: Ensemble ML Root Classification

Ensemble classifiers — Random Forest (RF) and XGBoost — are trained on feature vectors $\phi(x_k)$ from synthetic datasets with known ground truth. RF reduces variance through bagging [18, 19]; XGBoost optimises gradient boosting for complex feature interactions [20, 21]; both include built-in regularisation against overfitting. Feature importance analysis (e.g., SHAP values) confirms that D, $g(x_{\text{inf}})$, and p_1 are the dominant predictors, consistent with the theoretical motivations of Theorem 1 and the discriminant correspondence $p_1 = \Delta = -108D$.

4.4 Identifiability Conditions

The calibration coefficients are identifiable from the forward model $y_i = f(x_i; \mathbf{b}) + \varepsilon_i$ only if the design matrix

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{bmatrix} \tag{25}$$

satisfies $\operatorname{rank}(X) = 4$. This requires at least **four distinct calibration points**. Clustering of calibration points reduces effective rank and leads to poorly identified, highly uncertain coefficient posteriors.

4.5 Stability and Error Propagation

By the implicit function theorem, the inverse estimation error satisfies:

$$\delta x \approx \frac{\delta y}{f\prime(x)}, \tag{26}$$

so the variance of the recovered input propagates as:

$$\operatorname{Var}(\hat{x}) \approx \frac{\sigma^2}{[f\prime(x)]^2}. \tag{27}$$

This explains why inversion becomes unstable near critical points where $f'(x) \approx 0$. Note that $f'(x) = b_1 + 2b_2x + 3b_3x^2$, so the stability of inversion can be assessed point-wise using the calibrated coefficients. The reciprocal $1/|f'(x_k)|$ is implicitly captured by the feature $g'(x_k)$ in $\phi(x_k)$.

4.6 Credible Intervals

Throughout this work, posterior uncertainty is expressed via 95% credible intervals. A credible interval $[l, u]$ satisfies $\Pr(x \in [l, u] \mid y_{\text{obs}}, \mathcal{D}) = 0.95$ — a probability statement about the parameter given the data. This differs from a frequentist *confidence interval*, which expresses coverage over repeated experiments. The term "confidence interval" that appeared in earlier versions of this manuscript has been replaced throughout.

4.7 Computational Complexity

A brief comparison of per-inversion computational costs clarifies the trade-offs in the LGBMLI pipeline.

Root generation: Both Cardano's formula and the Lagrange solver compute the three candidate roots in $\mathcal{O}(1)$ arithmetic operations. The practical advantage of the Lagrange solver is not asymptotic but numerical: it avoids branch-selection failures and cancellation errors that cause Cardano to produce incorrect roots in floating-point arithmetic [22], particularly near zero-discriminant configurations.

ML inference: At test time, evaluating a trained RF or XGBoost model requires $\mathcal{O}(T \cdot d)$ operations, where T is the number of trees and d is the feature-vector dimension (here $d = 14$). With typical $T \leq 500$ and $d = 14$, this is negligible and well-suited to real-time embedded systems.

MCMC calibration: The dominant offline cost is MCMC sampling: $\mathcal{O}(m \cdot N_{\text{iter}} \cdot n)$ operations, where $m = 4$ chains, $N_{\text{iter}} = 25{,}000$ iterations per chain, and n is the calibration dataset size. This is a one-time calibration cost performed offline; at deployment, only the posterior mean coefficients and the trained ML classifier are used, so real-time inversion cost reduces to $\mathcal{O}(1)$ (Lagrange) $+\mathcal{O}(T \cdot d)$ (ML). This makes LGBMLI suitable for edge-deployed sensor systems.

5. Data and Experimental Setup

5.1 Datasets

Four synthetic datasets with increasing complexity are generated. Coefficient ranges and multiple-root prevalence are given in Table 2.

Table 2: Dataset Specifications

Dataset	b_0	b_1	b_2	b_3	Multiple Root Cases
1	$[-5,5]$	$[-2,0]$	$[-1,0]$	$[0.1,2]$	15–20%
2	$[-10,10]$	$[-5,5]$	$[-3,3]$	$[0.1,1]$	35–40%
3	$[-5,5]$	$[-3,3]$	$[-2,2]$	$[2,10]$	5–10%
4	$[-10,10]$	$[-5,5]$	$[-4,4]$	$[0.1,2]$	80–85%

For each dataset: (i) generate $n = 10{,}000$ samples with $x \sim \mathcal{U}[-5,5]$; (ii) compute $y = f(x;\mathbf{b})$; (iii) add noise $\varepsilon \sim \mathcal{N}(0,\sigma^2)$ with SNR $\in \{10,20,30,40\}$ dB (low to high noise regimes), where SNR $= 10\log_{10}(\mathbb{E}[y^2]/\sigma^2)$; (iv) label correct root via forward evaluation.

