Learning with Embedded Linear Equality Constraints via Variational Bayesian Inference

Matthew Marsh **Benoît Chachuat** **Antonio del Rio Chanona**
Sargent Centre for Process Systems Engineering, Imperial College London

Abstract

Machine Learning is becoming more prevalent in science and engineering, but many approaches do not provide meaningful uncertainty estimates and predictions may also violate known physical knowledge. We propose a Bayesian framework to embed linear relationships across inputs and outputs into the learning process, whilst characterizing full predictive uncertainty over both the model parameters and the domain knowledge. We evaluated our method on learning the single particle battery model subject to voltage and energy balances, showing its ability to provide reduced credible intervals and constraint violations compared to standard Bayesian neural networks based on variational inference.

1 Introduction

In many scientific and engineering domains, predictive models must satisfy known physical constraints. These constraints are often linear equalities arising from mass and energy balances. However, modern neural networks trained purely on data, of limited quantity or quality, with no consideration of constraints may frequently violate such relations, leading to physically inconsistent or infeasible predictions.

Bayesian neural networks (BNNs) provide a principled framework for uncertainty quantification by placing distributions over model parameters [Neal, 1996, Blundell et al., 2015]. Yet, standard BNN formulations do not explicitly enforce known constraints in the predictive distribution. While hard projection methods and penalty-based approaches exist [Raissi et al., 2019, Amos and Kolter, 2017, Donti et al., 2021], they typically treat constraints deterministically and do not account for overall uncertainty in constraint satisfaction.

To overcome these challenges, we propose a probabilistic framework for embedding linear equality constraints directly into BNNs, tractably evaluated using variational inference (VI). We show that when the predictive distribution is Gaussian and the constraints are linear, constraint enforcement reduces to closed-form Gaussian conditioning. This yields a posterior predictive distribution that satisfies constraints up to a defined tolerance level while preserving calibrated uncertainty.

Furthermore, we treat the constraint tolerance as a random variable and perform VI jointly over the network parameters and tolerance level. Alongside the constraint embedding, this allows the model to learn how strictly the constraints should be enforced from data, rather than assuming they hold exactly. The resulting framework integrates structured knowledge, uncertainty quantification, and tractable optimisation into a single differentiable training objective, providing a principled bridge between structured probabilistic modelling and modern deep learning.

2 Related Work

Neural networks are powerful approximators, however standard deterministic training yields only point predictions with no principled measure of confidence, and offers no mechanism to enforce known physical relationships. In science and engineering applications, both are important limitation: uncalibrated predictions may be overconfident, and physically inconsistent outputs undermine trust and decision-making. Bayesian inference and constraint enforcement address these shortcomings independently, but existing approaches rarely combine them. Whilst Bayesian methods quantify uncertainty without enforcing structure, constraint methods enforce structure without propagating uncertainty.

2.1 Bayesian Neural Networks

Bayesian inference provides a natural framework for uncertainty quantification (UQ) by treating model parameters as random variables whose posterior distribution reflects the evidence provided by data. Exact inference is intractable for deep networks, motivating scalable approximations such as VI [Graves, 2011, Blundell et al., 2015] and Monte Carlo dropout [Gal and Ghahramani, 2016]. These methods provide predictive uncertainty estimates, though without considering constraint satisfaction.

2.2 Constraining Neural Networks

Physics-informed neural networks (PINNs) incorporate physical knowledge by penalising physical constraint residuals during training [Raissi et al., 2019], and are amongst the most widely adopted example of *soft* constraint enforcement. However, this approach does not guarantee that the constraints are enforced everywhere, neither during training nor inference.

Hard constraints, on the other hand, augment model architectures to enforce constraints on all model outputs. Some approaches utilise projections onto feasible outputs [Chen et al., 2024]. OptNet [Amos and Kolter, 2017] uses quadratic programs as differentiable layers, expanded in subsequent work to general differentiable convex optimisation layers [Agrawal et al., 2019]. Subspace-based methods can also be used to reconstruct outputs consistent with constraints [Donti et al., 2021].

A common limitation shared by both soft and hard approaches is that constraint satisfaction is often treated deterministically, thereby omitting uncertainty propagation. Our work addresses this gap by incorporating constraints probabilistically via Gaussian conditioning [Hansen et al., 2023], treating constraints as noisy linear observations, with joint inference yielding a framework that enforces physical structure while preserving uncertainty estimates.

