Mechanistic and Data-Adaptive Bayesian Methods for Scientific Inference

by

Derek Hansen

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Statistics and Scientific Computing)
in the University of Michigan
2023

Doctoral Committee:

Professor Jeffrey Regier, Chair
Professor Camille Avestruz
Professor Yang Chen
Professor Edward Ionides

Derek Hansen
dereklh@umich.edu
ORCID iD: 0000-0001-8413-779X

To Rachael, the love of my life and partner in this journey.

ACKNOWLEDGMENTS

There are a number of people I would like to thank. First, I thank my advisor, Jeffrey Regier. This would not have been possible without your guidance, support, and expertise. It has been a pleasure learning from you and developing my research abilities through your example. I also thank my committee members Camille Avestruz, Yang Chen, and Edward Ionides for their advice and support throughout this process.

One of the best aspects of attending the University of Michigan the past five years was learning from exceptional peers. Though there are too many to list here, I would like to single out Drew Yarger, Ismael Mendoza, and Brian Manzo, who were co-authors on the works this dissertation is based on. It was an honor to work with you.

Part of this dissertation is based on work that I completed as an intern at Amazon Web Services AI Labs. I would like to thank Danielle Robinson, Shima Alizadeh, Gaurav Gupta, and Michael Mahoney for their mentorship and support throughout my time at Amazon and beyond.

Finally, I would like to thank my parents, Rachael, and my cats Mr. Snuffles and Sasha, for your unwavering emotional support.

This dissertation is based on work partially supported by the National Science Foundation Graduate Research Fellowship Program under grant no. 1256260.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

ABSTRACT

To draw rigorous conclusions from scientific data, Bayesian statistics requires computationally efficient methods for posterior inference as well as models that are both flexible and interpretable. This dissertation investigates the use of Bayesian methods in cases with both mechanistic and data-adaptive models. Mechanistic models are based on an interpretable understanding of the underlying process that generated the data, whereas data-adaptive models can adapt to unknown structure in the data but are often black-boxes that lack interpretability.

In the first half of this dissertation, we develop two methods for efficient and scalable Bayesian inference for well-understood mechanistic models. In the second chapter, we present a mechanistic state-space model of Argo float trajectories called ArgoSSM. ArgoSSM utilizes a physical model of floats' movement and incorporates daily ice-cover images and potential vorticity information to infer the missing locations of Argo floats while under ice in the Southern ocean. Inference is achieved by developing an efficient proposal distribution within sequential Monte Carlo (SMC). In the third chapter, we present the Bayesian Light Source Separator (BLISS), a new probabilistic method for detecting, deblending, and cataloging stars and galaxies. BLISS utilizes a mechanistic model for the placement of stars and galaxies and a deep generative model of galaxy shapes. By training neural networks via Forward Amortized Variational Inference (FAVI) for posterior inference, BLISS can perform fully Bayesian inference on megapixel images in seconds and produce highly accurate catalogs.

The latter two chapters of this dissertation describe methods that provide interpretable results while maintaining the flexibility of black-box data-adaptive models. In the fourth chapter, we propose the ProbConserv framework for incorporating physical constraints into a black-box probabilistic model. ProbConserv integrates the integral form of a conservation law into a Bayesian update, with a detailed analysis provided on learning the challenging Generalized Porous Medium Equation (GPME) family of partial differential equations. In the fifth chapter, we present FlowSelect, a new method for controlled feature selection from black-box predictive models. FlowSelect utilizes normalizing flows and a novel MCMC-based procedure to calculate p-values for each feature directly, demonstrating greater power and consistently controlling the false discovery rate (FDR) compared to competing knockoff-based approaches. FlowSelect also correctly infers genetic variants associated with specific soybean traits from GWAS data.

CHAPTER 1

Introduction

In modeling scientific phenomena, ambiguities frequently arise which cannot be resolved with available data. For example, in oceanography, Argo floats equipped with temperature and salinity sensors can disappear under ice cover for months. In astronomy, an apparent bright object on a telescope image could also be multiple dimmer objects that overlap visually. In genome-wide association studies, neighboring genes are highly correlated making it challenging to tell which gene is informative about the response.

By framing uncertainty and ambiguity in the language of probability, Bayesian methods offer a compelling answer to the demands of scientific applications. In the Bayesian paradigm, the practitioner specifies a generative model of the observed data and unobserved latent variables of interest. Inference is achieved via the posterior distribution of the unseen variables conditioned on the observed data. In this thesis, we explore two broad settings that differ in whether the generative model is *mechanistic* or *data-adaptive*. In mechanistic models, the process that generated the data is known and interpretable. For scientific applications, domain-awareness and interpretability come naturally from the underlying mechanism, and the main challenge is solving the inference problem of efficiently sampling the posterior distribution. We overcome this challenge either by utilizing an efficient proposal distribution within sequential Monte Carlo (SMC) for inferring Argo float trajectories in Chapter 2, or by carefully dividing an astronomical image into small "tiles" to enable scalable amortized variational inference in Chapter 3.

In the latter two chapters, we use flexible neural networks to learn a *data-adaptive* generative model. Neural networks have recently achieved remarkable results in generating high-fidelity examples of text and images (Vahdat and Kautz, 2020; Rombach et al., 2022). However, neural networks are black-boxes that lack the interpretability and domain-awareness of mechanistic models, limiting their acceptance within the scientific community (Baker et al., 2019). We develop two methods that address these limitations while maintaining the flexible power of neural networks. In Chapter 4 we develop a method for enforcing conservation laws as a Bayesian update to a black-box probabilistic model of physical functions. In Chapter 5 we develop a controlled

feature selection method that makes use of any black-box predictive model, allowing practitioners to interpret which features contain information about the response.

1.1 Bayesian generative models

In Bayesian inference, we formulate a generative model of the observed data $\mathbf{x} = x_{1:n}$ and a latent variable $\mathbf{z} = z_{1:m}$: [1]

$$p(\mathbf{z}, \mathbf{x}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z}). \tag{1.1}$$

The first density, $p(\mathbf{z})$, is the prior, which describes the process in which the the unseen variable z is generated. The second term, $p(\mathbf{x}|\mathbf{z})$, is the likelihood, which describes the distribution of the data $\mathbf{x}$ conditional on a particular value of $\mathbf{z}$. Inference is achieved via the posterior distribution $p(\mathbf{z}|\mathbf{x})$; i.e. the distribution of the latent variable z conditional on the data x. The form of $p(\mathbf{z}|\mathbf{x})$ follows from the application of Bayes' rule to Equation 1.1:

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{z})p(\mathbf{x}|\mathbf{z})}{p(\mathbf{x})}. \tag{1.2}$$

Performing inference through the posterior distribution in Equation 1.2 has multiple advantages compared to non-probabilistic methods. Bayesian inference encompasses uncertainty both from epistemic sources, such as model uncertainty, and aleatoric sources, such as irreducible noise in the data. Moreover, the posterior distribution in Equation 1.2 can be further updated with additional information if it becomes available. In particular, this presents a principled way to seamlessly incorporate different and possibly heterogeneous data sources containing information about $\mathbf{z}$.

Classically, $\mathbf{x}$ and $\mathbf{z}$ are described by an interpretable, parametric model with an explainable mechanism coming from the specific domain. In these *mechanistic* models, the latent variable $\mathbf{z}$ represents an unknown quantity of interest, such as market volatility (Shephard, 1996), astronomical catalogs (Regier et al., 2019; Liu et al., 2021), or states of infection in population (King et al., 2008). In each respective example, $\mathbf{x}$ could be the market returns, astronomical image of the night sky, or number of recorded infections in the population. We consider such mechanistic models in the domains of oceanography and astronomy in Chapters 2 and 3, respectively.

With recent advances in the capability of neural networks, black-box generative models have been proposed for the distribution of the data $\mathbf{x}$. In this setting, neural network weights θ are set to maximize the log-likelihood of the data $\log p_\theta(\mathbf{x})$. This deviates from a pure Bayesian setting that encapsulates all unknown parameters and variables into $\mathbf{z}$. In the case of the variational auto encoder (VAE, Kingma and Welling (2019)), the latent variable $\mathbf{z}$ is low-dimensional and the prior

[1] Here, we borrow the notation of Blei et al. (2017), which is more familiar to the VI audience, but variable names for may differ throughout different dissertation chapters.

$p(\mathbf{z})$ is set to follow an isotropic Gaussian distribution. A neural network, called the decoder, takes $\mathbf{z}$ as input and outputs distributional parameters for $p_\theta(\mathbf{x}|\mathbf{z})$. The latent z_i can be useful as a low-dimensional representation for visualization and compression. The VAE is a non-linear analogue to probabilistic PCA. In Chapter 3, we use a VAE to learn a low-dimensional representation of galaxy shapes.

Black-box generative models are also useful for interpolating functional data without knowledge of the underlying mechanics. In the neural process (Garnelo et al., 2018; Kim et al., 2019; Louizos et al., 2019), the input data $\mathbf{x} \equiv (x_1, y_1) \ldots, (x_T, y_T)$ contains observations of a function at particular input points, and $\mathbf{z}$ is a global state variable. Interpolation of the function at new inputs is facilitated through the posterior distribution of $\mathbf{z}$ conditioned on observed data. In Chapter 4, we use the attentive neural process (ANP, Kim et al. (2019)) to model physical systems of fluid dynamics governed by partial differential equations with differing parameter values.

Finally, normalizing flows (Kobyzev et al., 2020) have recently emerged as a state-of-the-art density estimation technique. Unlike the VAE and neural process, normalizing flows have no low-dimensional latent variable. Instead, a normalizing flow is a deterministic, invertible mapping from $\mathbf{x}$ to a random variable u of the same dimension with fixed distribution (Kobyzev et al., 2020). The trained normalizing flow is a generative model of $\mathbf{x}$, $p_\theta(\mathbf{x})$, which can be used for Bayesian missing data imputation. Specifically, if only part of $\mathbf{x}$ is observed, sampling the remaining components of $\mathbf{x}$ follows the form of Equation 1.2. This forms the basis for our controlled feature selection method in Chapter 5.

1.2 Posterior inference techniques

Unless a particular form for the prior and likelihood are chosen, the posterior distribution in Equation 1.2 is intractable. Thus, a method is needed to get an approximation. Formally, the goal of any Bayesian inference is to calculate the expectation of a given function $f : \mathbb{R}^m \to \mathbb{R}$ over the posterior distribution $p(\mathbf{z}|\mathbf{x})$:

$$\mathbb{E}\left(f(\mathbf{z})|\mathbf{x}\right) = \int f(\mathbf{z}) p(\mathbf{z}|\mathbf{x}) dz. \tag{1.3}$$

Quantities of interest such as the mean, variance, higher-order moments, median, and quantiles arise from different choices of f. In most cases, there is no analytical solution to the above expectation, and there is no way to directly sample the posterior distribution. The gold standard for Bayesian inference has been Markov chain Monte Carlo (MCMC). As the name implies, MCMC constructs a Markov chain whose stationary distribution is the target $p(\mathbf{z}|\mathbf{x})$. MCMC is widely used because the Cesaro average of f calculated over these draws converges almost surely to the target expectation above under mild conditions (Smith and Roberts, 1993). However, this convergence

can take a long time in high dimensions where neighboring draws of $\mathbf{z}_i$ are autocorrelated.

In cases where $\mathbf{z}$ and $\mathbf{x}$ have a sequential structure, methods such as Sequential Monte Carlo (SMC) can make use of this structure to more efficiently draw samples (Lopes and Tsay, 2011). SMC is an extension of importance sampling. In importance sampling, samples are drawn from a tractable proposal distribution, then reweighted with importance weights to match the intractable target distribution. As the number of particles are increased, importance sampling is a consistent estimator of the desired expectation (Owen, 2013). To avoid the distribution collapse that can occur in high-dimensions, SMC works by targeting the distribution of z_t at a particular time t given an existing sample at time $t-1$. New particles are drawn from a proposal distribution and then reweighted and resampled based on importance weights from the true distribution. This procedure produces consistent estimates as the number of particles increases. However, practical performance hinges on the quality of the proposal distribution. In situations with a high signal-to-noise ratio, lookahead strategies are necessary for good performance (Lin et al., 2013; Guarniero et al., 2017; Naesseth et al., 2019). We develop such a strategy in our implementation of SMC in Chapter 2.

Variational Inference (VI) takes a different approach than MCMC and SMC. In VI, the goal is to find a tractable distribution q_ϕ that minimizes the Kullback-Leibler (KL) divergence to $p(z|x)$. Finding the optimal parameter ϕ^* is an optimization problem:

$$\begin{aligned}\phi^* &= \text{argmin}_\phi \text{KL}(q_\phi(z)||p(z|x)), \\ &= \text{argmin}_\phi \mathbb{E}_{q_\phi}(\log q_\phi(z) - \log p(z|x)).\end{aligned} \tag{1.4}$$

The KL divergence in eq. (1.4) is non-negative and zero if and only if p and q_ϕ are equal in distribution. Since the density $p(z|x)$ is not tractable, we cannot directly optimize eq. (1.4). However, maximizing the evidence lower bound (ELBO) is equivalent to minimizing the KL divergence (Blei et al., 2017):

$$\phi^* = \text{argmax}_\phi \mathbb{E}_{q_\phi}(\log p(z,x) - \log q_\phi(z)). \tag{1.5}$$

For some settings, there are particular choices of the variational family for which the ELBO can be directly optimized via coordinate ascent (Blei et al., 2017). For other settings, the negative ELBO is minimized via stochastic gradient descent (SGD).

One choice for the variational distribution q is a mean-field approximation, where each latent parameter z_k has its own variational parameter ϕ_k and the distribution is factorized:

$$q_\phi(z) = \prod_{k=1}^{K} q_{\phi_k}(z_k). \tag{1.6}$$

However, when dealing with expanding data, a mean-field approximation requires fitting a new variational parameter ϕ_k for each object. A recent innovation to improve the scaling of VI to

Chapter	Setting	Generative model	Posterior Inference
Chapter 2	Oceanography	Mechanistic	SMC^2 (SMC+MCMC)
Chapter 3	Astronomy	Mechanistic/Data-adaptive	VI (FAVI)
Chapter 4	Fluid dynamics	Data-adaptive	VI (ELBO)
Chapter 5	Controlled feature selection	Data-adaptive	MCMC

Table 1.1: Summary of the settings and methods in each chapter of this dissertation. The first three chapters present a Bayesian model, while Chapter 5 utilizes a black-box generative model to implement a frequentist test of feature signifance (the conditional randomization test (CRT) from Candès et al. (2018)).

massive datasets is to use an *amortized* variational distribution (Gershman and Goodman, 2014; Kingma and Welling, 2019). In this scenario, the variational distribution still follows Equation 1.6. However, ϕ_k is not optimized directly, but rather it is the output of a function f_ϕ which is optimized.

$$\phi_k = f_\phi(x_k). \tag{1.7}$$

Here, f_ϕ is a flexible function such as a neural network which takes in as input the data point associated with the latent variable k. We utilize amortized variational inference trained on the ELBO in Chapter 4.

Amortization also allows for the use of alternative objective functions to the ELBO. This is desirable in cases where z is discrete, as optimizing discrete variables via stochastic gradient methods poses a challenge. The usual reparameterization trick, which allows for relatively low variance estimates of the gradient, does not work with discrete variables. However, if we consider instead the reverse KL divergence, this problem disappears. Instead of maximizing the ELBO, the variational parameters ϕ are optimized to minimize the expectation of the KL divergence between the distributions over the generative model

$$\begin{aligned} \hat{\phi} &= \text{argmin}_\phi E_{p(x)} KL(p(z|x)||q_\phi(z)) \\ &= \text{argmax}_\phi \mathbb{E}_{p(z,x)} \log q(z|x) \end{aligned} \tag{1.8}$$

Because the expectation in Equation 1.8 is not taken over q_ϕ, the gradient can be moved inside the expectation, and unbiased gradients can be calculated by simulating from the generative model $p(z, x)$. This optimization is called forward amortized variational inference (FAVI, Ambrogioni et al. (2019)), and it has also appeared as part of the wake-sleep algorithm (Hinton et al., 1995; Le et al., 2020). In particular, optimization of Equation 1.8 is called a "sleep-phase" update because the samples are "dreamed up" from simulations of the generative model, rather than from real data. The FAVI objective forms the basis of our inference model for detecting astronomical objects in Chapter 3.

1.3 Summary of contributions and organization of thesis

This dissertation presents Bayesian methods applied to a variety of scientific applications. These methods utilize either mechanistic or black-box generative models, and each method uses different posterior inference techniques tailored to the application. See Table 1.1 for a comparison.

In Chapter 2, we introduce ArgoSSM, a probabilistic state-space model of Argo ocean floats. ArgoSSM infers the posterior distribution of a float's position and velocity at each time based on GPS, daily ice cover images, and potential vorticity data. This inference is achieved using Sequential Monte Carlo equipped with an efficient proposal distribution, which is effective despite the high signal-to-noise ratio in the GPS data. We also infer a posterior distribution of model parameters using the SMC^2 algorithm (Chopin et al., 2013; Duan and Fulop, 2015), which combines an annealed importance sampling scheme with particle Markov chain Monte Carlo (Andrieu et al., 2010) to sample the posterior distribution of model parameters. Compared to existing interpolation approaches in oceanography, ArgoSSM more accurately predicts held-out GPS measurements. Moreover, because uncertainty estimates are well-calibrated in the posterior distribution, ArgoSSM enables more robust and accurate temperature and salinity field estimation. This is based on joint work with Drew Yarger under submission (Hansen and Yarger, 2022).

In Chapter 3, we present a new probabilistic method for detecting, deblending, and cataloging astronomical sources called the Bayesian Light Source Separator (BLISS). BLISS uses a mechanistic model for the placement of stars and galaxies and a deep generative model of galaxy shapes. For posterior inference, BLISS utilizes Forward Amortized Variational Inference (FAVI, Ambrogioni et al. (2019)). The BLISS inference routine is fast, requiring a single forward pass of the encoder networks on a GPU once the encoder networks are trained. BLISS can perform fully Bayesian inference on megapixel images in seconds, and produces highly accurate catalogs. This is based on work presented at the Machine Learning for Astronomical Sciences workshop at ICML 2022 (Hansen et al., 2022b).

In Chapter 4, we propose PROBCONSERV, a framework for incorporating physical constraints into a black-box probabilistic model. To do so, PROBCONSERV integrates the integral form of a conservation law into a Bayesian update. We provide a detailed analysis of PROBCONSERV on learning with the Generalized Porous Medium Equation (GPME), a widely-applicable parameterized family of PDEs that illustrates the qualitative properties of both easier and harder PDEs from the perspective of numerical methods. PROBCONSERV is effective for easy GPME variants, performing well with state-of-the-art competitors; and for harder GPME variants it outperforms other approaches that do not guarantee volume conservation. PROBCONSERV seamlessly enforces physical conservation constraints, maintains probabilistic uncertainty quantification (UQ), and deals well with shocks and heteroscedasticity. In each case, it achieves superior predictive performance

on downstream tasks. An abridged version of this work will appear at the Physics for Machine Learning Workshop at ICLR 2023 (Hansen et al., 2023).

Finally, in Chapter 5, we develop a method for controlled feature selection from black-box predictive models. Controlled feature selection aims to discover the features a response depends on while limiting the false discovery rate (FDR) to a predefined level. Recently, multiple deep-learning-based methods have been proposed to perform controlled feature selection through the Model-X knockoff framework. We demonstrate, however, that these methods often fail to control the FDR for two reasons. First, these methods often learn inaccurate models of features. Second, the "swap" property, which is required for knockoffs to be valid, is often not well enforced. We propose a new procedure called FLOWSELECT to perform controlled feature selection that does not suffer from either of these two problems. To more accurately model the features, FLOWSELECT uses normalizing flows, the state-of-the-art method for density estimation. Instead of enforcing the "swap" property, FLOWSELECT uses a novel MCMC-based procedure to calculate p-values for each feature directly. Asymptotically, FLOWSELECT computes valid p-values. Empirically, FLOWSELECT consistently controls the FDR on both synthetic and semi-synthetic benchmarks, whereas competing knockoff-based approaches do not. FLOWSELECT also demonstrates greater power on these benchmarks. Additionally, FLOWSELECT correctly infers the genetic variants associated with specific soybean traits from GWAS data. This is based on work in Hansen et al. (2022a) published in *Advances in Neural Information Processing Systems*.

CHAPTER 2

A Probabilistic Model of Ocean Floats under Ice

2.1 Introduction

Reliable measurements of the ocean are essential for scientific tasks such as modeling climate change (Lyman and Johnson, 2014), estimating temperature and salinity fields (Chang et al., 2009), and tracking the global hydrological cycle (Hosoda et al., 2009). However, measuring the vast reaches of the ocean is a challenging task. Historically, measurements were taken by sensors on board ships, but this limited most collection to popular routes, far from a comprehensive survey of the ocean.

To address this issue, the Argo project (Argo, 2020) was started in 1998 as a multinational collaboration to collect data about the world's oceans. Instead of using ships, the Argo project deploys floats that freely drift with the ocean's currents at a depth of one kilometer. Every ten days, a float descends to two kilometers and then ascends to the surface, measuring temperature, salinity, and pressure at multiple depths along the way. At the surface, the float records its position via GPS and transmits data via the Iridium satellite system to Argo data centers for processing. Each iteration of this data collection process is called a profile.

The implementation of an ice avoidance program in Klatt et al. (2007) has enabled floats to explore previously inaccessible regions such as the Southern Ocean near Antarctica. In these regions, when ice cover is detected, an Argo float does not attempt to resurface to avoid damage. While the float continues to take profiles as normal, GPS tracking can be lost for more than six months. Because the float moves freely, GPS tracking is critical for pinpointing its whereabouts when data is collected. Figure 2.1 illustrates both the extent and seasonality of missing locations in the Southern Ocean. Profiles with missing locations make up 16% of the Argo dataset in the Southern Ocean (Chamberlain et al., 2018; Reeve et al., 2019). Thus, simply removing the measurements without locations leads to both spatial and seasonal biases in estimation tasks (Gray and Riser, 2014; Reeve et al., 2016, 2019). In turn, these biases can affect our understanding of systems in which the Southern Ocean plays an important role, such as the global climate (e.g. Gray et al.,

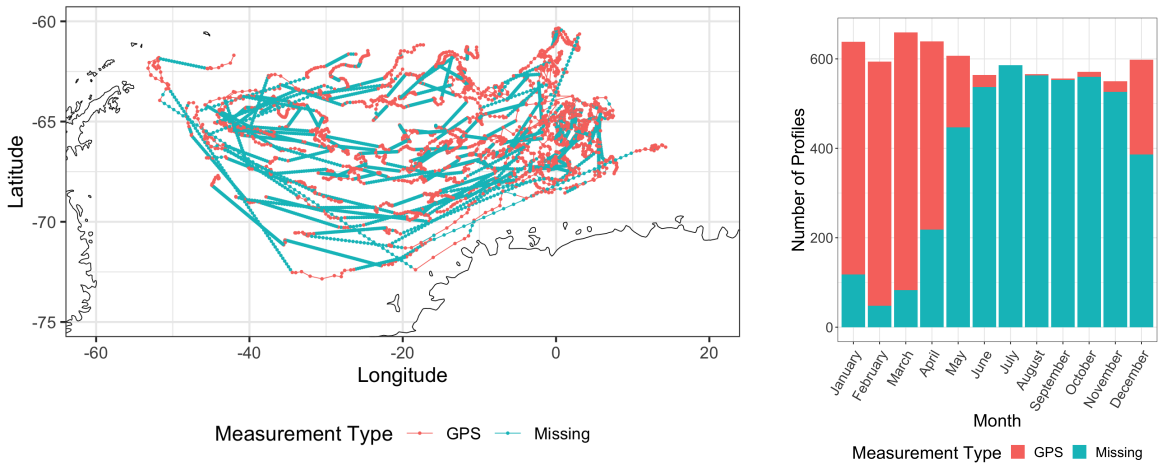


Figure 2.1: (Left) Recorded locations of 46 selected floats in the Weddell Gyre, a region of the Southern Ocean. Each dot represents a profile, a collection of temperature and salinity measurements. In this plot, missing locations of profiles are linearly interpolated between valid location measurements and do not reflect the floats' true positions (or the positions' uncertainties). (Right) Number of profiles collected by these floats in each month of the year. Nearly all profiles collected in the winter had missing locations.

2018).

The most common way to handle missing locations is to linearly interpolate between the nearest two observed locations (Wong et al., 2020). Chamberlain et al. (2018) improved this technique by interpolating a path based on local estimates of potential vorticity (PV), which takes into account the effect of the Earth's rotation on the ocean water column (see Section 7.7 of Talley et al., 2011, for more information). In parallel, Chamberlain et al. (2018) showed how positioning data from sound sources could constrain the locations using a velocity-driven linear state-space model (SSM), but with fixed parameters and no PV component. While their underlying model is probabilistic, they only use the predicted mean and do not report any model-estimated uncertainty.

All of these previous approaches fail to offer well-calibrated estimates of uncertainty alongside their predictions. Chamberlain et al. (2018) estimated that location uncertainty can be as much as 116km. However, this estimate came from the error in held-out data estimates and is not available for unseen data points actually under the ice. Moreover, these approaches do not offer a way to incorporate location uncertainty into downstream estimation tasks, leading to overconfident final estimates.

To solve both of these issues, we introduce a probabilistic state-space model of Argo float movements. Our model, ArgoSSM, combines available GPS measurements with local conservation of potential vorticity (PV) to more accurately model the floats' positions and velocities. In addition, we use data on ice concentration in the Southern Ocean to further constrain the floats' positions and directly characterize the missing data mechanism in our statistical model. Because

ArgoSSM is a generative model, we infer a posterior distribution of the trajectory of each float as well as model parameters given all available information. This inference is achieved via an efficient particle filtering scheme that fully adapts to the high signal-to-noise ratio in the GPS data. In a case study of Argo floats in the Southern Ocean, ArgoSSM more accurately predicts held-out GPS measurements than previous interpolation methods. Moreover, ArgoSSM illuminates where location data is particularly sparse and uncertain, quantifying a source of uncertainty for downstream estimates (e.g., temperature, salinity, or ocean circulation) that would have been ignored with imputed location estimates. By providing a principled and flexible statistical framework to handle missing location measurements, ArgoSSM can easily be incorporated into future scientific study of the Southern Ocean.

2.2 A generative model of ocean float movement

To motivate the probabilistic framework of ArgoSSM, we start with a simple model of how Argo floats move through the ocean. Profile data is collected at times $t_1 < t_2 < \dots t_N$, approximately ten days apart. At each time t_n, let X_n be the geographic position in latitude and longitude at time t_n. We take into account the elapsed time $\Delta t_n = t_n - t_{n-1}$ when updating the position from X_{n-1} to X_n. If the float's position at $n-1$ was X_{n-1}, we might expect that X_n will be close to X_{n-1} with noise proportional to the time passed. This can be written explicitly as a two-dimensional random walk (RW) model:

$$X_n = X_{n-1} + \epsilon_n^X, \tag{2.1}$$

where ϵ_n^X follows a multivariate Gaussian distribution with zero mean and covariance $\Delta t_n \Sigma_X$.

For a given index n, the expected value of X_n conditioned on X_{n-1} and X_{n+1} is a time-weighted average of X_{n-1} and X_{n+1}. Thus, the RW model is a generative model of float movement where the optimal predictor for an unseen point is linear interpolation. Linear interpolation might work well for short gaps in time, but it breaks down for large gaps in time that are seen in the Southern Ocean. This because linear interpolation ignores local information such as momentum. If we know the current has carried the float from position X_{n-1} to position X_n, that same current will likely carry the float further in the same direction. More specifically, each float has a velocity V_n that indicates where it is headed next. Thus, we modify Equation 2.1 to take velocity into account:

$$X_n = X_{n-1} + \Delta t_n V_{n-1} + \epsilon_n^X. \tag{2.2}$$

The velocity V_n also changes over time according to an auto-regressive (AR) model:

$$V_n = (1 - \alpha^{\Delta t_n}) v_0 + \alpha^{\Delta t_n} V_{n-1} + \epsilon_n^V, \tag{2.3}$$

where ϵ_n^V follows a multivariate Gaussian distribution with zero mean and covariance $\Delta t_n \Sigma_V$. Two parameters govern the velocity update: the long-run velocity of the float v_0 and the autoregressive term $\alpha \in [0, 1]$. The parameter α determines how quickly the velocity reverts to the long-run velocity v_0.

We refer to Equations 2.2 and 2.3 collectively as the AR model. While the AR model is more realistic than the RW model, it simplifies the true behavior of the floats. Notably, it ignores the float's vertical movement as it rises or drops in the ocean and intra-day movement on the ocean surface. Thus, the velocity state-variable should be interpreted as the average direction of the float over several days rather than a local estimate of the instantaneous velocity. The AR model is similar to the state-space model introduced in Chamberlain et al. (2018), though with key differences. The main modeling difference is that Chamberlain et al. (2018) updates the velocity according to a random walk (corresponding to $\alpha = 1$ in Equation 2.3). In Section 2.5, we find for many floats that the inferred autoregressive parameter α is significantly less than 1. Moreover, Chamberlain et al. (2018) fixes all parameter values, whereas we estimate them alongside the positions and velocities.

Information about the float's position comes from GPS measurements Y_n corresponding to each X_n:

$$Y_n = X_n + \epsilon_n^Y, \tag{2.4}$$

where ϵ_n^Y is the measurement error that follows a multivariate Gaussian with zero mean and covariance Σ_Y. With the Iridium satellite system, Argo GPS measurements are rated to be accurate to within eight meters (Wong et al., 2020), so the variability of the measurement error ϵ_n^Y in Equation 2.4 will typically be magnitudes lower than that of the transition error ϵ_n^X in Equation 2.2.

2.2.1 Missing due to ice cover

While GPS measurements accurately pin down the floats' locations, they may not always be available. To represent this availability, let A_n be an indicator variable that equals one if GPS is available at time t_n and zero otherwise. In the Southern Ocean, since the float only surfaces after three consecutive ice-free detections, A_n is mostly determined by the ice-avoidance algorithm (Klatt et al., 2007), which depends on the concentration of ice in the area.

To model the availability indicator A_n, we first require an estimated probability of detecting ice. We have available daily ice concentration estimates from Fetterer et al. (2017), which uses remotely-sensed data from microwave instruments on satellites. Let $E(x, t)$ be the concentration of ice at position x and time t. Accounting for imperfect ice detection due to limited resolution, the probability that the float detects ice at position x and time t is $\tilde{E}(x, t) = p^{\text{TPR}} E(x, t) + (1 - p^{\text{TNR}})E(x, t)$, where p^{TPR} is the "true positive rate" (correctly detecting ice) and p^{TNR} is the "true

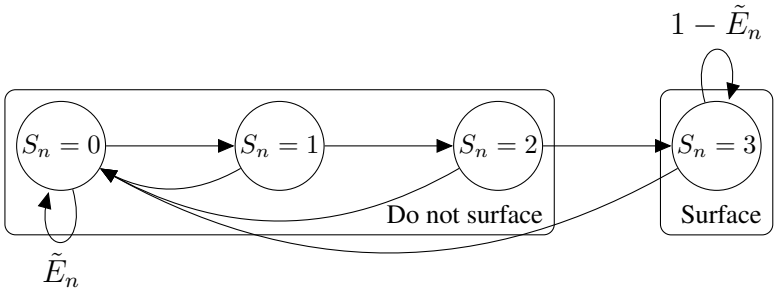


Figure 2.2: Transition diagram of the state S_n of the Argo ice detection algorithm. The transition to the next state is determined based on whether ice is detected or not. The probability of detecting ice, $\tilde{E}_n \equiv \tilde{E}(X_n, t_n)$, depends on both the geographic position X_n and the time t_n. Surfacing requires at least three consecutive ice-free detections.

negative rate" (correctly detecting no ice). We expect p^{TPR} and p^{TNR} to be close to 1, but since detections are based on the temperature of the water, we expect to see more false positives than false negatives (i.e. $p^{\text{TNR}} < p^{\text{TPR}}$).

With the probability of detecting ice at a particular time and location, we model the state of the ice-avoidance algorithm as a Markov chain, illustrated in Figure 2.2. The state of the algorithm, denoted S_n, can take one of four possible values in $\{0, 1, 2, 3\}$. If ice is detected, the state S_n resets to 0. Otherwise, the state increments by one (i.e. $S_n = (S_{n-1} + 1) \wedge 3$). The probability of transitioning to $S_n = 0$ is equal to the probability of detecting ice $\tilde{E}(X_n, t_n)$. Likewise, the probability of maintaining the streak of ice-free detections is $1 - \tilde{E}(X_n, t_n)$. The ice-avoidance state S_n determines whether the float surfaces, which directly impacts the availability of the GPS measurement A_n. For $S_n \in \{0, 1, 2\}$, the float will not surface, so the measurement is missing ($A_n = 0$) with probability 1. If $S_n = 3$, the ice-avoidance algorithm will no longer prevent the float from surfacing. In this case, $P(A_n = 1 | S_n = 3) = (1 - p^{\text{MAR}})$, where p^{MAR} is the probability that the GPS measurement is missing for reasons other than ice avoidance.

Even if not of direct interest, knowledge of the ice-avoidance state S_n provides information about the float's position. In particular, whenever S_n equals 0, there was most likely ice present at position X_n, so it is more likely the float is in a region with high concentration of ice. Similarly, if $S_n \in \{1, 2, 3\}$, then the float did not detect ice, so it is more likely the float is in a region with a low concentration of ice. This relationship is naturally captured in the joint posterior distribution of the three state variables (X_n, V_n, S_n) given the observed data. We discuss how this posterior distribution is estimated in Section 2.3.

2.2.2 Local conservation of potential vorticity

The motion of a freely-circulating object in the ocean conserves potential vorticity (PV) (Talley et al., 2011). PV is a function of the depth of the ocean and the vorticity, or local spin, which itself is a function of latitude. Incorporating PV conservation is important for both generating more realistic float trajectories and improving predictive performance in periods without GPS measurements. To incorporate local conservation of PV into the state-space model, we frame it as a probabilistic constraint. From a first-order Taylor approximation and Equation 2.2, we have that the difference in PV between two positions X_{n+1} and X_n is approximately $\epsilon_n^{\text{PV}} = \nabla\text{PV}(X_n) \cdot \Delta t_n V_n$, where $\nabla\text{PV}(X_n)$ is the gradient of PV with respect to position. Since PV is a conserved quantity, ϵ_n^{PV} should be close to zero, but not exactly zero due to the first-order approximation and imperfect estimation of PV. To account for this, we suppose ϵ_n^{PV} follows a univariate Gaussian distribution with mean 0 and standard deviation $\Delta t_n \sigma_{PV}$, with σ_{PV} determining the relative strictness of PV conservation.

Because ϵ_n^{PV} is linear with respect to V_n and Gaussian, it is an implicit measurement of V_n that can be incorporated as a Bayesian update of the autoregressive velocity update from Equation 2.3:

$$V_n | X_n, V_{n-1} = B\left((1-\alpha^{\Delta t_n})v_0 + \alpha^{\Delta t_n} V_{n-1}\right) + B^{\frac{1}{2}}\epsilon_n^V, \tag{2.5}$$

where $B = \left(I + \frac{1}{\sigma_{PV}^2}(\Delta t_n \Sigma_V)\nabla\text{PV}(X_n)\nabla\text{PV}(X_n)'\right)^{-1}$ is a matrix that encompasses the effect of PV conservation. The form of B follows from a conjugate Bayesian update with a Gaussian prior and likelihood (see Appendix A.5). Notably, B shrinks the component of velocity in the direction of $\nabla\text{PV}(X_n)$ while leaving the other component unchanged. This discourages large changes in PV, leading to more realistic predictions of under ice float trajectories. In turn, this improves predictive performance during periods with no GPS measurements (Section 2.5).

The value of $\nabla\text{PV}(X_n)$ used in the update is estimated from known ocean depth and latitude. We use bathymetry exported from the Southern Ocean State Estimate (SOSE) (Verdy and Mazloff, 2017). Potential vorticity is approximated by the expression $\text{PV}(X_n) = f(X_n)/h(X_n)$ where $h(X_n)$ is the ocean depth at X_n and $f(X_n) = 2\Omega \sin(\pi X_{n,lat}/180)$ is the Coriolis parameter based on the latitude of X_n and $\Omega = 7.292115 \cdot 10^{-5}$. See Talley et al. (2011) for more specifics on how PV is defined and its interpretation.

As in Chamberlain et al. (2018), we smooth the bathymetry to improve the results, and furthermore we estimate the gradient of PV using local quadratic regression (Fan and Gijbels, 1996). We use a bandwidth of 300 kilometers for local regression, though other bandwidths can easily be integrated into our analysis pipeline. These regressions result in the smoothed estimates $\hat{h}(\cdot)$ of

ocean depth and $\nabla\hat{h}(\cdot)$ of its gradient. With these values, the estimate of the PV gradient is

$$\nabla \text{PV}(X_n) = \frac{2\pi\Omega \cos\left(\frac{\pi X_{n,lat}}{180}\right)}{180\hat{h}(X_n)} \begin{pmatrix} 0 \\ 1 \end{pmatrix} - \frac{f(X_n)}{\hat{h}(X_n)^2} \nabla\hat{h}(X_n) \tag{2.6}$$

from the quotient derivative rule. These estimates are precomputed on a fine grid of locations, then bilinearly interpolated for efficient inference. Further details are in Appendix A.4.

2.3 State and parameter estimation with sequential Monte Carlo

As a state-space model, the distribution of float trajectories defined by ArgoSSM is composed of three parts: an initial distribution $\mu_\theta(X_1, V_1, S_1)$ in the float's initial state, a conditional transition distribution from the previous state to the current state $f_\theta(X_n, V_n, S_n | X_{n-1}, V_{n-1}, S_{n-1})$, and a measurement distribution $g_\theta(A_n, Y_n | S_n, X_n)$. The parameter θ encapsulates the parameters of each equation. The measurement distribution can be further decomposed into whether GPS is available $g^A_\theta(A_n|S_n)$ and the distribution of the GPS reading $g^Y_\theta(Y_n|X_n)$ when $A_n = 1$. This allows the distribution of the positions, velocities, and available measurements to factorize as follows:

$$\begin{aligned} p_\theta(X_{1:N}, V_{1:N}, S_{1:N}, A_{1:N}, Y_{n:A_n=1}) = \mu_\theta(X_1, V_1, S_1) \prod_{n=2}^{N} f_\theta(X_n, V_n, S_n | X_{n-1}, V_{n-1}, S_{n-1}) \\ \prod_{n=2}^{N} g^A_\theta(A_n|S_n) \prod_{A_n=1} g^Y_\theta(Y_n|X_n) \end{aligned} \tag{2.7}$$

For the RW and AR models, the initial, transition, and measurement distributions are Gaussian, the expected values of the transition and measurement distributions are affine functions of their dependencies, and the missingness indicator A_n is assumed to be independent of X_n and Y_n. This means that both $A_{1:N}$ and $S_{1:N}$ can be integrated out of Equation 2.7 above, leaving the marginal distribution of the remaining variables as a multivariate Gaussian with a sparse covariance matrix. Thus, sampling from the posterior $p_\theta(X_{1:N}, V_{1:N}|Y)$ is tractable and efficient via a Kalman smoother. See Appendix A.1 for more details of how the Kalman filter and smoother are derived.

Incorporating ice cover information and local conservation of PV introduces non-linearities that make the Kalman filter inapplicable. Instead, to sample from the posterior, we use sequential Monte Carlo (SMC), or a particle filter, followed by the forward filtering backward sampling (FFBS) procedure from Godsill et al. (2004). A full description of SMC and FFBS can be found in Appendices A.2.1 and A.2.2, respectively. Briefly, SMC works by approximating the distribution of the state $Z_n \equiv (X_n, V_n, S_n)$ conditional on all data up to and including index n, denoted

as Y^n. These approximations are formed by updating simulated particles targeting the distribution at index $n-1$ to particles targeting index n. Starting with an equally-weighted sample $\tilde{Z}_{i,n-1}, i = 1, \ldots, K$ targeting $\mathcal{P}_\theta(Z_{n-1}|Y^{n-1})$, new particles at index n are drawn from a proposal distribution $Z_n \sim q(\cdot|\tilde{Z}_{i,n-1})$. These are then reweighted and resampled to target the filtered distribution $\mathcal{P}_\theta(Z_{i,n}|Y^n)$. With enough particles, SMC gives an arbitrarily close approximation to the filtered distribution at each n. To recover the smoothed distribution of each state conditional on all data points, $Z_{1:N}|Y^N$, we employ backward sampling over the existing samples from SMC using the FFBS algorithm.

Choosing a good proposal distribution is important for efficient inference with SMC. A simple choice is to propose particles from the transition equation f_θ, which is blind to future observations (Gordon et al., 1993). This so-called "bootstrap filter" works well in many settings but poses a problem for the Argo data, because, as mentioned earlier, the GPS measurement noise in Equation 2.4 is much lower than the transition noise in Equation 2.3. The chance that a proposal from the transition distribution will land close to the actual observed point is quite low. As a result, the importance weights on each particle will be very uneven, frequently placing all probability mass on a single particle. This leads to sample degeneracy and high variance in both the estimates of the predicted path and the log-likelihood of the model for inference.

To overcome this, we utilize twisted target distributions at each index n (Guarniero et al., 2017; Naesseth et al., 2019), which amounts to weighting each particle by an additional function $\psi_n(Z_n)$ before resampling. In this framework, the ideal proposal distribution $q_n(Z_n|Z_{n-1})$ is equal to the posterior distribution conditional on all available data, denoted Y^N, and previous latent variables. The ideal pre-weight function $\psi_n(Z_n)$ is equal to the predictive density of all future observations, written $Y_{n+1:N}$.

$$\begin{aligned} q_n(Z_n|Z_{n-1}) &= \mathcal{P}(Z_n|Z_{n-1}, Y^N) \\ \psi_n(Z_n) &= \mathcal{P}(Y_{n+1:N}|Z_n). \end{aligned} \tag{2.8}$$

These settings will lead to importance sampling weights that are perfectly even and an exact estimate of the model likelihood (Guarniero et al., 2017), which we show in Appendix A.2.3.

Although both terms in Equation (2.8) are intractable, they offer guidance: a good proposal will be as close as possible to the posterior that looks ahead at all future data points. For ArgoSSM, we can use the tractable posterior distribution from the AR model as a proposal distribution in each step of SMC:

$$\begin{aligned} q_n(Z_n|Z_{n-1}) &= \mathcal{P}_{\theta,\text{AR}}(Z_n|Z_{n-1}, Y^N) \\ \psi_n(Z_n) &= \mathcal{P}_{\theta,\text{AR}}(Y_{n+1:N}|Z_n). \end{aligned} \tag{2.9}$$

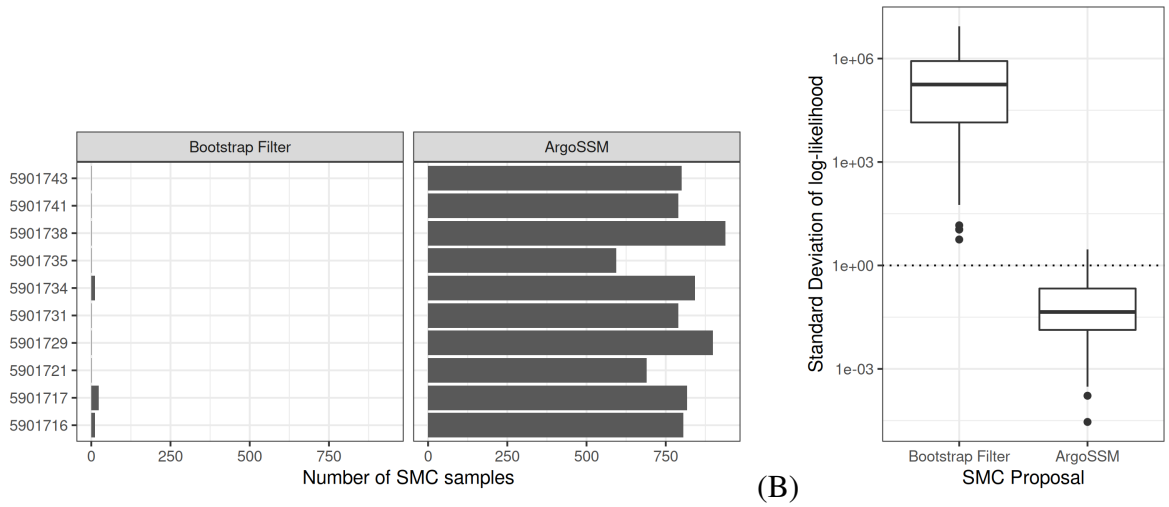


Figure 2.3: (A) Effective Sample Size (ESS) comparison from 1000 particles at the final observation for the bootstrap particle filter versus our adapted particle filter across each float used in the first holdout experiment. Floats are identified by their World Meteorological Organization (WMO) number. (B) The standard deviation of the estimated log-likelihood calculated across each float used in the first holdout experiment. The dotted line represents the target threshold for inference.

Here, $\mathcal{P}_{\theta,\text{AR}}$ is the posterior distribution of the AR model. The parameters of the AR model are set to be equal to the equivalent parameters of ArgoSSM. Both terms are Gaussian, so its mean and covariance can be efficiently calculated with a Kalman smoother (Lopes and Tsay, 2011) (c.f. Appendix A.1). By looking ahead to all high signal-to-noise ratio (SNR) GPS measurements and directly incorporating local PV information, our proposal allows for a more efficient sampling procedure than the baseline bootstrap particle filter (Gordon et al., 1993). Unlike other look-ahead methods such as Lin et al. (2013), our method takes advantage of the fact that the AR model is close to the ArgoSSM model and is fully tractable via the Kalman smoother.

In addition to adapting to all GPS measurements, there are a few properties of ArgoSSM that can be utilized to form an efficient proposal distribution. The ice-avoidance algorithm state S_n can be integrated out by calculating $P(S_{i,n} = k|X_{i,1:n})$ recursively:

$$\begin{aligned} P(S_n = k|X_{1:n}, A_{1:n}) &= \frac{g_\theta^A(A_n|S_n = k)P(S_n = k|X_{1:n}, A_{1:n-1})}{\sum_{k'} g_\theta^A(A_n|S_n = k')P(S_n = k'|X_{1:n}, A_{1:n-1})} \\ P(S_n = k|X_{1:n}, A_{1:n-1}) &= \sum_{k'} P(S_n = k|X_n, S_{n-1} = k')P(S_{n-1} = k'|X_{1:n-1}, A_{1:n-1}) \end{aligned} \tag{2.10}$$

Even if the probabilities of S_n were not of interest, they are necessary to calculate the log-probability of the missingness indicator A_n given the other state variables (X_n, V_n). See Appendix A.3 for more details.

While the likelihood is no longer exact after incorporating the non-linearities from ice cover

Parameter	Prior Distribution
α	Beta$(8.9, 0.99)$
σ^2_{PV}	LogNormal(0.0, 3.0)
$v_{0,1}$	Normal$(0.0, 0.01)$
$v_{0,2}$	Normal$(0.0, 0.01)$
p^{MAR}	Beta$(1.0, 9.0)$
p^{TPR}	Beta$(9.0, 1.0)$
p^{TNR}	Beta$(9.0, 1.0)$

Table 2.1: Table of priors used for each float parameter. The priors for the transition covariance Σ_X and observation variance Σ_Y were set by fitting a gamma distribution to maximum-likelihood estimates of the AR model on a small holdout set of floats.

and PV conservation, the adapted SMC procedure still yields low-variance estimates of the model likelihood. Figure 2.3 shows an empirical comparison between the bootstrap filter and the proposal used in AR across various floats. As expected, the bootstrap filter produces relatively few effective samples due to the high signal-to-noise ratio (SNR) in the data. This translates into unacceptably high variability in the Monte Carlo estimates of the model likelihood. With our proposal, the effective sample size is much higher, and the standard deviation in the estimated log-likelihood is well below 1 for the vast majority of floats, which is acceptable for parameter inference via SMC^2 described next.