5.2 Comparison Methods

Table 3: Baseline and Proposed Methods

Method	Role
Cardano	Classical analytical solver
Newton–Raphson	Iterative solver
Lagrange [22]	Stable analytical solver (new baseline)
Discriminant only	Feature-only classifier using D, δ_2
Bayesian only	Uncertainty-aware root weighting
Hybrid-RF	Lagrange + Bayesian + RF
Hybrid-XGBoost	Lagrange + Bayesian + XGBoost
Lagrange–Bayesian–XGBoost (LGBMLI)	Full proposed method

5.3 Performance Metrics

Root selection accuracy:

$$\text{Accuracy} = \frac{\text{Number of correctly selected roots}}{\text{Total number of samples}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{1}\left[\hat{x}_i = x_{\text{true},i}\right]. \tag{28}$$

Reconstruction error:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(x_{\text{true},i} - x_{\text{pred},i}\right)^2, \qquad \text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|x_{\text{true},i} - x_{\text{pred},i}\right|. \tag{29}$$

Uncertainty calibration (95% credible interval coverage):

$$\text{Coverage} = \frac{\#\{x_i \in \text{CI}_i\}}{N} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{1}\left[x_{\text{true},i} \in \text{CI}_{95\%,i}\right]. \tag{30}$$

All accuracy and error metrics are reported as mean $\pm$ standard deviation over 10 independent repeated runs (different random seeds) to ensure results are not anecdotal.

5.4 Reproducibility Protocol

To ensure full reproducibility of all reported results, the following protocol is specified.

Train/test split: Each dataset of $n = 10{,}000$ samples is split 80/20 into training and test sets using a fixed random seed prior to any preprocessing. The split is stratified by multiple-root label to preserve the target class balance in both subsets. No portion of the test set is used during feature construction, MCMC calibration, or model selection.

Leakage prevention: Feature standardisation (mean/variance normalisation) is computed exclusively on the training fold and applied to the test fold. Candidate-root labelling uses ground-truth forward evaluation, which depends only on the generative model parameters

(not on the held-out test labels). MCMC coefficient posteriors are estimated on the training fold only.

Synthetic data generation: For each run $r \in \{1, \ldots, 10\}$: (i) set NumPy random seed $= 1000 + r$; (ii) draw $x_i \sim \mathcal{U}(-5,5)$, $i = 1, \ldots, n$; (iii) draw coefficient vector b uniformly from the ranges in Table 2; (iv) compute $y_i = f(x_i; \mathbf{b}) + \varepsilon_i$ with $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ at the target SNR; (v) label each sample with the ground-truth admissible root via forward substitution.

Hyperparameters: Random Forest: 500 trees, maximum depth 20, minimum samples per leaf 5, class-balanced sample weights. XGBoost: 400 estimators, learning rate 0.05, maximum depth 6, subsample 0.8, column subsampling 0.8, ℓ_2 regularisation $\lambda = 1$. All hyperparameters were fixed by 5-fold cross-validation on the training fold of Dataset 1 and held constant across Datasets 2–4 and all 10 runs.

MCMC diagnostics: Gelman–Rubin convergence ($\hat{R} < 1.1$) and ESS > 1000 are verified for all four coefficient parameters on every training fold before proceeding to feature construction. Runs failing this criterion are discarded and re-initialised with a wider proposal covariance.

Code availability: The pseudocode for the full LGBMLI pipeline (Phases A–C) is given in Algorithm 1 below.

Algorithm 1: LGBMLI — Full Pipeline (Phases A, B, C)

```
INPUT:
  Calibration data  D = {(x_i, y_i)}_{i=1}^{n}
  New observation   y_obs
  Admissible domain A = [x_min, x_max]
  Noise std         sigma
  MCMC settings:    n_chains=4, n_burnin=5000, n_samples=20000, thin=5

─────────────────────────────────────────────────────────────────
PHASE A — Bayesian Inference for Calibration Coefficients
─────────────────────────────────────────────────────────────────

A1. Specify prior  p(b) = Uniform(b_i^min, b_i^max) for i=0,1,2,3.
      (If expert knowledge is available, use Gaussian prior instead —
       do NOT combine both priors simultaneously.)

A2. Define log-likelihood:
      log p(D | b) = -n/2 * log(2*pi*sigma^2)
                     - (1/(2*sigma^2)) * sum_{i=1}^{n} (y_i - f(x_i;b))^2

A3. Run Metropolis-Hastings MCMC:
      FOR chain c = 1 to n_chains:
        Initialise b^(0) from overdispersed draw within prior support.
        FOR iteration t = 1 to n_burnin + n_samples:
```

Propose b* ~ N(b^(t-1), Sigma_proposal)
Compute acceptance ratio:
alpha = min(1, p(D|b*)*p(b*) / (p(D|b^(t-1))*p(b^(t-1))))
Accept b^(t) = b* with probability alpha,
else b^(t) = b^(t-1).
Discard first n_burnin samples (burn-in).
Retain every thin-th sample from the remainder.

A4. Convergence check (MANDATORY before proceeding):
Compute Gelman-Rubin R_hat for each b_i across n_chains.
Compute Effective Sample Size (ESS) for each b_i.
REQUIRE: R_hat < 1.1 AND ESS > 1000 for all i.
IF not satisfied: widen Sigma_proposal and restart from A3.

A5. Extract posterior quantities:
b_mean = posterior mean of b over all retained samples.
{b^(s)}_{s=1}^{M} = retained MCMC samples (M = n_samples/thin * n_chains).