2.3 Post-Bayesian Inference and Structured Conditioning

In classical Gaussian models, conditioning under linear transformations admits closed-form solutions [Murphy, 2023]. Our approach integrates linear constraint conditioning directly into the variational objective, with the posterior learned jointly with a probabilistic tolerance over the constraints. The method can be interpreted as a post-Bayesian extension: rather than performing Bayesian inference over unconstrained function classes, uncertainty is conditioned and reshaped by known linear relationships, embedding inductive bias directly into the model.

3 Methodology

We consider supervised learning over a dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$, with $\mathbf{x} \in \mathbb{R}^{n_x}$, $\mathbf{y} \in \mathbb{R}^{n_y}$. The neural network outputs a diagonal Gaussian, with mean $\boldsymbol{\mu}_P \in \mathbb{R}^{n_y}$ and variance $\boldsymbol{\sigma}_P^2 \in \mathbb{R}^{n_y}$:

$$p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{\mu}_P(\mathbf{x}; \boldsymbol{\theta}), \operatorname{diag}(\boldsymbol{\sigma}_P^2(\mathbf{x}; \boldsymbol{\theta}))\right), \quad (1)$$

with the neural network parameters denoted by $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$.

3.1 Embedding Constraints with Linear-Gaussian System

We assume m known linear equality relationships between input ($\mathbf{x}$) and predicted output ($\mathbf{y}$) variables. As our inputs are treated as fixed, we introduce a small tolerance term ($\boldsymbol{\varepsilon}$) to account for measurement noise.

The constraint residual is given by:

$$\mathbf{z} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} - \mathbf{b} + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \operatorname{diag}(\mathbf{r})), \quad (2)$$

where $\mathbf{A} \in \mathbb{R}^{m \times n_x}, \mathbf{B} \in \mathbb{R}^{m \times n_y}, \mathbf{b} \in \mathbb{R}^m$, with $\mathbf{B}$ of full row rank, and $\mathbf{r} \in \mathbb{R}^m$ is the diagonal variance parameterizing the tolerance level.

Conditioning on the residual of the constraint, $\mathbf{z} = \mathbf{0}$ yields a Gaussian posterior predictive:

$$p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}, \mathbf{z} = \mathbf{0}) = \mathcal{N}(\boldsymbol{\mu}_C, \operatorname{diag}(\boldsymbol{\sigma}_C^2)), \quad (3)$$

with closed-form updated parameters given by:

$$\mathbf{S} = \mathbf{B} \operatorname{diag}(\boldsymbol{\sigma}_P^2) \mathbf{B}^\top + \boldsymbol{\varepsilon} \quad (4)$$

$$\boldsymbol{\mu}_C = \boldsymbol{\mu}_P + \left(\boldsymbol{\sigma}_P^2 \odot \mathbf{B}^\top\right) \mathbf{S}^{-1} \left(\mathbf{b} - \mathbf{A}\mathbf{x} - \mathbf{B}\boldsymbol{\mu}_P\right) \quad (5)$$

$$\boldsymbol{\sigma}_C^2 = \operatorname{diag}(\boldsymbol{\sigma}_P^2) - \left(\boldsymbol{\sigma}_P^2 \odot \mathbf{B}^\top\right) \mathbf{S}^{-1} \left(\mathbf{B} \odot (\boldsymbol{\sigma}_P^2)^\top\right). \quad (6)$$

This conditioning layer is fully differentiable and can be inserted into the network as a probabilistic projection operator. Importantly, although this update step induces cross correlation within the updated variance, we only consider the diagonal elements here to reduce number of learnable parameters.

The method recovers exact hard constraint enforcement as $\mathbf{r} \to 0$, while reverting to unconstrained predictions when $\mathbf{r}$ is large. Thus, $\mathbf{r}$ controls the strength of constraint enforcement and it becomes a learnable quantity within the Bayesian framework.