2.3.1 Bayesian parameter estimation via SMC^2

In addition to the uncertainty in the state variables, there is also uncertainty in the parameters which govern the dynamics, which we collectively refer to as θ. For ArgoSSM, these parameters are the initial parameters μ_1 and Σ_1; the transition parameters Σ_X, v_0, α, σ^2_{PV}, and Σ_V; the ice avoidance parameters $p^{\text{TPR}}, p^{\text{TNR}}, p^{\text{MAR}}$; and the observation variance Σ^2_Y.

Estimating parameters for a general state-space model is challenging, because the model likelihood, $\mathcal{P}(Y_A|\theta)$, requires evaluating an intractable integral over all state variables. A key feature of SMC is that it provides an unbiased estimate of the likelihood, which can be used for either maximum likelihood estimation or Bayesian inference on θ. See Lopes and Tsay (2011) for an overview of some of these approaches.

Our goal is to infer the posterior distribution of both θ and the state variables for each float. To start, we equip each model parameter with a prior distribution as shown in Table 2.1. Then, to sample the posterior distribution of parameters on each float, we implement the SMC^2 procedure introduced in Chopin et al. (2013). Starting with samples from the prior $\mathcal{P}(\theta)$, SMC^2 sequentially targets intermediate distributions $\mathcal{P}_1(\theta), \ldots, \mathcal{P}_K(\theta)$ via importance sampling such that $\mathcal{P}_K(\theta)$ is the

posterior distribution. Each intermediate distribution is constructed such that the Kullback-Leibler (KL) distance between adjacent distributions is small. To pick the sequence of distributions, we use the likelihood tempering scheme described in Duan and Fulop (2015). Each distribution is defined by exponentiating the likelihood with a value ξ between 0 and 1:

$$\begin{aligned} \mathcal{P}_k(\theta|Y_A) \propto \mathcal{P}(\theta)\mathcal{P}(Y_A|\theta)^{\xi_k}, \text{where} \\ \xi_0 = 0 < \xi_1 < \cdots < \xi_K = 1. \end{aligned} \tag{2.11}$$

When the importance weights drop below a set effective sample size (ESS), we resample and use particle Markov chain Monte Carlo (PMCMC) (Andrieu et al., 2010) to propose new θ to rejuvenate the sample.

2.4 Propagating uncertainty to spatiotemporal estimates

Through probabilistic modeling, ArgoSSM enables principled use of all data captured by Argo floats, even if the locations are missing. In turn, ArgoSSM allows analysis in previously under-explored areas of the ocean. For example, Gray and Riser (2014) remove parts of the Weddell Gyre and other areas from their global estimates of ocean circulation due to lack of data, and both Reeve et al. (2016) and Reeve et al. (2019) found it difficult to resolve seasonal effects in their models due to winter ice cover blocking many of the GPS measurements.

To demonstrate the benefits of ArgoSSM in downstream analysis of ocean properties, we consider two spatiotemporal estimation tasks: (1) temperature and salinity from Argo float data and (2) circulation estimation from Argo trajectory data. In each of these tasks, the goal is to use the irregularly-spaced Argo data to predict at unobserved locations, often on a regular grid. The temperature and salinity estimation problem is scientifically relevant and well-studied; see Roemmich and Gilson (2009) for a standard approach in oceanography and Reeve et al. (2016) for analysis specific to the Weddell Gyre. Approaches for this problem roughly correspond to the methodology for the task of spatial prediction in spatial statistics (Cressie, 1993), with specific application to Argo demonstrated in Kuusela and Stein (2018). For circulation (i.e., velocity) estimation, the Argo float trajectories between consecutive profiles are used to form an estimate of the horizontal ocean circulation at the parking depth (about 1 kilometer under the ocean surface) of the floats. Since the trajectory data is the primary object of interest here, missing location data can be more detrimental, and previous analyses choose to mask or discard missing trajectory data (Gray and Riser, 2014; Reeve et al., 2019). However, with ArgoSSM, velocity estimates are available at all time points and can be directly used since they are state variables.

We describe our model for temperature, though we use similar approaches for salinity and

ocean circulation. We combine data from all floats, using $j = 1, \ldots, J$ to index the float. At indices $n = 1, \ldots, N_j$, and conditional on the locations $X = \{X_{j,n}\}$ taken at times $\{t_{j,n}\}$, the profile temperatures $T_{j,n}$ at a particular depth in the ocean are generated from the following spatial distribution:

$$T_n = \mu(X_{j,n}, t_{j,n}) + \kappa(X_{j,n}, t_{j,n}) + \epsilon_{T,j,n}, \tag{2.12}$$

where μ is a smooth, deterministic mean function; κ is a zero-mean, spatiotemporally correlated random field; and $\epsilon_{T,j,n}$ is measurement error that follows a normal distribution $\mathcal{N}(0, \sigma^2_{\epsilon_T})$. We estimate $\mu(\cdot, \cdot)$ using locally-constant regression with four seasonal dynamic functions to capture seasonality in the estimates and a bandwidth of 250 kilometers. Following this, the covariance structure of the residuals are analyzed assuming a multivariate Gaussian distribution and using a Matérn covariance function in space and time (Guinness and Fuentes, 2016; Kuusela and Stein, 2018). Parameter estimates of the covariance function are obtained using maximum likelihood estimation using a Vecchia's approximation which eases computational burdens for spatial modeling with a large number of observations (Katzfuss et al., 2020; Katzfuss and Guinness, 2021). For velocity estimation, slight adjustments are made to this procedure since fewer floats will be used: a locally-constant estimator with no time dynamics was used for the mean estimation with a bandwidth of 400 kilometers, and the use of time in the covariance structure was removed so that we estimate one time-averaged value at each location.

With missing locations, estimation is done by first drawing the locations X from the posterior distribution of ArgoSSM, estimating parameters governing μ, κ, and ϵ conditional on each location set, and then predicting from the distribution of temperature at an unobserved location conditional on the observed data. By repeating this process for multiple samples of X from ArgoSSM, one can capture the influence of the location uncertainty into the downstream predictions of temperature in a spirit similar to multiple imputation (see Rubin, 1987). We make two simplifying assumptions about the posterior distribution of floats from ArgoSSM: the temperatures (T) are weakly informative about the locations, and the dependence between floats is accounted for in the available data:

$$\mathcal{P}(\{X_{k,1:N_k}, k = 1, \ldots, K\} | Y^{N_k}, T) = \prod_{k=1}^{K} \mathcal{P}(X_{k,1:N_k} | Y) \tag{2.13}$$

In reality, the trajectories of floats in similar spatial areas are positively correlated, which means the assumptions in Equation 2.13 will lead to an over-dispersed joint distribution of locations, which is acceptable for the task at hand. Incorporating dependencies between floats is a potential avenue for future work.

For spatiotemporal estimation tasks, ArgoSSM offers two key advantages over standard interpolation approaches for recovering missing locations. First, as demonstrated in held-out data

experiments in Section 2.5, ArgoSSM more accurately predicts float trajectories. Second, ArgoSSM propagates uncertainty in location data into downstream estimates, capturing a source of variability which would otherwise be ignored. After completing this analysis for multiple sample outputs from ArgoSSM, we can form estimates of the expectation $E(T_{j,n}) = E(E(T_{j,n}|X))$ (using the law of total expectation) that takes into account the uncertainty in the missing locations. More critically, as illustrated by the total law of variation, ArgoSSM allows us to quantify the full uncertainty in downstream estimates:

$$\text{Var}(T_{j,n}) = E(\text{Var}(T_{j,n}|X)) + \text{Var}(E(T_{j,n}|X)). \tag{2.14}$$

With fixed or imputed locations, only the first component of the total uncertainty would be accounted for in final estimates, and it would not be properly averaged over the distribution of X. With ArgoSSM, the second component is accounted for as well. As we will see in Section 2.5.2, these two components of uncertainty have quite different spatial characteristics. Areas with a low conditional variance can nonetheless have a high variance in their estimates because most floats in that area were under ice.

2.5 Analysis of Argo floats in the Southern Ocean

As a case study for our method, we consider 46 floats that traversed a specific area in the Southern Ocean, the Weddell Gyre, from 2002 to 2020 [1]. In this setting, our task is two-fold. First, we aim to reconstruct accurate trajectories of where floats went while under ice. Because it is impossible to verify how well ArgoSSM does on missing data points, we evaluate its performance on held-out data points adjacent to under-ice periods (Section 2.5.1). Second, we illustrate how ArgoSSM is useful for spatiotemporal estimation tasks by uncovering an additional source of uncertainty (Section 2.5.2). This applies both to measurements of temperature and salinity taken by the sensors and velocity estimates V_n calculated as part of the probabilistic model. Throughout this analysis, we draw 200 samples of model parameters θ and use $K = 2500$ particles per parameter sample to estimate the posterior distribution of float trajectories. The code used for performing this analysis is publicly available on Github at https://github.com/dereklhansen/ArgoSSM.

2.5.1 Holdout experiment

To evaluate the empirical predictive performance of ArgoSSM under ice, we observe how well the model predicts held-out GPS points. A single observation is held-out immediately before or after

[1] The float temperature, salinity, and trajectory data were downloaded from the Argo database on July 11, 2020

Model	RMSE	Median
PV Interpolation (Chamberlain et al., 2018)	33.1	13.2
Linear Interpolation (RW)	18.0	11.4
AR	16.1	**8.58**
AR+Ice	15.7	8.92
ArgoSSM (AR+Ice+PV)	**15.4**	8.64

Table 2.2: The observed prediction error of each method in kilometers in held-out data experiments. In the second, the floats had gone under the ice and additional measurements were held out. Note that the RW ("Random Walk") predictions are equivalent to linear interpolation.

the float has been under ice cover for a minimum of 36 days. This led to 197 independent holdout trials from 44 of the 48 floats. On each holdout trial, we compare ArgoSSM to several baseline models. These baseline models include the simple random walk (RW) model defined in Equation 2.1 and the autoregressive (AR) model defined in Equations 2.2 and 2.3. To further evaluate which parts of ArgoSSM (AR+Ice+PV) lead to improvement, we consider a variant that just includes the ice-cover data (AR+Ice). Finally, we compare to the PV interpolation method from Chamberlain et al. (2018).

The results of these two holdout experiments are summarized in Table 2.2. Across each holdout, we compare each model's prediction to the known true GPS measurements and calculate the root mean squared error (RSME) in kilometers. We also consider the median squared error, which suppresses the impact of extreme values. The AR and ArgoSSM outperform the baseline models of linear interpolation and PV interpolation in both measures. While incorporating ice-cover and PV into the model led to a slightly higher median RMSE than the baseline AR model, mean RMSE is reduced by a much larger amount, suggesting that ArgoSSM is particularly useful in reducing held-out measurements with larger sources of error.

Figure 2.4 provides a visual comparison of the performance of each model on float 5901717 [2], which was also investigated in Chamberlain et al. (2018). The random walk (RW) model, which amount to linear interpolation, performs poorly because it uses no local information about momentum, ice-cover, nor potential vorticity (PV). However, despite utilizing potential vorticity (PV), the PV interpolation method from Chamberlain et al. (2018) also misses the target. We see the autoregressive model (AR) does better than the baselines because it takes into account momentum. Compared to the difference between AR and the baseline models, the overall impact to predictive performance of adding ice cover and PV information is modest, but it makes a noticeable difference in Figure 2.4. Here, the AR model predicts the float swinging out too far. Including either or

[2]With this float and throughout, we refer to floats through their World Meteorological Organization (WMO) number.

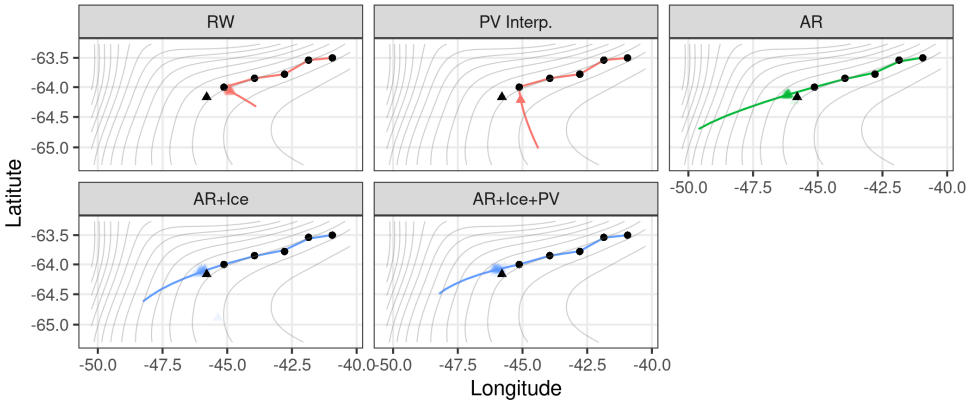


Figure 2.4: Comparison of each different model on Argo float 5901717 in predicting a held-out GPS measurement. The colored lines are the predicted paths from each model. Each black circle is a GPS observation given to the model and the black triangle is the held-out point.

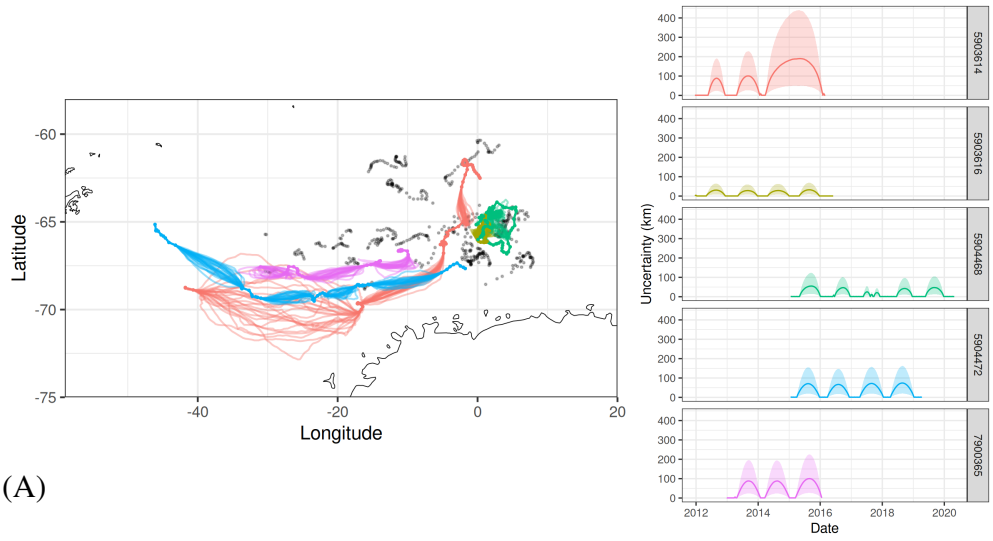
both ice-cover and PV information corrects for this, leading to a more accurate prediction.

2.5.2 Inference of spatiotemporal properties

With the posterior distribution of float trajectories, we show how the uncertainty in float locations shown in Figure 2.5 leads to additional uncertainty in estimated temperature, salinity, and velocity (circulation) fields that would otherwise be ignored by imputation methods. As a challenging example for temperature and salinity, we calculate estimates on August 1, 2015, a date in the middle of winter where many profiles have missing locations.

First, as described in Section 2.3.1, we infer the posterior distribution of model parameters for each float separately. Figure 2.6 shows the marginal posterior distribution of parameters for each float. This shows that the estimated parameter values for α and σ^2_{PV} can vary significantly between floats. For example, the high value of σ^2_{PV} for floats 5905995 and 5905381 indicates that the PV effect was not as strong as float 5901717. This illustrates the benefit of allowing different parameters per-float. Figure 2.6 also shows the inferred true-positive-rate (TPR) and true-negative-rate (TNR) ice detections. Like for the other parameters, these rates can differ significantly from float to float. The majority of floats display a higher TPR than TNR, highlighting the conservative nature of the ice-detection algorithm (Klatt et al., 2007).

Across multiple draws of model parameters, float locations are sampled using the FFBS method from Godsill et al. (2004). Figure 2.5 (A) shows that the estimated locations have varying degrees of uncertainty depending on location, time since the last observation, and float-specific parameters. The regional differences between locations are particularly striking. In the southwestern part of the map, profiles are both more sparse and have more missing locations, which adds significant uncertainty to downstream estimates. In Figure 2.5 (B), we display the uncertainty in kilometers



(A)

(B)

Figure 2.5: A view of selected Argo floats that had at least one observation in 2015, which were used in constructing spatial estimates. In subfigure (A), each point represents a GPS observation, while certain floats' predicted trajectories are plotted in color. In subfigure (B), the median uncertainty is plotted over time, with the shaded area representing the interval between the 5% and 95% quantiles. Float 5903014 had missing locations for nearly two consecutive years (2014-2015), while float 5904472 received GPS positioning twice during winter in late 2017.

as a random variable over time for several floats. We see that this uncertainty differs greatly between floats. This also underscores the need to incorporate downstream measurements, as median location uncertainty is as high as 200km for some floats.

Temperature and salinity For temperature and salinity, we focus on Argo data taken at 150 decibars of pressure, or about 150 meters deep in the ocean. As described in Section 2.4, we repeatedly estimate the temperature field conditional on samples of locations from the posterior distribution of ArgoSSM. Figure 2.7 shows that estimated temperatures do not change much between samples in areas that have many observations. However, in certain regions, there are noticeable differences between samples that come solely from the uncertainty in locations. Figure 2.8 shows the breakdown in temperature uncertainty using 20 samples from ArgoSSM. With one sample of locations, the conditional standard deviation of predicted temperatures depends on the estimated covariances and the proximity of nearby points. Figure 2.8 (A) shows an estimate of the mean conditional standard deviation across the 20 location samples, which estimates the first quantity in Equation 2.14. In contrast, Figure 2.8 (B) shows the standard deviation of temperature estimates between samples, corresponding to the second quantity in Equation 2.14. Most areas in this plot have relatively low standard deviation compared to those in Figure 2.8 (A). However, there is a

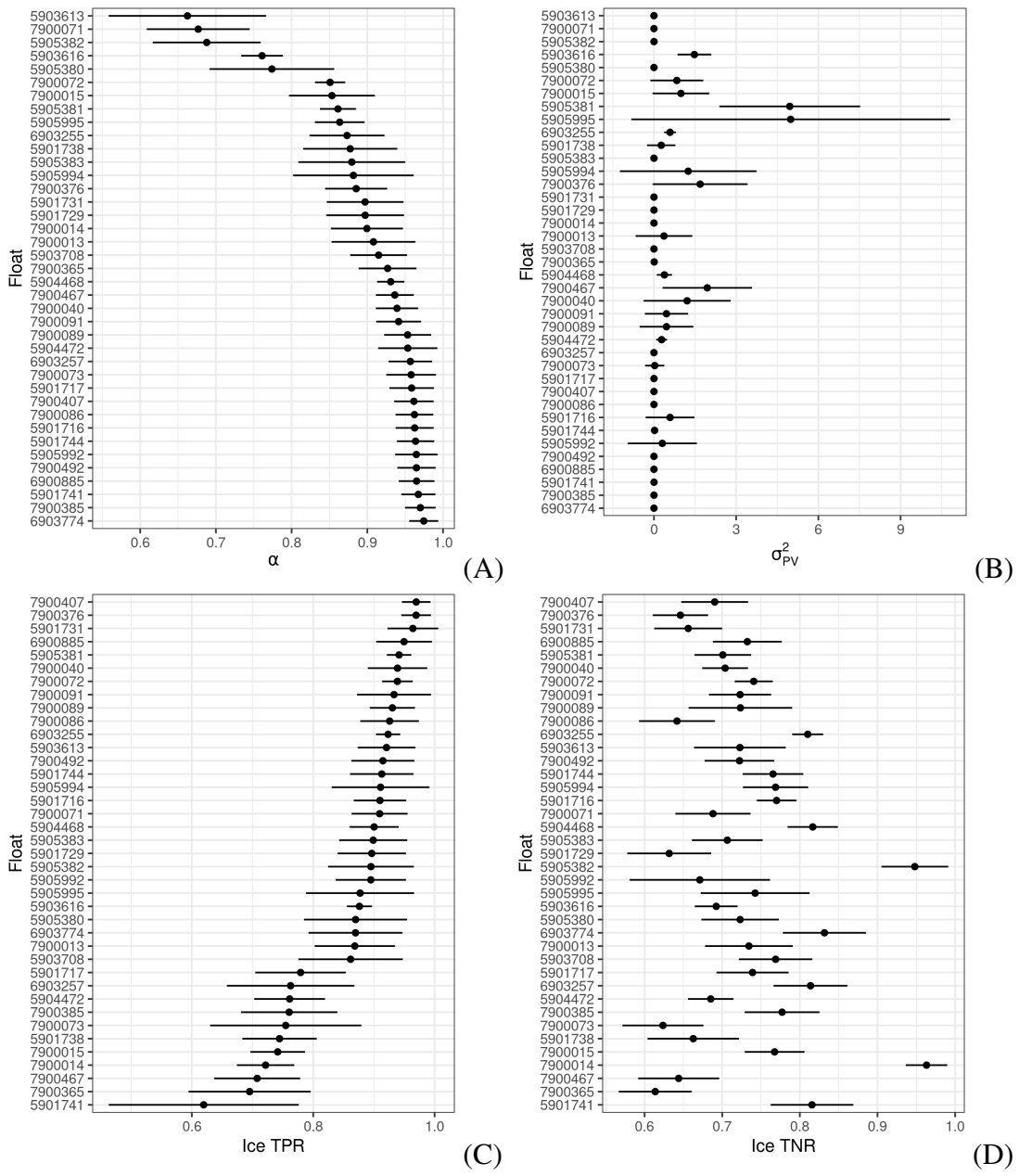


Figure 2.6: Posterior distributions for select parameters across the floats used for spatiotemporal function estimation. The black dot indicates the posterior mean while the black line indicates one standard deviation. The selected parameters are: (A) the autoregressive coefficient α; (B) the potential vorticity variance σ^2_{PV}; (C) the true positive rate (TPR) for the ice-detection algorithm; and (D) the true negative rate (TNR) for the ice detection algorithm.

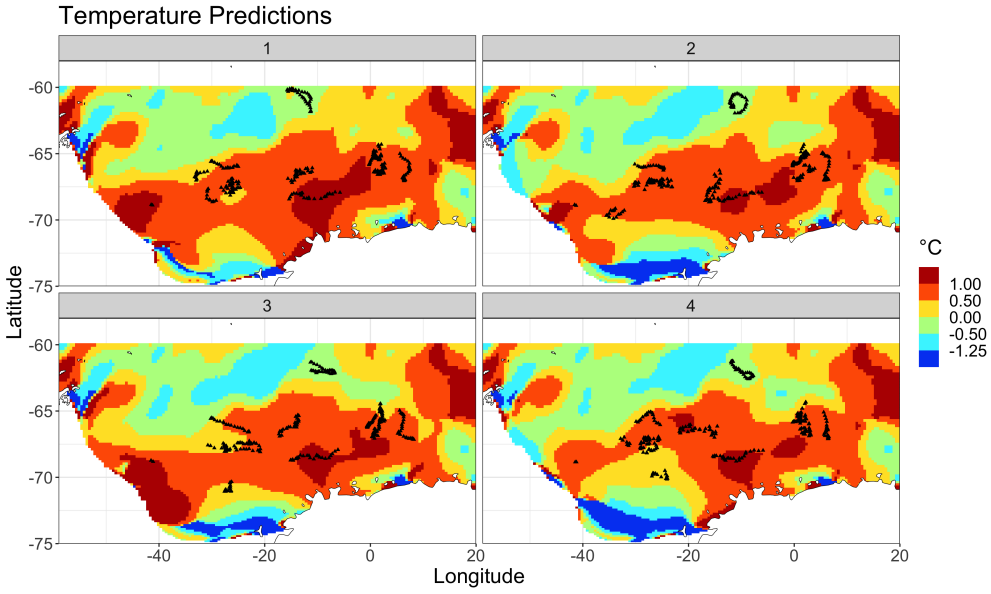


Figure 2.7: Mean temperature estimates on August 1, 2015, taken on four different samples of locations. The black dots show the ArgoSSM sample of missing locations that was used to construct the estimate. Estimates were constructed for a pressure of 150 decibars, corresponding to about 150 meters deep in the ocean.

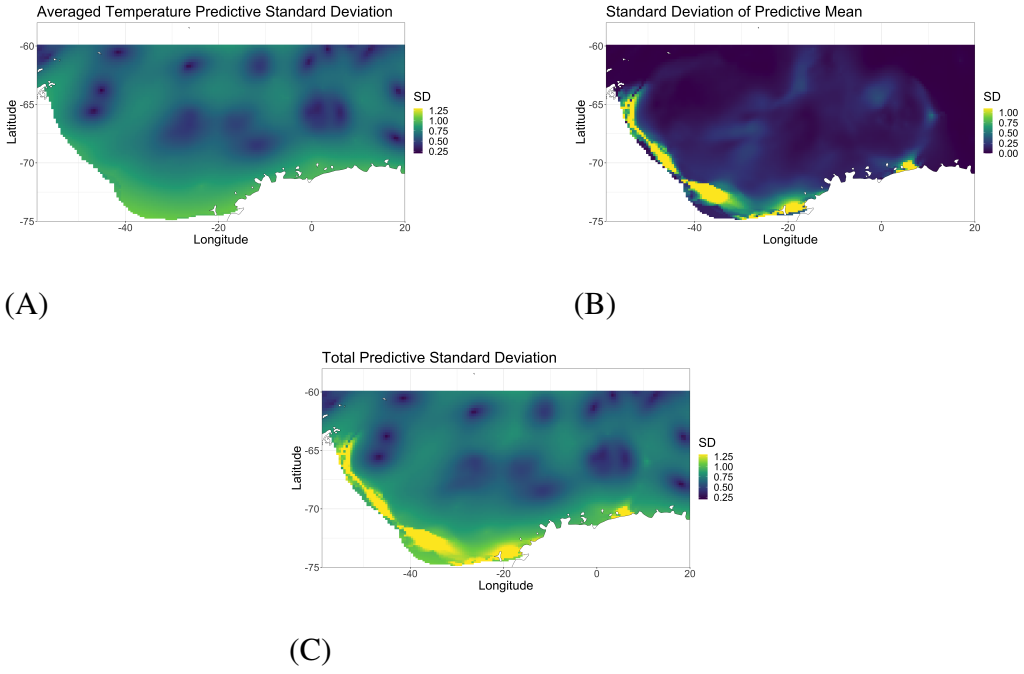


Figure 2.8: Spatial uncertainty in temperature estimated on August 1, 2015. (A) Average standard deviation conditional on locations. (B) The standard deviation of mean temperature estimates between samples of locations. (C) Total standard deviation accounting for both (A) and (B).

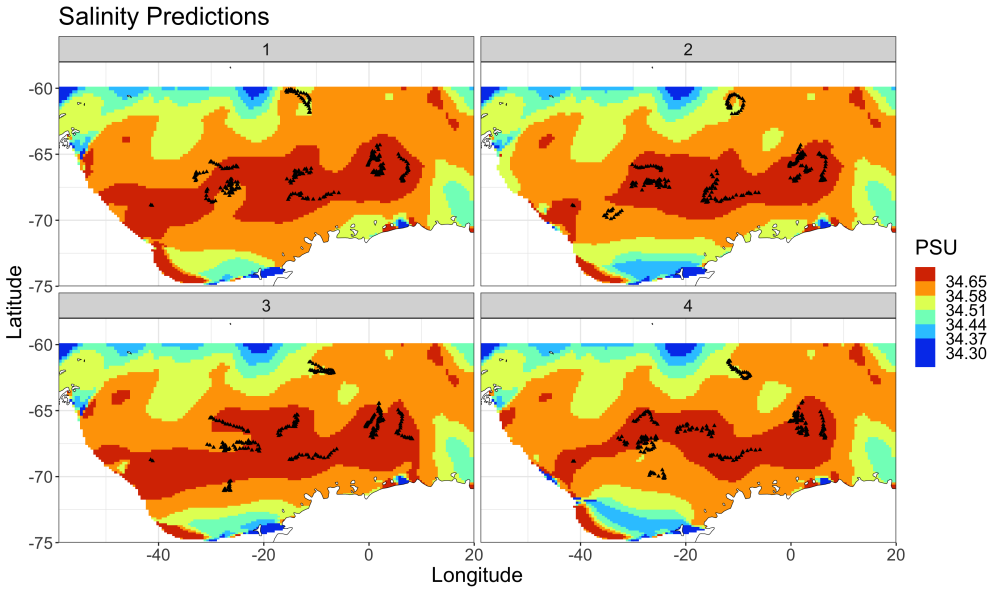


Figure 2.9: Mean salinity estimates in practical salinity units (PSU) on August 1, 2015 taken on four different samples of locations. The black dots show the ArgoSSM sample of missing locations that was used to construct the estimate.

striking amount of uncertainty in the southern edges of the map, where there are few observations. This uncertainty is large enough to significantly contribute to the total standard deviation shown in Figure 2.8 (C). Without ArgoSSM, this contribution would be ignored.

Returning briefly to Figure 2.8 (A), we also see that the areas with less predictive variability (near observed float locations) are more muted and indistinct when there is more location uncertainty for nearby floats. That is, this component of the variance results from properly averaging the spatially-predicted uncertainty across the distribution of missing locations. As a result, using one fixed set of locations for Figure 2.8 (A) would lead to overconfident predictions near the fixed locations and underconfident predictions near other plausible float locations. With ArgoSSM, improved estimates of Figure 2.8 (A) and Figure 2.8 (B) comprehensively characterize the variability of the estimated temperature field.

We plot the estimated salinity fields under four different samples from ArgoSSM in Figure 2.9. As with the temperature estimates, the location uncertainty has some effect on the estimated salinity field, especially near missing locations. In Figure 2.10 (A) and Figure 2.10 (B), we plot the respective plots of the variability in salinity suggested by Equation 2.14. In general, we see similar results: Figure 2.10 (A) properly averages the prediction error over the distribution of missing locations, smoothed over areas with uncertainty in float locations, and Figure 2.10 (B) represents uncertainty added based on the uncertainty in locations, which primarily contributes when there

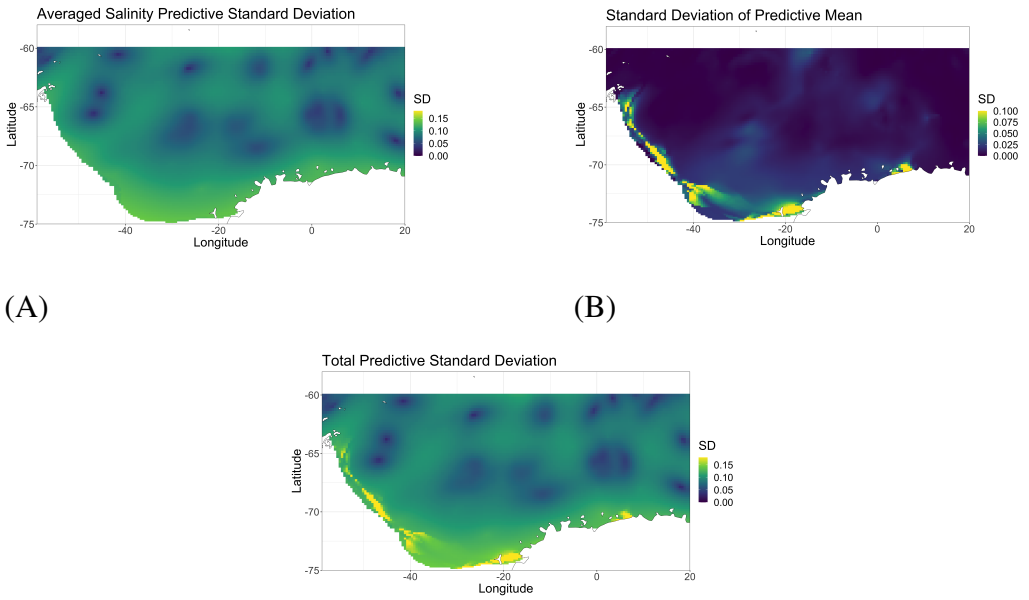
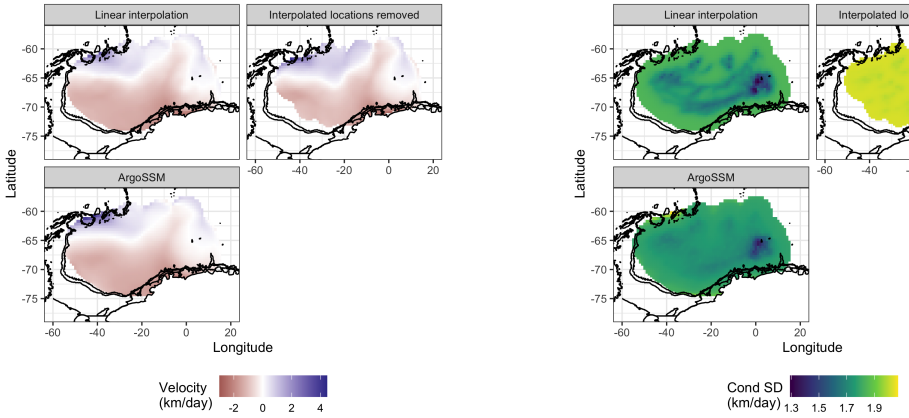


Figure 2.10: Spatial uncertainty in salinity estimated on August 1, 2015. (A) shows the average standard deviation conditional on locations. (B) shows the standard deviation of mean temperature estimates between samples of locations. (C) shows the total standard deviation accounting for both (A) and (B).

are few floats in the bottom-left of the plot. Together, these describe the variance of salinity in a principled manner that incorporates our understanding of the missing locations through ArgoSSM.

Ocean circulation For estimates of ocean circulation, we only use floats with parking depths between 950 and 1050 meters to provide an estimate of the ocean circulation at approximately 1000 meters resulting in 26 floats, with locations plotted in Figure A.1.

Figure 2.11 shows results for the estimated zonal (east-west) velocity used to estimate ocean circulation. As with temperature estimation, the zonal velocity is sensitive to missing locations. In the left of Figure 2.11, we plot the conditional expectation based on the estimated spatial model for zonal velocity, where a negative value represents movement west while a positive value represents movement east. We compare with linearly-interpolated locations, for which the float velocities are assumed constant while under-ice, as well as under-ice locations removed as done in Reeve et al. (2019). While there are only slight differences between the methods in their mean estimates of velocity, the conditional variance estimates show more drastic variation (Figure 2.11 B). Simply removing the missing estimates leads to high uncertainty, while filling in the missing estimates via linear interpolation leads to overconfident final estimates. Also, linear interpolation is more conservative on the basin's boundaries since Argo floats likely swing closer to the boundaries



(A) (B)

Figure 2.11: (A) Conditional expectation and (B) conditional standard deviation of zonal (east-west) velocity under different approaches. "ArgoSSM" refers to our estimates after using ArgoSSM to take into account the variability in the locations.

compared to the linearly interpolated paths. Meanwhile, ArgoSSM provides a joint distribution of the positions and velocities for each float at times when profiles are collected, allowing the use of PV information while fully characterizing the uncertainty in the missing data.

2.6 Discussion

Before the Argo project, the Southern Ocean had been disproportionately under-observed compared to other areas of the world's oceans, especially during the winter (Roemmich and Gilson, 2009). Improving the use of existing and future observations in this region through technological and methodological means at a low cost is an important scientific priority (Vernet et al., 2019; Riser et al., 2018). A critical aspect of this is the use of profiles with missing location data while under ice cover.

ArgoSSM improves upon existing approaches for under-ice location estimation in two ways. First, it more realistically models how floats drift, which is validated by held-out data experiments. Second, through inferring a posterior distribution of locations, ArgoSSM accounts for uncertainty stemming from missing locations. These improvements lend confidence to areas with low estimated uncertainty and signal areas where the variance is too high to draw conclusions. This could focus future data-gathering endeavors to areas of maximum impact.

ArgoSSM will enable scientists to incorporate location or velocity uncertainty into their esti-

mates. ArgoSSM posterior samples can be used to evaluate the sensitivity to location uncertainty, which may vary from task to task (Riser et al., 2018; Chamberlain et al., 2018). Future work could use ArgoSSM estimates to develop an "error-in-variables" approach to specific estimation tasks, removing the need to repeatedly estimate under different samples of locations (Cervone and Pillai, 2015). However, since this literature primarily focuses on independent and identically-distributed errors in the locations, such an approach may require care and methodological development. In this setting, for the same float in the same winter, the missing locations can be strongly correlated. We advocate for the sampling-based approach developed here since it considers this correlation and can be employed naturally by statisticians and scientists alike.

The current version of ArgoSSM may oversimplify the dynamics of float movement. However, ArgoSSM is a flexible framework that can be improved with better physical models and expanded to accommodate more data as it becomes available. For example, one could pair ArgoSSM with ocean circulation estimates instead of potential vorticity, though such estimates may be noisy in the winter due to sparse measurements (Reeve et al., 2019). Also, one could estimate model parameters and float positions using data from multiple floats simultaneously, which might further reduce the size of ArgoSSM's uncertainties. Additional location measurements could be incorporated as they become available, such as RAFOS sound source data (Chamberlain et al., 2018). Through these iterative improvements, ArgoSSM can utilize all available information to best predict float locations, improving existing downstream tasks and enabling new ones.

CHAPTER 3

Scalable Bayesian Inference for Detection and Deblending in Astronomical Images

3.1 Introduction

The forthcoming generation of astronomical surveys will peer deeper into space, revealing many more astronomical light sources than their predecessors. Because of the greater density of light sources in these surveys' images, many more light sources will visually overlap. Visually overlapping light sources, called "blends", are expected to make up 62% of the galaxies imaged by the upcoming Legacy Survey of Space and Time Sanchez et al. (2021). Blends are challenging for traditional (non-probabilistic) astronomical image processing pipelines because they introduce ambiguity into the interpretation of the image data.

We present a new probabilistic method for detecting, deblending, and cataloging astronomical objects called the Bayesian Light Source Separator (BLISS). BLISS is based on deep generative models, which embed neural networks within a Bayesian model and use deep learning to facilitate posterior inference. BLISS generalizes StarNet (Liu et al., 2021), which can only analyze images of starfields.

In the BLISS statistical model (Section 3.2), the latent space is interpretable: one random variable encodes the number of stars and galaxies imaged, a random vector encodes the locations and fluxes of these astronomical objects, and another random vector encodes the galaxy morphologies. Conditional on these random variables, the data (i.e., the pixel intensities in a collection of astronomical images) are modeled as Poisson or Gaussian. Owing to this Bayesian formulation, BLISS requires no special logic to analyze blended galaxies.

For posterior inference (Section 3.3), BLISS uses a new form of variational inference based on stochastic optimization, deep neural networks, and the forward Kullback-Leibler divergence. This new methodology is known as "Forward Amortized Variational Inference" (FAVI) Ambrogioni et al. (2019). In FAVI, a deep encoder network is trained on data simulated according to the generative model to solve the inverse problem: predicting the latent variables that generated a

particular synthetic astronomical image. FAVI has scaling advantages over Markov chain Monte Carlo and achieves improved fidelity of the posterior approximation compared with traditional variational inference in our application.

Algorithmically, inferences in BLISS are produced by a sequence of three deep convolutional encoder networks, each conditioned on the output of the earlier network (Section 3.3.1). The first encoder performs detection, estimating the number of light sources in particular regions. Conditional on samples from the first encoder, the second encoder probabilistically classifies each sampled source as a star or a galaxy. Conditional on sampling a galaxy by the second encoder, a third encoder network estimates the shape/morphology. No restrictive assumptions are made about the factorization of the posterior approximation, as is common in more traditional approaches to variational inference.

The BLISS inference routine is fast, requiring a single forward pass of the encoder networks on a GPU once the encoder networks are trained. BLISS can perform fully Bayesian inference on megapixel images in seconds, and produces more accurate catalogs than traditional methods do (Section 3.4). BLISS is highly extensible, and has the potential to directly answer downstream scientific questions in addition to producing probabilistic catalogs (Section 3.5).

3.2 The Statistical Model

Our generative model consists of two parts: the prior distribution over all possible astronomical catalogs and the likelihood of an image given a particular catalog.

3.2.1 Prior

Let $\mathcal{Z}$ be the collection of all possible catalogs, and let $z \in \mathcal{Z}$ be a particular realization. Our prior over $\mathcal{Z}$ is a marked spatial Poisson process. Light sources arrive according to a homogeneous Poisson process with rate μ, which is set based on prior knowledge. In other words, for a given image of size $H \times W$, the number of sources S in this image follows the Poisson distribution:

$$S \sim \text{Poisson}(\mu HW). \tag{3.1}$$

The locations of each of the sources $s = 1, \ldots, S$ are uniform in the image:

$$\ell_s \mid S \sim \text{Uniform}([0, H] \times [0, W]). \tag{3.2}$$

Each source s is either a star or galaxy:

$$a_s \mid S \sim \text{Bernoulli}(\xi), \tag{3.3}$$

where ξ is the proportion of imaged sources that are expected to be stars according to prior knowledge.

Given the point spread function (PSF) at a particular location in the image, a star's appearance is fully characterized by its flux f_s, which follows a truncated power law distribution:

$$f_{s,1} \mid S, a_s = 1 \sim \text{Pareto}(f_{min}, \alpha). \tag{3.4}$$

Unlike stars, the shape of galaxies can vary greatly within an image. To flexibly model this variety of shapes, we use a low-dimensional embedding following an uninformative prior to encode a galaxy's flux and shape:

$$v_s \mid S, a_s = 0 \sim \text{Normal}(0, I_{D\times D}), \tag{3.5}$$

where D is a user-defined embedding dimension.

3.2.2 Likelihood

Let $z_s = \{\ell_s, a_s, f_s, v_s\}$ denote the latent variables describing light source s. Let $z = \{S, \{z_s\}_{s=1}^S\}$ be a catalog sampled from the prior. Radiation from the light sources in catalog z is recorded as the observed photoelectron count x_n at each pixel n of the astronomical image. The photoelectron count x_n follows a Poisson distribution with rate $\lambda_n(z)+\gamma_n$, where $\lambda_n(z)$ is a deterministic function of the catalog and γ_n is background intensity. In practice, since the number of arrivals is large, we use a normal approximation to the Poisson distribution:

$$x_n \mid z \sim \text{Normal}(\lambda_n(z) + \gamma_n, \lambda_n(z) + \gamma_n). \tag{3.6}$$

The contribution of light source s to the intensity of pixel n, denoted $\lambda_n(z_s)$, depends on whether source s is a star or a galaxy. If source s is a star, the PSF and flux give its intensity at pixel n. If source s is a galaxy, then its contribution to pixel n comes from a decoder neural network $g_\theta(z_s)$, which is trained according to the procedure in subsection 3.3.2.

3.3 Variational Inference

We infer the posterior distribution of the catalog, $p(z \mid x)$, by using variational inference (VI), which allows for computationally efficient approximate inference (Blei et al., 2017; Zhang et al., 2018). Rather than drawing samples like MCMC, VI turns the problem of posterior inference into a numerical optimization problem. From a family of tractable distributions q_ϕ, parameterized by $\phi \in \Phi$, VI aims to find the approximating distribution q_{ϕ^*} that minimizes a divergence metric to the posterior distribution.

The generative model defined in Section 3.2 is *transdimensional*, because the number of light sources can vary significantly for a given image size. Inference on transdimensional probability distributions can be challenging. To make inference more tractable, we divide the image into sub-images called "tiles" (Liu et al., 2021), indexed by $t = 1, \ldots, T$. The number of objects that appear in each tile t, denoted a_t, are independent, as well as the latent variables associated with each tile, z_t. This leads to a variational distribution that factorizes across tiles:

$$\begin{aligned} q(\{a_t, z_t\}) &= \prod_{t=1}^{T} q_t(a_t, z_t) \\ q_t(a_t, z_t) &= (\alpha_t q_t(z_t))^{a_t} (1 - \alpha_t)^{1 - a_t} \\ q_t(z_t) &= q_t^\ell(\ell_t) q_t^b(b_t|\ell_t) q_t^f(f_t|\ell_t, b_t)^{b_t} q_t^v(v_t|\ell_t, b_t)^{1-b_t} \end{aligned} \tag{3.7}$$

For tractability, the distribution of a_t is truncated so that at most one object appears with probability α_t. Given the object appears, i.e. $a_t = 1$, its location within the tile ℓ_t follows a truncated normal distribution with mean μ_t and diagonal covariance Σ_t. The object type b_t follows a Bernoulli distribution and the shape variables f_t, v_t are normal.

3.3.1 Amortization and model architecture

With traditional VI, fitting a variational distribution would require running a computationally intensive iterative optimization procedure once for each tile. Instead, BLISS utilizes *amortized* variational inference (Kingma and Welling, 2014; Zhang et al., 2018). For each tile $t = 1, \ldots, T$, we let x_t denote the corresponding *padded tile*: the 52×52-pixel cropped sub-image centered around the 4×4-pixel tile with index t. In our amortized inference procedure, a sequence of encoder neural networks is trained to transform each padded tile x_t into distributional parameters for the latent variables a_t, z_t on that tile. The padded tile x_t is first fed to the *detection encoder*, which infers whether a source was generated (a_t), the location of that source ℓ_t, and the flux of that source

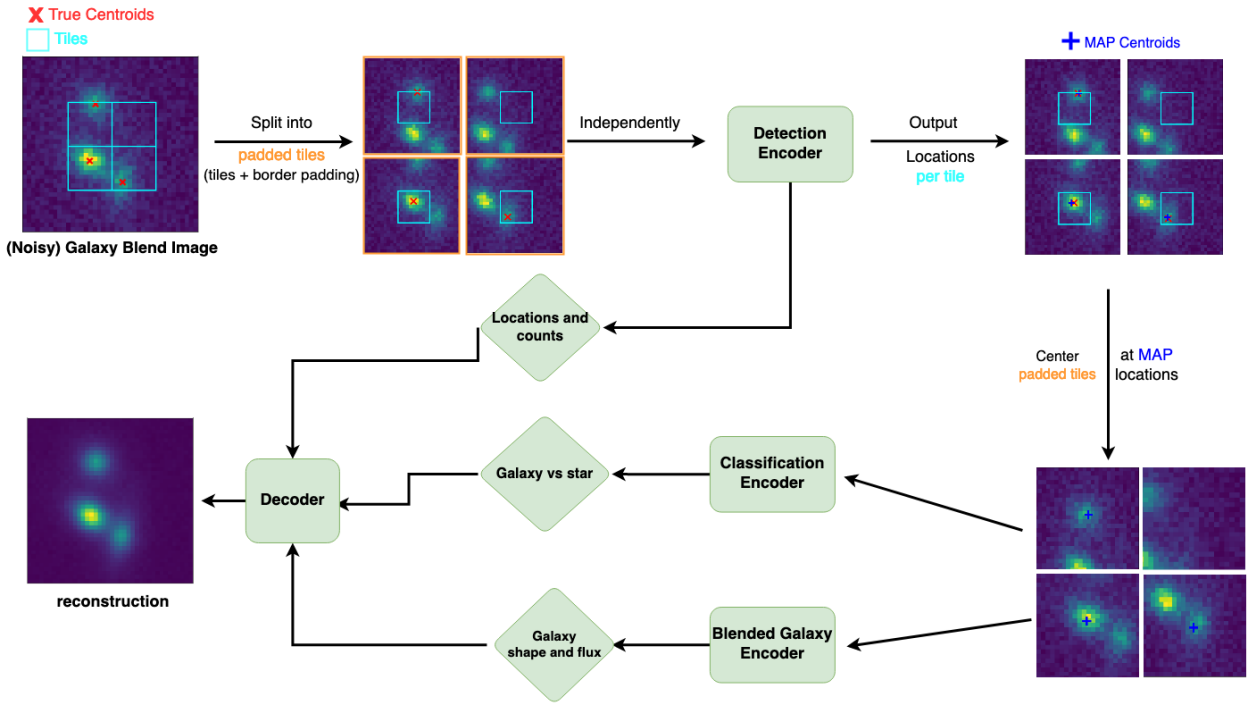


Figure 3.1: The BLISS encoder sequence.

if it is a star f_t:

$$\begin{aligned} a_t &\sim \text{Bernoulli}(\alpha_t) \\ \ell_t &\sim q_t^\ell \stackrel{D}{=} \mathcal{N}(\mu_t^\ell, \Sigma_t^\ell) \\ \alpha_t, \mu_t^\ell, \Sigma_t^\ell, \mu_t^f, \Sigma_t^f &= g_{\phi,\text{detect}}(x_t). \end{aligned} \tag{3.8}$$

After sampling the location, each padded tile is centered around that location and fed to the *classification encoder*, which infers whether the light source is a star or a galaxy.

$$\begin{aligned} b_t &\sim q_t^b \stackrel{D}{=} \text{Bernoulli}(\beta_t) \\ \beta_t &= g_{\phi,\text{class}}(x_t, \ell_t). \end{aligned} \tag{3.9}$$

Finally, fluxes $f_{i,t}$ are sampled from the distribution parameters if the source is a star:

$$\log f_t \sim q_t^f \stackrel{D}{=} \mathcal{N}(\mu_t^f, \Sigma_t^f) \tag{3.10}$$

and galaxy morphology parameters are sampled from the *galaxy encoder*:

$$\begin{aligned} v_t &\sim q_t^v \stackrel{D}{=} \mathcal{N}(\mu_t^v, \Sigma_t^v) \\ \mu_t^v, \Sigma_{i,t}^v &= g_{\phi,\text{galaxy}}(x_t, \ell_t). \end{aligned} \tag{3.11}$$

Further architectural details on each of the encoder networks are available in Appendix B.1.

