Bayesian Calibration & Inverse Prediction

Mathematical Derivation and Assumptions

McClelland Lab, University College London

February 24, 2026

Contents

1 Introduction

In analytical chemistry and bioassay work a common workflow is to prepare standards at known concentrations, measure the instrument response for each, and fit a calibration curve. The practical goal, however, is the *reverse*: given a new instrument reading, estimate the unknown concentration that produced it.

This is the **inverse prediction** problem. Classical approaches (Fieller's theorem, Wald intervals) provide approximate confidence intervals but rely on asymptotic normality, struggle

with nonlinear models, and do not fully propagate parameter uncertainty. A Bayesian treatment resolves these limitations naturally: we obtain the full joint posterior of the model parameters, then push every source of uncertainty—parameter estimation *and* measurement noise—through the inverse to get a complete distribution over the unknown concentration.

This document derives the method implemented in the accompanying Streamlit application.

2 Notation

Symbol	Meaning
n	Number of calibration points
x_i	Known standard value (e.g. concentration) for point i
y_i	Measured instrument response for point i
$\boldsymbol{x}^{\text{cal}} = (x_1, \ldots, x_n)^\top$	Calibration x-values
$\boldsymbol{y}^{\text{cal}} = (y_1, \ldots, y_n)^\top$	Calibration y-values
$f(x;\, \boldsymbol{\theta})$	User-specified calibration function
$\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)^\top$	Model parameters
σ	Noise standard deviation
y^*	New observed response
x^*	Unknown value to be estimated

3 The Calibration Model

We begin with three assumptions that define the generative model for the calibration data.

Assumption 1 (Calibration function)**.** The relationship between the known value x and the instrument response y is described by a parametric function $f : \mathbb{R} \times \mathbb{R}^p \to \mathbb{R}$, specified by the user. For example, $f(x;\, a, b) = a + bx$, or $f(x;\, a, b) = a\, e^{bx}$. We require f to be continuous and differentiable almost everywhere with respect to both x and $\boldsymbol{\theta}$.

Assumption 2 (Gaussian noise)**.** Each observation is the true calibration value plus independent, identically distributed Gaussian noise:

$$y_i \;=\; f(x_i;\, \boldsymbol{\theta}) \;+\; \varepsilon_i, \qquad \varepsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0,\, \sigma^2), \qquad i = 1, \ldots, n. \tag{1}$$

The noise terms are mutually independent and independent of the standard values x_i.

Assumption 3 (Differentiability)**.** The log-posterior is differentiable with respect to all continuous parameters. This is guaranteed by the SymPy-parsed equation and PyTensor's automatic differentiation, and is required by the gradient-based NUTS sampler (Section 6).

Under these assumptions the **likelihood** is

$$p(\boldsymbol{y}^{\text{cal}} \mid \boldsymbol{x}^{\text{cal}},\, \boldsymbol{\theta},\, \sigma) \;=\; \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\,\sigma}\, \exp\!\left[-\frac{\big(y_i - f(x_i;\, \boldsymbol{\theta})\big)^2}{2\,\sigma^2}\right], \tag{2}$$

or equivalently, writing $\boldsymbol{\mu} = \big(f(x_1;\boldsymbol{\theta}), \ldots, f(x_n;\boldsymbol{\theta})\big)^\top$,

$$\boldsymbol{y}^{\text{cal}} \mid \boldsymbol{\theta},\, \sigma \;\sim\; \mathcal{N}\big(\boldsymbol{\mu},\; \sigma^2\, \mathbf{I}_n\big). \tag{3}$$

4 Prior Distributions

We place weakly informative, independent priors on all parameters:

$$\theta_j \sim \mathcal{N}(0,\, 10^2), \qquad j = 1, \ldots, p, \tag{4}$$

$$\sigma \sim \mathcal{N}^{+}(10), \tag{5}$$

where $\mathcal{N}^{+}(\tau)$ denotes the half-normal distribution with scale τ, i.e. a $\mathcal{N}(0, \tau^2)$ truncated to the positive reals.