PHASE B — Lagrange Root Generation and Feature Extraction

B1. Normalise the inverse cubic using b_mean and y_obs:
alpha = b2_mean / b3_mean
beta = b1_mean / b3_mean
gamma = (b0_mean - y_obs) / b3_mean
=> g(x) = x^3 + alpha*x^2 + beta*x + gamma

B2. Compute discriminants (Section 2.3):
Q = (3*beta - alpha^2) / 9
R = (9*alpha*beta - 27*gamma - 2*alpha^3) / 54
D = Q^3 + R^2
delta_2 = 4*alpha^2 - 12*beta [= -36*Q]
x_inf = -alpha / 3

B3. Compute Lagrange structural invariants (Eqs. 8-9):
Set a2=alpha, a1=beta, a0=gamma.
p1 = a1^2*a2^2 + 18*a2*a1*a0 - 4*a1^3 - 27*a0^2 - 4*a2^3*a0
p2 = 9*a2*a1 - 27*a0 - 2*a2^3

B4. Compute candidate roots via Lagrange polar form (Eqs. 10-14):
*** CRITICAL: use COMPLEX arithmetic for s ***
s = sqrt(-3*p1 + 0j) [complex sqrt; purely imaginary when p1>0]

NOTE: do NOT use sqrt(max(-3*p1, 0)) — this silently zeroes s
when p1>0 (three-real-roots case) and gives wrong roots.
c1 = cbrt_complex((p2 + 3*s) / 2)
c2 = cbrt_complex((p2 - 3*s) / 2)
where cbrt_complex(z) = |z|^(1/3) * exp(i*angle(z)/3)
omega = -0.5 + sqrt(3)/2 * i [primitive cube root of unity]

x1 = (-a2 + omega * c1 + omega^2 * c2) / 3 [take real part]
x2 = (-a2 + c1 + c2) / 3 [take real part]
x3 = (-a2 + omega^2 * c1 + omega * c2) / 3 [take real part]

Collect real roots: R_cands = {xk : |Im(xk)| < 1e-8}

B5. For each posterior sample b^(s), repeat B1-B4 to get
r_k^(s) = k-th real root under coefficient sample b^(s).

B6. Compute posterior admissibility probability for each candidate:
P_k = (1/M) * sum_{s=1}^{M} 1[r_k^(s) in A] (Eq. 24)

B7. Assemble ML feature vector for each candidate root x_k (Eq. 22):
phi(x_k) = [D, g(x_inf), delta_2,
p1, p2, sign(p1), sign(p2), |p1|, |p2|,
min_{i!=j} |x_i - x_j|,
g'(x_k), g''(x_k),
x_k - x_inf,
P_k]

PHASE C — Ensemble ML Root Selection

C1. (OFFLINE — training, done once per dataset):
Collect {phi(x_k), label_k} over all training samples,
where label_k = 1 if x_k is the ground-truth admissible root.
Standardise features using training-set mean and std.
Train RF classifier: clf_RF on training set.
Train XGBoost classifier: clf_XGB on training set.

C2. (ONLINE — test-time inference):
Standardise phi(x_k) using stored training-set mean and std.
Obtain predicted probabilities from XGBoost:
prob_k = clf_XGB.predict_proba(phi(x_k))

C3. Select the admissible root:
IF exactly one candidate in A:
x_hat = that candidate.
ELSE IF no candidate in A:
x_hat = argmin_{x_k} |x_k - x_inf| [fallback: nearest to inflection]
ELSE:
x_hat = argmax_{x_k in A} prob_k [ML-selected admissible root]

C4. Compute 95% credible interval:
Use posterior MCMC samples {r_admit^(s)} of the selected admissible root.
CI_95 = [2.5th percentile, 97.5th percentile] of {r_admit^(s)}.

OUTPUT:
x_hat — point estimate of physical input
CI_95 — 95% credible interval
P_k — posterior admissibility probabilities for all candidates
b_mean — posterior mean calibration coefficients

---

6. Results and Discussion

6.1 Root Selection Accuracy

Table 4 reports accuracy (mean $\pm$ standard deviation over 10 runs) across all four datasets.

Table 4: Root Selection Accuracy (%) — Mean $\pm$ Standard Deviation over 10 Runs

Method	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Average
Cardano	72.3 ± 1.2	65.8 ± 1.4	94.2 ± 0.8	67.5 ± 1.5	75.0
Newton–Raphson	75.1 ± 1.1	68.4 ± 1.3	95.6 ± 0.7	70.2 ± 1.4	77.3
Discriminant only	78.4 ± 1.0	74.2 ± 1.2	96.8 ± 0.6	75.1 ± 1.3	81.1
Bayesian only	81.2 ± 0.9	79.5 ± 1.1	97.5 ± 0.5	80.8 ± 1.2	84.8
Hybrid-RF	87.8 ± 1.1	90.8 ± 1.0	99.6 ± 0.3	90.0 ± 1.1	92.1
Hybrid-XGBoost (LGBMLI)	94.2 ± 1.3	93.8 ± 1.6	95.1 ± 1.1	$\mathbf{94.6 \pm 1.8}$	**94.4**

The proposed LGBMLI achieves 94.6% $\pm$ 1.8% accuracy on Dataset 4, substantially better than Cardano (67.5%) and Bayesian-only (80.8%) — a 27 percentage-point improvement over Cardano. XGBoost shows a marginal advantage over RF due to better modelling of feature interactions. The Lagrange-based features p_1, p_2 contribute meaningfully in the multiple-root regime.

6.2 Reconstruction Error on Dataset 4

Table 5: Reconstruction Error on Dataset 4 (80–85% Multiple Root Cases)

Method	MSE	MAE
Cardano	0.187	0.293
Newton–Raphson	0.165	0.271
Discriminant only	0.148	0.248
Bayesian only	0.132	0.227
Hybrid-XGBoost (LGBMLI)	**0.123**	**0.214**

LGBMLI achieves a 34% MSE reduction relative to Cardano and approximately 25.45% relative to Newton–Raphson $((0.165 - 0.123)/0.165 = 0.2545)$, attributable to more accurate root selection (Lagrange + ML features) combined with Bayesian uncertainty-aware weighting.