3.2 Bayesian Inference over Constrained Neural Networks

Joint Variational Inference. Conventional BNNs perform inference over the sole model parameters, whereas we perform joint inference over both the parameters $\boldsymbol{\theta}$ and constraint tolerance $\mathbf{r}$ here. We aim to find a tractable mean-field variational approximation, $q(\boldsymbol{\theta}, \mathbf{r})$, to the true posterior, $p(\boldsymbol{\theta}, \mathbf{r} \mid \mathcal{D})$. We approximate both the priors and variational posterior jointly

over the network parameters and constraint tolerances as independent:

$$p(\boldsymbol{\theta}, \mathbf{r}) = p(\boldsymbol{\theta})p(\mathbf{r}), \qquad q(\boldsymbol{\theta}, \mathbf{r}) = q(\boldsymbol{\theta})q(\mathbf{r}),$$

noting that $\mathbf{r}$ must be positive as the prior variance on our constraint tolerance.

Modified ELBO. The predictive likelihood incorporates the conditioned Gaussian distribution in Eqn. (3). The variational objective is derived by minimizing the KL divergence between the variational and true posterior, yielding the negative ELBO:

$$\begin{aligned} q^*(\boldsymbol{\theta}, \mathbf{r}) &\in \arg\min_{q(\boldsymbol{\theta},\mathbf{r})} D_{\mathrm{KL}}(q(\boldsymbol{\theta}, \mathbf{r}) \,\|\, p(\boldsymbol{\theta}, \mathbf{r} \mid \mathcal{D})) \\ &= \mathbb{E}_{q(\boldsymbol{\theta},\mathbf{r})}\left[\log \frac{q(\boldsymbol{\theta}, \mathbf{r})}{p(\mathcal{D} \mid \boldsymbol{\theta}, \mathbf{r})p(\boldsymbol{\theta}, \mathbf{r})}\right] \\ &= \mathbb{E}_{q(\boldsymbol{\theta})q(\mathbf{r})}[-\log p(\mathcal{D} \mid \boldsymbol{\theta}, \mathbf{r})] \\ &\quad + D_{\mathrm{KL}}(q(\mathbf{r}) \,\|\, p(\mathbf{r})) + D_{\mathrm{KL}}(q(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta}))\,. \end{aligned}$$

The resulting objective balances data-fit with the embedded constraints, alongside a regularization term on the parameters and constraint tolerance from the priors. The ELBO is minimized using the usual reparameterization trick [Kingma and Welling, 2022].

Interpretation. The framework may be interpreted as learning how much to trust prior knowledge. Rather than assuming perfect validity of constraints, the model learns the appropriate enforcement strength from the data. This is particularly important in real-world systems where constraints may hold only approximately due to measurement noise or partial observability. The resulting predictive distribution is both uncertainty-aware and constraint consistent in expectation.

3.3 Uncertainty Decomposition Under Constraint Conditioning

Constraint conditioning modifies the structure of predictive uncertainty. For a standard BNN, the law of total variance enables the following decomposition into aleatoric and epistemic terms:

$$\mathrm{Var}(\mathbf{y} \mid \mathbf{x}, \mathcal{D}) = \mathbb{E}_{q(\boldsymbol{\theta})}[\boldsymbol{\sigma}_P^2] + \mathrm{Var}_{q(\boldsymbol{\theta})}[\boldsymbol{\mu}_P], \tag{7}$$

Under constraint conditioning the predictive distribution becomes:

$$p(\mathbf{y} \mid \mathbf{x}, \mathcal{D}) = \mathbb{E}_{q(\boldsymbol{\theta},\mathbf{r})}[\mathcal{N}(\boldsymbol{\mu}_C, \mathrm{diag}(\boldsymbol{\sigma}_C^2))], \tag{8}$$

with $\boldsymbol{\sigma}_C^2 = \boldsymbol{\sigma}_P^2 - \mathrm{diag}(\mathbf{KSK}^\top)$ following from Eqn. (6).

Then, using the update rules in Eqns. (4)–(6) and applying the law of total variance over $q(\boldsymbol{\theta})q(\mathbf{r})$ gives:

$$\begin{aligned} \mathrm{Var}(\mathbf{y}) = &\underbrace{\mathbb{E}_{q(\boldsymbol{\theta})}[\boldsymbol{\sigma}_P^2]}_{\text{Aleatoric}} - \underbrace{\mathbb{E}_{q(\boldsymbol{\theta},\mathbf{r})}[\mathrm{diag}(\mathbf{KSK}^\top)]}_{\text{Constraint reduction}} \\ &+ \underbrace{\mathrm{Var}_{q(\boldsymbol{\theta})}[\boldsymbol{\mu}_P]}_{\text{Epistemic}} + \underbrace{\mathrm{Var}_{q(\mathbf{r})}[\boldsymbol{\mu}_C]}_{\text{Tolerance uncertainty}} \\ &+ \underbrace{\mathrm{Cov}_{q(\boldsymbol{\theta},\mathbf{r})}[\boldsymbol{\mu}_P,\, \boldsymbol{\mu}_C - \boldsymbol{\mu}_P]}_{\text{Constraint–epistemic interaction}}\,. \end{aligned} \tag{9}$$