Amortized variational inference enables fast inference on unseen data. Because training the inference networks happens only once, the average cost of training per-data point is reduced as more data is processed. Also, amortized inference allows for stochastic optimization with subsamples of the image data; during each optimization iteration, it is far more efficient to load a small part of the image and to compute gradients with respect to it than do so for the entire image.

3.3.2 Training procedure

BLISS has two training stages. First, we learn a tractable generative model of single galaxy shapes by fitting a variational autoencoder (VAE) (Kingma and Welling, 2014) to simulations from GalSim (Rowe et al., 2015). Second, we train each of the aforementioned encoder networks (i.e., the location encoder, the classification encoder, and the galaxy encoder) on simulated data sampled from the generative model.

Galaxy VAE We use a variational autoencoder to learn a low-dimensional representation $v \in \mathbb{R}^D$ of centered galaxies. To generate centered galaxies for training, we place a prior on the flux, ellipticity, and size of the bulge and disk components as well as the angle of rotation. These parameters are rendered into single, centered galaxies using the GalSim simulation package. Collectively, this defines the empirical distribution $p(x)$ of centered galaxy images. Separate encoder g_ϕ and decoder f_θ networks are trained to minimize the evidence lower bound (ELBO):

$$
\begin{aligned}
\mathcal{L}(\theta, \phi) &= \mathbb{E}_{p(x)} \mathbb{E}_{q_t^v(v)} \left(\log p_\theta(x|v) + \log p(v) - \log q(v|x) \right) \\
p_\theta(x|v) &\overset{D}{=} \mathcal{N}(\mu^x, \Sigma^x) \\
\mu^X, \Sigma^x &= f_\theta(v) \\
q_\phi(v|x) &\overset{D}{=} \mathcal{N}(\mu^v, \Sigma^v) \\
\mu^V, \Sigma^v &= g_\phi(x).
\end{aligned}
\tag{3.12}
$$

The specific architectures for f_θ and g_θ are described in Figure B.3 in Appendix B.1. After training, the learned decoder network f_θ is subsequently included as a component of our overall generative model and used for training the BLISS galaxy encoder network. To account for the presence of other objects, the galaxy encoder is re-trained on centered images of galaxies from the generative model (Section 3.2) while holding the learned decoder f_θ fixed.

Encoder networks The location and classification encoders are trained using forward amortized variational inference (FAVI) (Ambrogioni et al., 2019). FAVI uses the expected forward KL diver-

gence as its training objective:

$$\phi^\star \equiv \text{argmin}_\phi \mathbb{E}_{p(x,a,z)} \left(\text{KL} \left(p(a,z|x) || q_\phi(a,z|x) \right) \right). \tag{3.13}$$

This is equivalent to maximizing the expected log-density of the variational distribution over full samples from the generative model:

$$\phi^\star = \text{argmax}_\phi \mathbb{E}_{p(x,a,z)} \log q_\phi(a,z|x) = \text{argmax}_\phi L(\phi). \tag{3.14}$$

The FAVI objective has several advantages. First, it leads to a better optimization path than the traditional VI objective. An unbiased estimate of the gradient of Equation 3.14 can be obtained by drawing (a, z, x) from the generative model described in Section 3.2 and then calculating the gradient of $q_\phi(a, z|x)$. This lets us avoid using the high-variance REINFORCE gradient estimator (Liu et al., 2021). Second, the learned distribution from FAVI tends to be overdispersed. This is typically more desirable than being underdispersed, which is a common problem when optimizing the ELBO (Blei et al., 2017; Liu et al., 2021). Third, training with FAVI leads to correct estimates of the marginal posterior distribution of latent variables, implicitly integrating over nuisance parameters such as the latent properties of light sources below the detection threshold (Ambrogioni et al., 2019).

3.4 Experiments

To illustrate the performance of BLISS, we run the trained encoder network on a 1489×2048 frame (run 94, camcol 1, field 12) from stripe 82 of the Sloan Digital Sky Survey (SDSS). After training for 5.5 hours with synthetic data (a one-time upfront training cost), BLISS inferred a probabilistic catalog for this SDSS frame in just 10.5 seconds.

Although BLISS is probabilistic and outputs a distribution of catalogs, we use the mode of the variational distribution as a point estimate for comparison to non-probabilistic catalogs. We let the coadd catalog from SDSS (henceforth, COADD) for this frame serve as a proxy for ground truth. COADD based its estimates on all the filter bands of numerous frames, whereas BLISS used just the r-band of one frame; we expected COADD to serve as reasonable, though imperfect, proxy for the ground truth in our benchmarks. This approach to benchmarking was developed in Regier et al. (2019), where it is described in great depth. We compare BLISS with PHOTO Lupton et al. (2005), which utilizes the same SDSS frame as BLISS, but has access to all five filter bands.

Table 3.1 compares the detection accuracy of BLISS and PHOTO. To qualify as a match, a source must be within one pixel of the other in L^∞ distance. Overall, BLISS detects 571 out of the

		BLISS		PHOTO	
Mag	Ground Truth	TP	FP	TP	FP
17 - 18	31	31	1	28	0
18 - 19	39	38	2	37	2
19 - 20	64	59	9	55	12
20 - 21	117	111	20	104	20
21 - 22	232	187	45	185	44
22 - 23	386	126	91	117	47
Overall	889	571	175	545	130

Table 3.1: Catalog comparison of BLISS to PHOTO, treating COADD as ground truth. Each row is a particular bin of galaxies based on their magnitude according to COADD. "TP" refers to true positives and "FP" refers to false positives.

	BLISS			PHOTO		
Mag	Tot	Gal	Star	Tot	Gal	Star
17 - 18	0.97	1.00	0.96	0.96	1.00	0.95
18 - 19	1.00	1.00	1.00	1.00	1.00	1.00
19 - 20	0.98	1.00	0.96	1.00	1.00	1.00
20 - 21	0.94	0.91	0.98	0.90	0.86	0.98
21 - 22	0.81	0.86	0.73	0.83	0.81	0.87
22 - 23	0.63	0.74	0.50	0.66	0.70	0.63
Overall	0.84	0.87	0.80	0.85	0.84	0.86

Table 3.2: Accuracy of classifications made by BLISS and PHOTO.

889 total sources in the SDSS frame. The vast majority of sources that BLISS did not find were faint (magnitude ≥ 21), making them harder to detect. Of the 7 sources greater than 20 magnitude that were not matched by BLISS, 4 sources were missed due to errors in COADD, 2 sources had unusual shapes, and 1 source was in a particularly difficult blend with many other sources. This was determined by manually checking discrepancies with the more recent DECaLS survey Dey et al. (2019). Similarly, of the 12 sources that BLISS detected that were unmatched in COADD, 10 were due to errors in COADD, and 2 were mistakes by BLISS. These latter two mistakes were both cases where a source's center straddled adjacent tiles, leading to a prediction in both. These types of errors are quite rare, and could be fixed by a post-processing step that conditions on neighboring tiles, or by a more complex variational approximation.

Table 3.2 compares the source-type classification accuracy of BLISS and PHOTO. Overall, BLISS correctly classifies most sources. When sources are dimmer and more ambiguous, BLISS

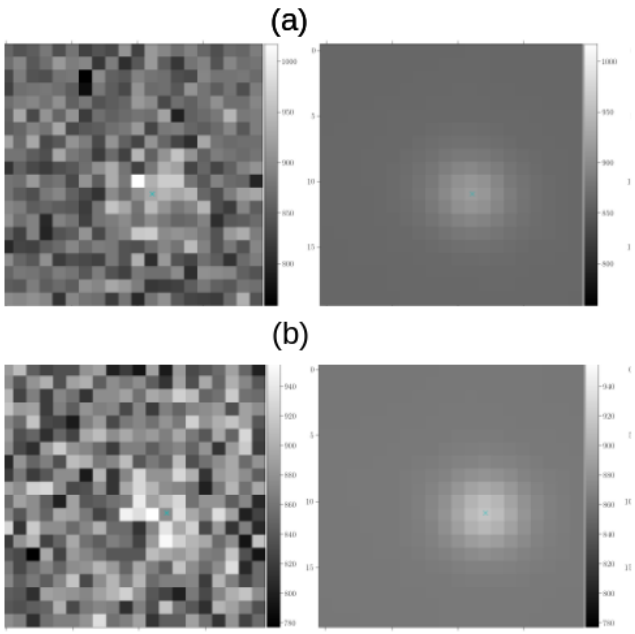


Figure 3.2: Images of BLISS detections near the 50% detection threshold in SDSS images (left) and their reconstructions (right). Example (a) was detected in COADD. (b) was not present in COADD but found in the DECaLS catalog.

sensibly defers to the prior, labeling most of them as galaxies. In SDSS images without much blending, BLISS identifies more sources than PHOTO, despite having less access to less data (only one of five bands), with the advantage of producing calibrated uncertainties for every prediction. While BLISS appears to produce more "false positives" than PHOTO, these all arise in the dimmest magnitude bin $22 - 23$ where more ambiguity is present. In this scenario, mistakes in COADD (our imperfect proxy of ground truth) likely favor PHOTO as both COADD and PHOTO are both produced by the same software pipeline. We found several cases where BLISS detected a dim object that was not present in COADD, but confirmed its existence in the DECaLS survey. Figure 3.2b is one example of a COADD mistake. More importantly, BLISS correctly identifies the ambiguity in these situations, as most of the "false positives" BLISS finds have probabilities close to the detection threshold. Setting this detection threshold allows practitioners to decide the level of certainty they require in sources for use in downstream tasks.

3.5 Conclusion

BLISS is a fundamentally different approach to interpreting astronomical images. It uses deep learning to enable scalable and accurate Bayesian inference. BLISS performs well at detecting

and deblending light sources in SDSS images. BLISS is also highly extensible in that its inference routine requires little modification if the underlying statistical model is revised or extended. The software used to run these experiments is available from https://github.com/prob-ml/bliss.

CHAPTER 4

Learning Physical Models that Respect Conservation Laws

4.1 Introduction

Conservation laws are ubiquitous in science and engineering, where they are used to model physical phenomena ranging from heat transfer to wave propagation to fluid flow dynamics, and beyond. These laws can be expressed in two complementary ways: in a differential form; or in an integral form. They are most commonly expressed as partial differential equations (PDEs) in a differential form,

$$u_t + \nabla \cdot F(u) = 0,$$

for an unknown u and a nonlinear flux function $F(u)$. This differential form of the conservation law can be integrated over a spatial domain Ω using the divergence theorem to result in an integral form of the conservation law,

$$U_t = -\int_\Gamma F(u) \cdot n d\Gamma,$$

where $U = \int_\Omega u(t,x) d\Omega$ and Γ denotes the boundary of Ω. As examples: in the case of heat transfer, u denotes the temperature, and U the conserved energy of system; and in the case of porous media flow, u denotes the density, and U the conserved mass of the porous media.

Global conservation states that the rate of change in time of the conserved quantity U over a domain Ω is given by the flux across the boundary Γ of the domain. Local conservation arises naturally in the numerical solution of PDEs. Traditional numerical methods (e.g., finite differences, finite elements, and finite volume methods) have been developed to solve PDEs numerically, with finite volume methods being designed for (and being particularly well-suited for) conservation laws (LeVeque, 1990, 2002, 2007). Finite volume methods divide the domain Ω into control volumes and apply the integral form locally. They enforce that the time derivative of the cell-averaged unknown is equal to the difference of the in-flux and out-flux over the control volume. (This

local conservation—so-called since the out-flux that leaves one cell equals the in-flux that enters a neighboring cell—can be used to guarantee global conservation over the whole domain.) This numerical approach should be contrasted with finite difference methods, which use the differential form directly, and which are thus not guaranteed to satisfy the conservation condition.

This discussion is relevant for machine learning (ML) since there has been an interest recently in Scientific ML (SciML) in incorporating the physical knowledge or physical constraints into neural network (NN) training. A popular example of this is the so-called Physics-Informed Neural Networks (PINNs) (Raissi et al., 2019). This approach uses a NN to approximate the PDE solution by incorporating the differential form of the PDE into the loss function, basically as a soft constraint or regularization term. Other data-driven approaches, including DeepONet (Lu et al., 2021) and Neural Operators (NOs) (Li et al., 2021a; Gupta et al., 2021), train on simulations and aim to learn the underlying function map from initial conditions or PDE coefficients to the solution. Other methods such as Physics-Informed Neural Operator (PINO) attempt to make the data-driven Fourier Neural Operator (FNO) "physics-informed," again by adding the differential form into the supervised loss function as a soft constraint regularization term (Li et al., 2021b; Goswami et al., 2022).

Challenges and limitations for SciML of this soft constraint approach on model training were recently identified (Krishnapriyan et al., 2021; Edwards, 2022). The basic issue is that, unlike numerical finite volume methods, these ML and SciML methods do *not* guarantee that the physical property of conservation is satisfied. This is a consequence of the fact that the Lagrange dual form of the constrained optimization problem does not in general satisfy the constraint. This results in very weak control on the physical conservation property, resulting in non-physical solutions that violate the governing conservation law.

In this work, we frame the problem of learning physical models that can respect conservation laws via a "finite-volume lens" from scientific computing. This permits us to use the integral form of the governing conservation law to enforce conservation conditions for a range of SciML problems. In particular, for a wide range of initial and boundary conditions, we can express the integral form as a time-varying linear constraint that is compatible with existing ML pipelines. This permits us to propose a two-step framework. In the first step, we use an ML model with a mean and variance estimate to compute a predictive distribution for the solution at specified target points. Possible methods for this step include: classic estimation methods (e.g., Gaussian Processes (Rasmussen and Williams, 2006)); methods designed to exploit the complementary strengths of classical methods and NN methods (e.g., Neural Processes (Kim et al., 2019)); as well as computing ensembles of NN models (to compute empirical estimates of means and variances). In the second step, we apply a discretization of the integral form of the constraint as a Bayesian update in order to enforce the physical conservation constraint on the black-box unconstrained output. We illustrate

our framework, PROBCONSERV, by using an Attentive Neural Process (ANP) (Kim et al., 2019) as the probabilistic deep learning model in the first step paired with a global conservation constraint in the second step. In more detail, the following are our main contributions:

- *Integral form for conservation*. We propose to use the integral form of the governing conservation law via finite volume methods, rather than the commonly used differential form, to enforce conservation subject to a specified noise parameter. Through an ablation study, we show that adding the differential form of the PDE as a soft constraint to the loss function does not enforce conservation in the underlying unconstrained ML model.
- *Strong control on the conservation constraint*. By using the integral form, we are able to enforce conservation via linear probabilistic constraints, which can be made arbitrarily binding or sharp by reducing the variance term σ_G^2. In particular, by adjusting σ_G^2, one can balance satisfying conservation with predictive metrics (e.g., MSE), with PROBCONSERV obtaining exact conservation when $\sigma_G^2 = 0$.
- *Effective for "easy" to "hard" PDEs*. We evaluate on a parametric family of PDEs, which permits us to explore "easy" parameter regimes as well as "medium" and "hard" parameter regimes. We find that our method and the baselines do well for "easy" problems (although baselines sometimes have issues even with "easy" problems, and even for "easy" problems their solutions may not be conservative), but we do seamlessly better as we go to "harder" problems, with a $5\times$ improvement in MSE.
- *Uncertainty Quantification (UQ) and downstream tasks*. We provide theoretical guarantees that PROBCONSERV increases predictive log-likelihood (LL) compared to the original black-box ML model. Empirically, we show that PROBCONSERV consistently improves LL, which takes into account both prediction accuracy and well-calibrated uncertainty. On "hard" problems, this improved control on uncertainty leads to better insights on downstream shock position detection tasks.

There is a large body of related work, too much to summarize here; see Appendix C.1 for a summary.

4.2 A Probabilistic Approach to Conservation Law Enforcement

In this section, we present our framework, PROBCONSERV, for learning physical models that can respect conservation laws. Our approach centers around the following two sources of information: an unconstrained ML algorithm that makes mean and variance predictions; and a conservation constraint (in the form of Equation 4.4 below) that comes from knowledge of the underlying physical

Algorithm 4.1 PROBCONSERV

Input: Constraint matrix G, constraint value b, non-zero noise σ_G and input points $(t_1, x_1), \ldots (t_N, x_N)$
Step 1: Calculate black-box prediction over output grid: $\mu, \Sigma = f_\theta((t_1, x_1), \ldots (t_N, x_N); D)$
Step 2: Calculate $\tilde{\mu}$ and $\tilde{\Sigma}$ according to Equation 4.8.
Output: $\tilde{\mu}, \tilde{\Sigma}$

system. See Algorithm 4.1 for details of our approach. In the first step, we compute a set of mean and variance estimates for the unconstrained model. In the second step, we use those mean and variance estimates to compute an update that respects the conservation law. The update rule has a natural probabilistic interpretation in terms of uncertainty quantification, and it can be used to satisfy the conservation constraint to a user-specified tolerance level. As this tolerance goes to zero, our method gracefully converges to a limiting solution that satisfies conservation exactly (Theorem 4.1).

4.2.1 Integral Form of Conservation Laws as a Linear Constraint

Here, we first derive the integral form of a governing conservation law from the corresponding differential form (a la finite volume methods), and we then show how this integral form can be expressed as a linear constraint (for PDEs with specific initial and boundary conditions, even for certain nonlinear differential PDE operators) for a broad class of real-world problems.

Consider the differential form of the governing equation:

$$\begin{cases} \mathcal{F}u(t,x) = 0, \ x \in \Omega, \\ u(0,x) = h(x), \qquad\qquad , \forall\, t \geq 0, \\ u(t,x) = g(t,x), \ x \in \Gamma, \end{cases} \tag{4.1}$$

where Γ denotes the boundary of the domain Ω, $h(x)$ the initial condition, and $g(t,x)$ the Dirichlet boundary condition. Recently-popular SciML methods, e.g., PINNs (Raissi et al., 2019), PINOs (Li et al., 2021b; Goswami et al., 2022), focus on incorporating this form of the constraint into the NN training procedure. In particular, the differential form of the PDE $\mathcal{F}u(t,x)$ could be added as a soft constraint to the loss function $\mathcal{L}$, as follows:

$$\min_\theta \mathcal{L}(u) + \lambda \|\mathcal{F}u\|,$$

where $\mathcal{L}$ denotes a loss function measuring the error of the NN approximated solution relative to the known initial and boundary conditions (and potentially any observed solution samples), θ denotes the NN parameters, and λ denotes a penalty or regularization parameter.

For conservation laws, the differential form is given as:

$$\mathcal{F}u = u_t + \nabla \cdot F(u), \tag{4.2}$$

for some given nonlinear flux function $F(u)$. The corresponding integral form of a conservation law is given as:

$$\int_\Omega u(t,x)d\Omega = \int_\Omega h(x)d\Omega - \int_0^t \int_\Gamma F(u)\cdot n d\Gamma dt. \tag{4.3}$$

See Appendix C.2 for a derivation.

In one-dimension, the boundary integral of the flux can be computed analytically, as the difference of the flux in and out of the domain:

$$\underbrace{\int_\Omega u(t,x)d\Omega}_{\mathcal{G}u(t,x)} = \underbrace{\int_\Omega h(x)d\Omega + \int_0^t (F_{\text{in}} - F_{\text{out}})dt}_{b(t)}, \tag{4.4}$$

where $\Omega = [x_0, x_N]$, $F_{\text{in}} = F(u,t,x_0)|_{u=g(t,x_0)}$, and $F_{\text{out}} = F(u,t,x_N)|_{u=g(t,x_N)}$. In two and higher dimensions, we do not have an analytic expression, but one can approximate this boundary integral as the sum over the spatial dimensions of the difference of the in and out fluxes on the boundary in that dimension. This methodology is well-developed within finite volume discretization methods, and we leave this extension to future work.

In many applications (including those we consider), by using the prescribed physical boundary condition $u(t,x) = g(t,x)$ for $x \in \Gamma$, it holds that the in and out fluxes on the boundary do *not* depend on u, and instead they only depend on t. This is known as a *boundary flux linearity assumption* since, when it holds, one can use a simple linear constraint to enforce the conservation law. This assumption holds for a broad class of problems—even including nonlinear conservation laws with nonlinear PDE operators $\mathcal{F}$ (See Appendix C.3 for the initial/boundary conditions, exact solutions, exact linear global conservation constraints and Table C.2 for a summary). In these cases, Equation 4.4 results in the following linear constraint equation:

$$\mathcal{G}u(t,x) = \int_\Omega u(t,x)d\Omega = b(t), \tag{4.5}$$

which can be used to enforce global conservation. See Appendix C.4.1 for details on how this integral equation can be discretized into a matrix equation.

In other applications, of course, the flux linearity assumption along the boundary of the domain will not hold. For example, the flux may not be known and/or the boundary condition may depend on $u(t,x)$. In these cases, we will not be able to not apply Equation 4.5 directly. However, nonlinear least squares methods may still be used to enforce the conservation constraint. This

methodology is also well-developed, and we leave this extension to future work.

4.2.2 Step 1: Unconstrained Probability Distribution

In Step 1 of PROBCONSERV, we use a supervised black-box ML model to infer the mean μ and covariance Σ of the unknown function u from observed data D. For example, D can include values of the function u observed at a small set of points. Over a set of N input points $(t_1, x_1), \ldots, (t_N, x_N)$, the probability distribution of $u := [u(t_1, x_1), \ldots u(t_N, x_N)] \in \mathbb{R}^N$ conditioned on data D has mean $\mu := \mathbb{E}(u|D)$ and covariance $\Sigma := \text{Cov}(u|D)$ given by the black-box model f_θ, i.e.,

$$\mu, \Sigma = f_\theta\left((t_1, x_1), \ldots, (t_N, x_N); D\right). \tag{4.6}$$

This framework is general, and there are possible choices for the model in Equation 4.6. Gaussian Processes (Rasmussen and Williams, 2006) are a natural choice, assuming that one has chosen an appropriate mean and kernel function for the specific problem. The ANP model (Kim et al., 2019), which uses a transformer architecture to encode the mean and covariance, is another choice. A third option is to perform repeated runs, e.g., with different initial seeds, of non-probabilistic black-box NN models to compute empirical estimates of mean and variance parameters.

4.2.3 Step 2: Enforcing Conservation Constraint

In Step 2 of PROBCONSERV, we incorporate a discretized and probabilistic form of the constraint given in Equation 4.5:

$$b = Gu + \sigma_G \epsilon, \tag{4.7}$$

where G denotes a matrix approximating the linear operator $\mathcal{G}$ (see Appendix C.4.1), b denotes a vector of observed constraint values, and ϵ denotes a noise term, where each component has unit variance. The parameter $\sigma_G \geq 0$ controls how much the conservation constraint can be violated (see Figure C.5 for details), with $\sigma_G = 0$ enforcing exact adherence. Step 2 outputs the following updated mean $\tilde{\mu}$ and covariance $\tilde{\Sigma}$ that respect conservation, given as:

$$\tilde{\mu} = \mu - \Sigma G^T(\sigma_G^2 I + G\Sigma G^T)^{-1}(G\mu - b), \tag{4.8a}$$

$$\tilde{\Sigma} = \Sigma - \Sigma G^T(\sigma_G^2 I + G\Sigma G^T)^{-1} G\Sigma, \tag{4.8b}$$

where μ and Σ denote the mean and covariance matrix, respectively, from Step 1 (Equation 4.6).

The update rule given in Equation 4.8 can be justified from two complementary perspectives.

From a Bayesian probabilistic perspective, Equation 4.8 is the posterior mean and covariance of the predictive distribution of u after incorporating the information given by the conservation constraint via Equation 4.7. Since both the prior and conservation constraint are Gaussian, Equation 4.8 takes the same form as the observation update step in the Kalman filter, where the Kalman gain is $\Sigma G^T(\sigma_G^2 I + G\Sigma G^T)^{-1}$ (Lopes and Tsay, 2011, Equation 8). From an optimization perspective, Equation 4.8 is the solution to a least-squares problem that places a binding inequality constraint on the conserved quantity Gu (i.e., $\|Gu - b\|_2 \leq c$ for some $c \in (0, \|G\mu - b\|_2)$). See Appendix C.6 for more details on these two complementary perspectives.

We emphasize that, for $\sigma_G > 0$, the final solution does not satisfy $Gu = b$ exactly. Adherence to the constraint can be gracefully controlled by shrinking σ_G. Specifically, if we consider a monotonic decreasing sequence of constraint values $\sigma_{G,n} \downarrow 0$, then the corresponding sequence of posterior means $\tilde{\mu}_n$ is well-behaved, and the limiting solution can be calculated. This is shown in the following theorem.

Theorem 4.1. *Let μ and Σ be the mean and covariance of u obtained at the end of Step 1. Let $\sigma_{G,n} \downarrow 0$ be a monotonic decreasing sequence of constraint values and let $\tilde{\mu}_n$ be the corresponding posterior mean at the end of Step 2 shown in Equation 4.8. Then:*

1. *The sequence $\tilde{\mu}_n$ converges to a limit $\tilde{\mu}^\star$ monotonically; i.e., $\|\tilde{\mu}_n - \tilde{\mu}^\star\|_{\Sigma^{-1}} \downarrow 0$.*
2. *The limiting mean $\tilde{\mu}^\star$ is the solution to a constrained least-squares problem: $\operatorname{argmin}_y \|y - \mu\|_{\Sigma^{-1}}$ subject to $Gy = b$.*
3. *The sequence $G\tilde{\mu}_n$ converges to b in L_2; i.e., $\|G\tilde{\mu}_n - b\|_2 \downarrow 0$.*

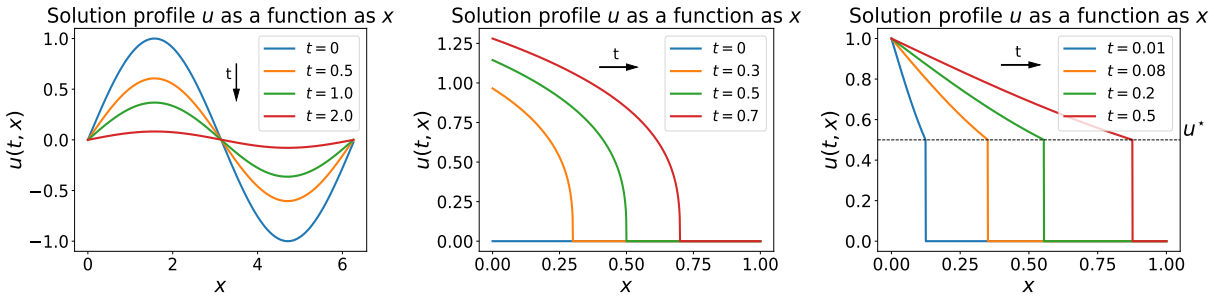
Moreover, if the conservation constraint $Gu = b$ holds exactly for the true solution u, then:

4. *The distance between the true solution u and the posterior mean $\tilde{\mu}_n$ decreases as $\sigma_{G,n} \to 0$, i.e., $\|\tilde{\mu}_n - u\|_{\Sigma^{-1}} \downarrow \|\tilde{\mu}^\star - u\|_{\Sigma^{-1}}$.*
5. *For sufficiently small $\sigma_{G,n}$, the log-likelihood $LL(u; \tilde{\mu}_n, \tilde{\Sigma}_n)$ is greater than $LL(u; \mu, \Sigma)$ and increases as $\sigma_{G,n} \to 0$.*

See section C.7 for a proof of Theorem 4.1. Importantly, Theorem 4.1 holds for any mean and covariance estimates μ, Σ, whether they come from a Gaussian Process, ANP, or repeated runs of a black-box NN. It also shows that we are guaranteed to improve in log-likelihood (LL), which we also verify in the empirical results (see Figure C.5).

We should also emphasize that, in addition to conservation, Equation 4.7 can incorporate other inductive biases, based on knowledge of the underlying PDE. To take but one practically-useful example, one typically desires a solution that is free of artificial high-frequency oscillations. This smoothing can be accomplished by penalizing large absolute values of the second derivative via a second order central finite difference discretization in the matrix $\tilde{G}$ (see Appendix C.4.2).

4.3 Empirical results



(a) easy: Diffusion equation ($k = 1$) (b) medium: PME ($k(u) = u^3$) (c) hard: Stefan (discont. $k(u)$)

Figure 4.1: Illustration of the "easy-to-hard" paradigm for PDEs, for the GPME family of conservation equations: (a) "easy" parabolic smooth (diffusion equation) solutions, with constant parameter $k(u) = k \equiv 1$; (b) "medium" degenerate parabolic PME solutions, with nonlinear monomial coefficient $k(u) = u^m$, with parameter $m = 3$ here; and (c) "hard" hyperbolic-like (degenerate parabolic) sharp solutions (Stefan equation) with nonlinear step-function coefficient $k(u) = \mathbf{1}_{u \geq u^\star}$, where $\mathbf{1}_{\mathcal{E}}$ is an indicator function for event $\mathcal{E}$.

In this section, we provide an empirical evaluation to illustrate the main aspects of our proposed framework PROBCONSERV. We choose the ANP model (Kim et al., 2019) as our black-box, data-driven model in Step 1, and we refer to this instantiation of our framework as PROBCONSERV-ANP.[1] Unless otherwise stated, we use the limiting solution described in Equation 4.8, with $\sigma_G = 0$, so that conservation is enforced exactly through the integral form of the PDE. We organize our empirical results around the following questions:

1. *Integral vs. differential form?*
2. *Strong control on the enforcement of the conservation constraint?*
3. *"Easy" to "hard" PDEs?*
4. *Uncertainty Quantification (UQ) for downstream tasks?*

Generalized Porous Medium Equation. The parametric Generalized Porous Medium Equation (GPME) is a *family* of conservation equations, parameterized by a nonlinear coefficient $k(u)$. It has been used in applications ranging from underground flow transport to nonlinear heat transfer to water desalination and beyond (Vázquez, 2007). The GPME is given as:

$$u_t - \nabla \cdot (k(u) \nabla u) = 0, \tag{4.9}$$

[1]The code is available at https://github.com/amazon-science/probconserv.

where $F(u) = -k(u)\nabla u$ is a nonlinear flux function, and where the parameter $k = k(u)$ can be varied. Even though the GPME is nonlinear in general, for specific initial and boundary conditions, it has closed form self-similar solutions (Vázquez, 2007; Maddix et al., 2018b,a). This enables ease of evaluation by comparing each competing method to ground truth solutions.

By varying the parameter $k(u)$ in the GPME family, one can obtain PDE problems with widely-varying difficulties, from "easy" (where finite element and finite difference methods perform well) to "hard" (where finite volume methods are needed), and exhibiting many of the qualitative properties of smooth/easy parabolic to sharp/hard hyperbolic PDEs. See Figure 4.1 for an illustration. In particular: the Diffusion equation is parabolic, linear and smooth, and represents an "easy" case (Sec. 4.3.1); the Porous Medium Equation (PME) has a solution that becomes sharper (as $m \geq 1$, for $k(u) = u^m$, increases), and represents an "intermediate" or "medium" case (Sec. 4.3.2); and the Stefan equation has a solution that becomes discontinuous, and represents a "hard" case (Sec. 4.3.3).

We consider these three instances of the GPME (Diffusion, PME, Stefan) that represent increasing levels of difficulty. In particular, the challenging Stefan test case illustrates the importance of developing methods that satisfy conservation conditions on "hard" problems, with non-smooth and even discontinuous solutions, as well as for downstream tasks, e.g., the estimation of the shock position over time. This is important, given the well-known inductive bias that many ML methods have toward smooth/continuous behavior.

See Appendix C.8 for more on the GPME; see Appendix C.9 for details on the PROBCONSERV-ANP model schematic (Figure C.4), model training, data generation and the ANP; and see Appendix C.10 for additional empirical results on the GPME and the hyperbolic linear advection equation.

Baselines. We compare our results to the following baselines:

- ANP: Base unconstrained ANP (Kim et al., 2019), trained to minimize the negative evidence lower bound (ELBO):

$$\mathcal{L} = -\mathbb{E}_{D,u\sim p}\mathbb{E}_{z\sim q_\phi} \log p_\theta(u, z|D) - \log q_\phi(z|u, D),$$

  where q_ϕ denotes the variational distribution of the data used for training, and p_θ denotes the generative model. The ANP learns a global latent representation z that captures uncertainty in global parameters, which influences the prediction of the reference solution u. At inference time, the distribution of u given z ($p_\theta(u|z, D)$) outputs a mean and diagonal covariance for Step 1.
- SOFTC-ANP: In this "Physics-Informed" Neural Process ablation, we include a soft con-

strained PDE in the loss function, as is done with PINNs (Raissi et al., 2019), to obtain:

$$\mathcal{L} + \lambda \mathbb{E}_{z \sim q_\phi} \|\mathcal{F} \mu_z\|_2^2,$$

where $\mathcal{F}$ denotes the underlying PDE differential form in Equation 4.1, μ_z denotes the output mean of the ANP, and λ denotes a hyperparameter controlling the relative strength of the penalty. (See Appendix C.10.1.2 for details on the hyperparameter tuning of λ.)

- HARDC-ANP: In this hard-constrained Neural Process ablation, we project the ANP mean to the nearest solution in L_2 satisfying the integral form of conservation constraint. This method is inspired by the approach taken in Négiar et al. (2022) that projects the output of a neural network onto the nearest solution satisfying a linear PDE system. HARDC-ANP is an alternative to Step 2 that solves the following constrained least-squares problem:

$$\begin{aligned} \mu_{HC} &= \text{argmin}_u \|u - \mu\|_2^2 \text{ s.t. } Gu = b \\ &= \mu - G^T (GG^T)^{-1} (G\mu - b). \end{aligned}$$

  HARDC-ANP is equivalent to the limiting solution of the mean of PROBCONSERV as $\sigma_G \to 0$ in Equation 4.8a, if the variance from Step 1 is fixed to be the same for each point, i.e., $\Sigma = I$.

Evaluation. At test time, we select a value of the PDE parameter α that lies within the range of PDE parameters used during training (i.e., $\alpha \in \mathcal{A}$). For each value of α, we generate multiple independent draws of (D_i, u_i, b_i) in the same manner as the training data. At a particular time-index t_j in the training window, we report the following prediction metrics: conservation error (CE) $(G\mu - b)_{t_j}$; mean-squared error (MSE) $\frac{1}{M}\|u_{t_j,\cdot} - \mu_{t_j,\cdot}\|^2$; and predictive log-likelihood (LL) $-\frac{1}{2M}\|u_{t_j,\cdot} - \mu_{t_j,\cdot}\|_{\Sigma^{-1}} - \frac{1}{2M}\sum_i \log \sigma^2_{t_j,i} - \log 2\pi$, where M denotes the number of spatial points. We report the average of each metric over $n_{\text{test}} = 50$ independent runs. Our convention for bolding the CE metric is binary on whether conservation is satisfied exactly or not. For the LL and MSE metrics, we bold the methods whose mean metric is within one standard deviation of the best mean metric.

4.3.1 Diffusion Equation: Constant k

The diffusion equation is the simplest non-trivial form of the GPME, with constant diffusivity coefficient $k(u) = k > 0$ (see Figure 4.1(a)). We train on values of $k \in \mathcal{A} = [1, 5]$. The diffusion equation is also known as the heat equation, where in that application the PDE parameter k denotes the conductivity and the total conserved quantity denotes the energy. In our empirical evaluations,

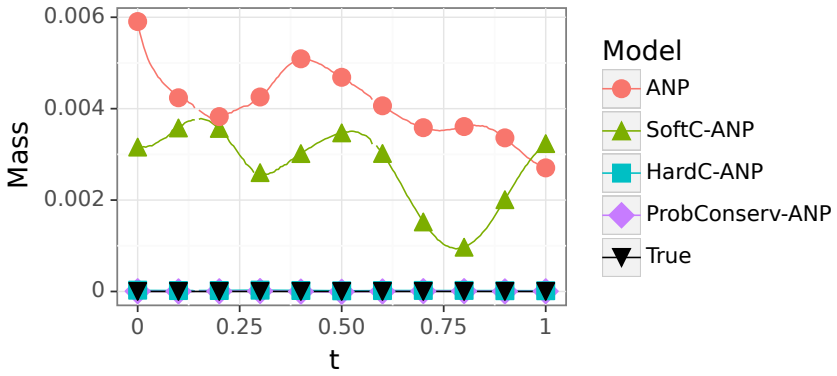


Figure 4.2: The total mass $U(t) = \int_\Omega u(t,x)d\Omega$ as a function of time t for the ("easy") diffusion equation with constant diffusivity coefficient $k \in \mathcal{A} = [1,5]$ and test-time parameter value $k = 1$. The true $U(t)$ is zero at all times since there is zero net flux from the domain boundaries and mass cannot be created or destroyed on the interior. Both PROBCONSERV-ANP and HARDC-ANP satisfy conservation of mass exactly. The other baselines violate conservation and result in a non-physical mass profile over time. ANP and SOFTC-ANP are not even zero at time $t = 0$.

	CE	LL	MSE
ANP	4.68 (0.10)	2.72 (0.02)	1.71 (0.41)
SOFTC-ANP	3.47 (0.17)	2.40 (0.02)	2.24 (0.78)
HARDC-ANP	**0** (0.00)	**3.08** (0.04)	**1.37** (0.33)
PROBCONSERV-ANP	**0** (0.00)	2.74 (0.02)	**1.55** (0.33)

Table 4.1: Mean and standard error for CE $\times 10^{-3}$ (should be zero), LL (higher is better) and MSE $\times 10^{-4}$ (lower is better) over $n_{\text{test}} = 50$ runs for the ("easy") diffusion equation at time $t = 0.5$ with variable diffusivity constant k parameter in the range $\mathcal{A} = [1,5]$ and test-time parameter value $k = 1$.

we use the diffusion equation notation, and refer to the conserved quantity as the mass.

Figure 4.2 illustrates that the unconstrained ANP solution violates conservation by allowing mass to enter and exit the system over time. Physically, there is no in-flux or out-flux on the boundary of the domain, and thus the true total mass of the system $U(t) = \int_\Omega u(t,x)d\Omega$ is zero at all times. Surprisingly, even incorporating the differential form of the conservation law as a soft constraint into the training loss via SOFTC-ANP violates conservation and the violation occurs even at $t = 0$.

Enforcing conservation as a hard constraint in our PROBCONSERV-ANP model and HARDC-ANP guarantees that the system total mass is zero, and also leads to improved predictive performance for both methods. In particular, Table 4.1 shows that these methods exactly obtain the lowest MSE and the highest LL. The success of these two approaches that enforce the integral form of the conservation law exactly, along with the failure of SOFTC-ANP that penalizes the

	$m=1$			$m=3$			$m=6$		
	CE	LL	MSE	CE	LL	MSE	CE	LL	MSE
ANP	6.67 (0.39)	3.49 (0.01)	0.94 (0.09)	−1.23 (0.29)	3.67 (0.00)	1.90 (0.04)	−2.58 (0.23)	3.81 (0.01)	**7.67** (0.09)
SOFTC-ANP	5.62 (0.35)	3.11 (0.01)	1.11 (0.14)	−0.65 (0.30)	3.46 (0.00)	2.06 (0.03)	−3.03 (0.26)	3.49 (0.00)	7.82 (0.09)
HARDC-ANP	**0** (0.00)	3.16 (0.04)	0.43 (0.04)	**0** (0.00)	3.44 (0.03)	**1.86** (0.03)	**0** (0.00)	3.40 (0.05)	**7.61** (0.09)
PROBCONSERV-ANP	**0** (0.00)	**3.56** (0.01)	**0.17** (0.02)	**0** (0.00)	**3.68** (0.00)	2.10 (0.07)	**0** (0.00)	**3.83** (0.01)	10.4 (0.04)

Table 4.2: Mean and standard error for CE $\times 10^{-3}$ (should be zero), LL (higher is better) and MSE $\times 10^{-4}$ (lower is better) over $n_{\text{test}} = 50$ runs for the ("medium") PME at time $t = 0.5$ with variable m parameter in the range $\mathcal{A} = [0.99, 6]$. For test-time parameter $m = 1$, where conservation by the unconstrained ANP is violated the most, PROBCONSERV-ANP leads to a substantial $\mathbf{5.5}\times$ improvement in MSE and log-likelihood. For test-time parameters $m = 3, 6$, the MSE for PROBCONSERV-ANP increases due to the error concentrated at the sharper boundary while the desired log-likelihood and conservation metrics improve.

differential form, demonstrates that physical knowledge must be properly incorporated into the learning process to improve predictive accuracy. Figure C.6 in Appendix C.10.1.1 illustrates that these conservative methods perform well on this "easy" case since the uncertainty from the ANP is relatively homoscedastic throughout the solution space; that is, the estimated errors are mostly the same size, and the constant variance assumption in HARDC-ANP holds reasonably well.

4.3.2 Porous Medium Equation (PME): $k(u) = u^m$

The Porous Medium Equation (PME) is a subclass of the GPME in which the coefficient, $k(u) = u^m$, is nonlinear and smooth (see Figure 4.1(b)). The PME is known to be degenerate parabolic, with differing behaviors depending on the value of m. We train on values of $m \in \mathcal{A} = [0.99, 6]$.

Table 4.2 compares the CE, MSE, and LL results for $m = 1, 3, 6$. These three values of m reflect "easy," "medium," and "hard" scenarios, respectively, as the solution profile becomes sharper. Despite achieving relatively low MSE for $m = 1$, the ANP model violates conservation the most. The error profiles as a function of x in Figure C.8 in Appendix C.10.1.2 illustrate the cause: the ANP consistently overestimates the solution to the left of the shock. Enforcing conservation consistently fixes this bias, leading to errors that are distributed around 0. Our PROBCONSERV-ANP method results in an $\approx 82\%$ improvement in MSE, and HARDC-ANP results in an $\approx 54\%$ improvement over the ANP. Since HARDC-ANP shifts every point equally, it induces a negative bias in the zero (degeneracy) region of the domain, leading to a non-physical solution.

For $m = 3, 6$, while the MSE for PROBCONSERV-ANP increases compared to the ANP, the LL for PROBCONSERV-ANP improves. The increase in LL for PROBCONSERV-ANP indicates that the uncertainty is better calibrated as a whole. Figure C.8 in Appendix C.10.1.2 illustrates that PROBCONSERV-ANP reduces the errors to the left of the shock point while increasing the error immediately to the right of it. This error increase is penalized more in the L_2 norm, which leads to an increase in MSE. The LL metric improves because our PROBCONSERV-ANP model takes into

account the estimated variance at each point. It is expected that the largest uncertainty occurs at the sharpest part of the solution, since that is the area with the largest gradient. This region is more difficult to be captured as the shock interface becomes sharper when m in increased.

For control on the enforcement of conservation constraint, see Figure C.2 in Figure C.5, where we show empirically that the log likelihood is always increasing, as stated in Theorem 4.1. Note that there are optimal values of σ_G^2, in which case the MSE can be better optimized.

4.3.3 Stefan Problem: Discontinuous Nonlinear $k(u)$

	CE	LL	MSE
ANP	-1.30 (0.01)	3.53 (0.00)	5.38 (0.01)
SOFTC-ANP	-1.72 (0.04)	**3.57** (0.01)	6.81 (0.15)
HARDC-ANP	**0** (0.00)	2.33 (0.06)	5.18 (0.02)
PROBCONSERV-ANP	**0** (0.00)	**3.56** (0.00)	**1.89** (0.01)

Table 4.3: Mean and standard error for CE $\times 10^{-2}$ (should be zero), LL (higher is better), and MSE $\times 10^{-3}$ (lower is better) over $n_{\text{test}} = 50$ runs for the ("hard") Stefan variant of the GPME at time $t = 0.05$. Each model is trained with the parameter $u^\star$ in the range $\mathcal{A} = [0.55, 0.7]$ and test-time parameter value $u^\star = 0.6$. PROBCONSERV-ANP leads to an increase in log-likelihood and a $\mathbf{3}\times$ decrease in MSE.

The most challenging case of the GPME is the Stefan problem. In this case, the coefficient $k(u)$ is a discontinuous nonlinear step function $k(u) = \mathbf{1}_{u \geq u^\star}$, where $\mathbf{1}_{\mathcal{E}}$ denotes an indicator function for event $\mathcal{E}$ and $u^\star \in \mathbb{R}_+$. The solution is degenerate parabolic and develops a moving shock over time (see Figure 4.1(c)). We train on values of $u^\star \in \mathcal{A} = [0.55, 0.7]$ and evaluate the predictive performances of each model at $u^\star = 0.6$.

Unlike the PME test case, where the degeneracy point ($x^*(t) = t$) is the same for each value of m, the shock position for the Stefan problem depends on the parameter $u^\star$ (See Figure C.1 in section C.3). This makes the problem more challenging for the ANP, as it can no longer memorize the shock position. On this "harder" problem, the unconstrained ANP violates the physical property of conservation by an order of magnitude larger in CE than in the "easier" diffusion and PME cases. By enforcing conservation of mass, PROBCONSERV-ANP results in substantial $\approx 65\%$ improvement in MSE (Table 4.3). In addition, Figure 4.3(a) shows that the solution profiles associated with ANP and the other baselines are smoothed and deviate more from the true solution than the solution profile of our PROBCONSERV-ANP model. Similar to our previous two case studies, adding the differential form of the PDE via SOFTC-ANP does not lead to a conservative solution (see Figure C.9 in Appendix C.10.1.3). In fact, Table 4.3 shows that surprisingly, conservation is violated more by SOFTC-ANP than with the ANP, with a corresponding increase in MSE. These results

demonstrate that physics-based constraints, e.g., conservation need be incorporated carefully (via finite volume based ideas) into ML-based models.

Table 4.3 shows that the LL for PROBCONSERV-ANP increases only slightly, compared to that of the ANP (3.56 vs 3.53), and it is slightly less than SOFTC-ANP. Figure 4.3(a) shows that enforcing conservation of mass creates a small upward bias in the left part of the solution profile for $x \in [0, 0.2]$. Since the variance coming from the ANP is smaller in that region, this bias is heavily penalized in the LL. This bias is worse for HARDC-ANP, which assumes an identity covariance matrix and ignores the uncertainty estimates from the ANP. HARDC-ANP adds more noticeable upward bias to the $x \in [0, 0.2]$ region, and it even adds bias to the zero-density region to the right of the shock. Compared to PROBCONSERV-ANP, HARDC-ANP only leads to a slight reduction in MSE (3%) and a much lower LL (2.33). This shows the benefit of using the uncertainty quantification from the ANP in our PROBCONSERV-ANP model for this challenging heteroscedastic case.