Rationale. The zero-centred normal priors on each θ_j are deliberately vague (standard deviation 10). The half-normal prior on σ enforces positivity while remaining uninformative over the plausible range of noise magnitudes. These defaults perform well across a broad class of calibration problems. Users with strong domain knowledge may substitute tighter priors in the source code.

Joint prior. Because the priors are independent,

$$p(\boldsymbol{\theta},\, \sigma) \;=\; \left[\prod_{j=1}^{p} p(\theta_j)\right] p(\sigma). \tag{6}$$

5 Posterior Distribution

Applying Bayes' theorem:

$$\boxed{p(\boldsymbol{\theta},\, \sigma \mid \boldsymbol{y}^{\text{cal}},\, \boldsymbol{x}^{\text{cal}}) \;=\; \frac{p(\boldsymbol{y}^{\text{cal}} \mid \boldsymbol{x}^{\text{cal}},\, \boldsymbol{\theta},\, \sigma)\;\; p(\boldsymbol{\theta},\, \sigma)}{p(\boldsymbol{y}^{\text{cal}} \mid \boldsymbol{x}^{\text{cal}})}} \tag{7}$$

where the marginal likelihood (evidence) is

$$p(\boldsymbol{y}^{\text{cal}} \mid \boldsymbol{x}^{\text{cal}}) \;=\; \int p(\boldsymbol{y}^{\text{cal}} \mid \boldsymbol{x}^{\text{cal}},\, \boldsymbol{\theta},\, \sigma)\, p(\boldsymbol{\theta},\, \sigma)\, \mathrm{d}\boldsymbol{\theta}\, \mathrm{d}\sigma. \tag{8}$$

For most nonlinear calibration functions this integral is analytically intractable, so we approximate the posterior using Markov chain Monte Carlo sampling.

6 MCMC Sampling via NUTS

6.1 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) [Neal, 2011] augments the parameter space with auxiliary momentum variables $\boldsymbol{r} \in \mathbb{R}^d$ (where $d = p + 1$) and defines the joint density

$$p(\boldsymbol{\theta},\, \sigma,\, \boldsymbol{r}) \;\propto\; \exp\!\big[\, -U(\boldsymbol{\theta}, \sigma) - \tfrac{1}{2}\, \boldsymbol{r}^{\top} \mathbf{M}^{-1} \boldsymbol{r} \,\big], \tag{9}$$

with *potential energy*

$$U(\boldsymbol{\theta}, \sigma) \;=\; -\log p(\boldsymbol{y}^{\text{cal}} \mid \boldsymbol{x}^{\text{cal}},\, \boldsymbol{\theta},\, \sigma) \;-\; \log p(\boldsymbol{\theta},\, \sigma) \tag{10}$$

and mass matrix $\mathbf{M}$ (adapted during warm-up to approximate the posterior covariance).

HMC simulates Hamiltonian dynamics using the leapfrog integrator with step size ϵ:

$$\boldsymbol{r}_{t+\epsilon/2} = \boldsymbol{r}_t - \tfrac{\epsilon}{2}\, \nabla_{\boldsymbol{\theta}}\, U(\boldsymbol{\theta}_t), \tag{11}$$

$$\boldsymbol{\theta}_{t+\epsilon} = \boldsymbol{\theta}_t + \epsilon\, \mathbf{M}^{-1}\, \boldsymbol{r}_{t+\epsilon/2}, \tag{12}$$

$$\boldsymbol{r}_{t+\epsilon} = \boldsymbol{r}_{t+\epsilon/2} - \tfrac{\epsilon}{2}\, \nabla_{\boldsymbol{\theta}}\, U(\boldsymbol{\theta}_{t+\epsilon}). \tag{13}$$

After L leapfrog steps the proposal is accepted with probability $\min\big(1,\, \exp(-\Delta H)\big)$, where ΔH is the change in the Hamiltonian.