Since each diagonal entry of $\mathbf{KSK}^\top$ is non-negative, constraint conditioning always reduces the marginal predictive variance of each output. As $\mathbf{r} \to 0$, the reduction is maximal (hard projection); and as $\mathbf{r} \to \infty$, it vanishes and the standard BNN decomposition is recovered.

4 Experiments

We evaluate our framework on learning the single particle model (SPM) of a lithium-ion battery, implemented via `PyBaMM` [Sulzer et al., 2021]. The SPM is governed by spherical-diffusion equations, coupled with lumped thermal dynamics. The learning task approximates input-output mapping of the SPM: *Given operating conditions, predict electrochemical and thermal outputs subject to known physical constraints and noisy data.* We benchmark this against a standard unconstrained BNN.

4.1 Problem Setup

Inputs and outputs. We aim to learn the interaction between 3 input variables and 8 output variables within the SPM model, as summarised in Table 3 in the Appendix. Data was generated by simulating full discharges across a grid of currents $C = \{0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}$ and temperatures $T = \{273, 283, 293, 298, 303, 313, 318\}$, yielding 500 state-of-charge points per combination. Of this, 60% was used for training, 20% for hyperparameter tuning and 20% as reserved for a test set. Gaussian noise was added to the outputs to reflect realistic measurement uncertainty (voltage quantities $\sigma \approx$ 2-5 mV; thermal quantities $\sigma \approx 30$–$50\,\mathrm{mW\,m^{-3}}$).

Constraints. The SPM satisfies two linear equality constraints across the outputs, which we embed within the framework. Writing the output vector as $\mathbf{y} \in \mathbb{R}^8$, these take the form $\mathbf{By} = \mathbf{0}$ with $\mathbf{B} \in \mathbb{R}^{2\times 8}$.

Kirchhoff's Voltage Law:

$$V = V_{\mathrm{OCV}} - \eta_+ - \eta_- - \Delta V_{\mathrm{IR}}. \tag{10}$$

Energy balance:

$$\dot{Q}_{\mathrm{tot}} = \dot{Q}_{\mathrm{rev}} + \dot{Q}_{\mathrm{irr}}. \tag{11}$$

Chosen Priors and Variational Posteriors. We placed a standard Gaussian prior over the network parameters:

$$p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \mathbf{I}_P),$$

where P denotes the number of parameters.

To ensure positivity of the constraint tolerance, we reparameterized $\mathbf{r}$ via a log-scale variable $\boldsymbol{\rho}$ such that

$$\mathbf{r} = \exp(\rho).$$

We also placed a Gaussian prior over $\boldsymbol{\rho}$:

$$p(\rho) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\rho}}, \boldsymbol{\Sigma}_{\boldsymbol{\rho}}),$$

with $\boldsymbol{\mu}_{\boldsymbol{\rho}} = [-2, -2]^\top$ and $\boldsymbol{\sigma}_{\boldsymbol{\rho}} = \mathrm{diag}([1, 1])$, reflecting a prior belief that the constraints hold approximately, with limited dispersion around the mean.

Both variational posteriors $q(\boldsymbol{\theta})$ and $q(\boldsymbol{\rho})$ were chosen as mean-field diagonal Gaussians.

4.2 Results

We compare all results on the test set. The standard BNN and the proposed Bayesian constrained probabilistic neural network (BCPNN) are comparable in terms of point predictive accuracy, and they achieve identical performance on coverage ratio (0.99), indicating that embedding constraints does not degrade regression performance.

When evaluating the predictive distributions (Table 1), the BCPNN also produces tighter coverage widths, reflecting the variance reduction predicted by the decomposition in Section 3.3. When sampling over the variational posteriors, we also see the BCPNN reduces both aleatoric uncertainty due to Gaussian conditioning on the constraint, and epistemic uncertainty, reflecting tighter posterior concentration and increased predictive confidence.