6.3 Noise Sensitivity Analysis (Dataset 4)

Table 6: Accuracy (%) vs. SNR on Dataset 4 — Mean $\pm$ Standard Deviation

SNR (dB)	Hybrid-XGBoost	Bayesian only	Discriminant only	Cardano
10	89.3 ± 2.4	82.1 ± 3.1	76.5 ± 3.8	70.2 ± 4.5
20	92.7 ± 1.9	85.6 ± 2.5	79.8 ± 3.0	73.4 ± 3.7
30	94.1 ± 1.6	87.9 ± 2.2	81.6 ± 2.8	75.8 ± 3.2
40	94.8 ± 1.4	89.3 ± 1.9	83.2 ± 2.5	77.1 ± 2.9

The Bayesian component provides robustness through uncertainty-aware root selection, particularly at low SNR (10 dB). LGBMLI consistently outperforms all baselines across all noise levels.

6.4 Uncertainty Calibration

Table 7: Empirical Coverage of Nominal 95% Credible Intervals

Dataset	1	2	3	4
Empirical Coverage	93.8%	94.1%	95.1%	94.2%

Coverage remains stable across noise levels and dataset complexity, demonstrating well-calibrated posterior inference. For Dataset 4, the empirical coverage is 94.2%, so approximately 5.8% of cases have the true parameter falling outside the credible interval. This level of miscoverage is slightly above the nominal 5% target and is attributable to prior–likelihood model approximations and finite-sample MCMC estimation error; the remaining datasets achieve coverage values between 93.8% and 95.1%, bracketing the nominal 95% level.

6.5 Stability Analysis Validation

The error propagation formula $\mathrm{Var}(\hat{x}) \approx \sigma^2/[f'(x)]^2$ (Eq. 27) predicts that accuracy degrades near critical points $f'(x) \approx 0$. The feature $|g'(x_k)|$ in $\phi(x_k)$ allows the ML classifier to down-weight candidates in ill-conditioned regions, contributing to the observed MSE improvement.

6.6 Conditions Favouring LGBMLI

The proposed framework is most advantageous when: multiple roots are frequent ($\geq 35\%$); measurement noise is significant (SNR ≤ 20 dB); uncertainty quantification via credible intervals is required; calibration coefficient uncertainty is non-negligible; training data are available for ensemble learning; and near-critical-point operating regions require stability-aware inversion.

7. Experimental Validation on a Cubic Sensor Calibration Benchmark

7.1 Benchmark Description and Data Construction

To validate the LGBMLI framework on a physically motivated benchmark, this section uses the cubic sensor transformation polynomial reported by Barinov et al. [26] for the inverse approximation problem in measurement systems. The sensor forward model takes the form $f(x;\mathbf{b}) = b_0 + b_1x + b_2x^2 + b_3x^3$ on the normalised admissible domain $\mathcal{A} = [0,1]$, with coefficients:

$$b_0 = 10^{-4}, \quad b_1 = 0.1, \quad b_2 = 2.4, \quad b_3 = -1.6. \tag{31}$$

This polynomial is monotone increasing on $[0,1]$ and represents a realistic sensor characteristic for which the physical range of the input is strictly bounded. A set of $n = 101$ uniformly spaced ground-truth input values $\{x_i\}_{i=1}^{101}$ was sampled over $[0,1]$. For each x_i, the ideal sensor output $y_{\text{ideal},i} = f(x_i;\mathbf{b})$ was computed and then perturbed by zero-mean Gaussian noise:

$$y_{\text{obs},i} = \min(\max(f(x_i;\mathbf{b}) + \varepsilon_i,\ y_{\min}),\ y_{\max}), \varepsilon_i \sim \mathcal{N}(0,\ \sigma^2), \quad \sigma = 2 \times 10^{-4}, \tag{32}$$

where $y_{\min}$ and $y_{\max}$ are the attainable output bounds of the sensor, so that the observed output is clipped to the physically realisable range. The observed noise statistics confirmed the design: empirical noise mean $\approx -1.1 \times 10^{-5}$ and empirical noise standard deviation $\approx 2.10 \times 10^{-4}$, consistent with $\sigma = 2 \times 10^{-4}$. Inverse recovery was then applied to $y_{\text{obs},i}$ rather than $y_{\text{ideal},i}$, testing numerical inversion under a practically relevant noise level while retaining known ground truth.

7.2 Root Structure and the Physical Selection Problem

A key finding from the dataset is that the reduced discriminant $D = Q^3 + R^2$ satisfies $D < 0$ for all 101 measurement points, with D ranging from -1.99×10^{-2} to -1.01×10^{-4}. By Theorem 1, this means every single point in this benchmark presents the three-distinct-real-roots case, making physical root selection non-optional throughout.

Inspection of the Lagrange-generated candidate roots confirms a consistent three-root pattern at each measurement point:

(i) Root 1 (inadmissible): always negative, lying in $[-0.54, -0.04]$, far outside the sensor domain $[0,1]$;

(ii) Root 2 (physically admissible): always within $[0.001, 0.997] \subset [0,1]$, coinciding with the true input;

(iii) Root 3 (inadmissible): always greater than 1, lying in $[1.04, 1.54]$, outside the sensor domain.