6.2 The No-U-Turn Sampler

NUTS [Hoffman and Gelman, 2014] removes the need to choose L by building a balanced binary tree of leapfrog steps. The tree doubles in size until a "U-turn" is detected:

$$\boldsymbol{r} \cdot (\boldsymbol{\theta}^{+} - \boldsymbol{\theta}^{-}) < 0 \qquad \text{or} \qquad \boldsymbol{r} \cdot (\boldsymbol{\theta}^{-} - \boldsymbol{\theta}^{+}) < 0, \tag{14}$$

where $\boldsymbol{\theta}^{+}$ and $\boldsymbol{\theta}^{-}$ are the trajectory endpoints. A multinomial scheme selects the next state from the trajectory, weighted by the unnormalised density. The step size ϵ is tuned during warm-up via dual averaging [Nesterov, 2009] to target an acceptance rate of ~ 0.8.

6.3 Warm-up, Sampling, and Diagnostics

1. **Warm-up.** The step size and mass matrix are adapted; these draws are discarded.
2. **Sampling.** S draws are collected from each of C independent chains, giving $N = S \times C$ posterior samples $\{(\boldsymbol{\theta}^{(s)}, \sigma^{(s)})\}_{s=1}^{N}$.
3. **Convergence checks.** The application reports:
   - $\hat{R}$ (Gelman–Rubin statistic): values $\lesssim 1.01$ indicate convergence [Gelman et al., 2013].
   - Effective sample size (n_{eff}): the number of effectively independent draws after accounting for autocorrelation.
   - Monte Carlo standard error (MCSE): the precision of the posterior mean estimate.

7 Inverse Prediction

7.1 Problem Statement

Given a new instrument reading y^*, we want the posterior predictive distribution of the unknown value x^* that produced it. We assume the new observation arises from the same process as the calibration data:

$$y^* \;=\; f(x^*;\, \boldsymbol{\theta}) \;+\; \varepsilon^*, \qquad \varepsilon^* \sim \mathcal{N}(0,\, \sigma^2). \tag{15}$$

7.2 Posterior Predictive Distribution of x^*

The target distribution is obtained by marginalising over the posterior:

$$\boxed{p(x^* \mid y^*,\, \boldsymbol{y}^{\text{cal}},\, \boldsymbol{x}^{\text{cal}}) \;=\; \int p(x^* \mid y^*,\, \boldsymbol{\theta},\, \sigma)\;\, p(\boldsymbol{\theta},\, \sigma \mid \boldsymbol{y}^{\text{cal}},\, \boldsymbol{x}^{\text{cal}})\, \mathrm{d}\boldsymbol{\theta}\, \mathrm{d}\sigma} \tag{16}$$

This integral propagates *both* parameter uncertainty and measurement noise into the prediction. We evaluate it by Monte Carlo:

Proposition 1 (Monte Carlo inverse prediction)**.** *For each posterior draw* $(\boldsymbol{\theta}^{(s)}, \sigma^{(s)})$, $s = 1, \ldots, N$*:*

1. ***Add noise:*** $\tilde{y}^{(s)} = y^* + \epsilon^{(s)}$ *where* $\epsilon^{(s)} \sim \mathcal{N}(0,\, \sigma^{(s)2})$.
2. ***Invert:*** $x^{*(s)} = f^{-1}\big(\tilde{y}^{(s)};\, \boldsymbol{\theta}^{(s)}\big)$.

The resulting collection $\{x^{*(s)}\}_{s=1}^{N}$ *is a sample from* $p(x^* \mid y^*,\, \boldsymbol{y}^{\text{cal}},\, \boldsymbol{x}^{\text{cal}})$.