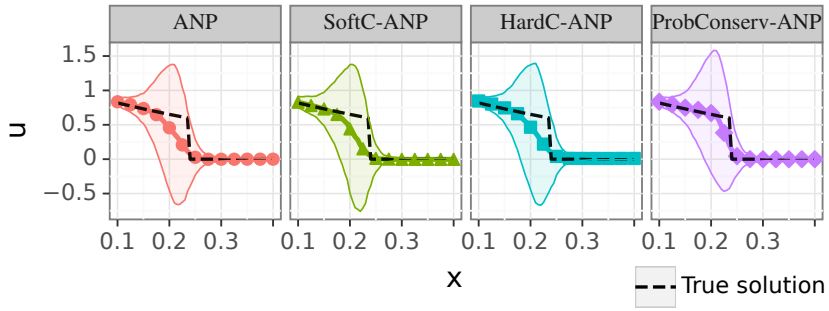
Downstream task: Shock point estimation. While quantifying predictive performance in terms of MSE or LL is useful in ML, these metrics are typically not of direct interest to practitioners. To this end, we consider the downstream task of shock point estimation, which is an important problem in fluids, climate, and other areas. The shock position for the Stefan problem $x^\star(t)$ depends on the parameter $u^\star$. Hence, for a given function at test-time, the shock position $x^\star(t)$ is unknown and must be predicted from the estimated solution profile.

We define the shock point at time t as the first spatial point (left-to-right) where the function equals zero:

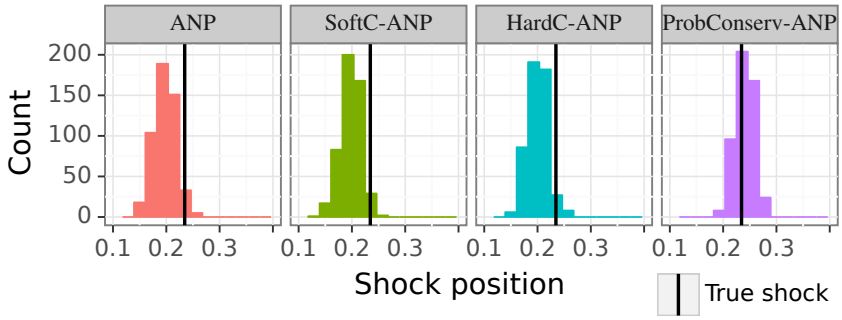
$$x^\star(t) = \inf_x \{u(t, x) = 0\}. \tag{4.10}$$

On a discrete grid, we approximate the infimum using minimum. The advantage of a probabilistic approach is that we can directly quantify the uncertainty of $x^\star(t)$ by drawing samples from the posterior distributions of our PROBCONSERV-ANP model and the baselines.

Figure 4.3(b) shows the corresponding histograms of the posterior of the shock position. We see that our PROBCONSERV-ANP posterior is centered around the true shock value. By underestimating the solution profile, the ANP misses the true shock position wide to the left, as do the other baselines SOFTC-ANP and HARDC-ANP. Remarkably, neither adding the differential form as a soft constraint (SOFTC-ANP) nor projecting to the nearest conservative solution in L_2 (HARDC-ANP) helps with the task of shock position estimation. This result highlights that both capturing the physical conservation constraint and using statistical uncertainty estimates in our PROBCONSERV-ANP model are necessary on challenging problems with shocks, especially when the shock position is unknown.



(a) Solution profile.



(b) Posterior of the shock position.

Figure 4.3: (a) Stefan solution profiles at time $t = 0.05$ with training parameter values $u^\star \in \mathcal{A} = [0.55, 0.7]$ and test-time parameter $u^\star = 0.6$. PROBCONSERV-ANP results in a sharper solution profile and the solution is mean-centered around the shock position. (b) The corresponding histogram of the posterior of the shock position computed as the mean plus or minus 3 standard deviations. PROBCONSERV-ANP reduces the level of underestimation and the induced negative bias at the shock interface to result in more accurate shock position prediction.

4.4 Conclusion

We have formulated the problem of learning physical models that can respect conservation laws from the finite volume perspective, by writing the governing conservation law in integral form rather than the commonly-used (in SciML) differential form. This permits us to incorporate the global integral form of the conservation law as a linear constraint into black-box ML models; and this in turn permits us to develop a two-step framework that first trains a black-box probabilistic ML model, and then constrains the output using a probabilistic constraint of the linear integral form. Our approach leads to improvements (in MSE, LL, etc.) for a range of "easy" to "hard" parameterized PDE problems. Perhaps more interestingly, our unique approach of using uncertainty quantification to enforce physical constraints leads to improvements in challenging shock point

estimation problems. Future extensions include support for local conservation in finite volume methods, where the same linear constraint approach can be taken by computing the fluxes as latent variables; imposing boundary conditions as linear constraints (Saad et al., 2022); and extension to other physical constraints, including nonlinear constraints, e.g., enstrophy in 2D and helicity in 3D, and inequality constraints, e.g., entropy (Tezaur et al., 2017).

CHAPTER 5

Normalizing Flows for Knockoff-free Controlled Feature Selection

5.1 Introduction

Researchers in machine learning have made much progress in developing regression and classification models that can predict a response based on features. In many application areas, however, practitioners need to know *which* features drive variation in the response, and they need to do so in a way that limits the number of false discoveries. For example, in genome-wide association studies (GWAS), scientists must consider hundreds of thousands of genetic markers to identify variants associated with a particular trait or disease. The cost of false discoveries (i.e., selecting variants that are not associated with the disease) is high, as a costly follow-up experiment is often conducted for each selected variant. Another example where controlled feature selection matters is analyzing observational data about the effectiveness of educational interventions. In this case, researchers may want to select certain educational programs to implement on a larger scale and require confidence that their selection does not include unacceptably many ineffective programs. As a result, researchers are interested in methods that model the dependence structure of the data while providing an upper bound on the false discovery rate (FDR).

Model-X knockoffs (Candès et al., 2018) is a popular method for controlled variable selection, offering theoretical guarantees of FDR control and the flexibility to use arbitrary predictive models. However, even with knowledge of the underlying feature distribution, the Model-X knockoffs method is not feasible unless the feature distribution is either a finite mixture of Gaussians (Gimenez et al., 2019) or has a known Markov structure (Bates et al., 2020). Hence, a body of research explores the use of empirical approaches that use deep generative models to estimate the distribution of X and sample knockoff features (Jordon et al., 2019; Liu and Zheng, 2018; Romano et al., 2020; Sudarshan et al., 2020).

The ability of these methods to control the FDR is contingent on their ability to correctly model the distribution of the features. By itself, learning a sufficiently expressive feature model

can be challenging. However, the knockoff procedure requires learning a knockoff distribution that satisfies the *swap property*, which is a much stronger requirement. Formally, let $X \in \mathbb{R}^D$ be a sample from the feature distribution and $\tilde{X} \in \mathbb{R}^D$ be a sample from the knockoff distribution conditioned on X. The swap property stipulates that the joint distribution $(X, \tilde{X}) \in \mathbb{R}^{2D}$ must be invariant to swapping the positions of any subset of features $S \in \{1, \dots, D\}$:

$$(X, \tilde{X})_{\text{swap}(S)} \overset{D}{=} (X, \tilde{X}) \tag{5.1}$$

Here, swap(S) means exchanging the positions of X_j and $\tilde{X}_j$ for all $j \in S$. For example, in the case $D = 3$ and $S = \{1, 3\}$, the joint distribution is $(X, \tilde{X}) = (X_1, X_2, X_3, \tilde{X}_1, \tilde{X}_2, \tilde{X}_3)$, and the swapped joint distribution is $(X, \tilde{X})_{\text{swap}(S)} = (\tilde{X}_1, X_2, \tilde{X}_3, X_1, \tilde{X}_2, X_3)$. Note that, for $S = \{1, \dots, D\}$, the swap property implies that $\tilde{X} \overset{D}{=} X$. See Candès et al. (2018) for a more detailed description of the swap property.

Even if a distribution were found satisfying the swap property, it may not provide enough power to make discoveries. For example, both properties are trivially satisfied by constructing exact copies of the features as knockoffs, but the resulting procedure has no power.

In situations where a valid knockoff distribution is available to sample from, knockoffs are computationally appealing because they require only one sample from a knockoff distribution to assess the relevance of all p features. However, in situations where the joint density of the features is unknown, we show that empirical approaches to knockoff generation (Jordon et al., 2019; Liu and Zheng, 2018; Romano et al., 2020; Sudarshan et al., 2020) fail to characterize a valid knockoff distribution and therefore do not control the FDR. We further show that even with a known covariate model, it is not straightforward to construct a valid knockoff distribution unless a specific model structure is known.

We propose a new feature selection method called FLOWSELECT (section 5.3), which does not suffer from these problems. FLOWSELECT uses normalizing flows to learn the joint density of the covariates. Normalizing flows is a state-of-the-art method for density estimation; asymptotically, it can approximate any distribution arbitrarily well (Papamakarios et al., 2021; Kobyzev et al., 2020; Huang et al., 2018). Additionally, FLOWSELECT circumvents the need to sample a knockoff distribution by instead applying a fast variant of the conditional randomization test (CRT) introduced in Candès et al. (2018). Samples from the complete conditionals are drawn using MCMC, ensuring they are unbiased with respect to the learned data distribution.

Asymptotically, FLOWSELECT computes correct p-values to use for feature selection (section 5.4). Our proof assumes the universal approximation property of normalizing flows and the convergence of MCMC samples to the Markov chain's stationary distribution. Under the same assumptions as the CRT, which includes a multiple-testing correction as in Benjamini and Hochberg

(1995), a selection threshold can be picked which controls the FDR at a pre-defined level. Empirically, on both synthetic (Gaussian) data and semi-synthetic data (real predictors and a synthetic response), FLOWSELECT controls the FDR where other deep-learning-based knockoff methods do not. In cases in which competing methods do control the FDR, FLOWSELECT shows higher power (section 5.5). Finally, in a challenging real-world problem with soybean genome-wide association study (GWAS) data, FLOWSELECT successfully harnesses normalizing flows for modeling discrete and sequential GWAS data, and for selecting genetic variants the traits depend on (section 5.5.4).

5.2 Background

FLOWSELECT brings together four existing lines of research, which we briefly introduce below.

Normalizing flows Normalizing flows is a general framework for density estimation of a multi-dimensional distribution with arbitrary dependencies (Papamakarios et al., 2021). A normalizing flow starts with a simple probability distribution (e.g., Gaussian or uniform), which is called the *base distribution* and denoted Z, and transforms samples from this base distribution through a series of invertible and differentiable transformations, denoted G, to define the joint distribution of $X \in \mathbb{R}^D \sim \mathcal{P}_X$. A normalizing flow with enough transformations can approximate any multivariate density, subject to regularity conditions detailed by Kobyzev et al. (2020). Compared to other density-estimation methods, normalizing flows are computationally efficient. Details about the specific normalizing flow architecture used in FLOWSELECT are provided in Appendix D.1.

Controlled feature selection Consider a response Y which depends on a vector of features $X \in \mathbb{R}^D$. Depending on how the features are chosen, it is plausible that only a subset of the features contains all relevant information about Y. Specifically, conditioned on the relevant features in X, Y is independent of the remaining features in X (i.e. the null features). The goal of the controlled feature selection procedure is to maximize the number of relevant features selected while limiting the number of null features selected to a predefined level. If we denote the total number of selected features R, then we can decompose R into V, the number of relevant features selected, and S, the number of null features selected.

Conditional randomization test Controlled feature selection can be seen as a multiple hypothesis testing problem where there are p null hypotheses, each of which says that feature X_j is conditionally independent of the response Y given all the other features X_{-j}. Explicitly, the test

of the following hypothesis is conducted for each feature $j = \{1, \ldots, D\}$:

$$H_0 : X_j \perp Y | X_{-j} \quad \text{versus} \quad H_1 : X_j \not\perp Y | X_{-j}. \tag{5.2}$$

To test these hypotheses, one can use a conditional randomization test (CRT) (Candès et al., 2018). For each feature tested in a conditional randomization test, a test statistic T_j (e.g., the LASSO coefficient or another measure of feature importance) is first computed on the data. Then, the null distribution of T_j is estimated by computing its value $\tilde{T}_j$ based on samples $\tilde{X}_j$ drawn from the conditional distribution of X_j given X_{-j}. Finally, the p-value is calculated based on the empirical CDF of the null test statistics, and features whose p-values fall below the threshold set by the Benjamini-Hochberg procedure (Benjamini and Hochberg, 1995) are selected. Though the CRT is introduced as a computationally inefficient alternative to knockoffs, the CRT nonetheless has appeal because it requires only knowledge of the feature distribution, which can be learned empirically by maximum likelihood.

Holdout randomization test The holdout randomization test (HRT) (Tansey et al., 2021) is a fast variant of the CRT; it uses a test statistic that requires fitting the model only once. Let θ represent the parameters of the chosen model, and let $T(X, Y, \theta)$ be an importance statistic calculated from the model with input data. For example, T, could be the predictive likelihood $\mathcal{P}_\theta(Y^{\text{test}} | X^{\text{test}})$ or the predictive score R^2. To use the HRT, first fit model parameters $\hat{\theta}$ based on the training data. Next, for each covariate j, calculate the test statistic $T_j^* \leftarrow T(X^{\text{test}}, Y^{\text{test}}, \hat{\theta})$. Then, generate k null samples and compute $T_{j,k} \leftarrow T(X^{\text{test}}_{(j \leftarrow j_k)}, Y^{\text{test}}, \hat{\theta})$, where $X^{\text{test}}_{(j \leftarrow j_k)}$ replaces the j-th covariate with the k-th generated null sample. Finally, calculate the p-value as in the CRT, based on the empirical CDF of the null test statistics.

5.3 Methodology

FLOWSELECT implements the CRT for arbitrary feature distributions by using a normalizing flow to fit the feature distribution and Markov chain Monte Carlo (MCMC) to sample from each complete conditional distribution. Performing controlled feature selection with FLOWSELECT consists of the three steps below.

Step 1: Model the predictors with a normalizing flow

Starting with the observed samples of the features $X_1, \ldots, X_N \sim \mathcal{P}_X$, we fit the parameters of a normalizing flow G_θ to maximize the log likelihood of the data with respect to a base distribution

Algorithm 5.1 Step 2 of the FLOWSELECT procedure for drawing K null features $\tilde{X}_{i,j}|X_{i,-j}$ for feature j at observation i.

Input: Feature matrix $X \in \mathbb{R}^{N\times D}$, observation index i, feature index j, number of samples K, fitted normalizing flow $p_{\hat{\theta}}$, MCMC proposal q_j
Output: Null features $\tilde{X}_{i,j,k}$ for $k = 1, \ldots, K$
for $k = 1, \ldots, K$ **do**
  Propose: $X^{\star}_{i,j,k} \sim q_j(\cdot|\tilde{X}_{i,j,k-1}, X_{i,-j})$
  $r_{i,j,k} \leftarrow \frac{p_{\hat{\theta}}(X^{\star}_{i,j,k}, X_{i,-j}) q_j(\tilde{X}_{i,j,k-1}|X^{\star}_{i,j,k}, X_{i,-j})}{p_{\hat{\theta}}(\tilde{X}_{i,j,k-1}, X_{i,-j}) q_j(X^{\star}_{i,j,k}|\tilde{X}_{i,j,k-1}, X_{i,-j})}$
  Sample: $U_{i,j,k} \sim \text{Bernoulli}(r_{i,j,k} \wedge 1)$
  if $U_{i,j,k} = 1$ **then**
    $\tilde{X}_{i,j,k} \leftarrow X^{\star}_{i,j,k}$
  else
    $\tilde{X}_{i,j,k} \leftarrow \tilde{X}_{i,j,k-1}$
  end if
end for

p_Z:

$$\begin{aligned} \hat{\theta} &= \arg\max_{\theta} \sum_{i=1}^{N} \log p_\theta(X_i) \\ \text{where } p_\theta(X_i) &= p_Z(G_\theta(X)) \left| \det \left(\frac{\partial G_\theta(X)}{\partial X} \right) \right|. \end{aligned} \tag{5.3}$$

The resulting density $p_{\hat{\theta}}$ is a fitted approximation to the true density $\mathcal{P}_X$. The specific normalizing flow architecture we use in our first two experiments consists of a single Gaussianization layer (Meng et al., 2020) followed by a masked autoregressive flow (MAF) (Papamakarios et al., 2017). The first layer can learn complex marginal distributions for each covariate, while the MAF learns the dependencies between them. More detail on normalizing flows and on this particular architecture can be found in appendix D.1.

Step 2: Sample from the complete conditionals with MCMC

For each feature j, we aim to sample corresponding null features $\tilde{X}_{i,j,k}$ for all $k \in \{1, \ldots, K\}$ that are equal in distribution to $p_{\hat{\theta}}(X_{i,j}|X_{i,-j})$, but independent of Y_i. However, directly sampling from this conditional distribution is intractable. Instead, we implement an MCMC algorithm that admits it as a stationary distribution. The samples drawn from MCMC are autocorrelated, but any statistic calculated over these samples will converge almost surely to the correct value. The choice of the MCMC proposal distribution q_j is flexible. Because each Markov chain is only one-dimensional, a Metropolis-Hastings Gaussian random walk with the standard deviation set based

on the covariance can be expected to mix rapidly. Alternatively, information from $p_{\hat{\theta}}$, such as higher-order derivatives, could be used to construct a more efficient proposal. algorithm 5.1 details how to implement step 2.

Step 3: Test for significance with the HRT

As in the CRT, feature j has high evidence of being significant if, under the assumption that j is a null feature, the probability of realizing a test statistic greater than the observed $T_j(X)$ is low. Formally, letting $[\tilde{X}_j, X_{-j}]$ be the observed feature matrix with the observed feature X_j swapped out with the null feature $\tilde{X}_j$, we can write this as a p-value α_j:

$$\alpha_j \equiv \mathcal{P}_{\tilde{X}_j|X_{-j}}\left(T_j(X) < T_j([\tilde{X}_j, X_{-j}])\right). \tag{5.4}$$

However, the above p-value α_j is not tractable. For each sample $\tilde{X}_{\cdot,j,k}$ drawn using MCMC, we calculate the corresponding feature statistic and compare it to the real feature statistic, leading to an approximated p-value $\hat{\alpha}_j$:

$$\hat{\alpha}_j \equiv \frac{1}{K+1}(1 + \sum_{k=1}^{K} \mathbf{1}[T_j(X) < T_j([\tilde{X}_{j,k}, X_{-j}])). \tag{5.5}$$

To control the FDR, we use the Benjamini-Hochberg procedure to establish a threshold for the observed p-values. Specifically, we set the threshold to $s(\gamma) \triangleq \max_j\{\hat{\alpha}_j : \hat{\alpha}_j \leq \frac{j}{D}\gamma\}$, and select all features j such that $\alpha_j \leq s(\gamma)$.

The Benjamini-Hochberg correction only guarantees FDR control provided that the p-values have either positive or zero correlation. Thus, the FDR control of FLOWSELECT depends on these assumptions being met. A more conservative correction from Benjamini and Yekutieli (2001) allows for arbitrary dependencies in p-values, but it suffers from low power. The Benjamini-Hochberg correction is widely used and empirically robust (Tansey et al., 2021), so we report results using it. Across our synthetic and semi-synthetic benchmarks in section 5.5, we also find that FLOWSELECT maintains empirical FDR control.

Provided that the Benjamini-Hochberg assumptions are met, the FDR will be controlled, but the power of the test depends on T_j being higher when j is a significant feature. For example, if Y is expected to vary approximately linearly with respect to X, $T_j(X)$ could be the absolute estimated regression coefficient $|\hat{\beta}_j|$ for the linear model $Y = X\beta + \epsilon$. Another choice is the HRT feature statistic described earlier.

5.4 Asymptotic results

The ability of FLOWSELECT to control the FDR relies on its ability to produce estimated p-values that converge to the correct p-values for the hypothesis test in eq. (5.2).

Theorem 5.1. *Let $X \in \mathbb{R}_{N\times D}$ be a random feature matrix, where each row $X_{i,\cdot}$ is independent and identically distributed; $x \in \mathbb{R}_{N\times D}$ be the observed feature matrix; and α_j be the p-value as defined in eq.* (5.4) *with test statistic $T_j(X)$. Suppose there exists a sequence of functions $(G^n)_{n=1}^{\infty}$ and a base random variable Z satisfying the following conditions:*

1. *Each G^n is continuously differentiable and invertible.*
2. *$G^n \to G$ pointwise for some map G that is triangular, increasing, continuously differentiable, and satisfies $G(X_{i,\cdot}) \overset{D}{=} Z$.*

For $n = 1, 2, \ldots$, let X^n be the random feature matrix where each row i is independent and has distribution $X^n_{i,\cdot} = (G^n)^{-1}(Z)$. Then, the p-value in eq. (5.5) *calculated using K MCMC samples targeting $X^n_{\cdot,j} \mid X^n_{\cdot,-j} = x_{\cdot,-j}$ converges to the correct p-value α_j with probability* 1.

Here we sketch the proof. A full proof can be found in Appendix D.2. First, by construction each G^n defines a distribution $X^n_{i,\cdot} \overset{D}{=} (G^n)^{-1}(Z)$ that in turn implies a conditional distribution $X^n_{\cdot,j}|X^n_{\cdot,-j} = x_{\cdot,-j}$. We show these conditional distributions converge to the true conditional distribution of $X_{\cdot,j}$ given $X_{\cdot,-j} = x_{\cdot,-j}$. Consequently, the probability of observing a higher test statistic under the approximated null distribution $\tilde{X}^n_{\cdot,j} \overset{D}{=} X^n_{\cdot,j}$, written α^n_j, will converge to the probability under the true null distribution $\tilde{X}_{\cdot,j}|X_{\cdot,-j} = x_{\cdot,-j}$, i.e. α_j. Next, the Cesaro average of K samples from an MCMC algorithm targeting $\tilde{X}^n_{\cdot,j}|X_{\cdot,-j} = x_{\cdot,-j}$, written $\hat{\alpha}_{j,K,n}$ will converge to α^n_j with probability 1 as $K \to \infty$. Combining these two convergences leads to the stated result.

Assuming the limiting p-values $\{\alpha_j\}$ satisfy the chosen multiple-hypothesis-testing assumptions, Theorem 5.1 specifies additional conditions that are sufficient for FDR control. These conditions are not strictly fewer than those required for empirical model-X knockoff-based methods to control FDR, but they may be easier to satisfy adequately in practice. For example, the condition that there exists a sequence $(G_n)_{n=1}^{\infty}$ converging to the true mapping G is satisfied asymptotically by many flow architectures that are universal distribution approximators, including the Gaussianization Flows and Masked Autoregressive Flows used in our experiments (Huang et al., 2018; Meng et al., 2020; Kobyzev et al., 2020). In practice, it is unlikely that an exact mapping G will be learned, as doing so could require infinite training data, infinitely deep transformations, and exact nonconvex optimization. Nonetheless, normalizing flows work extremely well in practice; **??** 5.1 gives intuition for the good performance of FLOWSELECT that we observe empirically.

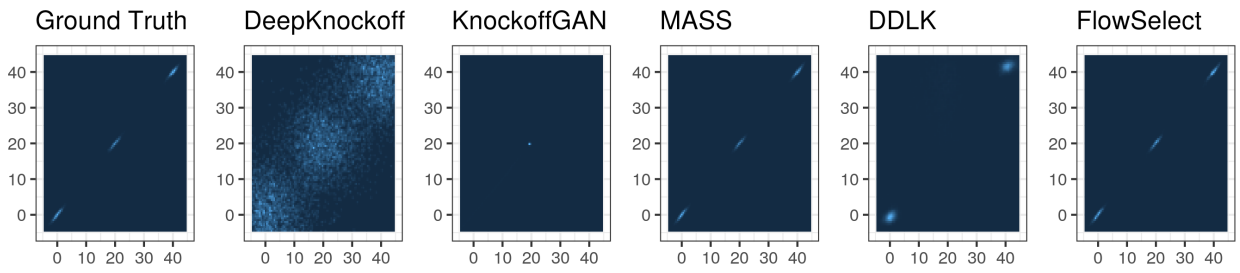


Figure 5.1: A density plot of the feature distribution with coordinate $j = 1$ on the x-axis and coordinate $j = 2$ on the y-axis. The ground truth density is compared to the normalizing flow fitted within FLOWSELECT and the distribution of each knockoff method (DeepKnockoff, KnockoffGAN, MASS, and DDLK). To have FDR control, each distribution should match the distribution of the features.

5.5 Experiments

5.5.1 Synthetic experiment with a mixture of highly correlated Gaussians

We compare FLOWSELECT to the aforementioned knockoff methods with synthetic data drawn from a mixture of three highly correlated Gaussian distributions with dimension $D = 100$.[1] For each knockoff method, we use the exact implementation described in their respective papers, and we utilize the code made publicly available by the authors (c.f. appendix D.2.2 for further details). For further comparison, we also implement the MASS knockoff procedure from Gimenez et al. (2019) and the RANK knockoff procedure from Fan et al. (2020). These methods estimate the unknown feature distribution using either a mixture of Gaussians (MASS) or a sparse precision matrix (RANK), and then sample the knockoffs directly as in Candès et al. (2018).

To generate the data, we draw $N = 100,000$ highly correlated samples. For $i = 1, \ldots, N$, we sample

$$X_i \overset{\text{i.i.d}}{\sim} \sum_{m=1}^{3} \pi_m p_{\mathcal{N}}(X_i; \mu_m, \Sigma_m), \tag{5.6}$$

with mixing weights $\pi = (0.371, 0.258, 0.371)$, mean vector $\mu = (0, 20, 40)$, and covariance matrices Σ_m. Each covariance Σ_m follows an AR(1) pattern such that $(\Sigma_m)_{i,j} = \rho_m^{|i-j|}$ where $\rho = (0.982, 0.976, 0.970)$. The response Y_i is linear in $f_i(X_i)$ for some function f_i and coefficient vector β i.e., $Y_i = f_i(X_i)\beta + \epsilon_i$. Each coefficient β_j equals $\frac{100}{\sqrt{N}} B_j$, where $B_j = 0$ with probability 0.8, $B_j = 1$ with probability 0.1, and $B_j = -1$ with probability 0.1. We consider two different schemes for the f_i that connect the features to the response. In our linear setting, f_i is equal to the identity function. In our nonlinear setting, $f_i(x)$ is set equal to $\sin(5x)$ for odd i and $f_i(x) =$

[1] Software to reproduce our experiments is available at https://github.com/dereklhansen/flowselect.

$\cos(5x)$ for even i.

The experimental setting we have described so far is adapted from Sudarshan et al. (2020). However, we found that the $N = 2000$ they used was too few observations for any of the methods to do well in a general non-linear setting. Moreover, in many situations where controlled feature selection is deployed, neighboring features will be highly correlated. To reflect this, we also increased the base correlation between features within each mixture to create a more challenging example. We show results under the original settings of Sudarshan et al. (2020) in appendix D.9.

For each model, we use 90% of the data for training to generate null features and the remaining 10% for calculating the feature statistics. To define the feature statistics, we use the holdout randomization test (HRT) described at the end of section 5.2. For the HRT, we employ different predictive models for each response type ("linear" and "nonlinear"). Specifically, for the linear response, we use the predictive log-likelihood from the LASSO (Tibshirani, 1996), and for the nonlinear response, we use the predictive negative mean-squared error from a random forest regressor (Breiman, 2001).

First, we look at how each procedure models the covariate distribution in fig. 5.1. In order to be valid knockoffs, the distribution of two knockoff features needs to be equal to that of the covariates. In this challenging example, each of the empirical knockoff methods fails to match the ground truth. In particular, DDLK and DeepKnockoffs are over-dispersed, while KnockoffGAN suffers from mode collapse. These findings for DeepKnockoffs and KnockoffGAN are similar to those reported by Sudarshan et al. (2020). Other than MASS, which directly fits a mixture of Gaussians, FLOWSELECT is the only method that matches the basic structure of the ground truth.

fig. 5.2 shows that the empirical knockoff procedures fail to control the FDR for both linear and nonlinear responses. One explanation for this lack of FDR control is the inability of the deep-learning-based methods to accurately model a knockoff distribution (c.f., fig. 5.1). As a result, the assumptions for the knockoff procedure will not hold, and FDR control is not guaranteed.

The effects of misspecification are clearly visible in the case of RANK, which approximates the mixture-of-Gaussians data with a multivariate Gaussian. However, even MASS, when given access to the correct data distribution, does not achieve across-the-board FDR control. This highlights the potential sensitivity of knockoffs to parameter misfit even when the underlying distributional family of the features is known. This is confirmed by the fact that, when provided with the true parameters, the oracle Model-X maintains FDR control, though with significantly less power than FLOWSELECT. (c.f. Appendix D.6).

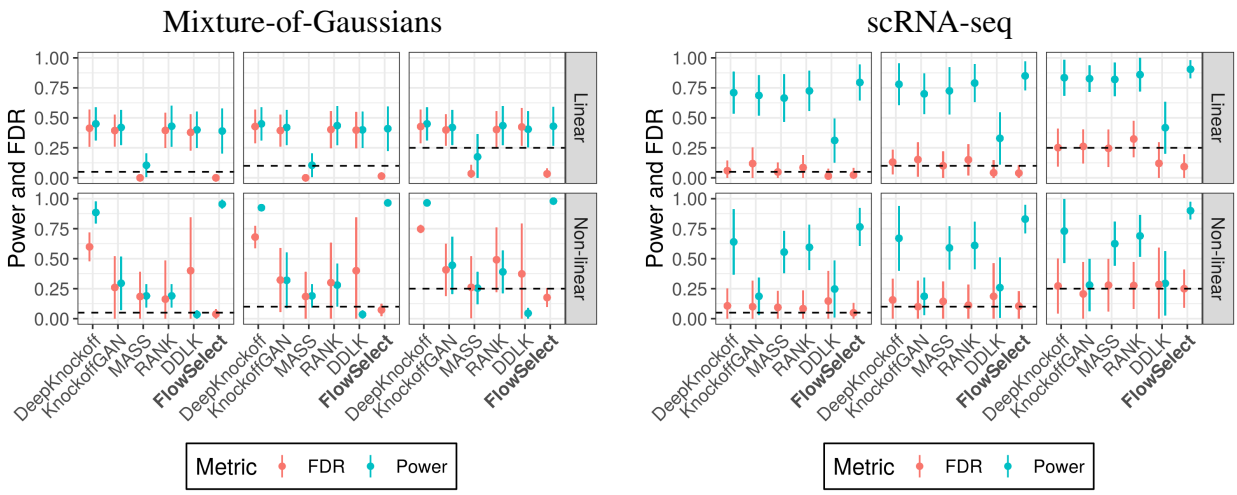


Figure 5.2: Comparison of power and false discovery rate (FDR) control of FLOWSELECT to knockoff methods on the Mixture-of-Gaussians dataset (left) and the scRNA-seq dataset (right) at targeted FDRs of 0.05, 0.1, and 0.25 (indicated by the dashed lines). Each point indicates the mean power and FDR across 20 replications and the error bars span one standard deviation either direction. In the top row, the response depends linearly on the features, and the feature statistics are calculated using the HRT with the LASSO. In the bottom row, the response depends non-linearly on the features, and the feature statistics are calculated using the HRT with random forest regression.

5.5.2 Semi-synthetic experiment with scRNA-seq data

In this experiment, we use single-cell RNA sequencing (scRNA-seq) data from 10x Genomics (Genomics, 2017). Each variable $X_{n,g}$ is the observed gene expression of gene g in cell n. These data provide an experimental setting that is both realistic and, because gene expressions are often highly correlated, challenging. More background information about scRNA-seq data can be found in Agarwal et al. (2020).

We normalize the gene expression measurements to lie in $[0, 1]$, and we add a small amount of Gaussian noise so that the data is not zero-inflated. As in the semi-synthetic experiment from Sudarshan et al. (2020), we pick the 100 most correlated genes to provide a challenging, yet realistic example. We simulate responses that are both linear and nonlinear in the features. fig. 5.2 shows that FLOWSELECT maintains FDR control across multiple FDR target levels, feature statistics, and generated responses. In cases in which the knockoff methods control FDR successfully, FLOWSELECT has higher power in discovering the features the response depends on.

An advantage of knockoffs over CRT-based methods like FLOWSELECT is that the predictive model only needs to be evaluated once. Hence, while FLOWSELECT has a faster runtime than DDLK for this experiment, it is slower than DeepKnockoff and KnockoffGAN. However, fig. 5.2 shows that these two models fail to reliably control FDR and have much less power than FLOWS-

ELECT; it is not clear how additional computational resources could be leveraged to improve the performance of these competing methods. A full table of runtimes on the scRNA-seq dataset can be found in appendix D.4.

The need to compute a different predictive model for each feature within the CRT is mitigated by using efficient feature statistics such as the HRT (Tansey et al., 2021) and the distilled CRT (Liu et al., 2020). These methods fit a larger predictive model once, then evaluate either the residuals or test mean-squared-error for each feature individually. Moreover, the ability to scale to large feature dimensions D is more limited by fitting the feature distribution than computational burden, a trait shared by both knockoff- and CRT-based methods.

FLOWSELECT provides asymptotic guarantees of FDR control assuming sufficient MCMC samples have been drawn for the p-values to converge. In this experiment, the consequence of terminating MCMC sampling before convergence is low power, rather than loss of FDR control (see fig. D.5 in appendix D.8). Even for small numbers of MCMC samples, the FDR stabilizes below the target rate, while the power steadily increases with the number of samples. Because the MCMC run is initialized at the true features, we speculate that the sampled features will be highly correlated with the true features in the beginning of the run, making it harder to reject the null hypothesis that a feature is unimportant.

5.5.3 Ablation Study

FLOWSELECT differs from the competing knockoff-based approaches in two ways: using normalizing flows with MCMC to model the feature distribution for sampling null features and using the CRT for feature selection. To illustrate the impact of each of these components separately, we compare to the procedure used in Tansey et al. (2021), which uses mixture density networks (MDNs) to model the complete conditional distribution of each feature $\mathcal{P}(X_j|X_{-j})$ separately. They then sample null features from these learned distributions directly and use the HRT for feature selection. Since both FLOWSELECT and this procedure utilize the HRT, this allows us to evaluate whether the performance improvement of FLOWSELECT over empirical knockoffs is solely due to use of the HRT.

We compare the MDN-based approach to FLOWSELECT on the mixture-of-Gaussians (section 5.5.1) and scRNA-seq (section 5.5.2) datasets. A plot of this comparison can be found in appendix D.5. While the MDN-based approach was able to match the performance of FLOWSELECT on the scRNA-seq dataset, it failed to control FDR at any level on the Mixture-of-Gaussians dataset, indicating that MDNs are less flexible than normalizing flows. In aggregate, these results show that both the normalizing flows paired with MCMC and the use of the HRT for significance testing are key to the performance of FLOWSELECT.

5.5.4 Real data experiment: soybean GWAS

Genome-wide association studies are a way for scientists to identify genetic variants (single-nucleotide polymorphisms, or SNPs) that are associated with a particular trait (phenotype). We tested FLOWSELECT on a dataset from the SoyNAM project (Song et al., 2017), which is used to conduct GWAS for soybeans. Each feature X_j takes on one of four discrete values, indicating whether a particular SNP is homozygous in the non-reference allele, heterozygous, homozygous in the reference allele, or missing. A number of traits are included in the SoyNAM data; we considered oil content (percentage in the seed) as the phenotype of interest in our analysis. There are 5,128 samples and 4,236 SNPs in total.

To estimate the joint density of the genotypes, we used a discrete flow (Tran et al., 2019). Modeling of genomic data is typically done with a hidden Markov model (Xavier et al., 2016); however, such a model may fail to account for long range dependence between SNPs, which a normalizing flow is better suited to handle. Having a more flexible model of the genome enables FLOWSELECT to provide better FDR control for assessing genotype/phenotype relationships. For the predictive model, we used a feed-forward neural network with three hidden layers. Additional details of training and architecture are presented in Appendix D.3.

A graphical representation of our results is shown as a Manhattan plot in fig. 5.3, which plots the negative logarithm of the estimated p-values for each SNP. At a nominal FDR of 20%, we identified seven SNPs that are associated with oil content in soybeans. We cross-referenced our discoveries with other publications to identify SNPs that have been previously shown to be associated with oil content in soybeans. For example, FLOWSELECT identifies one SNP on the 18th chromosome, `Gm18_1685024`, which is also selected in Liu et al. (2019). FLOWSELECT also selects a SNP on the 5th chromosome, `Gm05_37467797`, which is near two SNPs (`Gm05_38473956` and `Gm05_38506373`) identified in Cao et al. (2017) but which are not in the SoyNAM dataset. Sonah et al. (2014) identifies eight SNPs near the start of the 14th chromosome, and we select multiple SNPs in a nearby region on the 14th chromosome (seen in the peak of dots on chromosome 14 in fig. 5.3). However, the dataset in Sonah et al. (2014) is much larger ($\approx 47,000$ SNPs), which prevents an exact comparison. A list of all SNPs selected by our method is provided in Appendix D.3. For this experiment, FLOWSELECT tests over 4000 features in 10 hours using a single GPU. None of the empirical knockoff procedures (Sudarshan et al., 2020; Jordon et al., 2019; Romano et al., 2020) tested more than 387 features. This shows the potential for FLOWSELECT for high-dimensional feature selection with FDR control in a reasonable amount of time. Additional details about this experiment are available in Appendix D.3.

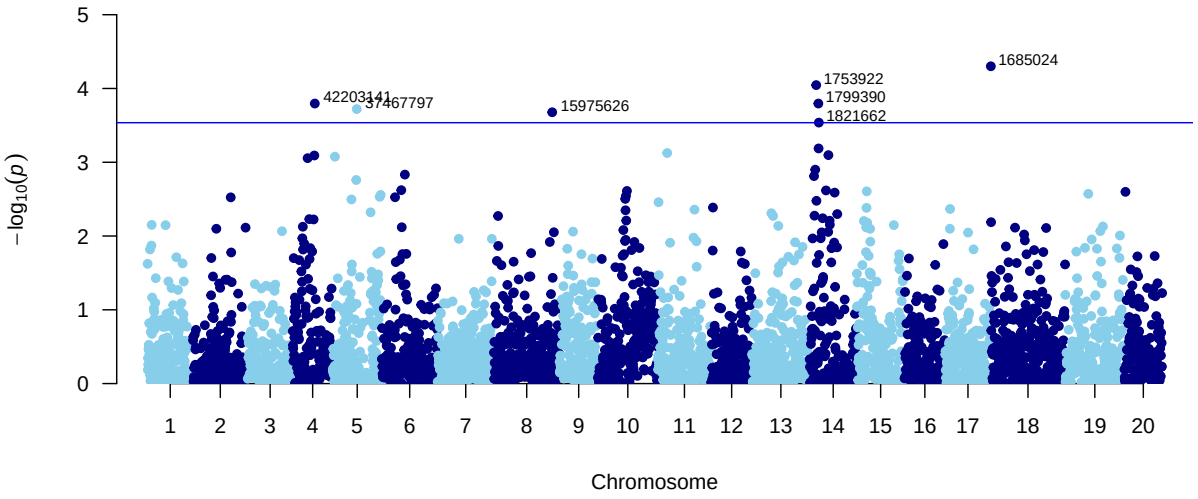


Figure 5.3: Manhattan plot for oil content in soybean GWAS experiment (Turner, 2018). p is the estimated p-value from the FLOWSELECT procedure, and the blue line indicates the rejection threshold for a nominal FDR of 20%.

5.6 Discussion

FLOWSELECT enables scientists and other practitioners to discover features a response depends on while controlling false discovery rate, using an arbitrary predictive model; even large-scale non-linear machine learning models can be utilized. By making fewer false discoveries for a fixed sensitivity level, FLOWSELECT can reduce the cost of follow-up experiments by limiting the number of irrelevant features considered. In contrast to the original model-X knockoffs method, FLOWSELECT does not require the feature distribution to be known a priori, nor does it require the feature distribution to have a particular form (e.g., Gaussian). Neither of these conditions are often satisfied in practice.

One limitation shared by both the conditional randomization test (CRT) and knockoffs is low power in cases in which important features are highly correlated with other important features. To mitigate this limitation, the CRT can be applied to test the significance of groups of correlated features rather than individual features. Within the FLOWSELECT framework, this entails modifying the MCMC step to draw null samples of groups of features conditioned on the others. The group's p-value can then be calculated with the same holdout randomization test (HRT) statistic used for testing individual features. Group feature selection has also been explored for knockoffs (Dai and Barber, 2016; Liu et al., 2020).

Another limitation of FLOWSELECT stems from its reliance on normalizing flows. The flexibil-

ity of normalizing flows, though often beneficial, comes at a cost: sufficient training examples are needed to learn the feature distribution, limiting applicability in data-starved regimes. Fortunately, as we show in appendix D.9, FLOWSELECT fares no worse than competing methods in low-data settings. In these regimes, FLOWSELECT could also use other density estimation techniques such as autoregressive models.

Furthermore, learning the feature distribution (potentially from limited data) is not the sole difficulty that the deep-learning-based knockoff methods face. To demonstrate that there are additional sources of difficult for knockoff-based methods, we gave DDLK, which typically fits the data distribution as part of its training procedure, access to the *the exact joint density*; neither the empirical FDR nor the power improved significantly (c.f. appendix D.7). This result points to a failure of DDLK to enforce the swap property, which is a challenging task as the number of swaps grows exponentially with the number of features. FLOWSELECT, on the other hand, achieves FDR control under a different set of conditions that often are simpler to satisfy adequately in practice.

CHAPTER 6

Conclusion and Future Work

Throughout this thesis, we have presented Bayesian methods for scientific applications that take advantage of the interpretability and robustness of Bayesian statistics. At several points, we illustrate that deep learning can alleviate the traditional challenges in implementing Bayesian methods, either by accelerating the inference of mechanistic models or by accomodating unknown structure via black-box generative models. On the other hand, Bayesian methods enrich the output of deep learning pipelines, whether by quantifying uncertainty or enforcing physical constraints. This is crucial for scientific applications as, in addition to predictive accuracy, they demand domain-awareness, interpretability, and robustness (Baker et al., 2019). Below, we present a few future directions based on the chapters of this work that expand upon these goals.

Variational inference for Argo floats The Bayesian inference procedure for predicting the trajectory of Argo floats presented in Chapter 2 was performed for each float separately using Sequential Monte Carlo (SMC). This procedure yields asymptotically exact results, and our Kalman smoother proposal already performs well by fully adapting to the high signal-to-noise-ratio GPS measurements. However, the algorithm is costly, especially as more particles are used, and the sequential nature of the algorithm limits how parallelizable the algorithm can be. A solution to better computational scalability is to use amortized variational inference to fit the float trajectories. Recent works have combined deep neural networks with Kalman filtering (Fraccaro et al., 2017; de Bézenac et al., 2020), which could be used to define a more expressive variational distribution.

Enforcement of local conservation in black-box models The PROBCONSERV framework presented in Chapter 4 is quite general, but we only consider global conservation of mass in experiments. A follow-up direction is to consider probabilistic enforcement of local conservation in an analogous manner to finite volume methods. This could lead to even more accurate solution profiles that satisfy physical conservation laws at multiple resolutions. While this accuracy would come at the cost of increased compute power, this would be a controllable trade-off in the hands of

the practitioner. In Chapter 2, we utilized knowledge about potential vorticity (PV) to locally constrain the velocity of floats via a Bayesian update. Applying this approach to a black-box model, however, will be more challenging, since we utilize a mechanistic model of float trajectories that treats velocity as a latent variable. Nevertheless, we could explore generalizing this approach into the black-box PROBCONSERV framework.

Multiple layers of tiles in BLISS The key to the performance of the BLISS encoder in Chapter 3 is decomposing a large astronomical image into small tiles that define the variational distribution. A challenge to this approach, however, is when an object appears right at the boundary of multiple tiles. The computed probabilities can change dramatically if the image is shifted by an amount less than the width of a tile (i.e. placing the object in the center of a tile rather than at the boundary). A solution to this problem could be considering multiple layers of tiles that overlap. By carefully constraining the variational distribution, one could favor placing objects in the center of tiles rather than at the boundaries, leading to less ambiguity in the estimation procedure.

APPENDIX A

Appendix for Chapter 2

A.1 Kalman filter equations

In this section, we show how a linear state-space model can be efficiently estimated via the Kalman filter. Consider the following set of equations:

$$Z_n = g_0 + GZ_{n-1} + \epsilon_n^Z, \tag{A.1}$$

$$Y_n = f + FZ_n + \epsilon_n^Y. \tag{A.2}$$

, where $Z_1 \sim \mathcal{N}(\mu_1, \Sigma_1)$, F is a $d_z \times d_z$ matrix, G is a $d_y \times d_z$ matrix, $\epsilon_n^Z \sim \mathcal{N}(0, \Sigma_Z)$ and $\epsilon_n^Y \sim \mathcal{N}(0, \Sigma_Y)$.

Together, $Z_{1:n}, Y_{1:n}$ follow a multivariate Gaussian. However, for large enough n, deriving conditional distributions we are interested in can become computationally challenging. The Kalman filter exploits the Markov structure of the problem to achieve efficient inference.

We start with the following lemma:

Lemma A.1. *Let $Z \sim \mathcal{N}(\mu, \Sigma)$ and $Y|Z \sim \mathcal{N}(f_0 + FZ, \Sigma_Y)$. Then $Z|Y$ is normally distributed with mean and variance as follows:*

$$\begin{aligned}\mathbb{E}(Z|Y) &= \mu - \Sigma F^T(\Sigma_Y + F\Sigma F^T)^{-1}F(F\mu - Y + f_0)\\ \mathrm{Cov}(Z|Y) &= \Sigma - \Sigma F^T(\Sigma_Y + F\Sigma F^T)^{-1}F\Sigma\end{aligned}$$

Proof A more detailed derivation for a similar result is shown in Appendix C via Lemma C.1 and Lemma C.2. The mean and covariance have the following updated form (Gelman et al., 2015, Chapter 3.5):

$$\begin{aligned}\mathrm{Cov}(Z|Y) &= \left(\Sigma^{-1} + F^T\Sigma_Y^{-1}F\right)^{-1} = A^{-1}\Sigma\\ \mathbb{E}(Z|Y) &= \left(\Sigma^{-1} + F^T\Sigma_Y^{-1}F\right)^{-1}\left(\Sigma^{-1}\mu + F^T\Sigma_Y^{-1}(Y - f_0)\right) = A^{-1}\left(\mu + \Sigma F^T\Sigma_Y^{-1}(Y - f_0)\right).\end{aligned}$$

where $A^{-1} = (\Sigma^{-1} + F^T\Sigma_Y^{-1}F)^{-1}\Sigma^{-1}$. Next, we transform A^{-1} using the following corollaries to the Woodbury identity: $(I + CB)^{-1} = I - C(I + BC)^{-1}C$ and $(C + BB^T)B = C^{-1}B(I + B^TC^{-1}B)^{-1}$ (Petersen et al., 2008):

$$\begin{aligned}
A^{-1} &= (I + \Sigma F^T\Sigma_Y^{-1}F)^{-1} \\
&= I - \Sigma F^T(I + \Sigma_Y^{-1}F\Sigma F^T)^{-1}\Sigma^{-1}F \\
&= I - \Sigma F^T(\Sigma_Y + F\Sigma F^T)^{-1}F. \\
A^{-1}\Sigma F^T\Sigma_Y^{-1} &= \left(\Sigma^{-1} + F^T\Sigma_Y^{-1}F\right)^{-1}F^T\Sigma_Y^{-1} \\
&= \left(\Sigma^{-1} + F^TLL^TF\right)^{-1}F^TLL^T \\
&= \Sigma F^TL\left(I + L^TF\Sigma F^TL\right)^{-1}L^T \\
&= \Sigma F^T\left((L^T)^{-1}L^{-1} + F\Sigma F^T\right)^{-1} \\
&= \Sigma F^T\left(\Sigma_Y + F\Sigma F^T\right)^{-1},
\end{aligned}$$

where $LL^T = \Sigma_Y$ is the Cholesky decomposition of Σ_Y. Substituting these two equalities into the expressions for $\mathrm{Cov}(Z|Y)$ and $\mathbb{E}(Z|Y)$ leads to the desired result. $\square$.