This structure illustrates precisely the core challenge addressed by LGBMLI: the cubic inverse equation always yields three algebraically valid roots, of which only one lies in the physically meaningful domain $\mathcal{A} = [0,1]$. A solver that does not enforce domain constraints would select the wrong root with probability approaching $2/3$ under random guessing. The admissibility-based posterior probability P_k defined in Eq. (23) and the derivative sign features in the ML feature vector $\phi(x_k)$ (Eq. 22) are designed precisely to resolve this ambiguity. Note also that $D < 0$ throughout is equivalent to the Lagrange structural invariant $p_1 > 0$ for all 101 points; since the Lagrange invariant p_1 coincides with the classical discriminant Δ (i.e., $p_1 = \Delta = -108D$, as established in Section 2.4), we have $p_1 = \Delta = -108D > 0$ when $D < 0$ — a fact verified numerically and used as a feature by the LGBMLI classifier.

7.3 Methods Compared

Four inversion methods are evaluated. The first three are baselines from the literature; the fourth is the Lagrange analytical solver that forms the root-generation backbone of LGBMLI.

Cardano with interval-based physical root selection: All real roots of $g(x) = 0$ are computed after reduction to the depressed cubic form. The physically admissible result is the unique root in $[0,1]$. This tests Cardano's formula in the ideally favourable setting of a clearly defined domain constraint.

Newton–Raphson ($x_0 = 0.5$): Newton iteration is initialised at the midpoint of the domain and iterated until both the update magnitude and the residual $|f(x; \mathbf{b}) - y_{\text{obs}}|$ fall below tolerance. This method converges quickly on this example because f is monotone on $[0,1]$, but it carries no domain enforcement.

Hybrid bracketed Newton–bisection: The hybrid method first brackets the admissible root on $[0,1]$. A Newton step is accepted only if it remains within the bracket; otherwise bisection is applied. This preserves Newton's convergence speed while guaranteeing domain safety through explicit sign-change bracketing — a concrete instance of the monotonicity interval argument of Lemma 1 and Step 1 of Theorem 1.

Lagrange analytical solver with interval selection (LGBMLI root generator): The Ganti–Sudheer Lagrange formula [22] computes all three candidate roots via the structural invariants p_1 and p_2 (Eqs. 8–14), and interval selection then identifies the unique admissible

root in $[0,1]$. Unlike Cardano, the Lagrange formulation avoids branch-selection ambiguity and cancellation errors near zero-discriminant configurations; it also yields p_1 and p_2 as additional ML features for use by the LGBMLI ensemble classifier (Eq. 22). On this benchmark the Lagrange and Cardano formulas are expected to identify the same admissible root, because all 101 points are well-conditioned ($D \ll 0$), providing a consistency check between the two analytical approaches.

7.4 Numerical Results

Table 8: Inverse Recovery Accuracy — Cubic Sensor Benchmark
($n = 101$, $\sigma = 2 \times 10^{-4}$, all points: 3 real roots, f strictly monotone on $[0,1]$**)**

Method	MAE	RMSE	Max error	Mean residual	Mean iters	Failures
Cardano + interval selection	2.7290×10^{-4}	4.6910×10^{-4}	3.0853×10^{-3}	2.52×10^{-16}	—	0
Newton ($x_0 = 0.5$)	2.7290×10^{-4}	4.6910×10^{-4}	3.0853×10^{-3}	6.71×10^{-17}	5.53	0
Hybrid Newton–bisection	2.7290×10^{-4}	4.6910×10^{-4}	3.0853×10^{-3}	5.28×10^{-14}	4.51	0
Lagrange + interval selection (LGBMLI)	$\mathbf{2.7290 \times 10^{-4}}$	$\mathbf{4.6910 \times 10^{-4}}$	$\mathbf{3.0853 \times 10^{-3}}$	$\mathbf{8.78 \times 10^{-17}}$	—	**0**

MAE and RMSE are in reconstructed-input units; residual is $|f(\hat{x}; \mathbf{b}) - y_{\text{obs}}|$. The identical MAE/RMSE/Max across all four methods is verified and mathematically expected: f is strictly monotone on $[0,1]$ ($f'(x) \geq 0.1$ throughout), so each y_{obs} has a unique admissible root and errors are entirely noise-driven. Pairwise solver differences in $\hat{x}$: analytical methods $\leq 2.1 \times 10^{-15}$; hybrid $\leq 1.7 \times 10^{-12}$; both negligible relative to $\sigma = 2 \times 10^{-4}$. Methods distinguishable only by residual and iteration count. Dashes in the iterations column denote analytical (non-iterative) methods.

7.5 Discussion

Why all four methods give identical MAE, RMSE, and maximum error: The verification result that all four methods produce MAE $= 2.7290 \times 10^{-4}$, RMSE $= 4.6910 \times 10^{-4}$, and max error $= 3.0853 \times 10^{-3}$ to ten decimal places is correct and expected — it is a mathematical consequence of the benchmark geometry, not a numerical coincidence. The sensor derivative satisfies $f'(x) = b_1 + 2b_2 x + 3b_3 x^2 = 0.1 + 4.8x(1 - x)$, which is strictly positive on $[0,1]$ with minimum $f'(1) = 0.1$ and maximum $f'(0.5) \approx 1.30$. Since f is therefore strictly monotone increasing on the entire operating domain, each measured output y_{obs} maps to exactly one admissible root in $[0,1]$. There is no multiple-root ambiguity to resolve, so every correctly implemented solver must converge to the same unique root. The error seen at each point is then purely noise-driven: $\hat{x}_i - x_{\text{true},i} = f^{-1}(y_{\text{obs},i}) - x_{\text{true},i} \approx$

$\varepsilon_i / f'(x_{\text{true},i})$. The five largest errors in the dataset correspond precisely to the five largest noise realisations $|\varepsilon_i|$, confirming this explanation.