Table 1: Predictive uncertainty decomposition, in normalized space.

Model	Aleatoric	Epistemic
BNN	0.0149 ± 0.0096	0.0074 ± 0.0042
BCPNN	$\mathbf{0.0140 \pm 0.0097}$	$\mathbf{0.0068 \pm 0.0041}$

However, the most significant distinction between these models lies in constraint adherence. The BNN exhibits large violations on the test set, as it has no mechanism to enforce the voltage decomposition or heat balance beyond what is implicitly learned from data. The BCPNN reduces median violation by over two orders of magnitude, for the voltage constraint, and over four on the heat balance (Table B.4), demonstrating that probabilistic conditioning enforces clear physical consistency.

The learned tolerance posteriors (Table 2) further show that the framework is able to automatically distinguish between constraints: Kirchhoff's voltage law (Eqn. 10) is enforced to a higher precision, while the energy balance (Eqn. 11) retains appreciable slack with high posterior uncertainty. This result is expected due to differing measurement noise across variables, illustrating the framework's ability to calibrate constraint confidence directly from the data.

Table 2: Learned constraint tolerance posteriors.

	$\boldsymbol{\mu}_{\boldsymbol{\rho}}$	$\boldsymbol{\sigma}_{\boldsymbol{\rho}}$	$\mathbb{E}[\mathbf{r}]$	$\mathbf{Std}[\mathbf{r}]$
Eqn. (10)	-11.21	0.23	1.36×10^{-5}	3.19×10^{-6}
Eqn. (11)	-1.99	1.00	1.37×10^{-1}	1.36×10^{-1}

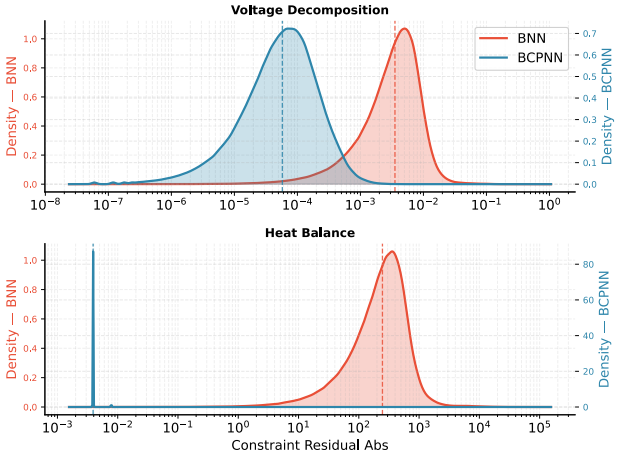

Figure 1: KDE over sampled posteriors of constraint violations

5 Conclusions

We presented a framework for incorporating linear equality constraints into BNNs via Gaussian conditioning. The resulting BCPNN preserves the predictive accuracy of a standard BNN while reducing constraint violations and producing much tighter and physically consistent uncertainty estimates. We also showed how the learned tolerance posteriors automatically can distinguish between constraints of different tolerances, providing interpretable diagnostics of constraint confidence without manual tuning.

The current formulation is applied to only linear equality constraints here. Extending the conditioning framework to handle inequality constraints and nonlinear relationships would broaden its applicability. Additionally, integrating the learned tolerance posteriors into active learning or experimental design pipelines could enable data collection strategies that are informed by both predictive, and constraint uncertainty could provide interesting avenues for further exploration.

Acknowledgements

Thanks to Dr. Laura Helleckes for her help reviewing the manuscript.

References

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter. Differentiable convex optimization layers. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, number 858, pages 9562–9574. Curran Associates Inc., Red Hook, NY, USA, December 2019.

Brandon Amos and J. Zico Kolter. OptNet: Differentiable Optimization as a Layer in Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 136–145, July 2017.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 1613–1622, Lille, France, 07–09 Jul 2015.

Hao Chen, Gonzalo E. Constante Flores, and Can Li. Physics-informed neural networks with hard linear equality constraints. *Computers & Chemical Engineering*, 189:108764, October 2024. doi: 10.1016/j.compchemeng.2024.108764.

Priya L. Donti, David Rolnick, and J. Zico Kolter. DC3: A learning method for optimization with hard constraints, April 2021. arXiv:2104.12225 [cs].

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1050–1059. PMLR, June 2016.