Proof. For fixed $(\boldsymbol{\theta}, \sigma)$ the forward model is deterministic, so $x^* = f^{-1}(y^* - \varepsilon^*;\, \boldsymbol{\theta})$ with $\varepsilon^* \sim \mathcal{N}(0, \sigma^2)$. Drawing $\tilde{y}^{(s)} = y^* + \epsilon^{(s)}$ where $\epsilon^{(s)} \sim \mathcal{N}(0, \sigma^{(s)2})$ is equivalent to sampling the noise-free response from the predictive distribution at draw s. Since the draws $(\boldsymbol{\theta}^{(s)}, \sigma^{(s)})$ come from the posterior $p(\boldsymbol{\theta}, \sigma \mid \boldsymbol{y}^{\text{cal}}, \boldsymbol{x}^{\text{cal}})$, the composition produces samples from the marginal (16). □

7.3 Inversion Methods

Symbolic inverse. When f is algebraically invertible, SymPy computes $x = f^{-1}(y;\, \boldsymbol{\theta})$ in closed form. If multiple real solutions exist (e.g. for a quadratic), the one closest to the centroid of the calibration x-values is selected.

Numerical inverse. When no closed form exists, Brent's root-finding method [Brent, 1973] solves

$$f(x;\,\boldsymbol{\theta}^{(s)}) - \tilde{y}^{(s)} = 0 \tag{17}$$

over the interval $\left[x_{\min} - 3\,\Delta x,\;\; x_{\max} + 3\,\Delta x\right]$ where $\Delta x = x_{\max} - x_{\min}$. Brent's method combines bisection, secant, and inverse quadratic interpolation, guaranteeing convergence whenever a sign change exists.

7.4 Credible Intervals and Point Estimates

From the N draws $\{x^{*(s)}\}$ (after discarding any non-finite values from failed inversions), the $100(1-\alpha)\%$ equal-tailed credible interval is

$$\mathrm{CI}_{1-\alpha} \;=\; \left[\,Q_{\alpha/2},\;\; Q_{1-\alpha/2}\,\right], \tag{18}$$

where Q_q is the q-th sample quantile. The application also reports:

$$\hat{x}_{\text{median}} = Q_{0.5}, \tag{19}$$

$$\hat{x}_{\text{mean}} = \frac{1}{N}\sum_{s=1}^{N} x^{*(s)}, \tag{20}$$

$$\hat{\sigma}_x = \sqrt{\frac{1}{N-1}\sum_{s=1}^{N}\left(x^{*(s)} - \hat{x}_{\text{mean}}\right)^2}. \tag{21}$$

8 Residual Diagnostics

After fitting the model, the application runs automated diagnostics to help the user assess whether the chosen equation and the noise assumptions are adequate.

8.1 Residuals

Residuals are evaluated at the posterior median parameters $\hat{\boldsymbol{\theta}} = \text{median}\{\boldsymbol{\theta}^{(s)}\}$:

$$e_i \;=\; y_i \;-\; f(x_i;\,\hat{\boldsymbol{\theta}}), \qquad i = 1, \ldots, n. \tag{22}$$

Two plots are shown: residuals versus fitted values $\hat{y}_i$ and residuals versus x_i.

8.2 Breusch–Pagan Test

The Breusch–Pagan test [Breusch and Pagan, 1979] checks whether the variance of the residuals depends on the fitted values:

$$H_0:\; \mathrm{Var}(\varepsilon_i) = \sigma^2 \;\; \forall\, i \qquad \text{vs.} \qquad H_1:\; \mathrm{Var}(\varepsilon_i) = h(\boldsymbol{z}_i^{\top}\boldsymbol{\gamma}). \tag{23}$$

The procedure regresses the squared residuals e_i^2 on the fitted values by OLS and computes the LM statistic $\mathrm{LM} = \frac{1}{2}\,\mathrm{ESS}$, which follows a χ^2_1 distribution under H_0. A p-value below 0.05 indicates that the noise spread changes with signal level. In that case, the application advises applying a variance-stabilising transformation (e.g. $\log y$ or $\sqrt{y}$) before re-fitting.