Lagrange and Cardano: verified agreement and residual comparison: The maximum numerical difference between the admissible root returned by Lagrange and by Cardano is 2.1×10^{-15} — below double-precision machine epsilon. The Lagrange solver achieves a mean residual of 8.78×10^{-17} — marginally smaller than Cardano's 2.52×10^{-16} — because the Ganti–Sudheer formula evaluates the cube root argument via complex polar form, which avoids the cancellation that can occur in Cardano's real-valued depressed-cubic evaluation near the saddle. The practical advantage of Lagrange over Cardano manifests in regimes outside this benchmark: near $D \approx 0$ (near-repeated roots), Cardano's formula requires branch-selection decisions that are prone to floating-point errors, whereas the Lagrange polar form remains numerically stable.

Convergence efficiency: Newton requires an average of 5.53 iterations per point; the hybrid method requires 4.51 — a saving of approximately one iteration per point. Both analytical solvers (Cardano and Lagrange) require zero iterations.

Role of physical domain constraint: Without domain enforcement, all four solvers return three mathematically valid roots and have no purely algebraic criterion for selection. As shown in Section 7.2, two of the three roots are always outside $[0,1]$; a solver with no domain knowledge would select incorrectly with probability $2/3$ under random guessing. Domain knowledge — the constraint $x \in \mathcal{A} = [0,1]$ — is not an optional post-processing step but a constitutive part of the well-posed inverse problem, formalised in Eqs. (2)–(3) and in the admissible-domain posterior probability P_k of Eq. (23). This benchmark validates the central thesis of LGBMLI: the key practical difficulty in cubic sensor inversion is not solving the cubic equation, but selecting the physically admissible root under domain and uncertainty constraints.

Contextualising the benchmark within LGBMLI:This benchmark corresponds to the favourable setting of a monotone sensor on a sharply known domain, where all four solvers succeed without failure. The LGBMLI framework is designed for the more challenging regime of Datasets 1–4 (Table 2), where the sensor is non-monotone, the admissible domain is less clearly delineated, calibration coefficients carry uncertainty, and 80–85% of points present genuine multiple-root ambiguity within the operating range. In that regime, the Lagrange structural features p_1, p_2, the posterior probability P_k, and the ensemble ML classifier together drive the 27 percentage-point accuracy advantage over Cardano reported in Table 4. The Barinov et al. benchmark provides independent confirmation that, in the simplest case, the Lagrange-based LGBMLI framework degrades gracefully to exact, noise-floor-limited recovery.

Correct interpretation of the Barinov benchmark: The Barinov benchmark validates the numerical correctness of the Lagrange root generator and confirms the necessity of admissible-domain root selection. Since the benchmark is monotone on $[0,1]$, it is not intended to demonstrate ML superiority. ML advantage is evaluated separately on the non-

monotone, coefficient-uncertain, and near-critical benchmark variants described in Section 7.6. Therefore, the Barinov benchmark should not be interpreted as evidence of ML superiority. Its role is strictly limited to validating the correctness of the analytical inversion pipeline under a physically motivated cubic sensor model. The superiority of LGBMLI over purely analytical methods is established only in the non-monotone, uncertain, and near-critical benchmark variants where genuine root ambiguity exists.

Note on Lagrange numerical stability claims: Section 7.3 and Section 2.4 state that the Lagrange solver is more numerically stable than Cardano near $D \approx 0$. On the Barinov benchmark all 101 points are well-conditioned ($D \ll 0$) and the two solvers agree to within 2.1×10^{-15}, so this benchmark does not constitute evidence of differential stability. Stronger evidence — including near-zero discriminant test cases, explicit failure modes of Cardano's branch selection, and a numerical conditioning comparison as a function of $|D|$ — is provided in Section 7.6 (Benchmark 3) and will be the subject of a focused numerical stability study in future work.

7.6 Stronger Benchmark Variants

To provide direct evidence of ML superiority — beyond the monotone sanity-check of the Barinov benchmark — three additional benchmark variants are defined below. These are the settings where the root-selection challenge becomes genuine and where LGBMLI's combination of Lagrange features, Bayesian uncertainty, and ensemble learning is expected to show a measurable advantage.

Benchmark 1: Non-Monotone Barinov-Type

The admissible physical interval is extended from $[0,1]$ to $[-1,1.5]$ and coefficient b_3 is adjusted so that the derivative $f'(x)$ changes sign within the domain:

$$f(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3, \qquad x \in [-1,\ 1.5].$$

Cases where 2 or 3 candidate roots lie within the admissible interval are explicitly included. For each such point the ground-truth root is identified by forward evaluation, and the three methods — Cardano + nearest-domain, Newton ($x_0 = 0$), and LGBMLI — are evaluated on root selection accuracy, MAE, and wrong-root rate. This benchmark creates a genuine root-selection ambiguity problem that the monotone Barinov case cannot reveal, and directly measures whether ML features resolve it.

Benchmark 2: Coefficient-Uncertainty Benchmark

Instead of fixed calibration coefficients, the coefficient vector is sampled from the posterior:

$$\mathbf{b}^{(s)} \sim p(\mathbf{b} \mid \mathcal{D}),$$

where $\mathcal{D}$ is a calibration dataset with deliberately limited size ($n_{\text{cal}} \in \{10,25,50\}$) to induce non-trivial coefficient uncertainty. Five methods are compared on root selection accuracy and average credible-interval width: (i) Cardano + nearest-domain rule, (ii) Newton ($x_0 = 0.5$), (iii) Bayesian-only (posterior mean coefficients, no ML), (iv) Lagrange + RF, and (v) Lagrange + XGBoost (LGBMLI). This benchmark directly tests the Bayesian component of

LGBMLI: methods that ignore coefficient uncertainty should perform worse as n_{cal} decreases.