Alex Graves. Practical variational inference for neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, page 2348–2356, Red Hook, NY, USA, 2011. doi: 10.5555/2986459.2986721.

Derek Hansen, Danielle C. Maddix, Shima Alizadeh, Gaurav Gupta, and Michael W. Mahoney. Learning physical models that can respect conservation laws. In *Proceedings of the 40th International Conference on Machine Learning*, pages 12469–12510, July 2023.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes, December 2022. arXiv:1312.6114 [stat].

Kevin P. Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023.

Radford M. Neal. *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer, New York, NY, 1996. doi: 10.1007/978-1-4612-0745-0.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library, December 2019.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi: 10.1016/j.jcp.2018.10.045.

Valentin Sulzer, Scott G. Marquis, Robert Timms, Martin Robinson, and S. Jon Chapman. Python battery mathematical modelling (PyBaMM). *Journal of Open Research Software*, 9(1):14, 2021. doi: 10.5334/jors.309.

Learning with Embedded Linear Equality Constraints via Variational Bayesian Inference: Supplementary Materials

A Derivations of Results

A.1 Derivation of Modified ELBO

We seek a tractable approximation to the true posterior, as the minimizer of the KL-divergence:

$$q^*(\boldsymbol{\theta}, \mathbf{r}) \in \arg\min_{q(\boldsymbol{\theta},\mathbf{r})} D_{\mathrm{KL}}(q(\boldsymbol{\theta}, \mathbf{r}) \,\|\, p(\boldsymbol{\theta}, \mathbf{r} \mid \mathcal{D}))$$
$$= \int\int q(\boldsymbol{\theta}, \mathbf{r}) \log \frac{q(\boldsymbol{\theta}, \mathbf{r})}{p(\boldsymbol{\theta}, \mathbf{r} \mid \mathcal{D})}\, d\boldsymbol{\theta}\, d\mathbf{r},$$

using Bayes' rule $p(\boldsymbol{\theta}, \mathbf{r} \mid \mathcal{D}) \propto p(\mathcal{D} \mid \boldsymbol{\theta}, \mathbf{r})\, p(\boldsymbol{\theta}, \mathbf{r})$, where $p(\mathcal{D} \mid \boldsymbol{\theta}, \mathbf{r})$ is our augmented likelihood.

Substituting into the KL objective gives:

$$\int\int q(\boldsymbol{\theta}, \mathbf{r}) \log \frac{q(\boldsymbol{\theta}, \mathbf{r})}{p(\mathcal{D} \mid \boldsymbol{\theta}, \mathbf{r})\, p(\boldsymbol{\theta}, \mathbf{r})}\, d\boldsymbol{\theta}\, d\mathbf{r} = \mathbb{E}_{q(\boldsymbol{\theta},\mathbf{r})}\left[\log \frac{q(\boldsymbol{\theta}, \mathbf{r})}{p(\mathcal{D} \mid \boldsymbol{\theta}, \mathbf{r})\, p(\boldsymbol{\theta}, \mathbf{r})}\right]$$
$$= \mathbb{E}_{q(\boldsymbol{\theta},\mathbf{r})}\left[\log \frac{q(\boldsymbol{\theta}, \mathbf{r})}{p(\boldsymbol{\theta}, \mathbf{r})}\right] - \mathbb{E}_{q(\boldsymbol{\theta},\mathbf{r})}\left[\log p(\mathcal{D} \mid \boldsymbol{\theta}, \mathbf{r})\ \right].$$

Then, using a mean-field factorization $q(\boldsymbol{\theta}, \mathbf{r}) = q(\boldsymbol{\theta})q(\mathbf{r})$ and $p(\boldsymbol{\theta}, \mathbf{r}) = p(\boldsymbol{\theta})p(\mathbf{r})$, we obtain:

$$\int\int q(\boldsymbol{\theta}, \mathbf{r}) \log \frac{q(\boldsymbol{\theta}, \mathbf{r})}{p(\mathcal{D} \mid \boldsymbol{\theta}, \mathbf{r})\, p(\boldsymbol{\theta}, \mathbf{r})}\, d\boldsymbol{\theta}\, d\mathbf{r} = \mathbb{E}_{q(\boldsymbol{\theta})q(\mathbf{r})}\left[\log \frac{q(\boldsymbol{\theta})q(\mathbf{r})}{p(\boldsymbol{\theta})p(\mathbf{r})}\right] - \mathbb{E}_{q(\boldsymbol{\theta})q(\mathbf{r})}\left[\log p(\mathcal{D} \mid \boldsymbol{\theta}, \mathbf{r})\ \right]$$
$$= \mathbb{E}_{q(\boldsymbol{\theta})q(\mathbf{r})}[-\log p(\mathcal{D} \mid \boldsymbol{\theta}, \mathbf{r})] + D_{\mathrm{KL}}(q(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta})) + D_{\mathrm{KL}}(q(\mathbf{r}) \,\|\, p(\mathbf{r})).$$