8.3 Wald–Wolfowitz Runs Test

The runs test [Wald and Wolfowitz, 1940] assesses whether the sign pattern of the residuals $(+/-)$ is random. Let n_+ and n_- be the counts of positive and negative residuals, and R the number of runs. Under H_0 (random ordering):

$$\mathbb{E}[R] = 1 + \frac{2\,n_+\,n_-}{n_+ + n_-}\,, \tag{24}$$

$$\mathrm{Var}(R) = \frac{2\,n_+\,n_-\,(2\,n_+\,n_- - n_+ - n_-)}{(n_+ + n_-)^2\,(n_+ + n_- - 1)}\,. \tag{25}$$

The standardised statistic $Z = (R - \mathbb{E}[R])/\sqrt{\mathrm{Var}(R)}$ is approximately standard normal. A p-value below 0.05 suggests systematic structure that the model does not capture; the user should consider a different functional form.

8.4 Practical Guidance

Based on the diagnostics the application presents the following advice:

- **Curved / trending residuals:** try a different model (e.g. add a polynomial term, switch to an exponential or power law).
- **Fan-shaped spread:** try a variance-stabilising transformation such as $\log(y)$ or $\sqrt{y}$ before fitting.
- **Isolated outliers:** check for data-entry errors; consider excluding the suspect point and re-fitting.
- **Random scatter around zero:** the model assumptions appear satisfied.

9 Summary of Assumptions

1. The user-specified function $f(x;\boldsymbol{\theta})$ adequately describes the calibration relationship (Assumption 1).
2. Observation noise is additive, Gaussian, and i.i.d. with constant variance σ^2 (Assumption 2).
3. The log-posterior is differentiable with respect to all continuous parameters (Assumption 3).
4. Prior distributions are weakly informative and mutually independent (Section 4).
5. New measurements follow the same generative process as the calibration data (no distribution shift).

When assumptions may be violated.

- *Wrong functional form.* The runs test will flag non-random residual patterns; try a different equation.
- *Non-constant variance.* The Breusch–Pagan test will flag this; apply a variance-stabilising transform (e.g. $\log y$) and re-fit.
- *Non-Gaussian noise.* Heavy tails may cause credible intervals to undercover. A Student-t likelihood could be added in future.
- *Correlated observations.* If measurements are time-dependent, the independence assumption is violated and intervals will be too narrow.

10 Implementation Notes

- **Equation parsing:** SymPy with implicit multiplication and `convert_xor` transformations.
- **Automatic differentiation:** PyTensor provides exact gradients for the NUTS sampler.
- **Sampling:** PyMC ≥ 5.10 with the default NUTS implementation.

- **Diagnostics:** ArviZ for convergence statistics; statsmodels for the Breusch–Pagan test; the Wald–Wolfowitz test is implemented directly using SciPy.
- **Numerical inversion:** SciPy's `brentq` with tolerance 10^{-10}.

References

T. S. Breusch and A. R. Pagan. A simple test for heteroscedasticity and random coefficient variation. *Econometrica*, 47(5):1287–1294, 1979.

R. P. Brent. *Algorithms for Minimization without Derivatives*. Prentice-Hall, 1973.

A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC, 3rd edition, 2013.

M. D. Hoffman and A. Gelman. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15:1593–1623, 2014.

R. M. Neal. MCMC using Hamiltonian dynamics. In S. Brooks, A. Gelman, G. L. Jones, and X.-L. Meng, editors, *Handbook of Markov Chain Monte Carlo*, chapter 5. Chapman & Hall/CRC, 2011.

Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120(1):221–259, 2009.

J. Salvatier, T. V. Wiecki, and C. Fonnesbeck. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, 2:e55, 2016.

A. Wald and J. Wolfowitz. On a test whether two samples are from the same population. *The Annals of Mathematical Statistics*, 11(2):147–162, 1940.