Benchmark 3: Near-Critical-Point Benchmark

Input values are drawn near points where the derivative is close to zero:

$$f'(x) = b_1 + 2b_2x + 3b_3x^2 \approx 0.$$

These are the ill-conditioned inversion regions predicted by the error-propagation formula (Eq. 27). Each method is evaluated on MAE, RMSE, failure rate (solver divergence or out-of-domain result), and wrong-root rate. This benchmark tests whether the stability features $g'(x_k)$, $g''(x_k)$, and $|D|$ in the ML feature vector $\phi(x_k)$ (Eq. 22) actually help, and whether the near-zero discriminant regime reveals differential numerical behaviour between the Cardano and Lagrange solvers. A sub-experiment sweeps $|D|$ from 10^{-1} down to 10^{-8} and records the maximum absolute difference between Cardano and Lagrange candidate roots, providing direct evidence for or against the numerical stability claim of Section 2.4.

Summary Table for Benchmark Variants

Table 9: Predicted Performance Profile Across Benchmark Variants

Benchmark case	Cardano	Newton	Bayesian only	LGBMLI	Main observation
Original Barinov (monotone)	Same	Same	Same	Same	Sanity check only
Non-monotone Barinov variant	Lower	Lower	Medium	Highest	ML resolves ambiguity
Coefficient uncertainty	Unstable	Sensitive	Better	Best	Bayesian + ML helps
Near-critical region	High error	Fails	Better	Best	Stability features help

8. Limitations and Extensions for Robustness

8.1 Current Limitations

Validation is limited to synthetic data where the generative noise model (Gaussian) matches the inference model. Future work should validate against real sensor hardware with manufacturing variability and data containing sparse outliers or missing measurements.

8.2 Extension to Outlier-Robust Inversion

To address bursty noise and sparse outliers, the Bayesian likelihood can be modified to incorporate heavy-tailed distributions:

Student-t likelihood: $\varepsilon \sim t_\nu(0, \sigma^2)$.

Contaminated normal:

$$p(\varepsilon) = (1-\pi_1)\,\mathcal{N}(0,\sigma^2) + \pi_1\,\mathcal{N}(0,\kappa\sigma^2), \quad \kappa \gg 1,$$

where π_1 is the outlier probability. Recent frameworks in fuzzy paranormed spaces [23, 24] provide density-based diagnostics that can filter sparse errors more effectively than MSE-based methods. Additional robustness requires Huber loss in place of MSE, adaptive regularisation, and imputation strategies for missing data.

8.3 Extension to Lacunary Measurements

Data-transmission losses and intermittent sensor failures result in incomplete datasets. Ideal-convergence theories [25] have shown that stability can be maintained even under missing-data scenarios, a critical consideration for IoT sensor networks with unreliable communication channels.

9. Conclusion

This paper proposes the Lagrange-Guided Bayesian Machine Learning Inversion (LGBMLI) framework for robust cubic inverse sensor calibration under multi-root ambiguity and coefficient uncertainty. The inverse problem is formulated as Bayesian posterior inference, where cubic roots correspond to posterior modes rather than exact algebraic solutions. Rigorous theoretical foundations are established via root-region classification and interval-wise monotonicity using the Intermediate Value Theorem. To address numerical instability, Cardano's formula is replaced by a Lagrange-based analytical solver that additionally yields informative structural features. Identifiability conditions (rank X=4) and a local sensitivity-based variance relation are derived. The framework achieves O(1) root generation and efficient inference complexity, with deployment decoupled from offline sampling.

On datasets with 80–85% multiple root cases, LGBMLI achieves 94.6% $\pm$ 1.8% root selection accuracy — a 27 percentage-point improvement over Cardano — with approximately 34% lower MSE reconstruction error (25.45% relative to Newton–Raphson; the corrected figure replaces the previously reported 28%). The 94.2% empirical coverage of nominal 95% credible intervals is slightly below the 95% nominal target (miscoverage $\approx$ 5.8%), attributable to prior–likelihood approximations; Datasets 1–3 bracket the nominal level more tightly (93.8%–95.1%). The experimental validation on the Barinov et al. [26] cubic sensor benchmark (Section 7) independently corroborates that all 101 test points present three distinct real roots ($D < 0$ throughout) and that physical domain enforcement is indispensable. Crucially, since the Barinov sensor is strictly monotone on $[0,1]$, this benchmark is a correctness and sanity check for the Lagrange root generator, not a proof of ML superiority; ML advantage is established in the non-monotone, coefficient-uncertain, and near-critical benchmark variants of Section 7.6.

References

[1] He, J., Yuan, L., Lei, H., Wang, K., Weng, Y., & Gao, H. (2024). A novel piecewise cubic Hermite interpolating polynomial-enhanced convolutional gated recurrent method

under multiple sensor feature fusion for tool wear prediction. *Sensors*, 24(4), 1129. https://doi.org/10.3390/s24041129

[2] Surantha, N., & Sutisna, N. (2025). Key considerations for real-time object recognition on edge computing devices. *Applied Sciences*, 15(13), 7533. https://doi.org/10.3390/app15137533

[3] Cardano, G. (1993). *Ars Magna or the Rules of Algebra* (T. R. Witmer, Trans.). Dover Publications.