Forward filtering The Kalman filter is defined recursively. Let $\mu_n \triangleq \mathbb{E}(Z_n|Y_{1:n})$ and $\Sigma_n \triangleq \mathrm{Cov}(Z_n|Y_{1:n})$ be the filtered mean and covariance, respectively. We show how to derive μ_{n+1} and Σ_{n+1}. We start by calculating the mean and covariance of Z_{n+1} conditioned on $Y_{1:n}$:

$$a_t \triangleq \mathbb{E}(Z_{n+1}|Y_{1:n}) = \mathbb{E}(g_0 + GZ_n + \epsilon_n^X|Y_{1:n}) = g_0 + G\mu_n, \tag{A.3}$$

$$R_t \triangleq \mathrm{Cov}(Z_{n+1}|Y_{1:n}) = \mathrm{Cov}(g_0 + GZ_n + \epsilon_n^X|Y_{1:n}) = G\Sigma_nG^T + \Sigma_Z. \tag{A.4}$$

From Lemma A.1 we have:

$$\mu_{n+1} = a_n - R_nF^T\left(\Sigma_Y + FR_nF^T\right)^{-1}(Fa_n - Y_n + f_0), \tag{A.5}$$

$$\Sigma_{n+1} = R_n - R_nF^T\left(\Sigma_Y + FR_nF^T\right)^{-1}FR_n. \tag{A.6}$$

The term $R_tF^T\left(\Sigma_Y + FR_tF^T\right)^{-1}$ is commonly called the Kalman gain. The conditional log-likelihood is

$$\log P(Y_n|Y_{1:n-1}) = \log P_N(Y_n; f_0 + Fa_n, \Sigma_Y + FR_nF^T), \tag{A.7}$$

where P_N denotes the multivariate normal density.

Backward smoothing Next, we want to derive the *smoothed* distribution of Z_n conditional on all observations, written $Z_n|Y_{1:N}$. This is also efficiently calculated by recursing backward from

the end of the filtering step.

Using our lemma,

$$h_n \triangleq \mathbb{E}(Z_n|Z_{n+1}, Y_{1:N}) = E(Z_n|Z_{n+1}, Y_{1:n}), \tag{A.8}$$
$$= \mu_n - \Sigma_n G^T \left(\Sigma_Z + G\Sigma_n G^T\right)^{-1} (g_0 + G\mu_t - Z_{n+1}), \tag{A.9}$$
$$= \mu_n - \Sigma_n G^T R_{n+1}^{-1} (g_0 + G\mu_t - Z_{n+1}), \tag{A.10}$$
$$H_n \triangleq \text{Cov}(Z_n|Z_{n+1}, Y_{1:N}) = \text{Cov}(Z_n|Z_{n+1}, Y_{1:n}), \tag{A.11}$$
$$= \Sigma_n - \Sigma_n G^T R_{n+1}^{-1} G\Sigma_n. \tag{A.12}$$

We can then integrate out Z_{n+1} to get the desired quantities.

$$\tilde{\mu}_n \triangleq \mathbb{E}(Z_n|Y_{1:N}) = E(h_n|Y_{1:n}), \tag{A.13}$$
$$= \mu_n - \Sigma_n G^T R_{n+1}^{-1} (g_0 + G\mu_t - \mathbb{E}(Z_{n+1}|Y_{1:N})), \tag{A.14}$$
$$= \mu_n - \Sigma_n G^T R_{n+1}^{-1} (g_0 + G\mu_t - \tilde{\mu}_{n+1}) \tag{A.15}$$
$$\tilde{\Sigma}_n \triangleq \text{Cov}(Z_n|Y_{1:N}) = \mathbb{E}(H_n|Y_{1:N}) + \text{Cov}(h_n|Y_{1:N}) \tag{A.16}$$
$$= H_n + \Sigma_n G^T R_{n+1}^{-1} \tilde{\Sigma}_{n+1} R_{n+1}^{-1} G^T \Sigma_n \tag{A.17}$$
$$= \Sigma_n - \Sigma_n G^T R_{n+1}^{-1} \left(R_{n+1} - \tilde{\Sigma}_{n+1}\right) R_{n+1}^{-1} G^T \Sigma_n \tag{A.18}$$

Finally, we need to draw from the conditional distribution $Z_{n+1}|Z_n, Y_{1:N}$ for downstream tasks and for use within the proposal distribution for SMC. The conditional mean and variance are:

$$\mathbb{E}(Z_{n+1}|Z_n, Y_{1:n}) = \tilde{\mu}_{n+1} - V_{n+1,n}\tilde{\Sigma}_n^{-1}(\tilde{\mu}_n - Z_n) \tag{A.19}$$
$$\text{Cov}(Z_{n+1}|Z_n, Y_{1:n}) = \tilde{\Sigma}_{n+1} - V_{n+1,n}\tilde{\Sigma}_n^{-1} V_{n+1,n}^T. \tag{A.20}$$

The term $V_{n+1,v}$ is the conditional covariance between Z_{n+1} and Z_n, which is derived as follows:

$$V_{n+1,n} \triangleq \text{Cov}(Z_{n+1}, Z_n|Y_{1:N}) \tag{A.21}$$
$$= \text{Cov}(Z_{n+1}, \mathbb{E}(Z_n|Z_{n+1}, Y_{1:N})|Y_{1:N}) \tag{A.22}$$
$$= \text{Cov}(Z_{n+1}, h_n|Y_{1:N}) \tag{A.23}$$
$$= \text{Cov}(Z_{n+1}, \mu_n - \Sigma_n G^T R_{n+1}^{-1}(g_0 + G\mu_n - Z_{n+1})|Y_{1:N}) \tag{A.24}$$
$$= \text{Cov}(Z_{n+1}, \Sigma_n G^T \mathbb{R}_{n+1}^{-1} Z_{n+1}|Y_{1:N}) \tag{A.25}$$
$$= \text{Cov}(Z_{n+1}, Z_{n+1}|Y_{1:N}) R_{n+1}^{-1} G^T \Sigma_n \tag{A.26}$$
$$= \tilde{\Sigma}_{n+1} R_{n+1}^{-1} G^T \Sigma_n. \tag{A.27}$$

A.2 Sequential Monte Carlo

Sequential Monte Carlo (SMC) is a technique for estimating a state-space model in the case that either the state transition or the observation kernel is non-linear and/or non-Gaussian, preventing the use of the Kalman filter. The states $Z_{1:N}$ and observations $Y_{1:N}$ come from the following model:

$$\begin{aligned} Z_1 &\sim \mu_\theta, \\ Z_n|Z_{n-1} &\sim f_\theta(\cdot|Z_{n-1}), \\ Y_n|Z_n &\sim g_\theta(\cdot|Z_n). \end{aligned} \tag{A.28}$$

Like the Kalman filter, the goal of SMC is to obtain an estimate of the filtered distribution $Z_n|Y_{1:n}$ via importance sampling. Samples from the smoothed distribution $Z_n|Y_{1:N}$ can be obtained via the forward-filtering backward sampling (FFBS) algorithm from Godsill et al. (2004). We briefly summarize the steps for forward filtering and backward smoothing below.

A.2.1 Forward filtering

Here, we briefly summarize the steps for forward filtering. Because our eventual goal is to sample the smoothed distribution, we utilize twisted Sequential Monte Carlo to target different intermediate distributions than the filtered distribution. See Naesseth et al. (2019) for a more detailed overview.

Initialization For particles $1, \ldots, K$, we sample

$$Z_{i,1} \sim q_1(\cdot), \tag{A.29}$$

where q_1 is the initial proposal distribution. We then calculate importance weights:

$$w_{i,1} = \frac{\mu_\theta(Z_{i,1}) g_\theta(Y_1|Z_{i,1})}{q_1(Z_1)}. \tag{A.30}$$

Together, the collection of particles and weights $\{(Z_{i,1}, w_{i,1})\}$ target the distribution $Z_1|Y_1$. Concretely, this means that, as $K \to \infty$, the weighted average over particles converges to the true expectation:

$$\lim_{K\to\infty} \frac{1}{\sum_{i=1}^K w_{i,1}} \sum_{i=1}^K w_{i,1} h(Z_{i,1}) = \mathbb{E}(h(Z_1)|Y_1) \tag{A.31}$$

In general, these importance sampling estimates are only asymptotically unbiased (consistent) as $K \to \infty$. However, an unbiased estimate of the likelihood $\mathcal{L}_1 \approx P(Y_1)$ can be obtained by

averaging the weights:

$$\mathcal{L}_1 = \frac{1}{K}\sum_{i=1}^{K} w_{i,1}. \tag{A.32}$$

Filtering For $n > 1$, we start with a collection of particles $(Z_{i,n-1}, w_{i,n-1})$ targeting the distribution $Z_n|Y_n$. We start by normalizing the weights attached to each particle:

$$\pi_{i,n-1} = \frac{w_{i,n-1}\psi_{n-1}(Z_{i,n-1})}{\sum_{j=1}^{k} w_{j,n-1}}. \tag{A.33}$$

We further modify the weights with a non-negative function ψ_{n-1}.

$$\pi^{\star}_{i,n-1} = \frac{\pi_{i,n-1}\psi_{n-1}(Z_{i,n-1})}{\sum_{j=1}^{k} \pi_{i,n-1}\psi_{n-1}(Z_{i,n-1})}. \tag{A.34}$$

We discuss later the optimal choice for ψ_{n-1}. If the effective sample size (ESS) of $\{\pi_{i,n-1}\}$ is below a set threshold, we resample:

$$\begin{aligned} \tilde{Z}_{i,n-1} &\sim \sum_{j=1}^{K} \pi^{\star}_{i,n-1}\delta_{Z_{j,n-1}}(\cdot), \\ \tilde{w}_{i,n-1} &= 1. \end{aligned} \tag{A.35}$$

Otherwise, we do not:

$$\begin{aligned} \tilde{Z}_{i,n-1} &= Z_{i,n-1} \\ \tilde{w}_{i,n-1} &= \pi^{\star}_{i,n-1}. \end{aligned} \tag{A.36}$$

New particles are proposed from the proposal distribution q_n:

$$Z_{i,n} \sim q_n(\cdot|\tilde{Z}_{i,n-1}). \tag{A.37}$$

Importance weights are calculated:

$$w_{i,n} = \tilde{w}_{i,n-1}\frac{f_\theta(Z_{i,n}|\tilde{Z}_{i,n-1})g_\theta(Y_n|Z_{i,n})}{q_n(Z_{i,n}|\tilde{Z}_{i,n-1})\psi_{n-1}(\tilde{Z}_{i,n-1})}. \tag{A.38}$$

An unbiased estimate of the conditional log-likelihood is:

$$\mathcal{L}_n = \left(\frac{1}{K}\sum_{i=1}^{K} w_{i,n}\right)\left(\sum_{i=1}^{K} \psi_{n-1}(Z_{i,n-1})\pi_{i,n-1}\right). \tag{A.39}$$

A.2.2 Backward smoothing

We briefly summarize the forward filtering backward sampling algorithm from Godsill et al. (2004). We start with a sequence of particles and weights $(Z_{i,n}, w_{i,n})$ targeting the filtered distribution $\mathcal{P}(Z_{i,n}|Y_{1:n})$. Recursing backwards, at index n, we suppose we have a sample $\tilde{Z}_{n+1} \sim \mathcal{P}(\cdot|Y_{1:N})$. For each i, we calculate the following importance weights and normalize them:

$$\begin{aligned} w_{i,n|n+1} &= w_{i,n} g_\theta(Z_{i,n}, \tilde{Z}_{n+1}) \\ \pi_{i,n|n+1} &= \frac{w_{i,n|n+1}}{\sum_{j=1}^K w_{j,n|n+1}}. \end{aligned} \tag{A.40}$$

A new sample $\tilde{Z}_j$ is then drawn from the empirical distribution $\sum_{i=1}^K \pi_{i,n|n+1}\delta_{Z_{i,n}}(\cdot)$. This procedure is repeated for each draw required from the smoothed distribution.

A.2.3 Derivation of optimal preweight and proposal

When estimating ArgoSSM, we set $q_n(Z_n|Z_{n-1}) \equiv \mathcal{P}_{\text{AR}}(Z_n|Z_{n-1}, Y_A)$ and $\psi_n(Z_n) \equiv \mathcal{P}_{\text{AR}}(Y_{A>n}|Z_n)$. These choices are motivated by Guarniero et al. (2017), since these are the optimal choice if we exclude the PV and ice-cover components from our model. We calculate each of the log-likelihood estimates $\mathcal{L}_n$ at each step $n = 1, \ldots, N$, and show that their product is equal to $\mathcal{P}(Y_A)$. In this scenario, since both the ArgoSSM and AR model follow the same probability law, we write the different distributions $(\mu_\theta, f_\theta, g_\theta, \mathcal{P}_{\text{AR}})$ as simply p_θ.

Case 1: $n = 1$ After sampling from initial state $Z_{i,1} \sim q_1 \equiv p_\theta(\cdot|Y_{1:N})$, we calculate the initial weights $w_{i,1}$:

$$\begin{aligned} w_{i,1} &= \frac{p_\theta(Z_{i,1})p_\theta(Y_1|Z_{i,1})}{p_\theta(Z_{i,1}|Y_{1:N})} \\ &= \frac{p_\theta(Z_{i,1})p_\theta(Y_1|Z_{i,1})p_\theta(Y_{1:N})}{p_\theta(Z_{i,1})p_\theta(Y_{1:N}|Z_{i,1})} \\ &= p_\theta(Y_{1:N})\frac{p_\theta(Y_1|Z_{i,1})}{p_\theta(Y_{1:N}|Z_{i,1})} \\ &= p_\theta(Y_{1:N})\frac{1}{p_\theta(Y_{2:N}|Z_{i,1})} \end{aligned}$$

Then, we have our estimated log-likelihood $\mathcal{L}_1$:

$$\mathcal{L}_1 = p_\theta(Y_{1:N})\frac{1}{K}\sum_{i=1}^K \frac{1}{p_\theta(Y_{2:N}|Z_{i,1})}$$

Moreover, the normalized weights $\pi_{i,1}$ are equal to:

$$\pi_{i,1} = \frac{1}{\sum_{i=1}^{K} \frac{1}{p_\theta(Y_{2:N}|X_{i,1},V_{i,1})}} \frac{1}{\mathcal{P}(Y_{2:N}|X_{i,1},V_{i,1})}.$$

Case 2: $n = 2, \ldots, N-1$ We use the inductive assumption that the previous normalized weights $\pi_{i,n-1}$ are equal to the following:

$$\pi_{i,n-1} = \frac{1}{\sum_{j=1}^{K} \frac{1}{p_\theta(Y_{n:N}|Z_{j,n-1})}} \frac{1}{p_\theta(Y_{n:N}|Z_{i,n-1})}.$$

The pre-weights are equal to:

$$\psi_{n-1}(Z_{i,n-1}) \equiv p_\theta(Y_{n:N}|Z_{i,n-1}).$$

By choice of pre-weights, the normalized resampling weights $\pi^\star_{i,n-1}$ are equal to 1. Thus, the effective sample size threshold is not triggered, and no resampling is performed.

$$\pi_{i,n-1}\psi_{n-1}(Z_{i,n-1}) = \frac{1}{\sum_{j=1}^{K} \frac{1}{p_\theta(Y_{n:N}|Z_{j,n-1})}},$$
$$\pi^\star_{i,n-1} \equiv 1.$$

After propagating the particles forward to n, the importance weights at time n are:

$$\begin{aligned}
w_{i,n} &= \frac{p_\theta(Z_{i,n}|Z_{i,n-1})p_\theta(Y_n|Z_{i,n})}{p_\theta(Z_{i,n}|Z_{i,n-1},Y_{1:N})\psi(Z_{i,n-1})} \\
&= \frac{p_\theta(Z_{i,n}|Z_{i,n-1})p_\theta(Y_n|Z_{i,n})p_\theta(Y_{n:N}|Z_{i,n-1})}{p_\theta(Z_{i,n}|Z_{i,n-1})p_\theta(Y_{n:N}|Z_{i,n},Z_{i,n-1})\psi(Z_{i,n-1})} \\
&= \frac{p_\theta(Y_n|Z_{i,n})p_\theta(Y_{n:N}|Z_{i,n-1})}{p_\theta(Y_{n:N}|Z_{i,n},Z_{i,n-1})\psi(Z_{i,n-1})} \\
&= \frac{p_\theta(Y_n|Z_{i,n})}{p_\theta(Y_{n:N}|Z_{i,n},Z_{i,n-1})} \\
&= \frac{p_\theta(Y_n|Z_{i,n})}{p_\theta(Y_{n:N}|Z_{i,n})} \\
&= \frac{1}{p_\theta(Y_{n+1:N}|Z_{i,n})}
\end{aligned}$$

This leads to the following estimate of the conditional log-likelihood $p_\theta(Y_n|Y_{1:n-1})$ (Pitt, 2002; Pitt et al., 2012):

$$\begin{aligned}\mathcal{L}_n &= \frac{1}{K}\sum_{i=1}^{K}\frac{1}{p_\theta(Y_{n+1:N}|Z_{i,n})}\sum_{i=1}^{K}\pi_{i,n-1},\\ &= \frac{\sum_{i=1}^{K}\frac{1}{p_\theta(Y_{n+1:N}|Z_{i,n})}}{\sum_{i=1}^{K}\frac{1}{p_\theta(Y_{n:N}|Z_{i,n-1})}}.\end{aligned}$$

We then calculate the estimated log-likelihood of observations up to and including n, $p_\theta(Y_{1:n})$:

$$\begin{aligned}\mathcal{L}^n = \mathcal{L}^{n-1}L_n &= p_\theta(Y_{1:N})\frac{1}{K}\sum_{i=1}^{K}\frac{1}{p_\theta(Y_{n:N}|Z_{i,n-1})}\frac{\sum_{i=1}^{K}\frac{1}{p_\theta(Y_{n+1:N}|Z_{i,n})}}{\sum_{i=1}^{K}\frac{1}{p_\theta(Y_{n:N}|Z_{i,n-1})}}\\ &= p_\theta(Y_{1:N})\frac{1}{K}\sum_{i=1}^{K}\frac{1}{p_\theta(Y_{n+1:N}|Z_{i,n})}\end{aligned}$$

Case 3: $n = N$ For the last time step at N, the derivations are almost the same as in Case 2. Since $Y_{n:N} = Y_n$, the terms in the numerator and denominator are equal, meaning:

$$w_{i,N} = 1.$$

Consequently, the conditional log-likelihood estimate is:

$$\begin{aligned}\mathcal{L}_N &= \frac{1}{K}\left(\sum_{i=1}^{K}1\right)\frac{K}{\sum\frac{1}{p_\theta(Y_N|Z_{i,N-1})}},\\ &= \frac{K}{\sum\frac{1}{p_\theta(Y_N|Z_{i,N-1})}}.\end{aligned}$$

Finally, the estimated log-likelihood of all observations (i.e. up to N) is exactly equal to the likelihood:

$$\begin{aligned}\mathcal{L} &= \prod_{n=1}^{N}\mathcal{L}_N = \mathcal{L}^{N-1}\mathcal{L}_N\\ &= p_\theta(Y_{1:N})\frac{1}{K}\sum_{i=1}^{K}\frac{1}{p_\theta(Y_N|Z_{i,N-1})}\frac{K}{\sum\frac{1}{p_\theta(Y_N|Z_{i,N-1})}}\\ &= p_\theta(Y_{1:N}).\end{aligned}$$

A.3 Derivation of ice update

We show how the filtered probabilities of the ice states can be integrated out to obtain Equation 2.10. First, we apply Bayes' rule to the filtered distribution of S_n given all previous positions $X_{1:n}$ and missingness indicators $A_{1:n}$ (where $A_n \in \{0, 1\}$):

$$\begin{aligned}\mathcal{P}(S_n = k|X_{1:n}, A_{1:n}) &= \frac{\mathcal{P}(A_n|S_n = k, X_{1:n})\mathcal{P}(S_n = k|X_{1:n}, A_{1:n-1})}{\sum_{k'} \mathcal{P}(A_n|S_n = k', X_{1:n})P(S_n = k'|X_{1:n}, A_{1:n-1})}. \\ &= \frac{g_\theta(A_n|S_n = k)\mathcal{P}(S_n = k|X_{1:n}, A_{1:n-1})}{\sum_{k'} g_\theta(A_n|S_n = k')P(S_n = k'|X_{1:n}, A_{1:n-1})}.\end{aligned} \tag{A.41}$$

The distribution $\mathcal{P}(A_n|S_n = k)$ is as summarized in Section 2.2.1 and repeated here:

$$\mathcal{P}(A_n = 1|S_n = k) = \begin{cases} 0 \text{ if } k \in \{0, 1, 2\} \\ 1 - p_{\text{MAR}} \text{ if } k = 3 \end{cases}. \tag{A.42}$$

Next, we derive the distribution $\mathcal{P}(S_n = k|X_{1:n}, A_{1:n-1})$ by integrating over S_{n-1}:

$$\mathcal{P}(S_n = k|X_{1:n}, V_{1:n}, A_{1:n-1}) = \sum_{k'} \mathcal{P}(S_n = k|X_n, S_{n-1} = k')\mathcal{P}(S_{n-1} = k'|X_{1:n}, V_{1:n}, A_{1:n-1}) \tag{A.43}$$

$$\begin{aligned}\mathcal{P}(S_{n-1} = k'|X_{1:n}, V_{1:n}, A_{1:n-1}) &= \frac{\mathcal{P}(S_{n-1} = k'|X_{1:n-1}, V_{1:n-1}, A_{1:n-1})\mathcal{P}(X_n, V_n|S_{n-1} = k', X_{1:n-1}, V_{1:n-1}, A_{1:n-1})}{\sum_{k''} \mathcal{P}(S_{n-1} = k''|X_{1:n-1}, A_{1:n-1})\mathcal{P}(X_n, V_n|S_{n-1} = k'', X_{1:n-1}, A_{1:n-1})} \\ &= \frac{\mathcal{P}(Sn - 1 = k'|X_{1:n-1}, V_{1:n-1}, A_{1:n-1})\mathcal{P}(X_n, V_n|X_{n-1}, V_{n-1})}{\sum_{k''} \mathcal{P}(S_{n-1} = k''|X_{1:n-1}, A_{1:n-1})\mathcal{P}(X_n, V_n|X_{n-1}, V_{n-1})} \\ &= \mathcal{P}(S_{n-1} = k'|X_{1:n-1}, V_{1:n-1}, A_{1:n-1}).\end{aligned} \tag{A.44}$$

Thus, all that is required is recording the values of $\mathcal{P}(S_{n-1} = k|X_{1:n}, V_{1:n}, A_{1:n-1})$ for each possible value of k. When applying measurement weights, we calculate the probability $\mathcal{P}(A_n|X_{1:n}, V_{1:n})$ by integrating over each state $S_n = k$:

$$\begin{aligned}\mathcal{P}(A_n = 1|X_{1:n}, V_{1:n}, A_{1:n-1}) &= \sum_k \mathcal{P}(A_n = 1|S_n = k)\mathcal{P}(S_n = k|X_{1:n}, V_{1:n}, A_{1:n-1}) \\ &= \sum_k \mathcal{P}(A_n = 1|S_n = k)\mathcal{P}(S_n = k|X_{1:n-1}, V_{1:n-1}, A_{1:n-1}).\end{aligned} \tag{A.45}$$

.

A.4 Calculation of potential vorticity

Here we will describe how the function $\text{PV}(X_n, V_{n-1})$, which appears in the ArgoSSM velocity update, is calculated from known ocean depth and latitude. We use bathymetry (ocean depth) data exported from the Southern Ocean State Estimate (SOSE) (Verdy and Mazloff, 2017). Potential vorticity is approximated by the expression $f(x_0)/h(x_0)$ where f is the Coriolis parameter and $h(x_0)$ is the water depth at location $x_0 = (x_{0,1}, x_{0,2})$. The values of $f(x_0)$ do not depend on longitude $x_{0,1}$ and are defined by

$$f(x_0) = 2\Omega \sin\left(x_{0,2} \cdot \frac{\pi}{180}\right)$$

where $x_{0,2}$ is the latitude (in degrees) and $\Omega = 7.292115 \cdot 10^{-5}$. See Talley et al. (2011) for more specifics on how PV is defined and its interpretation. While PV can be directly computed from the provided quantities, the resulting field is too rough, and as in Chamberlain et al. (2018) we smooth the bathymetry to improve the results. In addition, we provide a direct, smoothed estimate of the gradient of PV that is supplied to ArgoSSM.

From the quotient rule, an estimate of PV and its gradient can be written

$$\begin{aligned}
\text{PV}(x_0) &= \frac{f(x_0)}{\hat{h}(x_0)} \\
\nabla\text{PV}(x_0) &= \frac{2\pi\Omega \cos\left(x_{0,2} \cdot \frac{\pi}{180}\right)}{180 \cdot \hat{h}(x_0)} \begin{pmatrix} 0 \\ 1 \end{pmatrix} - \frac{f(x_0)}{\hat{h}(x_0)^2} \nabla\hat{h}(x_0)
\end{aligned}$$

where $\hat{h}(x_0)$ is a smoothed estimate of the ocean depth and $\nabla\hat{h}(x_0) \in \mathbb{R}^2$ an estimate of its gradient. We estimate these quantities using local quadratic regression as described below.

Let x_n be the location of the nth grid point, with $x_{n,1}$ and $x_{n,2}$ be its longitude and latitude, respectively. For a fixed location x_0, we compute $\hat{h}(x_0) = \hat{\beta}_0$, $\nabla\hat{h}(x_0) = (\hat{\beta}_{1,1}, \hat{\beta}_{2,1})$ which come from the solution $\hat{\boldsymbol{\beta}} = (\hat{\beta}_0, \hat{\beta}_{1,1}, \hat{\beta}_{2,1}, \hat{\beta}_{1,2}, \hat{\beta}_{2,2})$ of

$$\min_{\boldsymbol{\beta}} \sum_{n=1}^{N} K_b(x_n - x_0) \left(h(x_n) - \beta_0 - \sum_{d=1}^{2} \sum_{j=1}^{2} (x_{n,d} - x_{0,d})^j \beta_{d,j} \right)^2 \tag{A.46}$$

where $K_b(x) = \max\left\{0, \frac{1}{b}\frac{3}{4}(1 - \|x\|^2 / b^2)\right\}$ is the Epanechnikov kernel with bandwidth b based on distance in kilometers. The quadratic terms in Equation A.46 ensure that the gradient estimate is stable, and the solution is computed directly by weighted least squares. For more information on local polynomial estimation, see Fan and Gijbels (1996).

We have used $b = 300$ kilometers, though this could be more carefully chosen, and other band-

widths can easily be integrated into our analysis pipeline. These estimates are precomputed on a fine grid, then bilinearly interpolated on the fly in ArgoSSM, which uses the estimated PV to inform the velocity in ArgoSSM. Since the direction perpendicular to the gradient is only identifiable up to a sign, we take the direction closer to the direction of the previous velocity V_{n-1}.

A.5 Derivation of PV update

In this section, we derive the PV-informed velocity transition equation shown in Equation 2.5. First, we restate Equation 2.3 alongside the PV constraint:

$$V_n|X_n, V_{n-1} \sim \mathcal{N}\left((1-\alpha^{\Delta t_n})v_0 + \alpha^{\Delta t_n} V_{n-1}, \Delta t_n \Sigma_V\right), \tag{A.47}$$

$$b|X_n, V_n \sim \mathcal{N}(\nabla \mathrm{PV}(X_n) \cdot \Delta t_n V_n, (\Delta t_n)^2 \sigma_{PV}^2). \tag{A.48}$$

Conditioning on $b = 0$, and applying Lemma A.1, we get:

$$V_n|X_n, V_{n-1}, \{b=0\} = B\left((1-\alpha^{\Delta t_n})v_0 + \alpha^{\Delta t_n} V_{n-1}\right) + B^{\frac{1}{2}}\epsilon_n^V, \tag{A.49}$$

where

$$B = \left(I + \frac{1}{\sigma_{PV}^2}(\Delta t_n \Sigma_V)\nabla \mathrm{PV}(X_n)\nabla \mathrm{PV}(X_n)'\right)^{-1} \tag{A.50}$$

$$= \frac{1}{(\sigma_{PV}^2 + \|\nabla \mathrm{PV}(X_n)\|_2^2)}\Delta t_n \Sigma_V \nabla \mathrm{PV}(X_n)\nabla \mathrm{PV}(X_n)'. \tag{A.51}$$

B is a matrix that encompasses the effect of PV conservation. As $\sigma_{PV}^2 \to 0$, Equation A.51 shows that B converges to a projection onto the linear subspace where $\nabla PV(X_n)' V_n = 0$.

At a specific location, the PV is calculated as a function of the depth and latitude. A body freely circulating in the ocean tends to maintain PV. Thus, absent of other forces, a float will move perpendicular to the gradient of PV.

A.6 Approach for velocity

This section details the circulation estimation, which is mostly similar to the analysis of temperature and salinity. First, we describe the selected floats used in the analysis. Most Argo floats' parking depth is at 1000 meters, though floats often drift at 800 meters in the Weddell Gyre, which Reeve et al. (2019) consider. Since there are more floats with linear interpolated values at 1000 meters, we focus on these floats only, keeping only trajectories with parking depths between 950

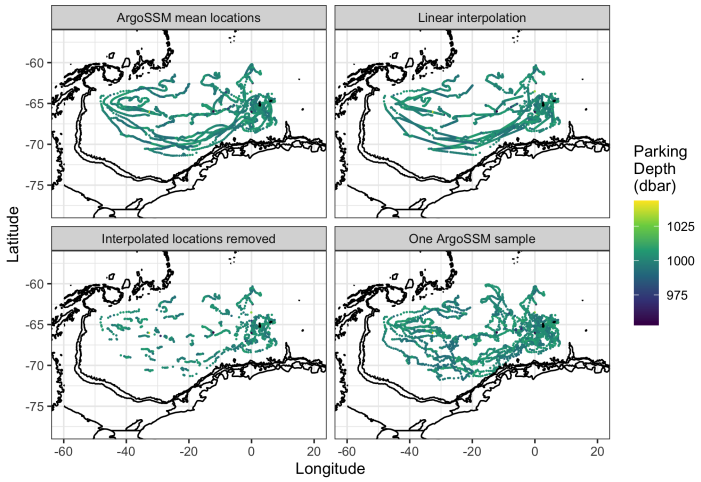


Figure A.1: Float trajectories used for velocity estimation, with (from top left, clockwise) ArgoSSM mean, linearly interpolated values, a sample from ArgoSSM, and with missing trajectories removed.

and 1050 meters. This leaves 26 floats from which to estimate velocities, plotted in Figure A.1. Also, we do not correct for drifting at the ocean surface during data transmission as is done in Reeve et al. (2019). Conceivably such an adjustment could be built into the ArgoSSM framework, but doing so is not straightforward or the main focus of this work.

Our approach for the meridional (north-south) and zonal (east-west) Argo velocities follows similarly from those presented in the temperature and salinity field estimation. We assume that the directions are independent; this is a simplification. Due to a reduction in the amount of data used, we make the following changes: a locally-constant estimator was used for the mean estimation with a bandwidth of 400 kilometers, and the use of time in the covariance structure was removed so that we estimate one time-averaged value at each location. In addition to using linear interpolation and ArgoSSM samples, we complete the analysis with linear interpolated velocities removed in a manner similar to Reeve et al. (2019).

APPENDIX B

Appendix for Chapter 3

B.1 Architecture details

In this section, we elaborate on the specific architecture used in each neural network encoder inside BLISS.

Detection encoder The detection encoder introduced in Equation 3.8, $g_\phi(x_{i,t})$ takes a padded tile $x_{i,t}$ as input. This padded tile has two channels: the astronomical image itself and the background value at each tile. It outputs the probability that a source is in the tile $\alpha_{i,t}$ and the mean and variance of the location of that source $\mu^\ell_{i,t}$, $\Sigma^\ell_{i,t}$. An initial convolution layer has an output of 8 channels, followed by batch normalization applied across the channel dimension (BatchNorm2D) (Ioffe and Szegedy, 2015) and the ReLU function $\mathrm{ReLU}(x) = \max(x, 0)$. We then apply three "ConvBlock" layers, shown in Figure B.1(a), which optionally halve the height and width of the input while doubling the number of channels. This output is then flattened and passed into a fully-connected neural network "FCNet", shown in Figure B.1(b).

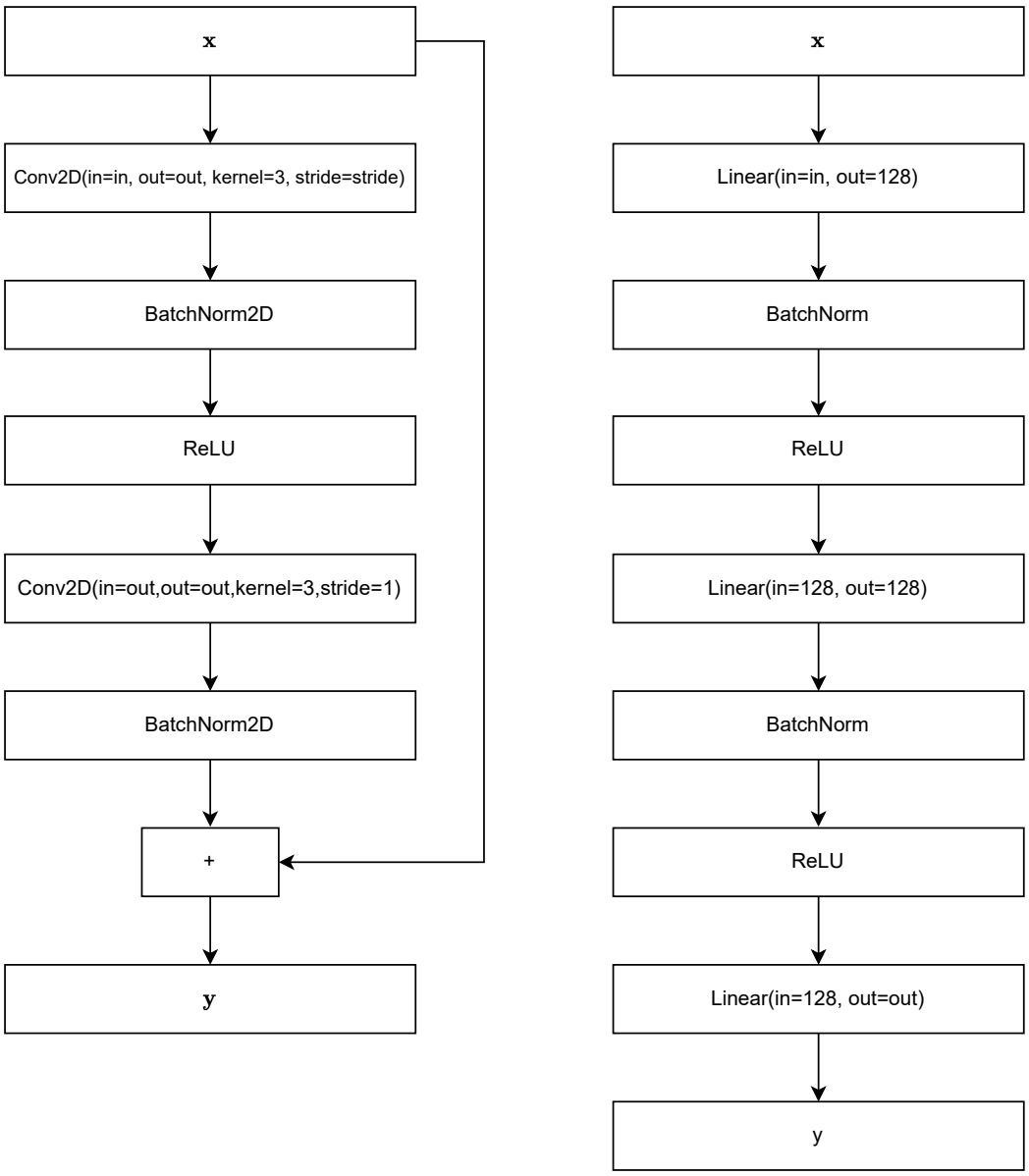
Binary encoder The binary encoder neural network introduced in Equation 3.9 has an almost identical architecture to the detection encoder, with two key differences. First, the padded tile is centered around the location of the object $\ell_{i,t}$ using bilinear interpolation. Second, the output of the "FCNet" is the probability that the object is a star $\beta_{i,t}$.

Galaxy variational autoencoder The galaxy variational autoencoder, shown in Figure B.3, consists of an encoder network and a decoder network. The encoder network consists of two convolutional 2d layers with a kernel size of 5 and a stride of 3 that double the number of channels. This is followed by a flattening step and two linear layers. Between each layer, we use the LeakyReLU

non-linearity citepmaasRectifierNonlinearitiesImprove2013:

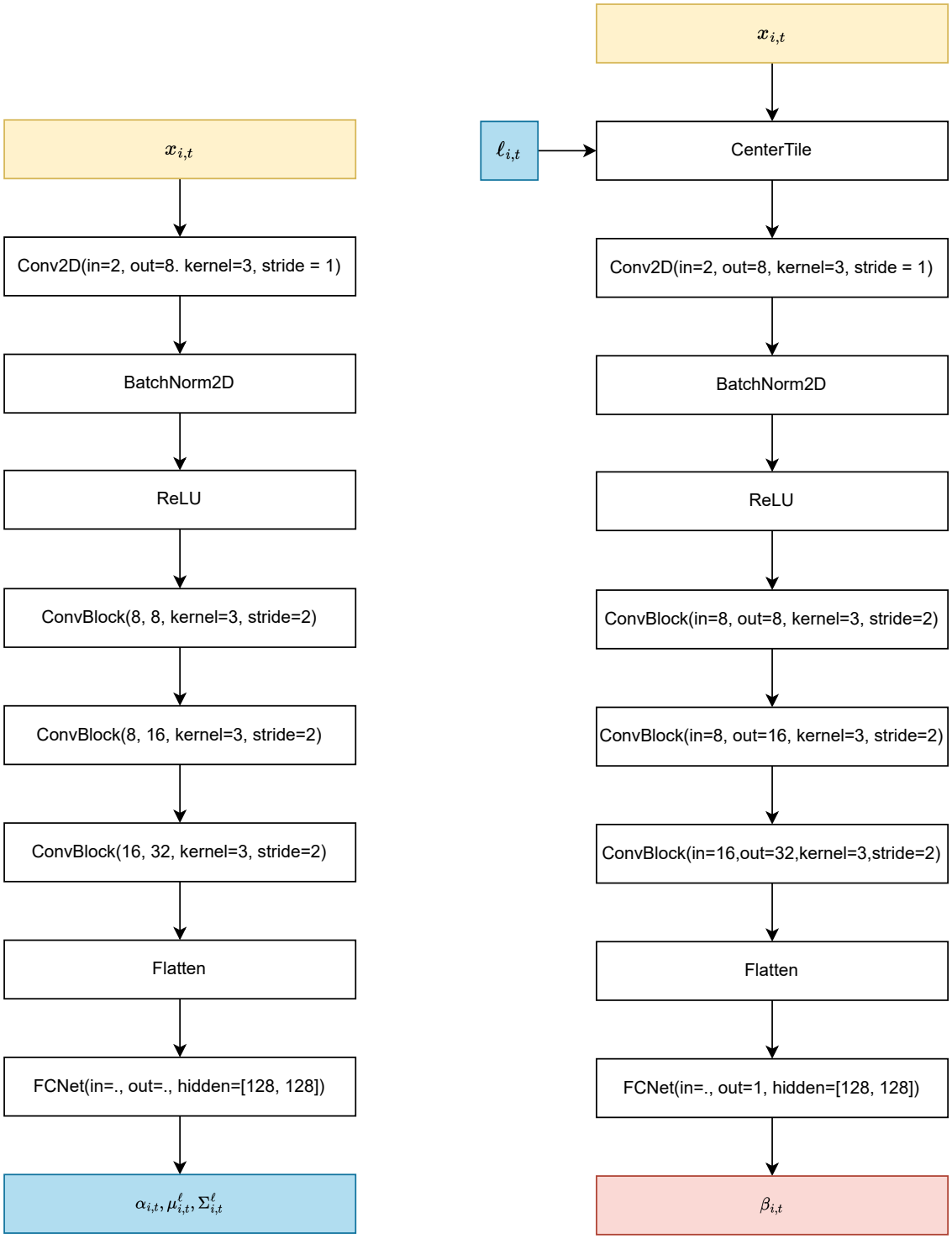
$$\text{LeakyReLU}(x) = \begin{cases} x \text{ if } x \geq 0 \\ 0.02x \text{ otherwise.} \end{cases}$$

The output of the final layer has double the dimension of the latent variable z since it is both the mean μ_z and scale σ_Z. To ensure σ_z is positive, we apply the softplus function $\log(1 + \exp(x))$. For the decoder, we use an analogous architecture to the encoder in reverse order; i.e. using ConvTranspose2D in place of Conv2D. For both the encoder and decoder, we re-parameterize the neural network weights using weight normalization for improved training (Salimans and Kingma, 2016).



(a) ConvBlock (b) FCNet

Figure B.1: Building block layers for the BLISS encoder. (a) The ConvBlock layer features two stacks of convolution layers followed by batch normalization and ReLU, with an additive skip connection at the end. The first convolutional layer may have a stride greater than 1, leading to an image with half the height and width; in this case, a convolution layer is applied to the input before the skip connection. (b) The FCNet layer is a standard two-layer multi-layer perceptron (MLP) with batch normalization and ReLU between dense linear layers.



(a) Detection encoder

(b) Binary encoder

Figure B.2: BLISS detection encoder architecture.

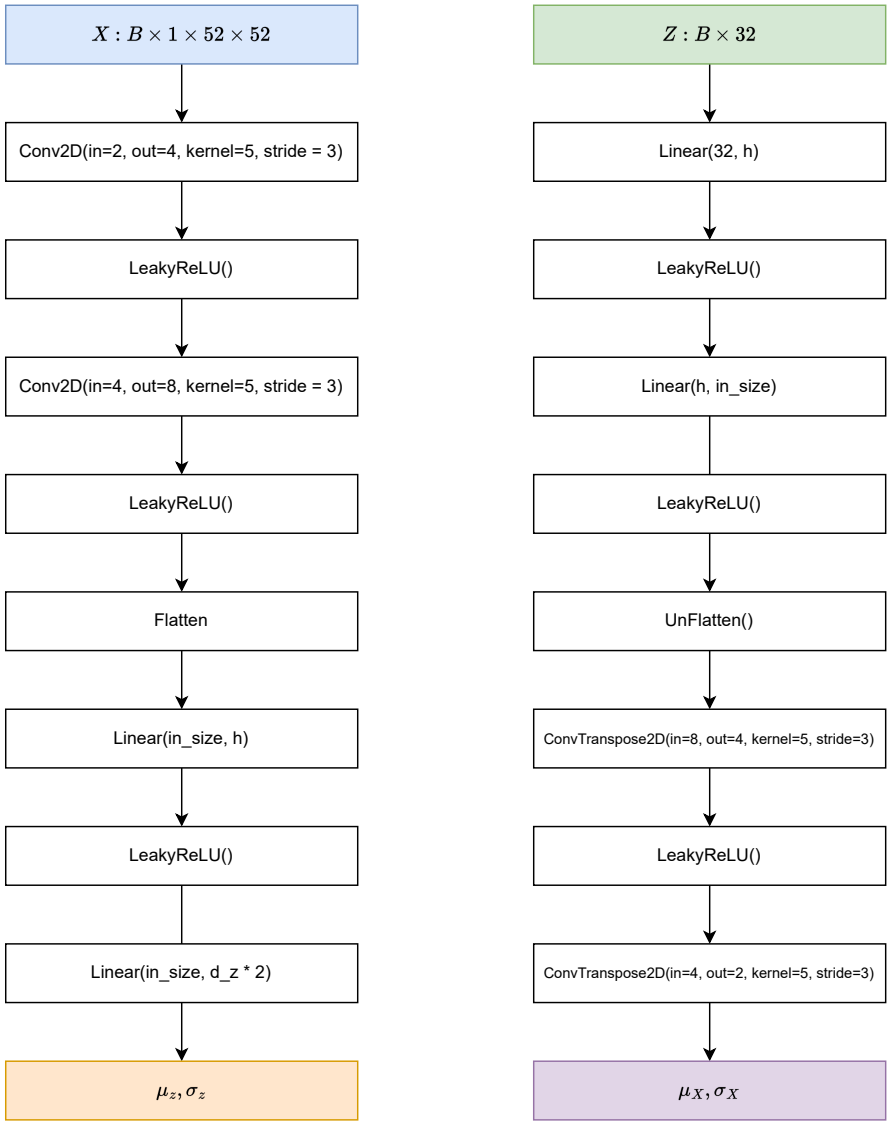


Figure B.3: Diagram of variational autoencoder architecture used for encoding and decoding centered galaxy images as part of BLISS.

APPENDIX C

Appendix for Chapter 4

C.1 Related Works

Our method involves combining in a novel way ideas from several different literatures. As such, there is a large body of related work, each of which approaches the problems we consider from somewhat different perspectives. Here, we summarize some of the most related. Table C.1 provides an overview of the comparisons of these methods.

Method	Conservative	UQ	Inference with different Initial Conditions	Inference with different PDE coefficients	Resolution independent
Numerical methods	✓	✗	✗	✗	✗
PINNs	✗	✗	✗	✗	✓
Neural Operators	✗	✗	✓	✓	✓
Conservative ML models	✓	✗	✓	✗	✗
PROBCONSERV (our approach)	✓	✓	✓	✓	✓

Table C.1: Summary of different properties of numerical and SciML methods for physical systems.

C.1.1 Numerical methods

Numerical methods aim to approximate the solution to partial differential equations (PDEs) by first discretizing the spatial domain Ω into N gridpoints $\{x_i\}_{i=1}^N$ with spatial step size Δx. Then, at each time step, we integrate the resulting semi-discrete ODE in time with temporal step size Δt to iteratively compute the solution at final time T, i.e., $\{u(T, x_i)\}_{i=1}^N$. By the Lax Equivalence theorem for linear problems, convergence to the true solution, i.e., the norm of the error tending to zero, can be proven to occur when $\Delta t, \Delta x \to 0$ ($N \to \infty$) for methods that are both stable and consistent (LeVeque, 2007).

Finite volume methods. Finite volume methods are designed for conservation laws. These methods divide the domain into control volumes, where the integral form of the governing equation

is solved (LeVeque, 1990, 2002). By solving the integral form at each control volume, these methods enforce flux continuity, i.e., that the out-flux of one cell is equal to the in-flux of its neighbor. This results in local conservation, which guarantees global conservation over the entire domain. Maddix et al. (2018b) show that the degenerate parabolic Generalized Porous Medium Equation (GPME) has presented challenges for classical averaged-based finite volume methods, e.g., arithmetic and harmonic averaging. These numerical artifacts include artificial temporal oscillations, and locking or lagging of the shock position. To eliminate these artifacts on the more challenging Stefan problem, Maddix et al. (2018a) show that information about the shock location needs to be incorporated into the scheme to satisfy the Rankine-Hugoniot condition. Other complex methods that explicitly track the front, e.g., front-tracking methods (Al-Rawahi and Tryggvason, 2002; Li et al., 2003) and level set methods (Osher and Sethian, 1988) that implicitly model the interface as a signed distance function, have also been applied to the Stefan problem for modeling crystallization (Sethian and Strain, 1992; Chen et al., 1997). A limitation of numerical methods is that to obtain higher accuracy, fine mesh resolutions must be used, which can be computationally expensive in higher dimensions. In addition, for changes in PDE parameters, the simulations need to be re-run. These classical methods are also deterministic, and they do not provide uncertainty quantification.

Reduced Order Models (ROMs). Reduced Order Models (ROMs) have been a popular alternative to full order model numerical PDE simulations for computational efficiency. ROMs aim to approximate the solution in a lower dimensional subspace by computing the proper orthogonal decomposition (POD) basis using the singular value decomposition (SVD). Similar to deep learning models, there is no way to enforce that unconstrained ROMs are conservative and non-oscillatory. Tezaur et al. (2017) investigate enforcing conservative, entropy and total variation diminishing (TVD) constraints for ROMs as constrained nonlinear least squares problems. These methods are coined "structure preserving" ROMs via physics-based constraints (Sargsyan, 2016).

C.1.2 Scientific Machine Learning (SciML) Models

Here we describe the recent work in using ML models to solve PDEs. At a high-level, these works can be divided into three categories: 1. Physics-Informed Neural Networks (PINNs), which aim to incorporate PDE information as a soft constraint in the loss function; 2. Neural Operators, which aim to learn the solution mapping from PDE coefficients or initial conditions to solutions; and 3. Hard-constrained conservative ML models, which aim to incorporate different types of constraints to enforce conservation into the architecture.