A.2 Derivation of the Uncertainty Decomposition

The predictive distribution of a new output $\mathbf{y}^*$ given input $\mathbf{x}^*$ is

$$p(\mathbf{y}^* \mid \mathbf{x}^*, \mathcal{D}) = \int\int p(\mathbf{y}^* \mid \mathbf{x}^*, \boldsymbol{\theta}, \mathbf{r})\, q(\boldsymbol{\theta})q(\mathbf{r})\, d\boldsymbol{\theta}\, d\mathbf{r}. \tag{12}$$

Since for fixed $(\boldsymbol{\theta}, \mathbf{r})$ the predictive distribution is Gaussian:

$$p(\mathbf{y}^* \mid \mathbf{x}^*, \boldsymbol{\theta}, \mathbf{r}) = \mathcal{N}\big(\boldsymbol{\mu}_C, \boldsymbol{\sigma}_C^2\big), \tag{13}$$

the law of total variance gives

$$\mathrm{Var}(\mathbf{y}^* \mid \mathbf{x}^*, \boldsymbol{\theta}, \mathbf{r}) = \mathbb{E}_{q(\boldsymbol{\theta})q(\mathbf{r})}\big[\boldsymbol{\sigma}_C^2\big] + \mathrm{Var}_{q(\boldsymbol{\theta})q(\mathbf{r})}[\boldsymbol{\mu}_C]. \tag{14}$$

Then, from Gaussian conditioning, we have:

$$\mathbf{S} = \mathbf{B}\,\mathrm{diag}(\boldsymbol{\sigma}_P^2)\mathbf{B}^\top + R, \tag{15}$$
$$\boldsymbol{\mu}_C = \boldsymbol{\mu}_P + \mathbf{K}\left(\mathbf{b} - \mathbf{A}\mathbf{x}^* - \mathbf{B}\boldsymbol{\mu}_P\right), \tag{16}$$
$$\boldsymbol{\sigma}_C^2 = \mathrm{diag}(\boldsymbol{\sigma}_P^2) - \mathbf{K}\mathbf{S}\mathbf{K}^\top, \tag{17}$$

where

$$\mathbf{K} = \mathrm{diag}(\boldsymbol{\sigma}_P^2)\mathbf{B}^\top\mathbf{S}^{-1}. \tag{18}$$

Since we only consider the diagonal elements of the covariance matrix, substituting $\boldsymbol{\sigma}_C^2$ into Eqn. (14) yields

$$\mathrm{Var}(\mathbf{y}^* \mid \mathbf{x}^*, \boldsymbol{\theta}, \mathbf{r}) = \mathbb{E}_{q(\boldsymbol{\theta})q(\mathbf{r})}\big[\mathrm{diag}(\boldsymbol{\sigma}_P^2)\big] - \mathbb{E}_{q(\boldsymbol{\theta})q(\mathbf{r})}\Big[\mathbf{K}\mathbf{S}\mathbf{K}^\top\Big] + \mathrm{Var}_{q(\boldsymbol{\theta})q(\mathbf{r})}[\boldsymbol{\mu}_C]. \tag{19}$$

The first expectation corresponds to aleatoric uncertainty inherited from the network likelihood, while the second term is a positive semi-definite reduction arising from constraint conditioning.