[4] Chavez-Pichardo, M., Martinez-Cruz, M. A., Trejo-Martinez, A., Vega-Cruz, A. B., & Arenas-Resendiz, T. (2023). On the practicality of analytical solutions for all third- and fourth-degree algebraic equations with real coefficients. *Mathematics*, 11(6), 1447. https://doi.org/10.3390/math11061447

[5] Lakshmikantham, V., & Sen, S. K. (2011). *Computational Error and Complexity in Science and Engineering* (Vol. 201, 1st ed.). Elsevier.

[6] Sharmin, S., Zahid, A. H., Bhattacharjee, S., Igwilo, C., Kim, M., & Le, W. (2025). Automatically detecting numerical instability in machine learning applications via soft assertions. In *Proc. FSE '25* (pp. 2806–2827). ACM. https://doi.org/10.1145/3729394

[7] Inderjeet, & Bhardwaj, R. (2025). A new iterative Newton–Raphson technique for the numerical simulation of nonlinear equations. *Journal of Integrated Science and Technology*, 13(4), 1080. https://doi.org/10.62110/sciencein.jist.2025.v13.1080

[8] Usman, M., Iqbal, J., Khan, A., Ullah, I., Khan, H., Alzabut, J., & Alkhawar, H. M. (2025). A new iterative multi-step method for solving nonlinear equations. *MethodsX*, 15, 103394. https://doi.org/10.1016/j.mex.2025.103394

[9] Casella, F., & Bachmann, B. (2021). On the choice of initial guesses for the Newton–Raphson algorithm. *Applied Mathematics and Computation*, 398, 125991. https://doi.org/10.1016/j.amc.2021.125991

[10] Deiters, U. K., & Macias-Salinas, R. (2014). Calculation of densities from cubic equation of state: Revisited. *Industrial & Engineering Chemistry Research*, 53, 2529–2536.

[11] Wu, L., & Williamson, S. A. (2024). Posterior uncertainty quantification in neural networks using data augmentation. In *Proc. AISTATS* (Vol. 238, pp. 3376–3384). https://doi.org/10.48550/arXiv.2403.12729

[12] Stanley, M., Kuusela, M., Byrne, B., & Liu, J. (2024). Posterior uncertainty estimation via a Monte Carlo procedure for 4D-Var data assimilation. *Atmospheric Chemistry and Physics*, 24, 9419–9433. https://doi.org/10.5194/acp-24-9419-2024

[13] Burman, E., Hansbo, P., & Larson, M. G. (2018). Solving ill-posed control problems by stabilized finite element methods. *Inverse Problems*, 34(3), 035004. https://doi.org/10.1088/1361-6420/aaa32b

[14] Speagle, J. S. (2019). A conceptual introduction to Markov Chain Monte Carlo methods. arXiv preprint arXiv:1909.12313.

[15] Brooks, S. P., & Gelman, A. (1998). General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, 7(4), 434–455. https://doi.org/10.1080/10618600.1998.10474787

[16] Bilionis, I., & Zabaras, N. (2014). Solution of inverse problems with limited forward solver evaluations: A Bayesian perspective. *Inverse Problems*, 30, 015004. https://doi.org/10.1088/0266-5611/30/1/015004

[17] Mohammed-Djafar, A. (2015). Bayesian approach with prior models which enforce sparsity in signal and image processing. *EURASIP Journal on Advances in Signal Processing*, 2015, 52. https://doi.org/10.1186/s13634-015-0235-4

[18] Altman, N., & Krzywinski, M. (2017). Ensemble methods: Bagging and random forests. *Nature Methods*, 14(10), 933–934. https://doi.org/10.1038/nmeth.4438

[19] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

[20] Zhu, X., Yang, F., & Zhang, J. (2024). A comprehensive review of gradient boosting decision trees: GBDT, XGBoost, LightGBM, and CatBoost. *Expert Systems with Applications*, 238, 122208. https://doi.org/10.1016/j.eswa.2023.122208

[21] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proc. KDD '16* (pp. 785–794). ACM. https://doi.org/10.1145/2939672.2939785

[22] Ganti, S., & Gopinathan, S. (2020). A note on the solutions of cubic equations of state in low temperature region. *Journal of Molecular Liquids*, 315, 113808. https://doi.org/10.1016/j.molliq.2020.113808

[23] Türkmen, M. R. (2025). Outlier-robust convergence of integer- and fractional-order difference operators in fuzzy-paranormed spaces: Diagnostics and engineering applications. *Fractal and Fractional*, 9(10), 667. https://doi.org/10.3390/fractalfract9100667

[24] Öğünmez, H., & Türkmen, M. R. (2025). Statistical convergence for Grünwald–Letnikov fractional differences: Stability, approximation, and diagnostics in fuzzy normed spaces. *Axioms*, 14(10), 725. https://doi.org/10.3390/axioms14100725

[25] Türkmen, M. R., & Öğünmez, H. (2025). Ideal ($\mathcal{I}_2$) convergence in fuzzy paranormed spaces for practical stability of discrete-time fuzzy control systems under lacunary measurements. *Axioms*, 14(9), 663. https://doi.org/10.3390/axioms14090663

[26] Barinov, I. N., Tikhonenkov, V. A., Volkov, V. S., & Kuchumov, E. V. (2016). Inverse problem of approximation for a polynomial cubic transformation function for a sensor. *Measurement Techniques*, 59(2), 127–132. https://doi.org/10.1007/s11018-016-0929-x