Physics-informed ML methods. Physics-informed neural networks (PINNs) (Raissi et al., 2019) parameterize the solution to PDEs with a neural network (NN). These methods impose physical

knowledge into neural networks by adding the differential form of the PDE to the loss function as a soft constraint or regularizer. Purely data-driven approaches include DeepONet (Lu et al., 2021) and Neural Operators (NOs) (Li et al., 2020, 2021a; Gupta et al., 2021), which aim to learn the underlying function map from initial conditions or PDE coefficients to the solution. Learning this mapping enables these methods to be resolution independent, i.e., train on a coarse resolution and perform inference on a finer resolution. These methods only use PDE knowledge implicitly by training on simulations. The Physics-Informed Neural Operator (PINO) attempts to address that the physics are not directly enforced in the model by making the data-driven Fourier Neural Operator (FNO) "physics-informed." To do so, they again add the differential form into the supervised loss function as a soft constraint regularization term (Li et al., 2021b; Goswami et al., 2022).

Recently Krishnapriyan et al. (2021); Edwards (2022) identified several challenges and limitations for SciML of this soft constraint approach on the training procedure for several PDEs with large parameter values. In particular, Krishnapriyan et al. (2021) show that the sharp and non-smooth loss surface loss surface created by adding the PDE directly as a regularizer can be more difficult to optimize. Relatedly, PINO has been shown to perform worse than the base FNO without the differential form of the PDE as a soft constraint in the loss (Li et al., 2021b; Saad et al., 2022). Motivated by these observations, Négiar et al. (2022) propose a solution for linear PDEs that enforces the differential form of the PDE as a hard constraint; and Subramanian et al. (2022) propose another solution using an adaptive update of collocation points. In addition, Wang et al. (2022) examine training issues associated with the spectral bias in PINNs (Jacot et al., 2018). Edwards (2022) discusses the broader-scale impacts of these results for the SciML field, and motivates the need for better solutions that capture the underlying continuous physics.

Machine Learning Models for Conservation Laws. Enforcing the PDE as a soft constraint gives very weak control on the physical conservation property, resulting in non-physical solutions that can violate governing conservation law. Jekel et al. (2022) aim to satisfy conservation by adding the continuity equation as a soft regularizer via the PINNs approach, and they show that this does not improve performance. To try to remedy this, Mao et al. (2020); Jagtap et al. (2020) propose conservative PINNs (cPINNs) for conservation laws, which aim to enforce flux continuity, i.e., the out-flux of one cell equals the in-flux of the neighboring cell, for a type of local conservation. Again, however, this condition on the flux is added to the loss function as a regularization term, i.e., as a soft constraint in a Lagrange dual form, and so the conservation condition is in general not exactly satisfied.

Motivated by the importance of satisfying conservation laws in climate applications, Bolton and Zanna (2019); Zanna and Bolton (2020); Beucler et al. (2021) have proposed building known linear physical constraints directly into deep learning architectures. Beucler et al. (2021) propose

a model that forces the output of a neural network into the null space of the constraint matrix. While the solution exactly satisfies the constraints, the constraints depend on the resolution of the data, and they are an approximation to the true physical quantity that needs to be constrained. Surprisingly, Beucler et al. (2021) also finds that the reconstruction error is not always improved with adding constraints. Other methods to enforce conservation include the following. Sturm and Wexler (2022) enforce the flux continuity equation in the last layer of the neural network to model the balance of atoms. Müller (2022) enforce conservation by encoding symmetries using Noether's theorem. Richter-Powell et al. (2022) propose so-called Neural Conservation Laws, to enforce conservation by design by using parametizations of deep neural networks similar to the approaches in Négiar et al. (2022); Sturm and Wexler (2022); Müller (2022). In particular, Richter-Powell et al. (2022) use a change of variables that combines time and space derivatives into the divergence operator to create a divergence-free model, and they then use auto-differentiation similar to the Neural ODEs approach (Chen et al., 2018). This optimize-then-discretize approach has been shown to have related difficulties (Krishnapriyan et al., 2022; Ott et al., 2021; Onken and Ruthotto, 2020).

C.2 Derivation of the Integral Form of a Conservation Law

To obtain the integral form of a conservation law, given in Equation 4.3 as:

$$\int_{\Omega} u(t,x)d\Omega = \int_{\Omega} h(x)d\Omega - \int_0^t \int_{\Gamma} F(u)\cdot n d\Gamma dt, \tag{C.1}$$

we first integrate the differential form of the conservation law, given in Equation 4.2 as:

$$\mathcal{F}u = u_t + \nabla \cdot F(u), \tag{C.2}$$

over the spatial domain Ω. From this, we obtain an expression for the rate of change in time of the total conserved quantity in terms of the fluxes on the boundary, given as:

$$\frac{d}{dt}\int_{\Omega} u(t,x)d\Omega = \int_{\Omega} u_t(t,x)d\Omega \tag{C.3a}$$

$$= -\int_{\Omega} \nabla \cdot F(u)d\Omega \tag{C.3b}$$

$$= -\int_{\Gamma} (F(u)\cdot n)d\Gamma, \tag{C.3c}$$

where the last step is obtained by applying the divergence theorem to the flux term, and n is the outward pointing unit normal on the boundary Γ.

We then integrate Equation C.3 over the temporal domain $[0, t]$. Doing this to Equation C.3a yields:

$$\int_0^t \int_\Omega u_t(t,x) d\Omega = \int_\Omega u(t,x) d\Omega - \int_\Omega u(0,x) d\Omega,$$

where $u(0,x) = h(x)$ denotes the initial condition. By equating this quantity to the temporal integral of the right hand side of Equation C.3c, we obtain the corresponding integral form of a conservation law:

$$\int_\Omega u(t,x) d\Omega = \int_\Omega h(x) d\Omega - \int_0^t \int_\Gamma F(u) \cdot n d\Gamma dt,$$

which is Equation C.1.

C.3 Exact solutions and Linear Conservation Constraints for Conservation Laws

In this section, we provide the exact solutions to to a wide range of conservation laws:

$$\begin{cases} \underbrace{u_t + \nabla \cdot F(u)}_{\mathcal{F}u} = 0, \ x \in \Omega, \\ u(0,x) = h(x), \\ u(t,x) = g(t,x), \ x \in \Gamma, \end{cases} \quad , \forall\, t \geq 0, \tag{C.4}$$

for general nonlinear flux $F(u)$, nonlinear differential operator $\mathcal{F}$, initial condition $h(x)$ and prescribed boundary conditions on the boundary Γ of the spatial domain Ω. These exact solutions are used to generate the solution samples for the training data in the experiment Figure 5.5.

The integral form of the conservation law in Equation 4.4 is given as:

$$\underbrace{\int_\Omega u(t,x) d\Omega}_{\mathcal{G}u(t,x)} = \underbrace{\int_\Omega h(x) d\Omega + \int_0^t (F_{\text{in}} - F_{\text{out}}) dt}_{b(t)}, \tag{C.5}$$

where $\Omega = [x_0, x_N]$, $F_{\text{in}} = F(u,t,x_0)|_{u=g(t,x_0)}$, $F_{\text{out}} = F(u,t,x_N)|_{u=g(t,x_N)}$ and $g(t,x)$ is the prescribed Dirichlet boundary condition in Equation C.4. We provide the exact formulation of our linear constraint $\mathcal{G}u(t,x) = b(t)$. Table C.2 provides a summary, showing that our boundary flux linearity assumption holds for a broad class of problems—even including nonlinear conservation laws with nonlinear PDE operators $\mathcal{F}$.

PDE	Type	$F(u)$	$h(x)$	$g(t,x)$	Ω	Γ	$b(t)$
Diffusion	Linear parabolic ("easy")	$-k\nabla u,\ k \in \mathbb{R}_+$	$\sin(x)$	$\{0,0\}$	$[0, 2\pi]$	$\{0, 2\pi\}$	0
PME	Nonlinear degenerate parabolic ("medium")	$-u^m\nabla u,\ m \in \mathbb{Z}_+$	0	$\{(mt)^{1/m}, 0\}$	$[0, 1]$	$\{0, 1\}$	$\frac{m^{1+1/m}}{m+1}t^{1+1/m}$
Stefan	Nonlinear degenerate parabolic ("hard")	$\begin{cases} -\nabla u, & u \geq u^\star \\ 0, & \text{otherwise} \end{cases}, u^\star \in \mathbb{R}_+$	0	$\{1,0\}$	$[0, 1]$	$\{0, 1\}$	$2c_1\sqrt{t/\pi}$
Advection	Linear hyperbolic ("medium")	$\beta u,\ \beta \in \mathbb{R}_+$	$\begin{cases} 1, & x \leq 0.5 \\ 0, & \text{otherwise} \end{cases}$	$\{1,0\}$	$[0, 1]$	$\{0, 1\}$	$\frac{1}{2} + \beta t$
Burgers'	Nonlinear hyperbolic ("hard")	$\frac{1}{2}u^2$	$\begin{cases} -ax, & x \leq 0, a \in \mathbb{R}_+ \\ 0, & \text{otherwise} \end{cases}$	$\{a,0\}$	$[-1, 1]$	$\{-1, 1\}$	$(a/2)(1+at)$

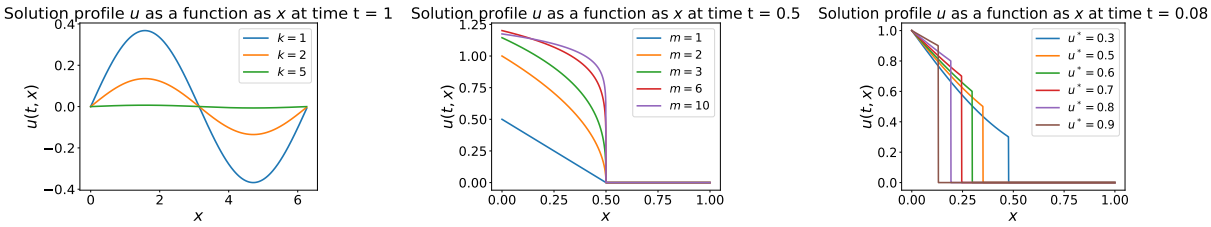
Table C.2: Classification of PDE conservation laws ranging from "easy" to "hard", and corresponding total time-varying conserved value $b(t)$ in the integral form of Equation C.5 for specified flux function $F(u)$, initial and boundary conditions $h(x)$ and $g(t,x)$, respectively in Equation C.4. See Section C.3.1.3 for the value of the constant $c_1 \in \mathbb{R}$.

C.3.1 GPME Family of Conservation Laws

In this subsection, we consider the (degenerate) parabolic GPME family of conservation laws given in Equation 4.9 as:

$$u_t - \nabla \cdot (k(u)\nabla u) = 0,$$

with flux $F(u) = -k(u)\nabla u$. Figure C.1 shows the effects of the various PDE parameters $k(u)$ at a fixed time t on the solution on three instances of the GPME ranging from the "easy" to "hard" cases, i.e., the diffusion equation, PME and Stefan, respectively.



(a) k: Diffusion equation at $t = 1$ ("easy"). (b) m: PME at $t = 0.5$ ("medium"). (c) $u^\star$: Stefan at $t = 0.08$ ("hard").

Figure C.1: Effect of PDE parameters on the three "easy" to "hard" instances of the GPME at fixed time t.

C.3.1.1 Diffusion Equation

The heat or diffusion equation is a simple linear parabolic PDE with constant coefficient $k(u) = k$, which represents an "easy" task. Figure C.1(a) illustrates the effect of the constant diffusivity

(conductivity) parameter k on solutions to the diffusion (heat) equation. For larger values of k, we see that the solution more quickly dissipates toward the constant smooth zero steady state.

Exact Solution. We use the same diffusion test problem from Krishnapriyan et al. (2021) with the following initial condition and periodic boundary conditions:

$$u(0,x) = h(x) = \sin(x), \forall x \in \Omega = [0, 2\pi],$$
$$u(t,0) = u(t,2\pi), \forall t \ \in [0,T],$$

respectively. The exact solution is given as

$$u(t,x) = FT^{-1}(FT(h(x))e^{-kn^2t}),$$

where FT denotes the Fourier transform, and n denotes the frequency in the Fourier domain.

Global Conservation. The total mass (energy) is constant and zero over all time, since there is no in or out flux to the domain. Then, Equation C.5 reduces to the following linear homogeneous system:

$$\mathcal{G}u(t,x) = \int_{x_0}^{x_N} u(t,x)dx = 0 = b(t). \tag{C.6}$$

To derive the above relation, we see by using separation of variables that the solution $u(t,x) = \sin(x)T(t)$ is a damped sine curve over time. The flux $F(u) = -k\nabla u = -\cos(x)T(t)$, where $T(t)$ denotes a decaying exponential function. Then, the integral form in Equation C.5 is given as:

$$\begin{aligned}\int_\Omega u(t,x)d\Omega &= \int_\Omega h(x)d\Omega + \int_0^t [F(u,t,x_0=0) - F(u,t,x_N=2\pi)]dt \\ &= \int_0^{2\pi} \sin(x)d\Omega - k\int_0^t [\cos(0)T(t) - \cos(2\pi)T(t)]dt = 0,\end{aligned}$$

by periodicity.

C.3.1.2 Porous Medium Equation

In the Porous Medium Equation (PME), the nonlinearity and small values of the coefficient $k(u) = u^m$, for $m \geq 1$, cause challenges for current state-of-the-art SciML baselines as well as classical numerical methods on this degenerate parabolic equation. The difficulty increases as the exponent m increases, and the solution forms sharper corners. In particular, the solution gradient is finite for $m = 1$, and it approaches infinity near the front for $m > 1$. Figure C.1(b) illustrates the effect of

the parameter m on the solution, with solutions for $m > 1$ being sharper, and having a different profile than those for the piecewise linear solution for $m = 1$.

Exact Solution. We test the locking problem (TLP) of the PME from Lipnikov et al. (2016); Maddix et al. (2018b) with the following initial and growing in time Dirichlet left boundary conditions for some final time $T \leq 1$:

$$\begin{aligned} u(0, x) &= h(x) = 0, \forall x \in \Omega = [0, 1], \\ u(t, 0) &= g(t, 0) = (mt)^{1/m}, \forall t \in [0, T], \\ u(t, 1) &= g(t, 1) = 0, \forall t \in [0, T], \end{aligned}$$

respectively. The exact solution is given as:

$$u(t, x) = (m(t - x)_+)^{1/m}. \tag{C.7}$$

Global Conservation. We write the specific form of the linear conservation constraint in Equation C.5 for the PME as:

$$\mathcal{G}u(t, x) = \int_{x_0}^{x_N} u(t, x)dx = \frac{m^{1+1/m}}{m + 1} t^{1+1/m} = b(t), \tag{C.8}$$

by using the fact that the total mass of the initial condition is zero, and that $u(t, x_N = 1) = 0$ on the right boundary for $t \leq x_N = 1$.

Global conservation is driven by the in-flux at the growing in left boundary, where

$$F_{\text{in}} = F(u, t, x_0)|_{u=g(t,x_0),x=x_0} = -g(t, x_0)^m \nabla u|_{x=x_0} = -mt\nabla u|_{x=x_0}.$$

The boundary flux at the right boundary is 0, since we assume that the shock is contained in the domain and $t > x$, hence $u(t, 1) = 0$ and

$$F_{\text{out}} = F(u, t, x_N)|_{u=g(t,x_N),x=x_N} = -g(t, x_N)^m \nabla u|_{x=x_N} = 0.$$

The first integral on the righthand side in Equation C.5 consisting of the initial mass is 0, since $h(x) = 0$, and we are left only with the in-flux term:

$$\begin{aligned}\int_\Omega u(t,x)d\Omega &= \int_0^t F_{\text{in}}(t)dt \\ &= -\int_0^t g(t,x_0=0)^m \nabla u|_{x=x_0} dt \\ &= \int_0^t (mt)(mt)^{1/m-1} dt \\ &= m^{1/m}\int_0^t t^{1/m} dt \\ &= \frac{m^{1+1/m}}{m+1} t^{1+1/m},\end{aligned}$$

where $\nabla u|_{x=x_0} = -(m(t-x))^{1/m-1}|_{x=x_0} = -(mt)^{1/m-1}$.

C.3.1.3 Stefan Problem

The Stefan problem is the most challenging problem in the GPME degenerate parabolic family of conservation equations since the coefficient $k(u)$ is a nonlinear step function of the unknown u, given as:

$$k(u) = \begin{cases} k_{\max},\ u \geq u^\star, \\ k_{\min},\ u < u^\star, \end{cases} \tag{C.9}$$

for constants $k_{\max}, k_{\min} \in \mathbb{R}$ and $u(t, x^*(t)) = u^* \in \mathbb{R}_+$ for shock position $x^*(t)$. In this problem, the solution is a shock or moving interface with a finite speed of propagation that does not dissipate over time. Figure C.1(c) illustrates the effect of the parameter $u^\star$ on the solution and shock position, with smaller values of $u^\star$ resulting in a faster shock speed.

Exact Solution. We use the Stefan test case from van der Meer et al. (2016); Maddix et al. (2018a) with $k_{\max} = 1$, $k_{\min} = 0$ in Equation C.9, and the following initial and Dirichlet boundary conditions for some final time T:

$$\begin{aligned} u(0,x) &= h(x) = 0, \forall x \in \Omega = [0,1], \\ u(t,0) &= g(t,0) = 1, \forall t \in [0,T], \\ u(t,1) &= g(t,1) = 0, \forall t \in [0,T], \end{aligned}$$

respectively. The exact solution is given as:

$$u(t,x) = \mathbf{1}_{u \geq u^\star} \left(1 - c_1 \Phi[x/(2\sqrt{k_{\max} t})]\right), \tag{C.10}$$

where $\mathbf{1}_{\mathcal{E}}$ denotes an indicator function for event $\mathcal{E}$, $\Phi(x) = \text{erf}(x) = \int_0^x \phi(y)dy$ denotes the error function with $\phi(y) = (2/\sqrt{\pi})\exp(-y^2)$, and constant $c_1 = (1-u^*)/\Phi[\alpha/(2\sqrt{k_{\max}})]$. A nonlinear solve for $\tilde{\alpha}$: $(1-u^*)/\sqrt{\pi} = u^*\Phi(\tilde{\alpha})\tilde{\alpha}\exp(\tilde{\alpha}^2)$, is used to compute $\alpha = 2\sqrt{k_{\max}}\tilde{\alpha}$. The exact shock position is $x^*(t) = \alpha\sqrt{t}$.

Global Conservation. We write the linear $\mathcal{G}$ conservation constraint in Equation C.5 for the Stefan equation as:

$$\mathcal{G}u(t,x) = \int_{x_0}^{x_N} u(t,x)dx = 2c_1\sqrt{\frac{k_{\max}t}{\pi}} = b(t). \tag{C.11}$$

We use the fact that the solution is monotonically non-increasing to compute the coefficient values at the boundaries, i.e., $u(t,x_0) \geq u^\star \geq u(t,x_N)$, where $0 = x_0 \leq x^\star \leq x_N = 1$ and $x^*(t)$ denotes the shock position. It follows that $k(u(t,x_0)) = k_{\max}$ and $k(u(t,x_N)) = 0$. Then the out-flux $F_{\text{out}} = k(u(t,x_N))\nabla u = 0$. The first integral on the righthand side of Equation C.5 consisting of the initial mass is 0, since $h(x) = 0$, and we are left only with the in-flux term as follows:

$$\begin{aligned}
\int_\Omega u(t,x)d\Omega &= \int_0^t F_{\text{in}}(t)dt \\
&= -k_{\max}\int_0^t \nabla u|_{x=x_0}dt \\
&= c_1\sqrt{\frac{k_{\max}}{\pi}}\int_0^t t^{-1/2}dt \\
&= 2c_1\sqrt{\frac{k_{\max}t}{\pi}},
\end{aligned}$$

where

$$\begin{aligned}
\nabla u|_{x=x_0} &= -c_1\Phi'[x_0/(2\sqrt{k_{\max}t})]/(2\sqrt{k_{\max}t}) \\
&= -c_1/\sqrt{\pi k_{\max}t}\exp[x_0^2/(4k_{\max}t)] \\
&= -c_1/\sqrt{\pi k_{\max}t},
\end{aligned}$$

for $x_0 = 0$.

C.3.2 Hyperbolic Conservation Laws

In this section, we consider hyperbolic conservation laws, where solutions exhibit shocks and smooth initial conditions self-sharpen over time (LeVeque, 1990, 2002).

C.3.2.1 Linear Advection

The linear advection (convection) equation:

$$u_t + \beta u_x = 0, \tag{C.12}$$

is a hyperbolic conservation law with flux $F(u) = \beta u$, where a fluid with density u is transported or advected by some constant velocity $\beta \in \mathbb{R}$. For larger values of β, the shock moves faster.

Exact Solution. Here we consider the test case with the following initial and boundary conditions:

$$\begin{aligned} u(0, x) &= h(x) = \mathbf{1}_{x \leq 0.5}, \forall x \in \Omega = [0, 1], \\ u(t, 0) &= g(t, 0) = 1, \forall t \in [0, T], \\ u(t, 1) &= g(t, 1) = 0, \forall t \in [0, T], \end{aligned}$$

respectively, and $\mathbf{1}_{\mathcal{E}}$ denotes an indicator function for event $\mathcal{E}$. Note that the linear advection (convection) problem is also studied in Krishnapriyan et al. (2021) with smooth $h(x) = \sin(x)$ and periodic boundary conditions. Here we consider the more challenging case, where the initial condition is already a shock.

In our case, the exact solution,

$$u(t, x) = h(x - \beta t),$$

is simply the initial condition shifted to the right, which is a shock wave traveling to the right with speed $\beta > 0$.

Global Conservation. We write the linear conservation constraint in Equation C.5 for linear advection as:

$$\mathcal{G}u(t, x) = \int_{x_0}^{x_N} u(t, x)dx = \frac{1}{2} + \beta t = b(t). \tag{C.13}$$

The out-flux $F_{\text{out}} = u(t, 1) = g(t, 1) = 0$, by the fixed right Dirichlet boundary condition, and we are left with the following terms:

$$\begin{aligned} \int_{\Omega} u(t, x)d\Omega &= \int_{\Omega} h(x)dx + \int_0^t F_{\text{in}}(t)dt \\ &= \int_0^{0.5} dx + \beta \int_0^t u(t, 0)dt \\ &= \frac{1}{2} + \beta t, \end{aligned}$$

by using the Dirichlet boundary condition $u(t,0) = g(t,0) = 1$ in the second term in the last step. We see that the time rate of change in total mass is constant over time.

C.3.2.2 Burgers' Equation

Burgers' Equation, given as:

$$u_t + \frac{1}{2}(u^2)_x = 0, \tag{C.14}$$

is a commonly used nonlinear hyperbolic conservation law with flux $F(u) = \frac{1}{2}u^2$. Among other things, it is used in traffic modeling.

Exact Solution. We consider the test case from Tezaur et al. (2017), where $a = 1$, with the following initial and boundary conditions:

$$u(0,x) = h(x) = \begin{cases} a, \ x \leq -1, \\ -ax, \ -1 \leq x \leq 0, \\ 0, \ x \geq 0, \end{cases} \quad \forall x \in \Omega = [-1,1],$$
$$u(t,-1) = g(t,-1) = a, \forall t \in [0,T],$$
$$u(t,1) = g(t,1) = 0, \forall t \in [0,T],$$

respectively for constant, positive parameter slope $a \geq 1$. For larger values of a, the slope of the initial condition is steeper, and a shock is formed faster.

We write the nonlinear Burgers' Equation C.14 in non-conservative form as

$$u_t + uu_x = 0.$$

We see that this is the advection Equation C.12 with speed $\beta = u$. Hence, similarly the exact solution is given by $u(t,x) = h(x - ut)$ when the characteristics curves do not intersect, by using the method of characteristics (Evans, 2010). We then obtain the following solution:

$$u(t,x) = \begin{cases} a, \ x - ut \leq -1, \\ -a(x - ut), \ -1 \leq x - ut \leq 0, \\ 0, \ x - ut \geq 0. \end{cases}$$

We use the second case to solve this implicit equation explicitly for u, i.e., $u = -a(x - ut) \iff u = \frac{-ax}{1-at}$. Then $x - ut = \frac{x}{1-at}$, where the denominator $1 - at > 0$ for $t < 1/a$. We then solve the

inequalities and substitute this in to obtain:

$$u(t,x) = \begin{cases} a, & x \leq at-1, \\ \frac{ax}{at-1}, & at-1 \leq x \leq 0, \\ 0, & x \geq 0, \end{cases}$$

for $0 \leq t < 1/a$. We see that as time increases the linear part of the solution self-sharpens with a steeper slope until the characteristics intersect at breaking time

$$t_b = \frac{-1}{\inf_x h'(x)} = 1/a,$$

and a shock is formed. This is known as the waiting time phenomenon (Maddix et al., 2018a). The rightward moving shock forms with weak solution given as:

$$u(t,x) = \begin{cases} a, \ x \leq \frac{1}{2}(at-1), \\ 0, \ x \geq \frac{1}{2}(at-1), \end{cases}$$

for $t \geq 1/a$. The shock speed $x'(t)$ is given by the Rankine-Hugoniot (RH) condition (Evans, 2010). The RH condition simplifies for Burgers' Equation as follows:

$$x'(t) = \frac{f(u_R) - f(u_L)}{u_R - u_L} = \frac{1}{2}\frac{u_R^2 - u_L^2}{u_R - u_L} = \frac{1}{2}\frac{(u_R - u_L)(u_R + u_L)}{u_R - u_L} = \frac{u_R + u_L}{2} = \frac{a}{2},$$

where $u_L = a$ denotes the solution value to the left of the shock and $u_R = 0$ denotes the solution value to the right of the shock. Lastly, to obtain the shock position $x(t)$, we solve the simple ODE $x'(t) = a/2$ with initial condition $x(t_b = 1/a) = 0$ to obtain $x(t) = \frac{at}{2} + c$, where $x(1/a) = \frac{1}{2} + c = 0$, and so $c = -\frac{1}{2}$. This results in $x(t) = \frac{1}{2}(at-1)$, as desired.

Global Conservation. We write the linear conservation constraint in Equation C.5 for Burgers' equation as:

$$\mathcal{G}u(t,x) = \int_{x_0}^{x_N} u(t,x)dx = \frac{a}{2}(1+at) = b(t). \tag{C.15}$$

The out-flux is $F_{\text{out}} = \frac{1}{2}u(t,1)^2 = \frac{1}{2}g(t,1)^2 = 0$, by the fixed right Dirichlet boundary condition, and we are left with the following terms:

$$\begin{aligned}\int_\Omega u(t,x)d\Omega &= \int_\Omega h(x)dx + \int_0^t F_{\text{in}}(t)dt \\ &= -a\int_{-1}^0 x dx + \frac{1}{2}\int_0^t u(t,-1)^2 dt \\ &= \frac{a}{2}(1+at),\end{aligned}$$

by using the Dirichlet boundary condition $u(t,-1) = g(t,-1) = a$ in the second term in the last step. We again see that the time rate of change in total mass is constant over time.

C.4 Discretizations of the Integral Operator $\mathcal{G}$ for Conservation and Additional Linear Constraints

In this section, we first describe common discretization schemes G for the integral operator $\mathcal{G}$ in Equation 4.5 given as:

$$\mathcal{G}u(t,x) = \int_\Omega u(t,x)d\Omega = b(t), \tag{C.16}$$

to form a linear matrix constraint equation $Gu = b$. Then, we show how to incorporate other types of linear constraints into our framework PROBCONSERV. In particular, we consider artificial diffusion, which is a common numerical technique to smooth numerical artifacts through the matrix $\tilde{G}$ arising from the second order central finite difference scheme of the second derivative.

C.4.1 Discretizations of the Integral Operator $\mathcal{G}$

Here, we provide examples of the discrete matrix $G \in \mathbb{R}^{T\times MT}$, which approximates the continuous integral operator $\mathcal{G}$ in Equation C.16. We use M to denote the number of spatial points, T to denote the number of time points, and we set $N = MT$.

We form a discrete linear system from the continuous integral conservation law, i.e,. $Gu = b$, where each row i of G acts as a Riemann approximation to the integral $\mathcal{G}u(t,x)$ at time t_i. At inference time, we assume we have an ordered output grid $\{(t_1,x_1),\ldots,(t_1,x_M),\ldots,(t_T,x_1),\ldots,(t_T,x_M)\}$ with spatial grid spacing $\Delta x_j = x_{j+1} - x_j$ for $j = 1,\ldots,M-1$. We want to compute the solution at these corresponding grid points given as:

$$u = [u(t_1,x_1),\ldots,u(t_1,x_M),\ldots,u(t_T,x_1),\ldots,u(t_T,x_M)]^T \in \mathbb{R}^{MT}.$$

The known right-hand side is given as:

$$b = [b(t_1), \ldots, b(t_T)]^T \in \mathbb{R}^T.$$

We now proceed to provide examples of specific matrices G corresponding to common numerical spatial integration schemes (Burden et al., 2016).

Left Riemann Sum. For G arising from the common first-order left Riemann sum

$$\sum_{j=1}^{M-1} u(t_i, x_j)\Delta x_j,$$

at time t_i, we have the following expression:

$$G_{ij} = \begin{cases} \Delta x_j, \ (i-1)M + 1 \leq j \leq iM - 1, \\ 0, \ \text{otherwise.} \end{cases}$$

In other words, it uses the left function value $u(t, x_j)$ on the interval $[x_j, x_{j+1}]$. The right Riemann sum ($\sum_{j=2}^{M} u(t, x_j)\Delta x_{j-1}$ at time t) is a simple extension that shifts the column indices by 1 to $(i-1)M + 2 \leq j \leq iM$ to use the right value $u(t_i, x_{j+1})$ on the interval $[x_j, x_{j+1}]$.

Trapezoidal Rule. For G arising from the second order trapezoidal rule

$$Gu = \sum_{j=1}^{M-1} \frac{u(t_i, x_j) + u(t_i, x_{j+1})}{2} \Delta x_j,$$

at time t_i, we have the following expression:

$$G_{ij} = \begin{cases} \frac{\Delta x_j}{2}, \ j = (i-1)M + 1, \\ \frac{\Delta x_{j-1} + \Delta x_j}{2}, \ (i-1)M + 2 \leq j \leq iM - 1, \\ \frac{\Delta x_{j-1}}{2}, \ j = iM, \\ 0, \ \text{otherwise.} \end{cases} \tag{C.17}$$

We use the trapezoidal discretization of G in Equation C.17 in our experiments. Note that higher order schemes, e.g., Simpson's Rule may also be used, as well as more advanced numerical techniques. These can help to reduce the error in the spatial integration approximation, including shock tracking schemes in Maddix et al. (2018a) on the more challenging sharper problems with shocks that we see for high values of m in the PME and Stefan.

C.4.2 Adding artificial diffusion into the discretization

In addition to various discretization schemes to compute the integral operator $\mathcal{G}$, our PROBCONSERV framework can incorporate other inductive biases based on the knowledge of the underlying PDE, e.g., to bypass undesirable numerical artifacts. One common technique that has been used widely in numerical methods for this purpose is adding artificial diffusion (Maddix et al., 2018b). This artificial diffusion can act locally at sharp corners such as shock interfaces, where numerical methods tend to suffer from high frequency oscillations. Other common numerical methods to avoid numerical oscillations include total variation diminishing (TVD), i.e., $\text{TV}(u(t_{i+1}, x)) \leq \text{TV}(u(t_i, x))$, $\forall i$, or total variation bounded (TVB), i.e., $\text{TV}(u(t_{i+1}, x)) \leq C$, $C > 0$, $\forall i$, where $\text{TV}(u) = \int_\Omega |\frac{\partial u}{\partial x}| d\Omega$ and is approximated as $\sum_{j=1}^{M-1} |u(t_i, x_{j+1}) - u(t_i, x_j)|$ (LeVeque, 1990; Tezaur et al., 2017). Note that enforcing these inequality constraints is a direction of future work.

In machine learning, artificial diffusion is analogous to adding a regularization penalty on the L_2 norm of the second derivative $\int \{\frac{\partial^2}{\partial x^2} u(t_i, x)\}^2 dx$ (Hastie et al., 2013). This can be written as the L_2 norm of a linear operator applied to u, $\|\tilde{\mathcal{G}}(u)\|_2^2$, where $\tilde{\mathcal{G}}(u)(t_i) := \frac{\partial^2}{\partial x^2} u(t_i, x)$. Thus, we can incorporate this penalty term into PROBCONSERV in the same manner as the integral operators by discretizing $\tilde{\mathcal{G}}$ via a matrix $\tilde{G}$. Let $\tilde{G}$ be the second order central finite difference three-point stencil at time t_i over M spatial points:

$$(\tilde{G}u)_j = \left(\frac{u(t_i, x_{j+2}) - u(t_i, x_{j+1})}{\Delta x_{j+1}} \right) - \left(\frac{u(t_i, x_{j+1}) - u(t_i, x_j)}{\Delta x_j} \right).$$

for $j = 1, \ldots, M-2$. For simplicity of notation, we assume $\Delta x_j := \Delta x$ for all x_j, though this need not be the case in general. This results in the following three-banded matrix:

$$\tilde{G} = \frac{1}{\Delta x} \begin{bmatrix} 1 & -2 & 1 & 0 & \ldots \\ 0 & 1 & -2 & 1 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}. \tag{C.18}$$

Since our goal is to penalize large differences in the solution, we set the constraint value b to zero:

$$\tilde{G}u + \sigma_{\tilde{G}} \epsilon = 0,$$

where $\sigma_{\tilde{G}} > 0$ denotes the constraint value for the artificial diffusion. Since the mechanism is exactly the same with a linear constraint, artificial diffusion can be applied using Equation 4.8a with $b = 0$, where $\tilde{\mu}$ and $\tilde{\Sigma}$ are the mean and covariance after applying the conservation constraint

as follows:

$$\tilde{\mu}_{\text{diffusion}} = \tilde{\mu} - \tilde{\Sigma}\tilde{G}^T(\sigma_{\tilde{G}}^2 I + G\tilde{\Sigma}G^T)^{-1}(\tilde{G}\tilde{\mu}),$$
$$\tilde{\Sigma}_{\text{diffusion}} = \tilde{\Sigma} - \tilde{\Sigma}\tilde{G}^T(\sigma_{\tilde{G}}^2 I + G\tilde{\Sigma}G^T)^{-1}(\tilde{G}\tilde{\Sigma}).$$

Moreover, the guarantees of Theorem Theorem 4.1 still hold. Smaller values of $\sigma_{\tilde{G}}$ lead to smaller values of $\|\tilde{G}\tilde{\mu}_{\text{diffusion}}\|_2^2$, which results in a smoother solution.

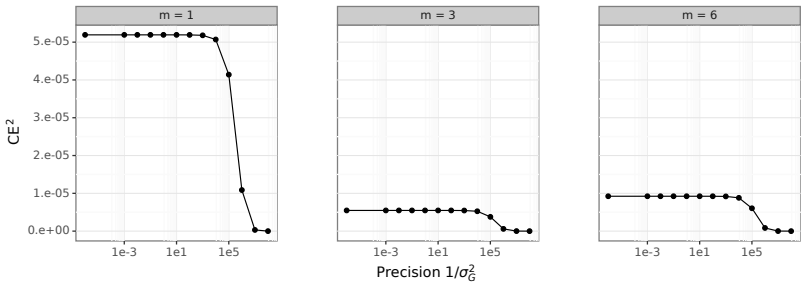
Unlike the case of enforcing conservation, it is typically not desirable when applying artificial diffusion to set $\sigma_{\tilde{G}}$ to zero, as this will lead to a simple line fit (Hastie et al., 2013). We set the variance for each row of $\tilde{G}$ as follows: Let $\sigma_i^2 := \text{Var}(u_n)$ be the variance of target value u_n from the Step 1 procedure:

$$\begin{aligned}\sigma_{\tilde{G},i}^2 := \text{Var}((\tilde{G}u)_i) &= \text{Var}(u_i - 2u_{i+1} + u_{i+2}) \\ &= \sigma_i^2 + 4\sigma_{i+1}^2 + \sigma_{i+2}^2 - 4\rho\left(\sigma_i\sigma_{i+1} + \sigma_{i+1}\sigma_{i+2}\right) + 2\rho^2\sigma_i\sigma_{i+2},\end{aligned}$$

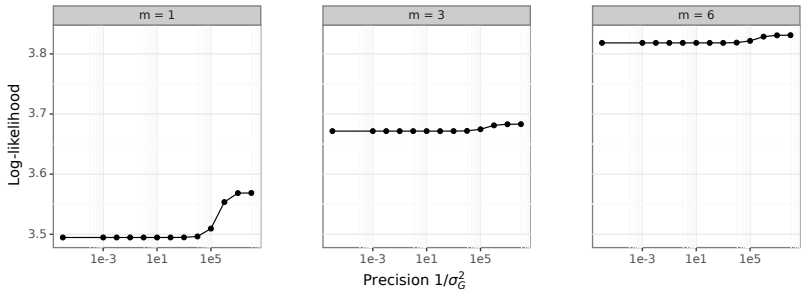
where $\rho \in [0, 1]$ determines the level of auto-correlation between neighboring points. Higher values of ρ lead to lower values of $\sigma_{\tilde{G},i}^2$, and hence a higher penalty.

C.5 Control on Conservation Constraint

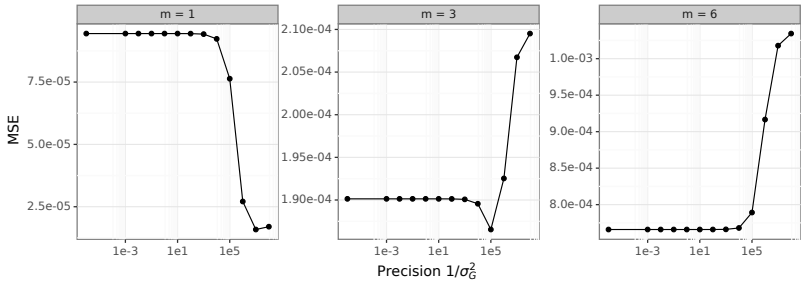
Figure C.2 illustrates that Theorem 4.1 holds empirically for the the PME in subsection 4.3.2, where both the norm of the conservation error (CE^2) monotonically decreases to zero and the predictive log likelihood (LL) monotonically increases as the constraint precision $\sigma_G^2 \to 0$ ($1/\sigma_G^2 \to \infty$). We that for MSE, the trend depends on the difficulty of the problem. For "easy" scenarios, where $m = 1$, the MSE also monotonically improves (decreases) as $\sigma_G^2 \to 0$ ($1/\sigma_G^2 \to \infty$). For "medium" difficulty problems, where $m = 3$, we see that there is an optimal value for σ_G^2 around 10^{-5}, and enforcing the constraint exactly does not result in the lowest MSE. For the "harder" $m = 6$ case, we see that a looser tolerance on the constraint results in better MSE. However, in this case the solution is non-physical since it does not satisfy conservation. Note that in the sharper $m = 6$ case, the accuracy may be able to be improved by using more advanced approximations for the integral operator $\mathcal{G}$ that take the sharp corners into account (Maddix et al., 2018a).



(a) Norm of the conservation error: $\text{CE}^2 = \|G\mu - b\|_2^2$.



(b) Log-likelihood: $\text{LL}(u; \mu, \Sigma) = -\frac{1}{2M}\|u_{t_j,\cdot} - \mu_{t_j,\cdot}\|_{\Sigma^{-1}} - \frac{1}{2M}\sum_i \log \sigma_{t_j,i}^2 - \log 2\pi$.



(c) Mean-squared error (MSE): $\frac{1}{M}\|u_{t_j,\cdot} - \mu_{t_j,\cdot}\|^2$.

Figure C.2: Illustration of the norm of the conservation error CE^2 (lower is better) in top row, the predictive log likelihood (LL) in middle row (higher is better), and the mean-squared error (MSE) (lower is better) in the bottom row, as a function of the constraint precision $\frac{1}{\sigma_G^2}$ for the PME in subsection 4.3.2, where M denotes the number of spatial points and t_j denotes the time-index in the training window at which the metrics are reported. Each column indicates results for a different values of PDE parameter $m \in \{1, 3, 6\}$, corresponding to "easy", "medium", and "hard" scenarios, respectively. In all three cases, CE^2 monotonically decreases to zero and LL monotonically increases as $\sigma_G^2 \to 0$ ($1/\sigma_G^2 \to \infty$), illustrating Theorem 4.1. The biggest gains in log-likelihood are for $m = 1$, where conservation was also violated the most. In contrast, the relationship between MSE and $\frac{1}{\sigma_G^2}$ is not guaranteed to be monotonic, and it qualitatively changes, depending on the value of m.

C.6 Derivation of constrained mean and covariance

In this section, we provide two interpretations for the Step 2 procedure of PROBCONSERV from Equation 4.8 given as:

$$\tilde{\mu} = \mu - \Sigma G^T(\sigma_G^2 I + G\Sigma G^T)^{-1}(G\mu - b), \tag{C.19a}$$

$$\tilde{\Sigma} = \Sigma - \Sigma G^T(\sigma_G^2 I + G\Sigma G^T)^{-1}G\Sigma. \tag{C.19b}$$

While Equation 4.8 is well-defined in the case that $\sigma_G^2 = 0$, for simplicity we assume $\sigma_G^2 > 0$ throughout this section. In Lemma C.1, we show how Step 2 is justified as a Bayesian update of the unconstrained normal distribution from Step 1 by adding information about the conservation constraint contained in Equation 4.7, i.e., $b = Gu + \sigma_G\epsilon$ in Step 2. In Lemma C.2, we show how the posterior mean $\tilde{\mu}$ and $\tilde{\Sigma}$ can be re-expressed in a numerically stable and computationally efficient form given in Equation C.19. This form is equivalent to the observation update step seen in the Kalman filter. Finally, Lemma C.3 shows that this is equivalent to a least-squares optimization with an upper bound on the conservation error.

Note: $\mu \in \mathbb{R}^{MT}, \Sigma \in \mathbb{R}^{MT\times MT}, G \in \mathbb{R}^{T\times MT}, b \in \mathbb{R}^T$, where $N = MT$ denotes the number of spatio-temporal output points, M denotes the number of spatial points and T denotes number of constraints or in this case time steps to enforce the conservation constraint.

Lemma C.1 (Step 2 as Bayesian update)**.** *Assume the predictive distribution of u conditioned only on observed data D is normal with mean μ and covariance Σ. Let b be the known conservation quantity that follows a normal distribution with mean Gu and covariance $\sigma_G^2 I$, where $\sigma_G^2 > 0$. Then the posterior distribution of u conditional on both data D and conservation quantity b is normal with mean $\tilde{\mu}$ and covariance $\tilde{\Sigma}$ given as:*

$$\begin{aligned} u|b, D &\sim \mathcal{N}(\tilde{\mu}, \tilde{\Sigma}), \\ \tilde{\Sigma} &= A^{-1}\Sigma, \\ \tilde{\mu} &= A^{-1}(\mu + \frac{1}{\sigma_G^2}\Sigma G^T b), \end{aligned}$$

where $A = I + \frac{1}{\sigma_G^2}\Sigma G^T G$.

Proof This follows the same logic as a standard multivariate normal model with known covariance; see Chapter 3.5 of Gelman et al. (2015). We outline the derivation below. Note that we mark

the terms that are independent of the unknown u as constants.

$$
\begin{aligned}
\log p(u|b,D) &= \log \underbrace{p(u|D)}_{\text{Step 1}} \underbrace{p(b|u)}_{\text{Step 2}} - \log \int p(b|u)dp(u|D) \text{ (Bayes' Rule)} \\
&= \log p(u|D) + \log p(b|u) + C_1 \\
&= \log \mathcal{N}(u;\mu,\Sigma) + \log \mathcal{N}(b;Gu,\sigma_G^2 I) + C_1 \\
\\
&= -\frac{1}{2}\left(\|u-\mu\|^2_{\Sigma^{-1}} + \frac{1}{\sigma_G^2}\|Gu-b\|_2^2 \right) + C_2 \\
&= -\frac{1}{2}\left(u^T\Sigma^{-1}u - 2u^T\Sigma^{-1}\mu + u^T(\frac{1}{\sigma_G^2}G^TG)u - 2u^T\frac{1}{\sigma_G^2}G^Tb \right) + C_3 \\
&= -\frac{1}{2}\left(u^T(\Sigma^{-1} + \frac{1}{\sigma_G^2}G^TG)u - 2u^T(\Sigma^{-1}\mu + \frac{1}{\sigma_G^2}G^Tb) \right) + C_3 \\
&= -\frac{1}{2}\Bigg(u^T \underbrace{(\Sigma^{-1} + \frac{1}{\sigma_G^2}G^TG)}_{\tilde{\Sigma}^{-1}} u \\
&\quad - 2u^T \underbrace{(\Sigma^{-1} + \frac{1}{\sigma_G^2}G^TG)}_{\tilde{\Sigma}^{-1}} \underbrace{(\Sigma^{-1} + \frac{1}{\sigma_G^2}G^TG)^{-1}(\Sigma^{-1}\mu + \frac{1}{\sigma_G^2}G^Tb)}_{\tilde{\mu}} \Bigg) + C_3 \\
&= -\frac{1}{2}\left(u^T\tilde{\Sigma}^{-1}u - 2u^T\tilde{\Sigma}^{-1}\tilde{\mu} \right) + C_3 \\
&= \log \mathcal{N}(u;\tilde{\mu},\tilde{\Sigma}) + C_4,
\end{aligned}
$$

where

$$
\tilde{\Sigma} = (\Sigma^{-1} + \frac{1}{\sigma_G^2}G^TG)^{-1} = (I + \frac{1}{\sigma_G^2}\Sigma G^TG)^{-1}\Sigma = A^{-1}\Sigma, \tag{C.20a}
$$

$$
\tilde{\mu} = (\Sigma^{-1} + \frac{1}{\sigma_G^2}G^TG)^{-1}(\Sigma^{-1}\mu + \frac{1}{\sigma_G^2}G^Tb) = \tilde{\Sigma}(\Sigma^{-1}\mu + \frac{1}{\sigma_G^2}G^Tb) \tag{C.20b}
$$

$$
= A^{-1}\Sigma(\Sigma^{-1}\mu + \frac{1}{\sigma_G^2}G^Tb) = A^{-1}(\mu + \frac{1}{\sigma_G^2}\Sigma G^Tb), \tag{C.20c}
$$

$$
C_1 = -\log \int p(b|u)dp(u|D), \tag{C.20d}
$$

$$
C_2 = C_1 - \frac{1}{2}\left(MT\log 2\pi + \log\det\Sigma + T\log\pi + \log\sigma_G^2 \right), \tag{C.20e}
$$

$$
C_3 = C_2 - \frac{1}{2}\left(\mu^T\Sigma^{-1}\mu + \frac{1}{\sigma_G^2}b^Tb \right), \tag{C.20f}
$$

$$
C_4 = 0. \tag{C.20g}
$$

Note that $C_4 = 0$ since the left-hand side and right-hand side are log-probability densities, so we have the desired expression. □

Lemma C.2 (Numerically stable form for Step 2)**.** *Assume that $\sigma_G^2 > 0$. The posterior mean and covariance $\tilde{\mu}$ and $\tilde{\Sigma}$ can be written in a numerically stable form as:*

$$\tilde{\mu} = \mu - \Sigma G^T(\sigma_G^2 I + G\Sigma G^T)^{-1}(G\mu - b),$$
$$\tilde{\Sigma} = \Sigma - \Sigma G^T(\sigma_G^2 I + G\Sigma G^T)^{-1}G\Sigma.$$

Proof We use the following two Searle identities (corollaries of the Woodbury identity) (Petersen et al., 2008):

$$(I + CB)^{-1} = I - C(I + BC)^{-1}C, \tag{C.21a}$$
$$(C + BB^T)^{-1}B = C^{-1}B(I + B^TC^{-1}B)^{-1}, \tag{C.21b}$$

for some matrices B, C. Using Equation C.21a, we re-write A^{-1}:

$$A^{-1} = (I + \frac{1}{\sigma_G^2}\Sigma G^T G)^{-1} \tag{C.22a}$$
$$= I - \Sigma G^T(I + \frac{1}{\sigma_G^2}G\Sigma G^T)^{-1}\frac{1}{\sigma_G^2}G \tag{C.22b}$$
$$= I - \Sigma G^T(\sigma_G^2 I + G\Sigma G^T)^{-1}G. \tag{C.22c}$$

The desired expression for $\tilde{\Sigma}$ immediately follows by combining Equation C.22c with Lemma C.1. For $\tilde{\mu}$, we break the expression into two parts, and then use the Searle identity shown in Equation C.21b as follows:

$$\tilde{\mu} = A^{-1}(\mu + \frac{1}{\sigma_G^2}\Sigma G^T b) \tag{C.23a}$$
$$= A^{-1}\mu + A^{-1}\frac{1}{\sigma_G^2}\Sigma G^T b, \tag{C.23b}$$
$$A^{-1}\mu = (I - \Sigma G^T(\sigma_G^2 I + G\Sigma G^T)^{-1}G)\mu, \tag{C.23c}$$
$$A^{-1}\frac{1}{\sigma_G^2}\Sigma G^T b = (\Sigma^{-1} + \frac{1}{\sigma_G^2}G^T G)^{-1}\frac{1}{\sigma_G^2}G^T b \tag{C.23d}$$
$$= \frac{1}{\sigma_G^2}\Sigma G^T(I + \frac{1}{\sigma_G^2}G\Sigma G^T)^{-1}b \tag{C.23e}$$
$$= \Sigma G^T(\sigma_G^2 I + G\Sigma G^T)^{-1}b. \tag{C.23f}$$

Adding the expressions in Equation C.23c and Equation C.23f yields the desired form for $\tilde{\mu}$. □

Observe that the matrix $\sigma_G^2 I + G\Sigma G^T \in \mathbb{R}^{T\times T}$ is invertible for all values of σ_G^2 (including zero), since it is square in the smaller dimension and has full rank T. In addition, inverting $\sigma_G^2 I + G\Sigma G^T \in \mathbb{R}^{T\times T}$ has reduced computational complexity compared to inverting A. The matrix $\Sigma G^T(\sigma_G^2 I + G\Sigma G^T)^{-1}$ is commonly refered to as the *Kalman gain*, as it specifies how the prior μ should be updated based on the observation error $b - G\mu$ (Lopes and Tsay, 2011).