We now expand the mean-variance term. Writing

$$\boldsymbol{\mu}_C = \boldsymbol{\mu}_P + \Delta(\boldsymbol{\theta}, \mathbf{r}), \tag{20}$$

where

$$\Delta(\boldsymbol{\theta}, \mathbf{r}) = \mathbf{K}\left(\mathbf{b} - \mathbf{A}\mathbf{x}^* - \mathbf{B}\boldsymbol{\mu}_P\right), \tag{21}$$

we obtain

$$\mathrm{Var}_{q(\boldsymbol{\theta})q(\mathbf{r})}(\boldsymbol{\mu}_C) = \mathrm{Var}_{q(\boldsymbol{\theta})}(\boldsymbol{\mu}_P) + \mathrm{Var}_{q(\boldsymbol{\theta})q(\mathbf{r})}(\Delta) + 2\ \mathrm{Cov}_{q(\boldsymbol{\theta})q(\mathbf{r})}\left(\boldsymbol{\mu}_P, \Delta\right). \tag{22}$$

Combining terms, the predictive variance decomposes as

$$\mathrm{Var}(\mathbf{y}^*) = \underbrace{\mathbb{E}_{q(\boldsymbol{\theta})}\left[\mathrm{diag}(\boldsymbol{\sigma}_P^2)\right]}_{\text{Aleatoric}} + \underbrace{\mathrm{Var}_{q(\boldsymbol{\theta})}(\boldsymbol{\mu}_P)}_{\text{Epistemic}} - \underbrace{\mathbb{E}_{q(\boldsymbol{\theta})q(\mathbf{r})}\left[\mathbf{K}\mathbf{S}\mathbf{K}^\top\right]}_{\text{Constraint Reduction}} + \underbrace{\mathrm{Var}_{q(\boldsymbol{\theta})q(\mathbf{r})}(\Delta)}_{\text{Constraint Tolerance}} + \underbrace{2\ \mathrm{Cov}_{q(\boldsymbol{\theta})q(\mathbf{r})}(\boldsymbol{\mu}_P, \Delta)}_{\text{Interaction}}. \tag{23}$$

Since $\mathbf{K}\mathbf{S}\mathbf{K}^\top$ is positive semi-definite, constraint conditioning always reduces predictive uncertainty in directions aligned with the constraint. The remaining terms quantify uncertainty induced by posterior variability in the constraint tolerance and its interaction with parameter uncertainty.

B Numerical Experiments

B.1 Computational Setup

All experiments were carried out on a PC, equipped with 32GB DDR5 RAM, with an Intel Core i9 CPU and an NVIDIA RTX5090 GPU with 32GB of DDR6 VRAM. All models were implemented in `PyTorch` Paszke et al. [2019]

B.2 Input and Output Variables

The variables used within the SPM model, and their applicability to our learning framework, are given in Table 3. Simulations were performed under heteroscedastic observation noise applied independently to each output variable.

Table 3: SPM inputs and outputs used in the learning.

Symbol	Description	Unit	Range
Inputs			
I	Applied current	A	0.5–3.0
SOC	State of charge	–	0.05–0.95
T	Ambient temperature	K	273–318
Outputs			
V	Terminal voltage	V	
V_{OCV}	Open-circuit voltage	V	
η_+	Positive electrode overpotential	V	
η_-	Negative electrode overpotential	V	
ΔV_{IR}	Ohmic (IR) drop	V	
$\dot{Q}_{\mathrm{tot}}$	Total volumetric heating	$\mathrm{W\,m^{-3}}$	
$\dot{Q}_{\mathrm{rev}}$	Reversible heating	$\mathrm{W\,m^{-3}}$	
$\dot{Q}_{\mathrm{irr}}$	Irreversible heating	$\mathrm{W\,m^{-3}}$	

B.3 Predictive Performance

We evaluate predictive performance across all outputs using mean squared error (MSE) and credible interval width (CW). Results are reported as mean $\pm$ 1.96σ across output variables and are shown in Table 4.

Table 4: Model predictive performance ($\mu \pm 1.96\sigma$ over outputs).

Model	MSE	CW
BNN	$1.13 \times 10^{-4} \pm 1.88 \times 10^{-3}$	0.067 ± 0.022
BCPNN	$\mathbf{1.11 \times 10^{-4} \pm 2.21 \times 10^{-3}}$	$\mathbf{0.059 \pm 0.022}$

B.4 Constraint Satisfaction

To assess structural consistency, we sampled 10,000 Monte Carlo posterior draws and computed the corresponding constraint residuals across the test set. The mean violation magnitude is reported in Table B.4.

Model	Constraint Violation
BNN	174.82 ± 90.78
BCPNN	$\mathbf{8.81 \times 10^{-4} \pm 4.3 \times 10^{-5}}$