Lemma C.3 (Solution to constrained optimization)**.** *The expression for the posterior mean $\tilde{\mu}$ with $\sigma_G^2 > 0$ is equivalent to solving the following constrained least-squares problem for some value of $c > 0$:*

$$\tilde{\mu} = \text{argmin}_y \frac{1}{2}\|y - \mu\|^2_{\Sigma^{-1}},$$

subject to $\frac{1}{2}\|Gy - b\|_2^2 < c$, where $c < \frac{1}{2}\|G\mu - b\|_2^2$.

Proof This is a standard result from ridge regression (Hastie et al., 2013).

Since $c < \frac{1}{2}\|G\mu - b\|_2^2$, the complementary slackness condition requires that $c = \frac{1}{2}\|Gy - b\|_2^2$. Thus, we get the following Lagrangian:

$$L(y, \lambda) = \frac{1}{2}\|y - \mu\|^2_{\Sigma^{-1}} + \lambda\left(\frac{1}{2}\|Gy - b\|_2^2 - c\right).$$

Observe that, if we re-label $y := u$ and $\lambda := 1/\sigma_G^2$, then $L(y, \lambda)$ is equal to $-\log p(u|b, D) + C_2$, where C_2 is a constant with respect to y. Thus, the optimal value of y is the posterior mean from Equation C.19a, i.e.,

$$\nabla_y L(y, \lambda) = 0 \iff y = \tilde{\mu},$$

where

$$\tilde{\mu} = \mu - \Sigma G^T(\frac{1}{\lambda} I + G\Sigma G^T)(G\mu - b).$$

Next, we substitute the above expression for $\tilde{\mu}$ into the remaining feasibility condition:

$$\begin{aligned}
c &= \frac{1}{2}\|G\tilde{\mu} - b\|_2^2 \\
&= \|G\left(\mu - \Sigma G^T(\frac{1}{\lambda} I + G\Sigma G^T)^{-1}(G\mu - b)\right) - b\|_2^2 \\
&= \|G\mu - G\Sigma G^T(\frac{1}{\lambda} I + G\Sigma G^T)^{-1}(G\mu - b) - b\|_2^2 \\
&= \|\left(I - G\Sigma G^T(\frac{1}{\lambda} I + G\Sigma G^T)^{-1}\right)(G\mu - b)\|_2^2.
\end{aligned}$$

The eigenvalues of matrix $I - G\Sigma G^T[(1/\lambda)I + G\Sigma G^T]^{-1}$ shrink to 0 as $1/\lambda \to 0$. This establishes that c and $1/\lambda$ have a monotonic relationship. Hence, one can find a value of c such that $\lambda = 1/\sigma_G^2$.

C.7 Proof of Theorem 4.1

In this section, we provide the proof for Theorem 4.1. We begin by first restating Theorem 4.1.

Theorem 4.1. *Let μ and Σ be the mean and covariance of u obtained at the end of Step 1. Let $\sigma_{G,n} \downarrow 0$ be a monotonic decreasing sequence of constraint values and let $\tilde{\mu}_n$ be the corresponding posterior mean at the end of Step 2 shown in Equation 4.8. Then:*

1. *The sequence $\tilde{\mu}_n$ converges to a limit $\tilde{\mu}^\star$ monotonically; i.e., $\|\tilde{\mu}_n - \tilde{\mu}^\star\|_{\Sigma^{-1}} \downarrow 0$.*
2. *The limiting mean $\tilde{\mu}^\star$ is the solution to a constrained least-squares problem: $\operatorname{argmin}_y \|y - \mu\|_{\Sigma^{-1}}$ subject to $Gy = b$.*
3. *The sequence $G\tilde{\mu}_n$ converges to b in L_2; i.e., $\|G\tilde{\mu}_n - b\|_2 \downarrow 0$.*

Moreover, if the conservation constraint $Gu = b$ holds exactly for the true solution u, then:

4. *The distance between the true solution u and the posterior mean $\tilde{\mu}_n$ decreases as $\sigma_{G,n} \to 0$, i.e., $\|\tilde{\mu}_n - u\|_{\Sigma^{-1}} \downarrow \|\tilde{\mu}^\star - u\|_{\Sigma^{-1}}$.*
5. *For sufficiently small $\sigma_{G,n}$, the log-likelihood LL$(u; \tilde{\mu}_n, \tilde{\Sigma}_n)$ is greater than LL$(u; \mu, \Sigma)$ and increases as $\sigma_{G,n} \to 0$.*

For the proof of Theorem 4.1, recall the following expression for the posterior mean from Equation 4.8a:

$$\tilde{\mu}_n = \mu - \Sigma G^T (\sigma_{G,n}^2 I + G\Sigma G^T)^{-1}(G\mu - b).$$

Proof of 1. Define $\tilde{\mu}^\star \equiv \mu - \Sigma G^T (G\Sigma G^T)^{-1}(G\mu - b)$. We will show that $\tilde{\mu}_n$ converges monotonically to $\tilde{\mu}^\star$ as follows:

$$\begin{aligned}
\tilde{\mu}_n - \tilde{\mu}^\star &= \Sigma G^T \left[(G\Sigma G^T)^{-1} - (\sigma_{G,n}^2 I + G\Sigma G^T)^{-1}\right](G\mu - b) && \text{(C.24a)}\\
&= \Sigma G^T \left[(G\Sigma G)^{-1}(-\sigma_{G,n}^2 I)(-\sigma_{G,n}^2 I - G\Sigma G^T)^{-1}\right](G\mu - b) && \text{(C.24b)}\\
&= \sigma_{G,n}^2 \Sigma G^T \left[(G\Sigma G^T)^{-1}(\sigma_{G,n}^2 I + G\Sigma G^T)^{-1}\right](G\mu - b) && \text{(C.24c)}\\
&= \sigma_{G,n}^2 \Sigma G^T \left[\sigma_{G,n}^2 G\Sigma G^T + I\right]^{-1}(G\mu - b). && \text{(C.24d)}
\end{aligned}$$

The above follows from the Searle identity

$$C^{-1} + B^{-1} = C^{-1}(C + B)B^{-1},$$

where $C = G\Sigma G^T$, $B = -(\sigma_{G,n}^2 I + G\Sigma G^T)$, and $C + B = -\sigma_{G,n}^2 I$. Then,

$$\|\tilde{\mu}_n - \tilde{\mu}^\star\|_{\Sigma^{-1}}^2 = (G\mu - b)^T \left[\sigma_G^2 (G\Sigma G^T) + I\right]^{-1} \sigma_G^2 G\Sigma\Sigma^{-1}\Sigma G^T \sigma_G^2 \left[\sigma_G^2 (G\Sigma G^T) + I\right]^{-1}(G\mu - b). \tag{C.25}$$

Focusing on the matrix, we obtain:

$$
\begin{aligned}
Q_n &:= \left[\sigma_G^2(G\Sigma G^T) + I\right]^{-1} \sigma_G^2 G\Sigma\Sigma^{-1}\Sigma G^T \sigma_G^2 \left[\sigma_G^2(G\Sigma G^T) + I\right]^{-1} && \text{(C.26a)}\\
&= \left[\sigma_G^2(G\Sigma G^T) + I\right]^{-1} \sigma_G^2 G\Sigma G^T \sigma_G^2 \left[\sigma_G^2(G\Sigma G^T) + I\right]^{-1} && \text{(C.26b)}\\
&= \sigma_G^2 \left[\sigma_G^2(G\Sigma G^T) + I\right]^{-1} \sigma_G^2 G\Sigma G^T \left[\sigma_G^2(G\Sigma G^T) + I\right]^{-1} && \text{(C.26c)}\\
&= \sigma_G^2 \left[\sigma_G^2(G\Sigma G^T) + I\right]^{-1} \left[\frac{1}{\sigma_{G,n}^2}(G\Sigma G^T)^{-1} + I\right]^{-1} && \text{(C.26d)}\\
&= \sigma_{G,n}^2 \left[I + \sigma_G^2 G\Sigma G^T + \frac{1}{\sigma_G^2}(G\Sigma G)^{-1} + I\right]^{-1} && \text{(C.26e)}\\
&= \sigma_{G,n}^4 \left[2\sigma_{G,n}^2 I + \sigma_G^4 G\Sigma G^T + (G\Sigma G^T)^{-1}\right]^{-1}. && \text{(C.26f)}
\end{aligned}
$$

Let λ_i, v_i be an eigenvalue and associated eigenvector of $G\Sigma G^T$, respectively. Then v_i is also an eigenvector of matrix Q_n with associated eigenvalue

$$
\frac{\sigma_{G,n}^4}{2\sigma_{G,n}^2 + \sigma_{G,n}^4\lambda_i + \lambda_i^{-1}} = \frac{1}{2\sigma_{G,n}^{-2} + \lambda_i + \lambda_i^{-1}\sigma_{G,n}^{-4}}.
$$

Since all the eigenvalues are strictly decreasing as $\sigma_{G,n} \to 0$, the value $\|\tilde{\mu}_n - \tilde{\mu}^\star\|_{\Sigma^{-1}}^2 = (G\mu - b)^T Q_n (G\mu - b) \downarrow 0$, as required. □

Proof of 2. Now, we show that $\tilde{\mu}^\star = \text{argmin}_y \|y - \tilde{\mu}^\star\|_{\Sigma^{-1}}^2$ subject to $Gy = b$. This constrained least-squares problem can be cast into the following constrained least-norm problem:

$$
\text{minimize} \|u\|_2^2, \text{ subject to } G\Sigma^{1/2}u = b - \Sigma^{-1/2}\mu,
$$

with the transformation $u = \Sigma^{-\frac{1}{2}}(y - \mu)$ or $y = \mu + \Sigma^{\frac{1}{2}}u$.

The final solution is

$$
\mu - \Sigma G^T (G\Sigma G^T)^{-1}(G\mu - b),
$$

which equals $\tilde{\mu}^\star$. □

Proof of 3. We show that the L_2 norm between the predicted conservation value and the true value, $\|G\tilde{\mu}_n - b\|_2^2$, converges monotonically to 0 as $\sigma_{G,n}^2 \to 0$. We start by substituting the expression for Equation 4.8a:

$$
\begin{aligned}
G\tilde{\mu}_n - b &= G\mu - G\Sigma G^T(\sigma_{G,n}^2 I + G\Sigma G^T)^{-1}(G\mu - b) - b\\
&= (I - G\Sigma G^T(\sigma_{G,n}^2 I + G\Sigma G^T)^{-1})(G\mu - b).
\end{aligned} \qquad \text{(C.27)}
$$

Let v_i be an eigenvector of $G\Sigma G^T$ and λ_i the associated eigenvector. Then v_i is also an eigenvector of $(I - G\Sigma G^T(\sigma_{G,n}^2 I + G\Sigma G^T)^{-1})$ with eigenvalue $1 - \lambda_i/(\sigma_{G,i}^2 + \lambda_i)$. Since all the eigenvalues are monotonically decreasing to zero as $\sigma_{G,n}^2 \to 0$ monotonically, $\|G\tilde{\mu}_n - b\|_2^2 \downarrow 0$. For $\sigma_G^2 = 0$, $G\tilde{\mu}_n - b = 0$. $\square$

Proof of 4. Define $P := \Sigma G^T(G\Sigma G^T)^{-1}G$, which is an orthogonal projection matrix since

$$P^2 = \Sigma G^T(G\Sigma G^T)^{-1}G\Sigma G^T(G\Sigma G^T)^{-1}G = P$$

and

$$\langle x, Py\rangle_{\Sigma^{-1}} = x^T\Sigma^{-1}Py = x^TG^T(G\Sigma G^T)^{-1}Gy = x^TP^T\Sigma^{-1}y = \langle Px, y\rangle_{\Sigma^{-1}}.$$

The norm $\|\tilde{\mu}_n - u\|_{\Sigma^{-1}}$ can be decomposed into two parts:

$$\|\tilde{\mu}_n - u\|_{\Sigma^{-1}} = \|P(\tilde{\mu}_n - u)\|_{\Sigma^{-1}} + \|(I - P)(\tilde{\mu}_n - u)\|_{\Sigma^{-1}}.$$

First, we show that the second term $\|(I-P)(\tilde{\mu}_n - u)\|_{\Sigma^{-1}}$ equals $\|\tilde{\mu}^\star - u\|_{\Sigma^{-1}}$ for all n as follows:

$$\begin{aligned}
(I-P)\tilde{\mu}_n &= (I-P)\mu - (I-P)\Sigma G^T(\sigma_G^2 I + G\Sigma G^T)^{-1}(G\mu - b)\\
&= (I-P)\mu - \Sigma G^T(\sigma_G^2 I + G\Sigma G^T)^{-1}(G\mu - b) + P\Sigma G^T(\sigma_G^2 I + G\Sigma G^T)^{-1}(G\mu - b)\\
&= (I-P)\mu - \Sigma G^T(G\Sigma G^T)^{-1}(G\mu - b) + \Sigma G^T(G\Sigma G^T)^{-1}G\Sigma G^T(\sigma_G^2 I + G\Sigma G^T)^{-1}(G\mu - b)\\
&= (I-P)\mu\\
&= \tilde{\mu}^\star - \Sigma G^T(G\Sigma G^T)^{-1}b,\\
(I-P)u &= u - \Sigma G^T(G\Sigma G^T)^{-1}Gu\\
&= u - \Sigma G^T(G\Sigma G^T)^{-1}b,
\end{aligned}$$

Therefore,

$$(I-P)\tilde{\mu}_n - (I-P)u = \tilde{\mu}^\star - u.$$

Next, we show that the first term $\|P(\tilde{\mu}_n - u)\|_{\Sigma^{-1}}$ is equal to the distance between $\tilde{\mu}_n$ and $\tilde{\mu}^\star$. We first compute:

$$P\tilde{\mu}_n = P\mu - P\Sigma G^T(\sigma_G^2 I + G\Sigma G^T)^{-1}(G\mu - b) \tag{C.28a}$$
$$= \Sigma G^T(G\Sigma G^T)^{-1}G\mu - \Sigma G^T(\sigma_G^2 I + G\Sigma G^T)^{-1}(G\mu - b), \tag{C.28b}$$
$$Pu = \Sigma G^T(G\Sigma G^{-1})Gu \tag{C.28c}$$
$$= \Sigma G^T(G\Sigma G^{-1})b. \tag{C.28d}$$

Then subtracting Equation C.28d from Equation C.28a gives:

$$P\tilde{\mu}_n - Pu = \Sigma G^T(G\Sigma G^T)^{-1}G\mu - \Sigma G^T(\sigma_G^2 I + G\Sigma G^T)^{-1}(G\mu - b) - \Sigma G^T(G\Sigma G^T)^{-1}b \tag{C.29a}$$
$$= \left(\Sigma G^T(G\Sigma G^T)^{-1} - \Sigma G^T(\sigma_G^2 I + G\Sigma G^T)^{-1}\right)(G\mu - b) \tag{C.29b}$$
$$= \tilde{\mu}_n - \tilde{\mu}^\star. \tag{C.29c}$$

From part 1, $\|\tilde{\mu}_n - \tilde{\mu}^\star\|_{\Sigma^{-1}} \downarrow 0$ monotonically as $\sigma_{G,n}^2 \downarrow 0$. Thus,

$$\|\tilde{\mu}_n - u\|_{\Sigma^{-1}}^2 = \|\tilde{\mu}_n - \tilde{\mu}^\star\|_{\Sigma^{-1}}^2 + \|\tilde{\mu}^\star - u\|_{\Sigma^{-1}}^2 \downarrow \|\tilde{\mu}^\star - u\|_{\Sigma^{-1}}^2.$$

□

Proof of 5. Recall that the predictive log-likelihood (LL) is defined as:

$$\mathrm{LL}(u; \tilde{\mu}_n, \tilde{\Sigma}_n) = -\frac{1}{2M}\|u - \tilde{\mu}_n\|_{\tilde{\Sigma}^{-1}} - \frac{1}{2}\sum_i \log \tilde{\Sigma}_{n,i,i} - \frac{1}{2M}\log 2\pi,$$

where M denotes the total number of points. Also recall that the precision is well-defined as:

$$\tilde{\Sigma}_n^{-1} = \Sigma^{-1} + \frac{1}{\sigma_{G,n}^2}G^T G,$$

so the first term of the predictive likelihood can be further decomposed as:

$$\|\tilde{\mu}_n - u\|_{\tilde{\Sigma}_n^{-1}}^2 = (\tilde{\mu}_n - u)^T\tilde{\Sigma}_n^{-1}(\tilde{\mu}_n - u) = (\tilde{\mu}_n - u)^T\Sigma^{-1}(\tilde{\mu}_n - u) + (\tilde{\mu}_n - u)^T\frac{1}{\sigma_G^2}G^T G(\tilde{\mu}_n - u)$$
$$= \|\tilde{\mu}_n - u\|_{\Sigma^{-1}}^2 + \frac{1}{\sigma_G^2}\|G\tilde{\mu}_n - b\|_2^2$$
$$= \|\tilde{\mu}_n - u\|_{\Sigma^{-1}}^2 + \|\frac{1}{\sigma_G}(G\tilde{\mu}_n - b)\|_2^2.$$

Substituting the expression from Equation C.27, we get:

$$\frac{1}{\sigma_G}(G\tilde{\mu}_n - b) = \frac{1}{\sigma_G}(I - G\Sigma G^T(\sigma_{G,n}^2 I + G\Sigma G^T)^{-1})(G\mu - b). \tag{C.30}$$

Let v_i be an eigenvector of $G\Sigma G^T$ and λ_i the associated eigenvector. Then v_i is also an eigenvector of $\frac{1}{\sigma_G}(I - G\Sigma G^T(\sigma_{G,n}^2 I + G\Sigma G^T)^{-1})(G\mu - b)$ with eigenvalue:

$$\frac{1}{\sigma_G}\left(1 - \frac{\lambda_i}{\sigma_{G,i}^2 + \lambda_i}\right) = \frac{\sigma_{G,n}}{\sigma_{G,n}^2 + \lambda_i} = \frac{1}{\sigma_{G,n} + \lambda_i \sigma_{G,n}^{-1}}.$$

For sufficiently small $\sigma_{G,n}$, the eigenvalues are monotonically decreasing to zero as $\sigma_{G,n}^2 \to 0$.

Finally, $\log(\tilde{\Sigma}_n)_{i,i}$ is non-increasing with respect to $\sigma_{G,n}^2$. From Equation 4.8b,

$$\begin{aligned}\tilde{\Sigma}_n &= \Sigma - \Sigma G^T(\sigma_{G,n}^2 I + G\Sigma G^T)G^T\Sigma,\\ (\tilde{\Sigma}_n)_{i,i} &= \Sigma_{i,i} - e_i^T\Sigma G^T(\sigma_{G,n}^2 I + G\Sigma G^T)G^T\Sigma e_i,\end{aligned}$$

where e_i denotes the i-th elementary vector. Since $\Sigma G^T(\sigma_G^2 I + G\Sigma G^T)^{-1}G\Sigma$ is positive definite with positive diagonal entries, and the eigenvalues of $(\sigma_G^2 I + G\Sigma G^T)^{-1}$ increase monotonically as $\sigma_{G,n} \to 0$, the entry $(\tilde{\Sigma}_n)_{i,i}$ decreases as $\sigma_{G,n} \to 0$.

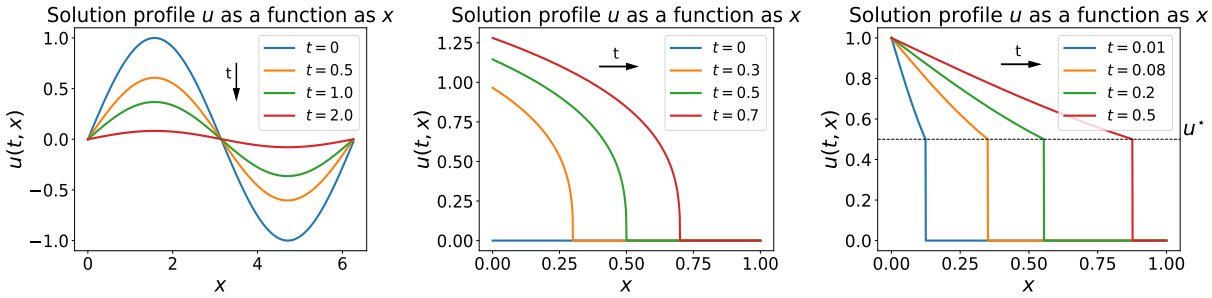
C.8 Additional Details on the Generalized Porous Medium Equation

In this section, we discuss in more detail the parametric Generalized Porous Medium Equation (GPME). The GPME is a *family* of conservation equations, parameterized by a nonlinear coefficient $k(u)$, and it has been used in several applications ranging from underground flow transport to nonlinear heat transfer to water desalination and beyond. Among other things, it has the parametric ability to represent pressure, diffusivity, conductivity, or permeability, in these and other applications (Vázquez, 2007). From the ML/SciML methods perspective, it has additional advantages, including closed-form self-similar solutions, structured nonlinearities, and the ability to choose the parameter $k(u)$ to interpolate between "easy" and "hard" problems (analogous to but distinct from the properties of elliptical versus parabolic versus hyperbolic PDEs).

The GPME Equation. The basic GPME is given as:

$$u_t - \nabla \cdot (k(u)\nabla u) = 0, \tag{C.31}$$

where $F(u) = -k(u)\nabla u$ is a nonlinear flux function, and where the parameter $k = k(u)$ can be varied (to model different physical phenomena, or to transition between "easy" PDEs and "hard" PDEs). Even though the equation appears to be parabolic, for small values of $k(u)$ in the nonlinear case, it exhibits degeneracies, and it is is called "degenerate parabolic." By varying k, solutions span from "easy" to "hard," exhibiting many of the qualitative properties of smooth/nice parabolic to sharp/hard hyperbolic PDEs. Among other things, this includes discontinuities associated with self-sharpening occurring over time, even for smooth initial conditions.



(a) easy: Diffusion equation ($k = 1$) (b) medium: PME ($k(u) = u^3$) (c) hard: Stefan (discont. $k(u)$)

Figure C.3: Illustration of the "easy-to-hard" paradigm for PDEs, for the GPME family of conservation equations: (a) "easy" parabolic smooth (diffusion equation) solutions, with constant parameter $k(u) = k \equiv 1$; (b) "medium" degenerate parabolic PME solutions, with nonlinear monomial coefficient $k(u) = u^m$, with parameter $m = 3$ here; and (c) "hard" hyperbolic-like (degenerate parabolic) sharp solutions (Stefan equation) with nonlinear step-function coefficient $k(u) = \mathbf{1}_{u \geq u^\star}$, where $\mathbf{1}_{\mathcal{E}}$ is an indicator function for event $\mathcal{E}$.

Figure C.3 (Figure 4.1 repeated here) provides an illustration of this "easy-to-hard" paradigm for PDEs for the three classes of the GPME considered in the main text. In particular, Figure C.3(a) illustrates an "easy" situation, with $k(u) \equiv 1$, where we have a simple parabolic solution to the linear heat/diffusion equation, where a sine initial condition is gradually smoothed over time. Figure C.3(b) illustrates a situation with "medium" difficulty, namely the degenerate parabolic Porous Medium Equation (PME) with nonlinear differentiable monomial coefficient $k(u) = u^m$. Here, for $m = 3$, a constant zero initial condition self-sharpens, and it develops a sharp gradient that does not dissipate over time (Maddix et al., 2018b). Finally, Figure C.3(c) illustrates an example of the "hard" Stefan problem, where the coefficient $k(u)$ is a nonlinear discontinuous step-function of the unknown u defined by the unknown value $u^\star = u(t, x^\star(t)) = 0.5$ at the discontinuity location $x^\star(t)$. In this case, the solution evolves as a rightward moving shock or moving interface over time (Maddix et al., 2018a).

Here, we provide more details on these and other classes of the GPME.

Heat/Diffusion Equation. Perhaps the simplest non-trivial form of the GPME, where the conductivity or diffusivity coefficient

$$k(u) = k > 0,$$

is a constant, corresponds to the heat (or diffusion) equation. In this case, Equation 4.9 reduces to the linear parabolic equation, $u_t = k\Delta u$, where Δ denotes the Laplacian operator. Solutions of this equation are smooth due to the diffusive nature of the Laplacian operator, and even sharp initial condition are smoothed over time.

Variable Coefficient Problem. The linear variable coefficient problem

$$k(u, x) = k(x),$$

is also a classical parabolic equation. The variable coefficient problem is commonly used in reservoir simulations to model the interface between permeable and impermeable materials, where $k(u)$ denotes the step-function permeabilities that depends on the spatial position x.

Porous Medium Equation (PME). Another subclass of the GPME, in which the coefficient is nonlinear but smooth, is known as the Porous Medium Equation (PME). The PME is known to be degenerate parabolic, and it becomes more challenging as m increases. The PME with $m = 1$ has been widely used to model isothermal processes, e.g., groundwater flow and population dynamics in biology. For $m > 1$, the PME results in sharp solutions, and it has been used to describe adiabatic processes and nonlinear phenomena such as heat transfer of plasma (ionized gas).

Super-slow Diffusion Problem. Another subclass of the GPME, known as super-slow diffusion, occurs when

$$k(u) = \exp(-1/u).$$

Here, the diffusivity $k(u) \to 0$ as $u \to 0$ faster than any power of u. This equation models the diffusion of solids at different absolute temperatures u. The coefficient $k(u)$ represents the mass diffusivity in this case, and it is connected with the Arrhenius law in thermodynamics.

Stefan Problem. The most challenging case of the GPME is when the coefficient $k(u)$ is a discontinuous nonlinear step function:

$$k(u) = \begin{cases} k_{\max},\ u \geq u^\star \\ k_{\min},\ u < u^\star, \end{cases} \tag{C.32}$$

for given constants $k_{\max}$, $k_{\min}$ and $u^\star \in \mathbb{R}$, in which case it is known as the Stefan problem. The Stefan problem has been used to model two-phase flow between water and ice, crystal growth, and more complex porous media such as foams (van der Meer et al., 2016).

We conclude by noting that, even though the GPME is nonlinear in general, for specific initial and boundary conditions, it has closed form self-similar solutions. For details, see Vázquez (2007); Maddix et al. (2018b,a). This enables ease of evaluation by comparing each competing method to the ground truth.

C.9 Detailed Experiment Settings

In this section, we review the basics of the Attentive Neural Process (ANP) (Kim et al., 2019) that we use as the black-box deep learning model in Step 1 of our model PROBCONSERV-ANP in the empirical results Figure 5.5. Figure C.4 illustrates a schematic for PROBCONSERV-ANP that shows how in the first step the mean and covariance estimates μ, Σ from the ANP are fed into our probabilistic constraint in the second step to output the updated mean and covariance estimates $\tilde{\mu}, \tilde{\Sigma}$.

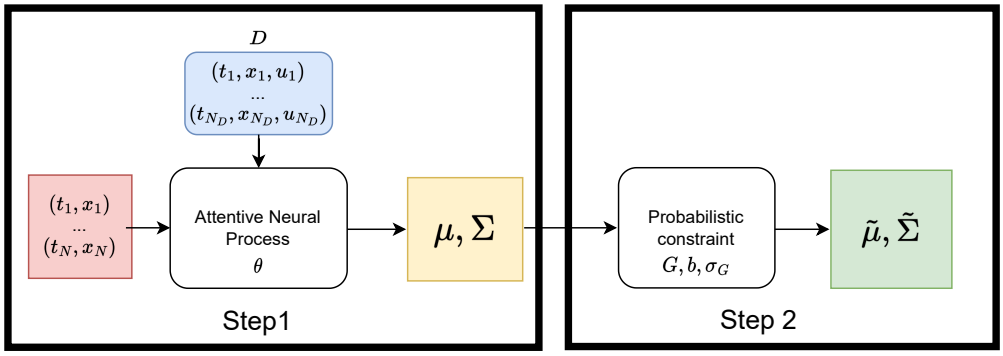


Figure C.4: Schematic for the instantiation of our framework PROBCONSERV with the ANP (PROBCONSERV-ANP) as the data-driven black box model in Step 1 that is used in the empirical results. In Step 1, the ANP outputs a mean μ and covariance Σ (yellow) of the solution profile u evaluated at the N target points (red). The ANP takes as input the context set D that comprises N_D labelled points (blue). The parameter θ encapsulates the neural network weights within the ANP. In Step 2, the probabilistic constraint in Equation 4.8 is applied yielding an updated mean $\tilde{\mu}$ and covariance $\tilde{\Sigma}$ (green). The probabilistic constraint is determined by the matrix G, value b, and variance σ_G^2 in Equation 4.7.

Model training. The model from Step 1 is data-driven, with parameter θ that needs to be learned from data. Given an empirical data distribution, written as $(u, b, D) \sim p$, we maximize the expected joint likelihood of the function u and the constraint b, conditioned on data D, as a function

of the Step 1 parameter θ and Step 2 parameters σ_G and G as follows:

$$\begin{aligned} L(\theta, \sigma_G, G) &= \mathbb{E}_{u,b,D\sim p} \log p(u, b|D) \\ &= \underbrace{\mathbb{E}_{u,D\sim p} \log p_\theta(u|D)}_{\text{Step 1}} + \underbrace{\mathbb{E}_{u,b} \log p_{\sigma_G,G}(b|u)}_{\text{Step 2}}. \end{aligned} \tag{C.33}$$

This follows because the joint probability can be broken into conditional distributions using Bayes' Rule. The Step 2 constraint only depends on the value u.

The Step 1 parameter θ is only present in the first term of the summation in Equation C.33. Then, the optimal value for $\theta^\star$ is found by optimizing the unconstrained log-likelihood from Step 1 over the empirical data distribution and is given as follows:

$$\begin{aligned} \theta^\star &= \arg\max_\theta L(\theta, \sigma_G, G) \\ &= \arg\max_\theta \mathbb{E}_{u,D\sim p} \log p_\theta(u|D). \end{aligned} \tag{C.34}$$

Equation C.34 is simply the optimization target of several generative models, e.g., Gaussian processes and the ANP. This justifies training the Step 1 black-box model with its original training procedure before applying our Step 2.

Data Generation. For each PDE instance, we first generate training data for the data-driven model in Step 1. We generate these samples, indexed by i, by randomly sampling n_{train} values of the PDE parameters α_i from an interval $\mathcal{A}$. To create the input data D_i, the solution profile corresponding to α_i is evaluated on a set of N_D points uniformly sampled from the spatiotemporal domain $[0, t] \times \Omega$. Then, the reference solution for u with parameter α_i, denoted u_i, is evaluated over another set of N_{train} uniformly-sampled points. The Step 1 model (ANP) is then trained on these supervised input-output pairs, (D_i, u_i). Using Equation 4.5, the conservation value b in Step 2 is calculated given the parameter α_i. At inference time, we fix specific values of the PDE parameters α that are of interest and generate new input-output pairs to evaluate the predictive performance. The settings are the same as those at training, except that the reference solution is evaluated on a fixed grid that evenly divides the time domain $[0, t]$ into T_{test} points and the spatial domain Ω into M_{test} points for a spatio-temporal grid of $N_{\text{test}} = T_{\text{test}} \times M_{\text{test}}$ points. For consistent results, we repeat this procedure over n_{test} independent datasets for each α.

Table C.3 provides the training settings and Table C.4 provides the cor responding test settings.

We describe here how the input data D; input points $(t_1, x_1), \ldots, (t_N, x_N)$; and solution u are created for a particular draw of PDE parameter $\alpha \in \mathcal{A}$. The input data (a.k.a. the context

PDE	Parameter	$\mathcal{A}$	Time domain $[0, t]$	Spatial domain Ω	n_{train}	N_D	N_{train}
Diffusion	k	$[1, 5]$	$[0, 1]$	$[0, 2\pi]$	10,000	100	100
PME	m	$[1, 6]$	$[0, 1]$	$[0, 1]$	10,000	100	100
Stefan	$u^\star$	$[0.55, 7]$	$[0, 0.1]$	$[0, 1]$	10,000	100	100

Table C.3: Training details for each instance of the GPME (Diffusion, PME, Stefan) used in the experiments.

PDE	Parameter values	Test time	Spatial domain Ω	n_{test}	N_D	T_{test}	M_{test}	N_{test}
Diffusion	$k \in \{1, 5\}$	0.5	$[0, 2\pi]$	50	100	201	201	$40,401$
PME	$m \in \{1, 3, 6\}$	0.5	$[0, 1]$	50	100	201	201	$40,401$
Stefan	$u^\star \in \{0.6\}$	0.05	$[0, 1]$	50	100	201	201	$40,401$

Table C.4: Testing details for each instance of the GPME (Diffusion, PME, Stefan) used in the experiments.

set) D is generated as follows. First, draw samples from the spatiotemporal domain $(t_n, x_n) \sim \text{Uniform}([0, t] \times \Omega)$, for $n = 1, \ldots, N_D$. For each sample (t_n, x_n), evaluate the reference solution $u_n := u(t_n, x_n)$ for α. Then $D = \{(t_n, x_n, u_n)\}_{n=1,\ldots,N_D}$.

We create input points $(t_1, x_1), \ldots, (t_N, x_N)$ differently depending on whether we are training or testing. At train-time, the input points are sampled uniformly from the spatiotemporal domain

$$(t_n, x_n) \sim \text{Uniform}([0, t] \times \Omega),$$

for $n = 1, \ldots, N_{\text{train}}$. At test-time, we divide up the time domain $[0, t]$ into T_{test} evenly-spaced points and the spatial domain Ω into M_{test} evenly-spaced points. We then take the cross product of these as the set of input points, whose size is $N_{\text{test}} = T_{\text{test}} \times M_{\text{test}}$. Finally, over the set of input points, we evaluate the reference solution for α as: $u = [u(t_n, x_n)]_{n=1,\ldots,N_{\text{train}}}$.

Attentive Neural Processes (ANP). The Attentive Neural Process (ANP) (Kim et al., 2019) models the conditional distribution of a function u at target input points $\{x_n\} := x_1, \ldots, x_N$ for $x_i \in \mathbb{R}^{D+1}$ given a small set of context points $D := \{x_i, u_i\}_{i \in C}$. The function values at each target point x_n, written as u_n, are conditionally independent given the latent variable z with the following

distribution for u_n:

$$\begin{aligned} p_\theta(u_n|D) &= \int_z p_\theta(u_n|z, D)p_\theta(z|D)dz, \\ p_\theta(u_n|z, D) &= p_\mathcal{N}(u_n|\mu_n, \sigma_n^2), \\ p_\theta(z|\mu_z, \Sigma_z) &= p_\mathcal{N}(z|\mu_z, \Sigma_z), \\ \mu_n, \sigma_n &= f_\theta^u(x_n, z, f_\theta^r(x_n, D)), \\ \mu_z, \Sigma_z &= f_\theta^z(D). \end{aligned} \tag{C.35}$$

Here, $p_\mathcal{N}(u|\mu, \sigma^2) := (2\pi\sigma^2)^{-1/2} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$ denotes the univariate normal distribution with mean μ and variance σ^2 and f_θ^z, f_θ^u, and f_θ^r are neural networks whose architecture is described in more detail below.

As standard in variational inference, the attentive neural process (ANP) is trained to maximize the evidence lower bound (ELBO), which is a tractable lower bound to the marginal likelihood $\mathbb{E}_{u,D\sim p} \log p_\theta(u|D)$ that we want to maximize in Equation C.34:

$$\begin{aligned} \mathbb{E}_{u,D\sim p} \log p_\theta(u|D) &\geq \mathbb{E}_{u,D\sim p}\mathbb{E}_{z\sim q_\phi} \log p_\theta(u, z|D) - \log q_\theta(z|u, D), \\ q_\theta(z|u, D) &= p_\mathcal{N}(z|\mu_z^q, \Sigma_z^q), \\ \mu_z^q, \Sigma_z^q &= f_\theta^z(D \cup \{(t_1, x_1, u_1), \dots (t_N, x_N, u_N)\}). \end{aligned} \tag{C.36}$$

By concatenating the context set D with the target set, the ANP can use the same networks for both the generative model p_θ and the variational model q_θ. This differs from methods such as the variational auto-encoder (VAE) that train a separate network for the variational model.

In the experiments, we train the ELBO in Equation C.36 using stochastic gradient descent over random mini-batches of the supervised pairs (u, D) and a sample of the latent variable z (using the reparameterization trick for an unbiased gradient estimate). Specifically, we use the ADAM optimizer with a learning rate of 1×10^{-4} and a batch size of 250.

Further architectural details. Here, we briefly describe the architecture of the ANP used in experiments; a more thorough description of the ANP in general can be found in the original paper (Kim et al., 2019).

Symbol	Value	Description
d_x	2	Input dimension
d_u	1	Output dimension
d_z	128	Latent dimension
h	128	Size of hidden layer
n_{heads}	4	Number of heads in MultiHead
d_h	128	Column dimension in MultiHead layers

Table C.5: ANP hyperparameters.

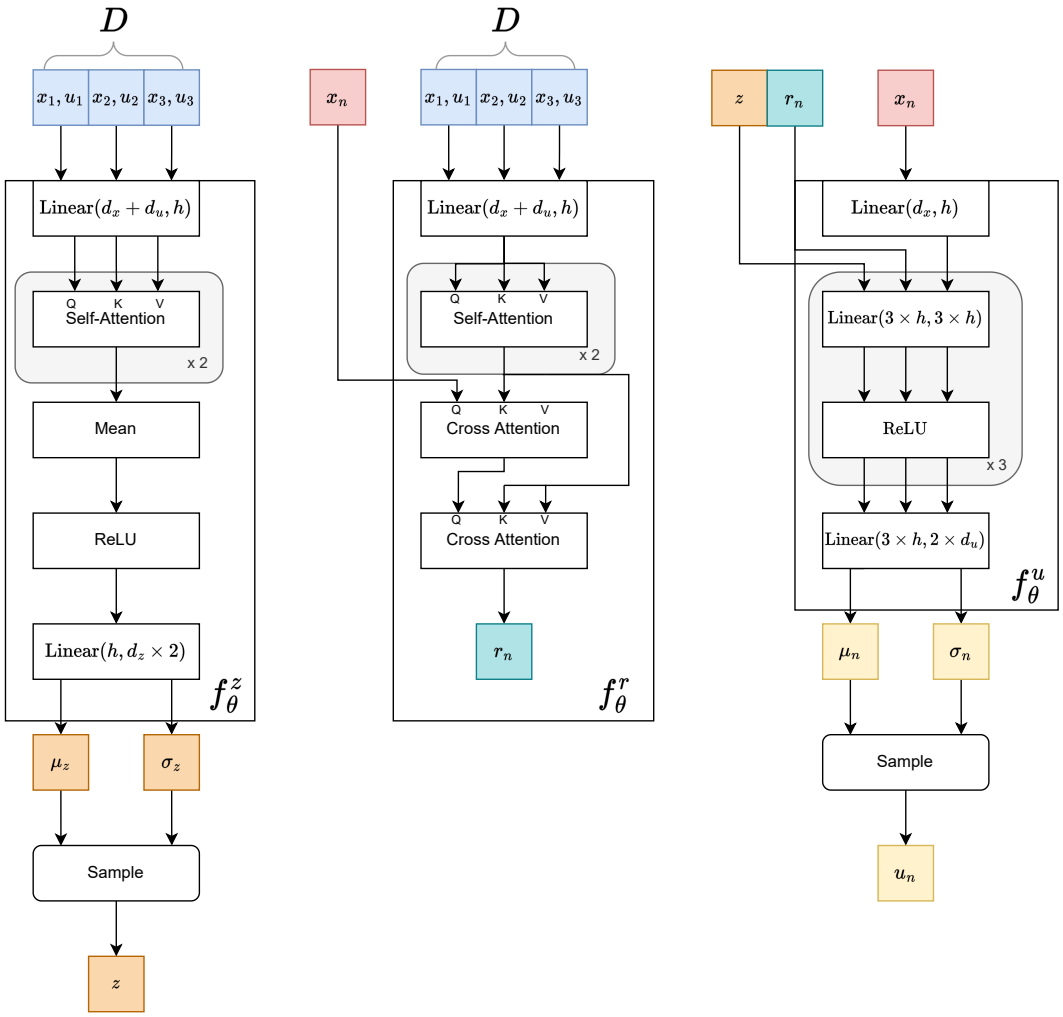


Figure C.5: Architectural diagram of the three main networks that make up the Attentive Neural Process (ANP) that is used in the experiments as the Step 1 black-box model.

The ANP consists of three distinct networks:

1. The *latent encoder* f_θ^z takes the context set $D = \{x_i, u_i\}_{i \in C}$ as input and outputs a mean μ_z and diagonal covariance Σ_z for the latent representation z. Note that f_θ^z is invariant to the order of the context set inputs in D.

2. The *deterministic encoder* f_θ^r takes the context set $D = \{x_i, u_i\}_{i \in C}$ and the target points $\{x_n\}$ as input, and outputs a set of deterministic representations $\{r_n\}$ corresponding to each target point. Note that f_θ^r is permutation-invariant to the order of the context set inputs in D, and is applied pointwise across the target inputs $\{x_n\}$.

3. The *decoder* f_θ^u takes the outputs from the latent encoder, deterministic encoder, and the target points $\{x_n\}$ as input, and outputs a set of mean and variances $\{\mu_n, \sigma_n\}$ corresponding to each target point. The decoder is applied pointwise across the target inputs $\{x_n\}$ and deterministic representation $\{r_n\}$.

For reproducibility, Figure C.5 shows how each network is constructed. Each building blocks is also briefly described below:

- Linear($d_{\text{in}}, d_{\text{out}}$): dense linear layer $xA + b$.
- Mean: Averages the inputs of the input set; i.e., Mean$(\{s_i\}) = \frac{1}{|\{s_i\}|} \sum_i s_i$.
- ReLU: Applies ReLU activation pointwise.
- Cross-Attention and Self-Attention. These are multi-head attention blocks first introduced in Vaswani et al. (2017). The three inputs to the multi-head attention block are the queries $Q = [q_1|\dots|q_{d_q}]^\top$, keys $K = [k_1|\dots|q_{d_k}]^\top$, and values $V = [v_1|\dots|v_{d_k}]^\top$. The hyperparameters are the number of heads, n_{heads} and the number of columns of the matrices W_i^Q, W_i^K, W_i^V, denoted as d_h. We summarize the notations below:

$$
\begin{aligned}
\text{Self-Attention}(Q) &:= \text{MultiHead}(Q, Q, Q), \\
\text{Cross-Attention}(Q, K, V), &:= \text{MultiHead}(Q, K, V), \\
\text{MultiHead}(Q, K, V) &:= [H_1|\dots|H_{n_{\text{heads}}}]W^O, \\
H_i &:= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V), \\
\text{Attention}(Q, K, V) &:= \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V, \\
\text{softmax}\left(\begin{bmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,n} \end{bmatrix}\right) &:= \begin{bmatrix} \frac{\exp(x_{1,1})}{\sum_{i=j}^n \exp(x_{1,j})} & \dots & \frac{\exp(x_{1,n})}{\sum_{j=1}^n \exp(x_{1,j})} \\ \vdots & \ddots & \vdots \\ \frac{\exp(x_{m,1})}{\sum_{i=1}^m \exp(x_{m,j})} & \dots & \frac{\exp(x_{m,n})}{\sum_{j=1}^n \exp(x_{m,j})} \end{bmatrix}.
\end{aligned}
$$

C.10 Additional Empirical Results

In this section, we provide additional empirical results for the degenerate parabolic Generalized Porous Medium (GPME) family of conservation laws as well as for hyperbolic conservation laws.

C.10.1 GPME Family of Conservation Laws

Here, we include additional solution profiles and conservation profiles over time for the GPME family of equations, ranging from the "easy" diffusion (heat), "medium" PME, to the "hard" Stefan equations.

C.10.1.1 Diffusion (Heat) Equation

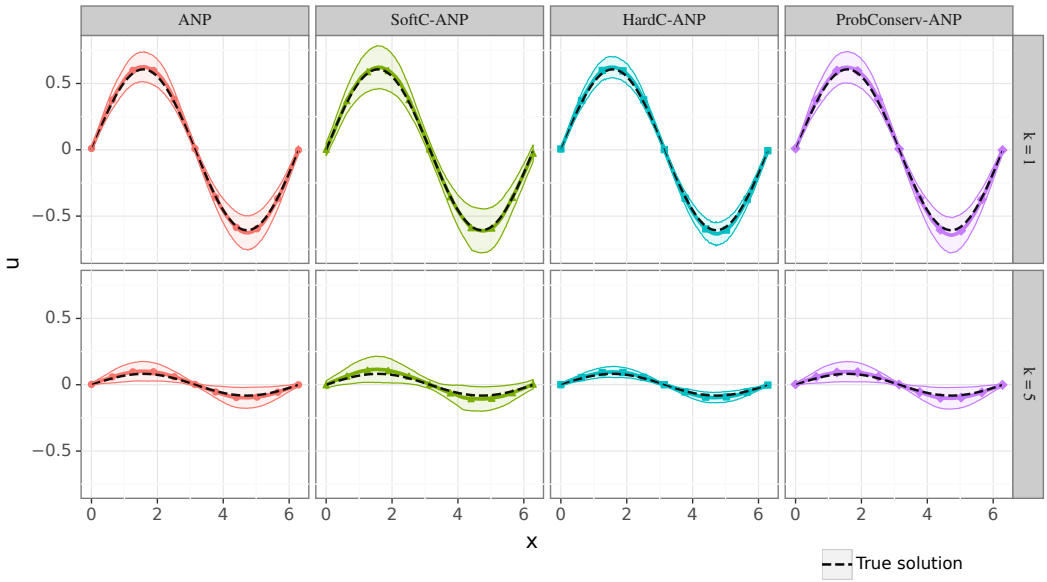


Figure C.6: Solution profiles for the diffusion (heat) equation at time $t = 0.5$ for diffusivity (conductivity) test-time parameter $k = 1$ in the top row and $k = 5$ in the bottom row. Each model is trained on samples of $k \in \mathcal{A} = [1, 5]$. The shaded region illustrates ± 3 standard deviation uncertainty intervals. PROBCONSERV-ANP and HARDC-ANP both display tighter uncertainty bounds than the baseline ANP, while SOFTC-ANP is more diffuse. The uncertainty is relatively homoscedastic on this "easy" case.

Solution profiles. Figure C.6 shows the solution profiles for the "easy" diffusion equation, at time $t = 0.5$, where a sine curve is damped over time for test-time parameter $k = 1, 5 \in \mathcal{A} = [1, 5]$. Table C.6 shows the corresponding metrics.

	$k=1$			$k=5$		
	CE	LL	MSE	CE	LL	MSE
ANP	4.68 (0.10)	2.72 (0.02)	1.71 (0.41)	1.76 (0.04)	3.28 (0.02)	0.547 (0.08)
SOFTC-ANP	3.47 (0.17)	2.40 (0.02)	2.24 (0.78)	2.86 (0.05)	2.83 (0.02)	1.75 (0.24)
HARDC-ANP	**0** (0.00)	**3.08** (0.04)	**1.37** (0.33)	**0** (0.00)	**3.64** (0.03)	**0.461** (0.07)
PROBCONSERV-ANP	**0** (0.00)	**2.74** (0.02)	**1.55** (0.33)	**0** (0.00)	**3.30** (0.02)	**0.485** (0.07)

Table C.6: Mean and standard error for CE $\times 10^{-3}$ (should be zero), LL (higher is better) and MSE $\times 10^{-4}$ (lower is better) over $n_{\text{test}} = 50$ for the ("easy") diffusion equation at time $t = 0.5$ with variable diffusivity constant k parameter in the range $\mathcal{A} = [1, 5]$ and test-time parameter values $k = 1, 5$.

C.10.1.2 Porous Medium Equation (PME)

Results for different λ for SOFTC-ANP. As is the case with PINNs (Raissi et al., 2019), the SOFTC-ANP method has a hyper-parameter λ that controls the balance in the training loss between the reconstruction and differential term. A higher value of λ places more emphasis on the residual of the PDE term and less emphasis on the evidence lower bound (ELBO) from the ANP.

To investigate whether tuning λ will lead to significantly different results, we report results for different values of λ for the SOFTC-ANP on the Porous Medium Equation (PME). Since these results are presented on the same test dataset used in Table 4.2, it provides an optimistic case on how tuning λ could improve the results for SOFTC-ANP. Table C.7 shows that the predictive performance is roughly the same across different values of λ, with both MSE and LL worse than the original ANP across the board and the conservation error (CE) $G\mu - b$ at the final time worse for $m = 6$.

	$m=1$			$m=3$			$m=6$		
	CE	LL	MSE	CE	LL	MSE	CE	LL	MSE
ANP ($\lambda = 0$)	6.67	**3.49**	**0.94**	-1.23	**3.67**	**1.90**	$\mathbf{-2.58}$	**3.81**	**7.62**
SOFTC-ANP ($\lambda = 0.01$)	5.58	3.11	1.11	-0.61	3.46	2.03	-3.00	3.49	7.76
SOFTC-ANP ($\lambda = 0.1$)	5.58	3.11	1.11	-0.67	3.46	2.07	-3.01	3.49	7.87
SOFTC-ANP ($\lambda = 1$)	5.62	3.11	1.11	-0.65	3.46	2.06	-3.03	3.49	7.82
SOFTC-ANP ($\lambda = 10$)	**5.52**	3.11	1.08	$\mathbf{-0.56}$	3.46	2.04	-3.02	3.49	7.76
SOFTC-ANP ($\lambda = 100$)	5.62	3.11	1.11	-0.59	3.46	2.03	-3.03	3.49	7.69

Table C.7: Investigation of the effect of the soft constraint penalty parameter λ in the SOFTC-ANP baseline. The metrics CE $\times 10^{-3}$ (should be zero), LL (higher is better) and MSE $\times 10^{-4}$ (lower is better) are reported for the ("medium") PME at time $t = 0.5$ with variable m parameter in the range $\mathcal{A} = [0.99, 6]$ and test-time parameters $m \in \{1, 3, 6\}$. We see that the performance is not significantly changed as a function of λ, and, surprisingly, that the unconstrained ANP ($\lambda = 0$) performs better in most metrics than SOFTC-ANP.

PROBCONSERV-ANP with diffusion. As described in subsection C.4.2, we explore adding numerical diffusion for eliminating artificial small-scale noises when enforcing conservation. Table C.8 shows that adding artificial diffusion improves both MSE and LL compared to the conservation constraint alone. Figures C.7-C.8 illustrate that by removing such artificial noises, PROBCONSERV-ANP with diffusion leads to tighter uncertainty bounds as well as higher LL than the other baselines.

	$m = 1$			$m = 3$			$m = 6$		
	CE	LL	MSE	CE	LL	MSE	CE	LL	MSE
ANP	6.67 (0.39)	3.49 (0.01)	0.94 (0.09)	−1.23 (0.29)	3.67 (0.00)	1.90 (0.04)	−2.58 (0.23)	3.81 (0.01)	**7.67** (0.09)
SOFTC-ANP	5.62 (0.35)	3.11 (0.01)	1.11 (0.14)	−0.65 (0.30)	3.46 (0.00)	2.06 (0.03)	−3.03 (0.26)	3.49 (0.00)	7.82 (0.09)
HARDC-ANP	**0** (0.00)	3.16 (0.04)	0.43 (0.04)	**0** (0.00)	3.44 (0.03)	1.86 (0.03)	**0** (0.00)	3.40 (0.05)	**7.61** (0.09)
PROBCONSERV-ANP	**0** (0.00)	3.56 (0.01)	**0.17** (0.02)	**0** (0.00)	3.68 (0.00)	2.10 (0.07)	**0** (0.00)	3.83 (0.01)	10.4 (0.04)
PROBCONSERV-ANP (w/diff)	**0** (0.00)	**4.04** (0.02)	**0.15** (0.02)	**0** (0.00)	**3.96** (0.00)	**1.43** (0.05)	**0** (0.00)	**4.03** (0.01)	7.91 (0.03)

Table C.8: Mean and standard error for CE $\times 10^{-3}$ (should be zero), LL (higher is better) and MSE $\times 10^{-4}$ (lower is better) over $n_{\text{test}} = 50$ runs for the ("medium") PME at time $t = 0.5$ with variable m parameter in the range $\mathcal{A} = [0.99, 6]$. We see that PROBCONSERV-ANP (w/diff) improves the performance on PROBCONSERV-ANP by applying smoothing at the sharp boundary as the test-time parameter m is increased.

Solution and error profiles. Figures C.7-C.8 illustrate the differing solution profiles and errors for the PME for various values of $m \in \{1, 3, 6\}$, respectively. As expected, we see a gradient for $m > 1$ that becomes sharper and approaches infinity for $m = 6$. Increasing m results in smaller values of the PDE parameter denoting the pressure $k(u) = u^m$, which increases the degeneracy for smaller values of $k(u)$, i.e., larger values of m. In this case the problem also becomes more challenging. For $m = 1$, we have a piecewise linear solution, and for $m = 3, 6$ we see sharper oscillatory uncertainty bounds at the front or free boundary, resulting in some negative values at this boundary as well. We see the value of the uncertainty quantification to reflect that the model is certain in the parabolic regions to the left and right of the sharp boundary especially in the zero (degeneracy) region, and is most uncertain at the boundary (degeneracy) point.

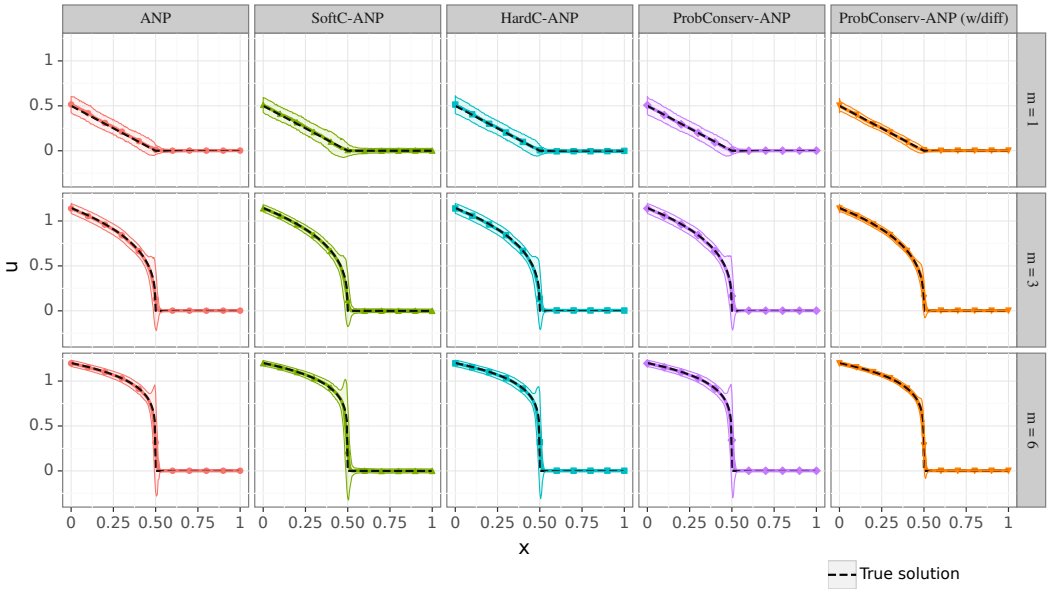


Figure C.7: Solution profiles and uncertainty intervals for the PME predicted by our PROBCONSERV-ANP and other baselines. The solutions are obtained for three scenarios with increasing sharpness in the profile as m is increased from $m = 1$ to $m = 6$ from left to right, respectively. The HARDC-ANP model, which assumes constant variance for the whole domain, results in too high uncertainty in the zero (degenerate) region, unlike our proposed PROBCONSERV-ANP approach that incorporates the variance information to effectively handle this heteroscedasticity. Adding diffusion to PROBCONSERV-ANP removes the oscillations locally at the degeneracy, as desired.

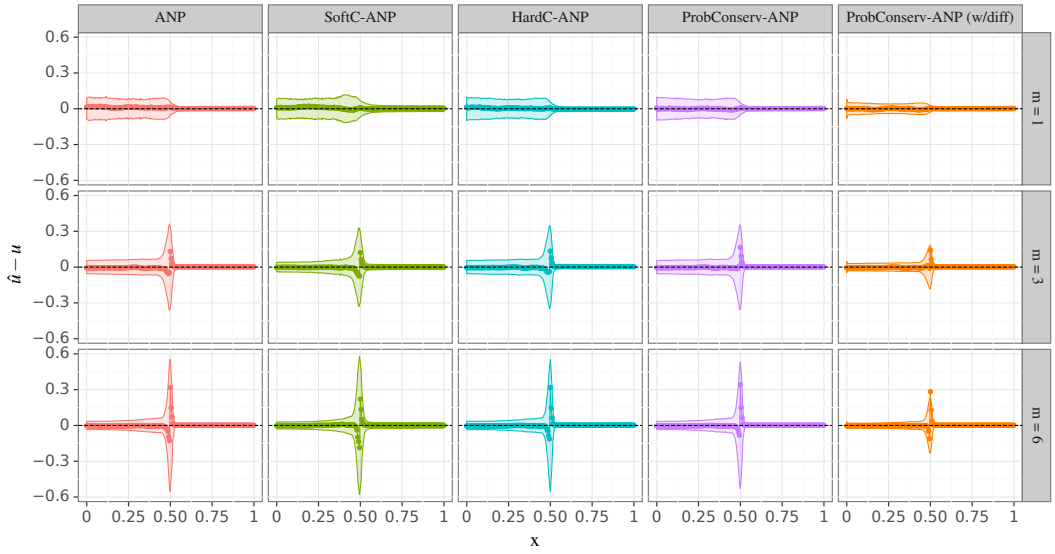


Figure C.8: The solution errors with uncertainty bounds as a function of x for our PROBCONSERV-ANP and other baselines for the PME with parameter $m \in \{1, 3, 6\}$ after training on $m \in \mathcal{A} = [0.99, 6]$. The shaded region indicates ± 3 standard deviations as estimated by each model. For $m = 1$, both PROBCONSERV-ANP with and without diffusion result in solutions with smaller errors. While HARDC-ANP model reduces the error scale, it underestimates the zero portion of the solution, which is nonphysical, as the solution quantity cannot be negative. For $m \in \{3, 6\}$, while the error magnitude becomes dominant at the shock position for all methods, PROBCONSERV-ANP with diffusion provides the lowest errors with the tightest confidence interval.

C.10.1.3 Stefan

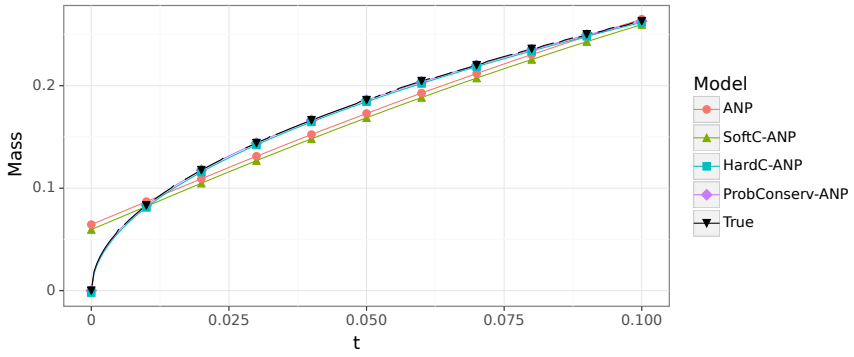


Figure C.9: True mass over time for each model. The true mass conservation profile over time is matched exactly by our proposed method PROBCONSERV-ANP and the hard-constrained HARDC-ANP by design. The unconstrained ANP and surprisingly even the differential form soft-constrained SOFTC-ANP have a non-physical linear mass profile over time.

Figure C.9 shows PROBCONSERV-ANP follows the true profile of conserved mass in the system over time by design. We also see that the unconstrained ANP and surprisingly the soft-constrained SOFTC-ANP that applied the differential form as a soft constraint does not result in conservation being satisfied since it does not enforce it exactly. For these baselines, the mass profile over time is linear and does not match the true profile which is proportional to $\sqrt{t}$.

C.10.2 Hyperbolic Equations

Here, we demonstrate that our approach PROBCONSERV-ANP also works for hyperbolic conservation laws by considering the linear advection problem ("medium") and Burgers' equation ("hard"), which are both introduced in Table C.2 in section C.3.

C.10.2.1 Linear Advection

Figure C.10 displays the system total mass, $U(t) = \int_\Omega u(t,x)d\Omega$ as a function of time, obtained by our PROBCONSERV-ANP model and the other baselines and compared against the true curve. The results are obtained for two values of test-time parameter $\beta = 1, 3$ denoting the velocity with training range $\beta \in \mathcal{A} = [1, 5]$. The unconstrained ANP contradicts the system true mass at all times including $t = 0$. By proper incorporation of the conservation constraint, both PROBCONSERV-ANP and HARDC-ANP methods are able to predict the system mass and capture the actual trend over time exactly while the soft-constrained differential form SOFTC-ANP baseline results in little improvement.

Figure C.11 shows the predicted solution profiles and corresponding uncertainty intervals for time $t = 0.1$ and test-time parameter $\beta = 1, 3$. Our PROBCONSERV-ANP model predicts sharper shock profile centered around the actual shock position. On the contrary, both ANP and HARDC-ANP lead to highly diffusive profiles which are shifted toward the left of actual shock interface, leading to the under-estimation of the shock position on this downstream task. This under-estimation becomes more evident in Figure C.12, which indicates the corresponding histograms of shock position. The histograms associated with the ANP and HARDC-ANP models are skewed to the left and both result in the averaged shock positions which are lower than the actual value depicted by the solid vertical line. By proper leveraging of our finite volume based physical constraint, our PROBCONSERV-ANP results in proper uncertainty quantification which leads to accurate prediction of shock location compared to the other baseline models. Table C.9 also shows this accuracy improvement with a maximum improvement of $2.86\times$ in MSE for $\beta = 1$.

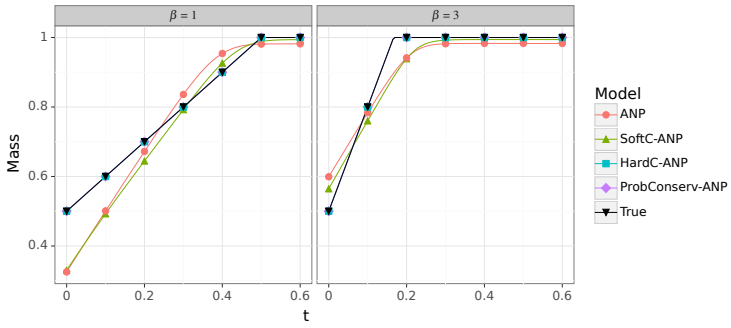


Figure C.10: System total mass as a function of time t for the linear advection problem with test-time parameter $\beta = 1, 3$ and training parameter range $\mathcal{A} = [1, 5]$. Both PROBCONSERV-ANP and HARDC-ANP satisfy conservation of mass while the unconstrained ANP and soft-constrained SOFTC-ANP baselines deviate from the actual trend completely at all times.

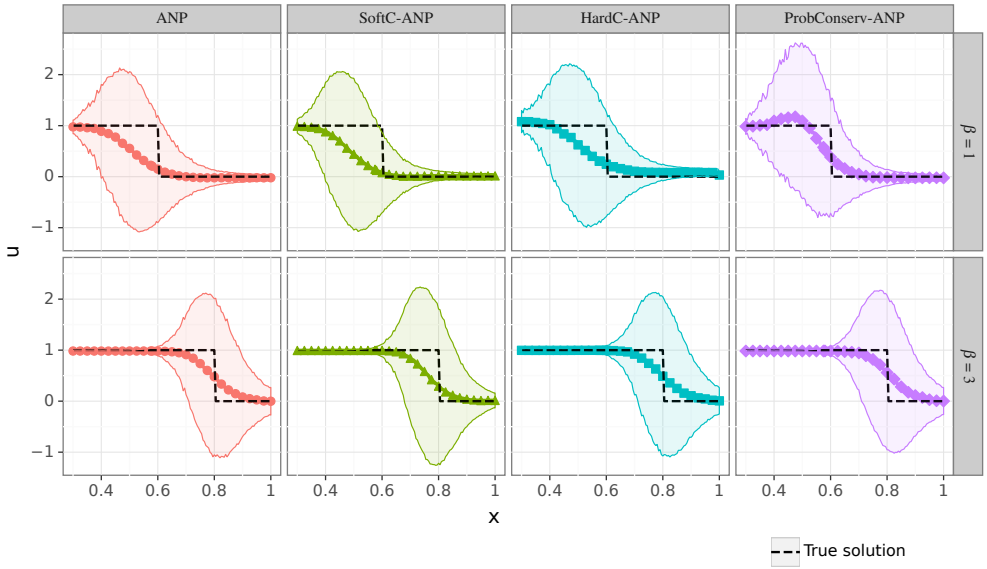


Figure C.11: Solution profiles and uncertainty intervals for linear advection problem at time $t = 0.1$ for test-time parameter $\beta = 1, 3$ and training parameter range $\mathcal{A} = [1, 5]$. Despite satisfaction of conservation constraint, HARDC-ANP predicts a highly diffusive profile and remarkable underestimation of shock interface region especially for $\beta = 1$. The prediction error is even higher for the unconstrained ANP model which does not enforce the conservation, and the shock interface is shifted further away from the true solution. PROBCONSERV-ANP results in a sharper profile than other the baselines and the predicted shock interface is around the actual shock position leading to more accurate shock position estimation.

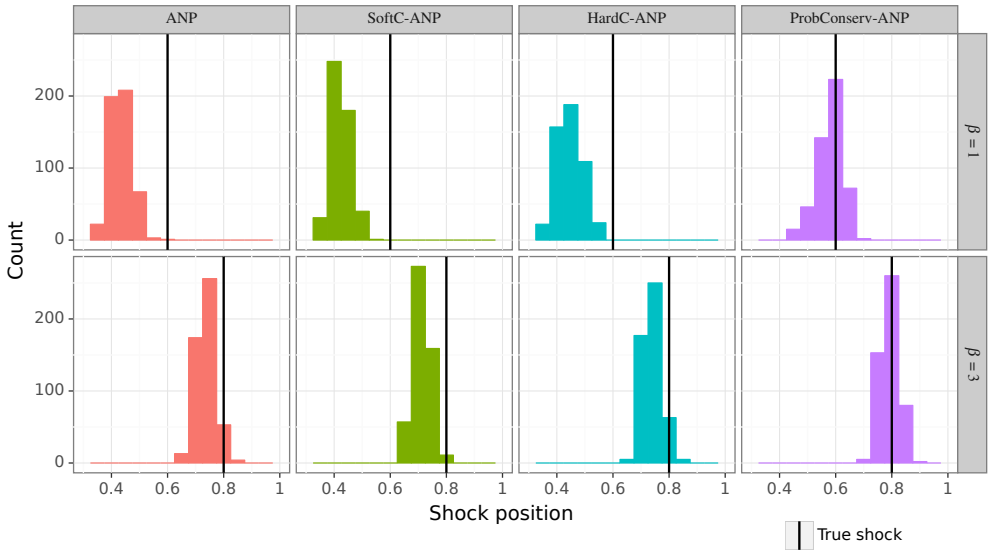


Figure C.12: The histogram of shock position for the linear advection problem, computed as the mean plus or minus 3 standard deviations. Due to the shift in the shock interface, both the ANP and HARDC-ANP models underestimate the position of the shock, and the underestimation is more significant for $\beta = 1$. The PROBCONSERV-ANP model provides a histogram distributed almost symmetrically around the true shock interface and thus leads to an accurate estimate of the shock position.

	$\beta = 1$			$\beta = 3$		
	CE	LL	MSE	CE	LL	MSE
ANP	-0.136 (0.004)	0.96 (0.01)	5.72 (0.25)	0.042 (0.003)	0.51 (0.01)	2.03 (0.01)
SOFTC-ANP	-0.137 (0.004)	**1.58** (0.03)	7.64 (0.34)	0.013 (0.003)	**2.31** (0.02)	2.87 (0.20)
HARDC-ANP	**0** (0.00)	-2.96 (0.34)	4.59 (0.17)	**0** (0.00)	1.34 (0.21)	1.93 (0.07)
PROBCONSERV-ANP	**0** (0.00)	1.06 (0.01)	**2.00** (0.06)	**0** (0.00)	0.52 (0.01)	**1.62** (0.01)

Table C.9: Mean and standard error for CE (should be zero), LL (higher is better) and MSE $\times 10^{-2}$ (lower is better) over $n_{\text{test}} = 50$ runs for the hyperbolic linear advection problem at time $t = 0.1$ with variable β parameter in the range $\mathcal{A} = [1, 5]$.

C.10.2.2 Burgers' Equation

Figure C.13 illustrates that the total mass is linear over time, and in this case is approximately satisfied by our PROBCONSERV-ANP and the baselines. Figure C.14 shows the waiting time phenomenon, where the piecewise linear initial condition self-sharpens until the breaking time $t_b = 1/a$, where it forms a rightward moving shock. We see that the breaking time is inversely proportional to the slope, and that the shock forms sooner for larger values of a.

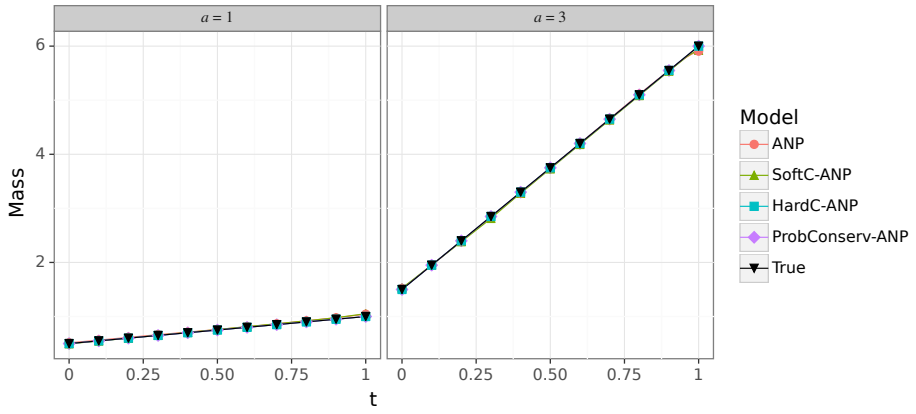


Figure C.13: True mass as a function of time t for the Burgers' equation with test-time parameter $a = 1, 3$ and training parameter range $\mathcal{A} = [1, 4]$.

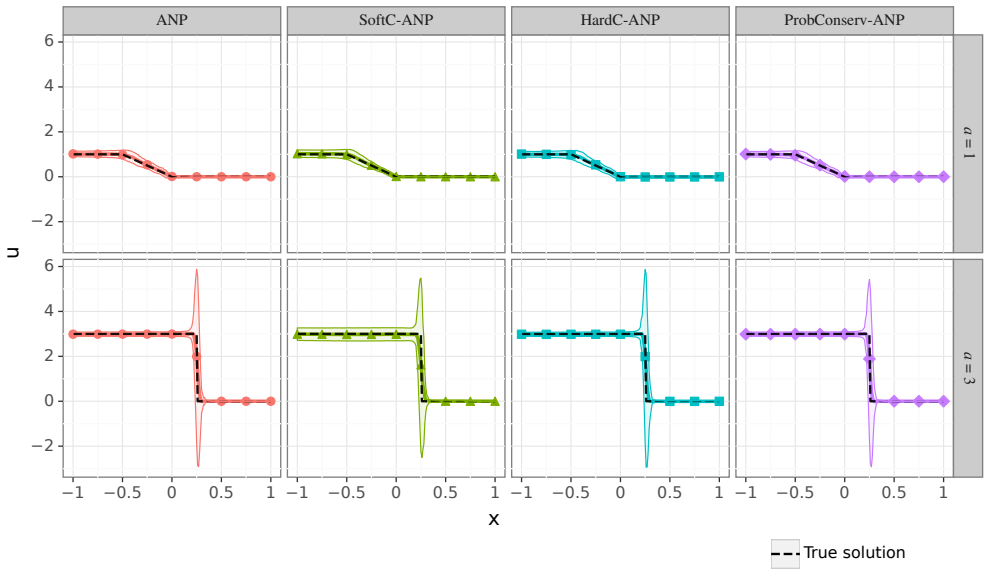


Figure C.14: Solution profiles and uncertainty intervals for Burgers' equation at time $t = 0.5$ for test-time parameter $a = 1, 3$ and training parameter range $\mathcal{A} = [1, 4]$.

APPENDIX D

Appendix for Chapter 5

D.1 Normalizing flows

Normalizing flows (Papamakarios et al., 2021) represent a general framework for density estimation of a multi-dimensional distribution with arbitrary dependencies. Briefly, suppose $X \sim \mathcal{P}_X$ is a random variable in $\mathbb{R}^d$. Now, let $Z \sim \mathcal{N}(0, I_d)$ be a multivariate standard normal distribution. We assume there exists a mapping G that is triangular, increasing, and differentiable such that

$$G(X) = Z.$$

A formal treatment of when such a G exists can be found in Bogachev et al. (2005). However, a sufficient condition is that the density of X is greater than 0 on $\mathbb{R}^d$ and the cumulative density function of X_j, conditional on the previous components $X_{\leq j}$, is differentiable with respect to $X_j, X_{\leq j}$ (Papamakarios et al., 2021):

$$U_i = G_i(X) \equiv F_i(X_i | X_{\leq i})$$

From this construction, each U_i is independent of all previous U_i and has distribution $\mathrm{Unif}[0, 1]$. From there, we simply set $Z_i = \Phi^{-1}(U_i)$, where Φ is the CDF of the standard normal.

Since $G_i(X)$ depends only on the elements in X up to i, it is triangular. Because $p_X > 0$, the conditional cdfs are strictly increasing, so G is an increasing map. Finally, since each cdf is differentiable, the entire map G is differentiable, and its Jacobian is non-zero.

Because of the inverse mapping theorem, G is invertible and we can write

$$X = G(Z).$$

Normalizing flows are a collection of distributions that parameterize a family of invertible, differentiable transformations G_θ from a fixed base distribution Z to an unknown distribution X.

Using the change-of-variables theorem, we can express the distribution of X in terms of the base distribution density p_Z and the transformation G_θ:

$$p_\theta(X) = p(G_\theta(X)) \left| \det \left(\frac{\partial G_\theta(X)}{\partial X} \right) \right|$$

where $\frac{\partial G_\theta(X)}{\partial X}$ is the Jacobian of G. The goal is to find a parameter value $\hat{\theta}$ that maximizes the likelihood of the observed X:

$$\hat{\theta} = \arg\max_\theta p_\theta(X).$$

A key feature of normalizing flows is that they are composable.

D.1.1 Flow Architecture

In experiments, the first layer G is a Gaussianization flow (Meng et al., 2020) applied elementwise:

$$G_j(X_j) = \Phi^{-1} \left(\sum_{m=1}^{M} \sigma \left(\frac{X_j - \mu_{j,m}}{s_{j,m}} \right) \right),$$

where Φ^{-1} is the standard normal inverse CDF. With sufficiently large M, this Gaussianization layer can approximate any univariate distribution. This is composed with a Masked Autoregressive Flow (MAF) F (Papamakarios et al., 2017), which consists of MADE layers interspersed with batch normalization and reverse permutation layers:

$$\begin{aligned}
\text{MADE}_{j,k} &= (X_j - \mu_{j,k}) \exp(-\alpha_{j,k}) \\
\text{where } \mu_j &= f_{\mu_{j,k}}(X_{<j}) \\
\alpha_j &= f_{\alpha_{j,k}}(X_{<j}) \\
F &= \text{MADE}_{j,K} \circ \text{BatchNorm} \circ \text{Reverse} \circ \text{MADE}_{j,K-1} \circ \cdots \text{BatchNorm} \circ \text{Reverse} \circ \text{MADE}_{j,1}
\end{aligned}$$

Here, f_{μ_j} and f_{α_j} are fully connected neural networks.

D.2 Proof of convergence

Theorem 5.1. *Let $X \in \mathbb{R}_{N\times D}$ be a random feature matrix, where each row $X_{i,\cdot}$ is independent and identically distributed; $x \in \mathbb{R}_{N\times D}$ be the observed feature matrix; and α_j be the p-value as defined in eq.* (5.4) *with test statistic $T_j(X)$. Suppose there exists a sequence of functions $(G^n)_{n=1}^{\infty}$ and a base random variable Z satisfying the following conditions:*

1. *Each G^n is continuously differentiable and invertible.*
2. *$G^n \to G$ pointwise for some map G that is triangular, increasing, continuously differentiable, and satisfies $G(X_{i,\cdot}) \overset{D}{=} Z$.*

For $n = 1, 2, \ldots$, let X^n be the random feature matrix where each row i is independent and has distribution $X^n_{i,\cdot} = (G^n)^{-1}(Z)$. Then, the p-value in eq. (5.5) *calculated using K MCMC samples targeting $X^n_{\cdot,j} \mid X^n_{\cdot,-j} = x_{\cdot,-j}$ converges to the correct p-value α_j with probability* 1.

Proof of Theorem 1. Without loss of generality, we consider the first feature, which is indexed by $j = 1$. Let p_X be the density of each row of the matrix $X_{i,\cdot}$ and p_Z the density of the base variable Z. For each i.i.d observation at $i = 1, \ldots, N$, we define F to be the cumulative distribution function of $X_{i,1}$ conditional on the other features $X_{i,-1} = x_{i,-1}$:

$$\begin{aligned} F(x_1) \triangleq \mathcal{P}(X_{i,1} \le x_1 | X_{i,-1} = x_{i,-1}) &= \frac{\int_{-\infty}^{x_1} p_X(x_1', x_{i,-1}) dx_1'}{\int_{-\infty}^{\infty} p_X(x_1', x_{i,-1}) dx_1'} \\ &= \frac{\int_{-\infty}^{x_1} p_Z(G(x_1', x_{i,-1}))|\partial G(x_1', x_{i,-1})| dx_1'}{\int_{-\infty}^{\infty} p_Z(G(x_1', x_{i,-1}))|\partial G(x_1', x_{i,-1})| dx_1'}. \end{aligned} \tag{D.1}$$

For a particular mapping G^n, we define F^n analogously:

$$F^n(x_1) \triangleq \frac{\int_{-\infty}^{x_1} p_Z(G^n(x_1', x_{i,-1}))|\partial G^n(x_1', x_{i,-1})| dx_1'}{\int_{-\infty}^{\infty} p_Z(G^n(x_1', x_{i,-1}))|\partial G^n(x_1', x_{i,-1})| dx_1'}. \tag{D.2}$$

Since G^n and G are continuously differentiable,

$$p_Z(G^n(x_1', x_{i,-1}))|\partial G^n(x_1', x_{i,-1})| \to p_Z(G(x_1', x_{i,-1}))|\partial G(x_1', x_{i,-1})| \text{ as } n \to \infty. \tag{D.3}$$

Then, by the dominated convergence theorem, $F^n \to F$ pointwise.

Let $X^n_{i,1} \sim F^n$. Since $F^n \to F$ pointwise, and F is a distribution function, $X^n_{i,1}$ converges in distribution to $X_{i,1} \mid X_{i,-1} = x_{i,-1}$. Likewise, the joint distribution across all independent observations, written $X^n_{\cdot,1}$, converges in distribution to $X_{\cdot,1} \mid X_{\cdot,-1} = x_{\cdot,-1}$.

Now, let $\tilde{X}^n_{\cdot,1}$ be equal in distribution to $X^n_{\cdot,1}$, but sampled such that it is independent of the outcome Y. It follows from the reasoning above that $\tilde{X}^n_{\cdot,1}$ converges to the desired null distribution $\tilde{X}_{\cdot,1}|X_{\cdot,-1}$ as $n \to \infty$. Define $g_1(\tilde{x}_{\cdot,1}) \triangleq 1[T_1(X) < T_1([\tilde{x}_{\cdot,1}, X_{\cdot,-1}])]$. With the regularity condition that T_1 is discontinuous on a set of measure zero, the expectation converges:

$$\lim_{n\to\infty} \mathbb{E}_{\tilde{X}^n_{\cdot,1}}(g_1) \to \mathbb{E}_{\tilde{X}_{\cdot,1}|X_{\cdot,-1}=x_{\cdot,-1}}(g_1) = \alpha_1. \tag{D.4}$$

The Cesaro average of g calculated over MCMC samples that target the distribution of $\tilde{X}^n_{\cdot,1}$ under the probability law of G_n converges almost surely to $\mathbb{E}_{\tilde{X}^n_{\cdot,1}}(g_1)$ (Smith and Roberts, 1993). That is,

$$\lim_{K\to\infty} \hat{\alpha}_{j,K,n} = \lim_{K\to\infty} \frac{1}{K} \sum_{k=1}^{K} g_1(\tilde{X}_{\cdot,1,k}) = \mathbb{E}_{\tilde{X}^n_{\cdot,1}}(g_1)\ w.p.1. \tag{D.5}$$

Combining eq. (D.4) and eq. (D.5) gives the desired result. □

D.2.1 Variable selection methods

Linear For the linear response, we estimate a linear model with an L1 penalty (aka the LASSO) on training data:

$$\hat{\beta} = \arg\min_{\beta} \frac{1}{N}\|X\beta - Y\|_2^2 + \lambda \sum_{j=1}^{D} |\beta_j| \tag{D.6}$$

The penalization term λ is selected via 5-fold cross-validation.

Nonlinear For the nonlinear response, we fit a random forest on the training data. The hyperparameters are the defaults in the scikit-learn implementation.

Feature statistic If $\hat{f}(X)$ is the fitted regression function, then the feature statistic is the negative mean-squared error:

$$T(X, Y) = -\frac{1}{N}\|\hat{f}(X) - Y\|_2^2.$$

D.2.2 Competing methods

For DDLK (Sudarshan et al., 2020), KnockoffGAN (Jordon et al., 2019), and DeepKnockoffs, (Romano et al., 2020), we used the exact architecture and hyperparameter settings from their respective papers. For the ablation study in section 5.5.3, we use the exact implementation in Tansey et al. (2021). For these methods, we used the code that the researchers graciously made publicly available:

Method	Link
DDLK	https://github.com/rajesh-lab/ddlk/
DeepKnockoffs	https://github.com/msesia/deepknockoffs/
HRT (MDN)	https://github.com/tansey/hrt/
KnockoffGAN	https://github.com/firmai/tsgan/tree/master/alg/knockoffgan

For MASS (Gimenez et al., 2019), we followed their described procedure and fit a mixture of Gaussians to the feature distribution using scikit-learn, selecting the number of components via the Akiake Information Criterion (AIC). We then used the `knockoffs` R package, available on CRAN, to sample knockoffs using the estimated parameters for each component.

For RANK (Fan et al., 2020), we estimate the sparse precision matrix using the Graphical LASSO (Friedman et al., 2008) implemented in sci-kit learn, using cross-validation to tune the regularization parameter. We then use the `knockoffs` R package to sample the knockoffs with this covariance.

D.3 Architecture and training details for soybean GWAS

Discrete flows For the discrete flows in the soybean example, we use a single layer of MADE which outputs a dimension of size 4. μ is then set equal to the argmax of this output.

For training the flows, we use a relaxation of argmax with temperature equal to 0.1.

Discrete MCMC Each feature has $K = 4$ values, so we can enumerate all four possible states for each proposal and sample in proportional to these probabilities via a Gibbs Sampling procedure. Setting the probabilities leads to an acceptance rate of 1, and the samples are uncorrelated since the previous sample doesn't enter into the proposal distribution

Predictive model For the predictive model of each trait conditional on the SNPs, we use a fully connected neural network. This network has three hidden layers of size 128, 256, and 128. ReLU activations are used between each fully connected layer. Dropout is used on both the input layer and after each hidden layer with $p = 0.2$. The learning rate in ADAM was set to 1×10^{-5}, with early stopping implemented using a held-out validation set.

The feature statistic for each sample is the negative mean-squared error (MSE) for each observation.

Runtime To obtain sufficient resolution on roughly 4200 simultaneous tests, we drew 100,000 samples from our model. The runtime was 10 hours using a single NVIDIA 2080 Ti.

Selected SNPs table D.1 shows the SNPs selected by FLOWSELECT that are associated with oil content in soybeans.

Chromosome	SNP	p-value
4	Gm04_42203141	1.60e-04
5	Gm05_37467797	1.90e-04
8	Gm08_15975626	2.10e-04
14	Gm14_1753922	9.00e-05
14	Gm14_1799390	1.60e-04
14	Gm14_1821662	2.90e-04
18	Gm18_1685024	5.00e-05

Table D.1: Selected SNPs for soybean GWAS experiment.

D.4 Runtime comparison of each controlled feature selection method

Method	Runtime (min)
DeepKnockoff	3.0
KnockoffGAN	3.73
MASS	12.6
DDLK	91.9
FLOWSELECT	59.4

Table D.2: The median runtime for each method on the scRNA-seq data with $D = 100$ features and $N = 100,000$ observations. All experiments were implemented using PyTorch, except for KnockoffGAN, which was implemented in Tensorflow, and MASS, which we implemented using scikit-learn and the knockoffs R package. The experiments were conducted using an Intel Xeon Gold 6130 CPU and an NVIDIA GeForce RTX 2080 Ti GPU.

D.5 Comparison to Holdout Randomization Test of Tansey et al. (2021)

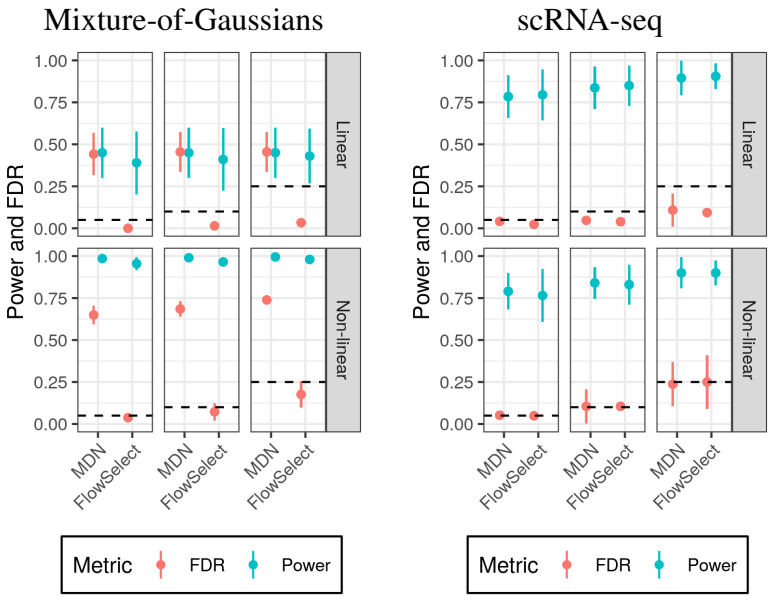


Figure D.1: Comparison of FLOWSELECT to the HRT procedure, which samples the complete conditionals using multiple mixture-density-networks (MDNs). Each column shows the power and observed false discovery rate (FDR) at targeted FDRs of 0.05, 0.1, and 0.25 (indicated by the dashed lines). The experimental settings for each dataset are the same as in Figure 5.2.

D.6 Oracle Model-X

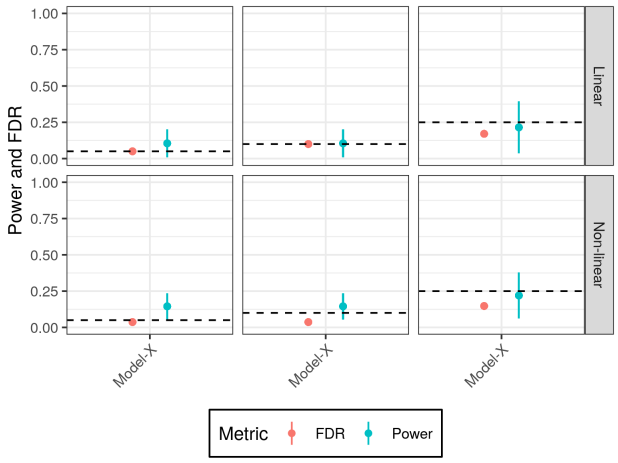


Figure D.2: FDR control and power of Oracle Model-X knockoffs on the mixture-of-Gaussians dataset (compare to Figure 5.2).

D.7 DDLK with true joint distribution

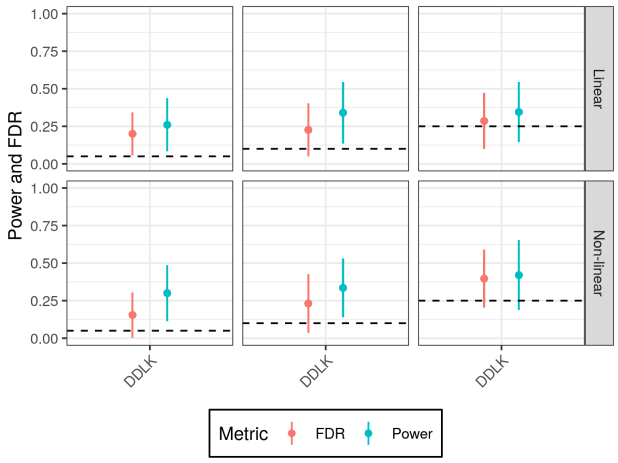


Figure D.3: FDR control and power of DDLK on the mixture-of-Gaussians dataset using the ground truth feature density in training (compare to Figure 5.2).

D.8 Observed Power and FDR control for given number of MCMC samples

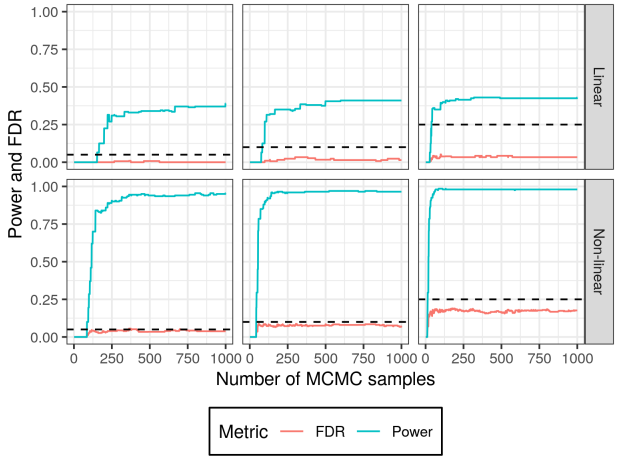


Figure D.4: FDR control and power of FLOWSELECT on the mixture-of-Gaussians dataset for a given number of MCMC samples at targeted FDRs of 0.05, 0.1, and 0.25 (indicated by the dashed lines). This suggests that the consequence of terminating the MCMC chain prematurely leads to a drop in power but FDR control is still maintained.

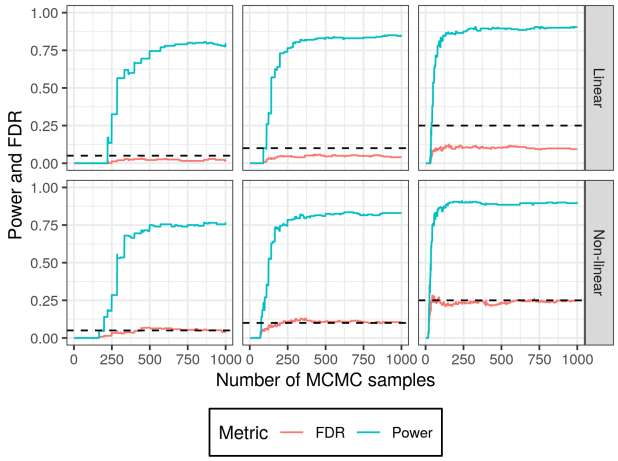


Figure D.5: Power and FDR control of FLOWSELECT on the scRNA-seq dataset as a function of the number of MCMC samples at targeted FDRs of 0.05, 0.1, and 0.25 (indicated by the dashed lines). This suggests that the consequence of terminating the MCMC chain prematurely leads to a drop in power but FDR control is still maintained.

D.9 Mixture-of-Gaussians results for FDR and Power under Sudarshan et al. (2020) settings

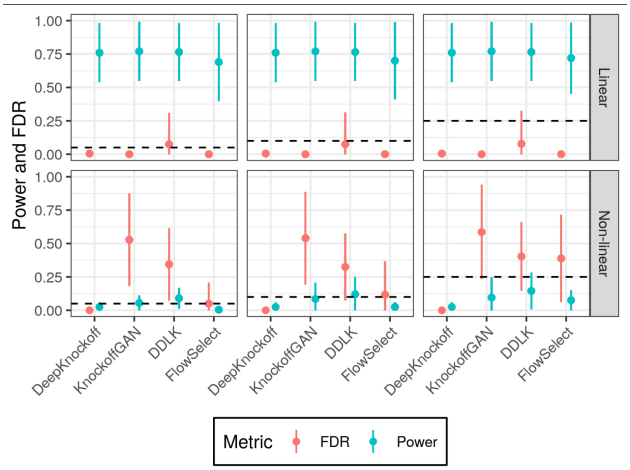


Figure D.6: Mixture-of-gaussians setup with $\rho = (0.6, 0.4, 0.2)$ and $N = 2000$ to match the settings in Sudarshan et al., 2020. In the linear response setting, which matches the data-generating process of Sudarshan et al., 2020, all competing knockoff-based methods (i.e., DDLK, KnockoffGAN, and DeepKnockoff) as well as FlowSelect control the FDR at 5%, 10% and 25% levels and achieve a power of about 0.75. In the non-linear response setting, none of the methods control FDR, except for DeepKnockoffs which had nearly zero power. The good performance in the linear setting can be explained by the LASSO feature statistic shrinking most null features to zero since they have relatively low correlation. Since FDR control should hold for any response setting, these findings suggest that none of the methods do well in modeling the underlying distribution with $N = 2000$ observations.

D.10 Learned normalizing flow mapping on mixture-of-Gaussians and scRNA-seq datasets

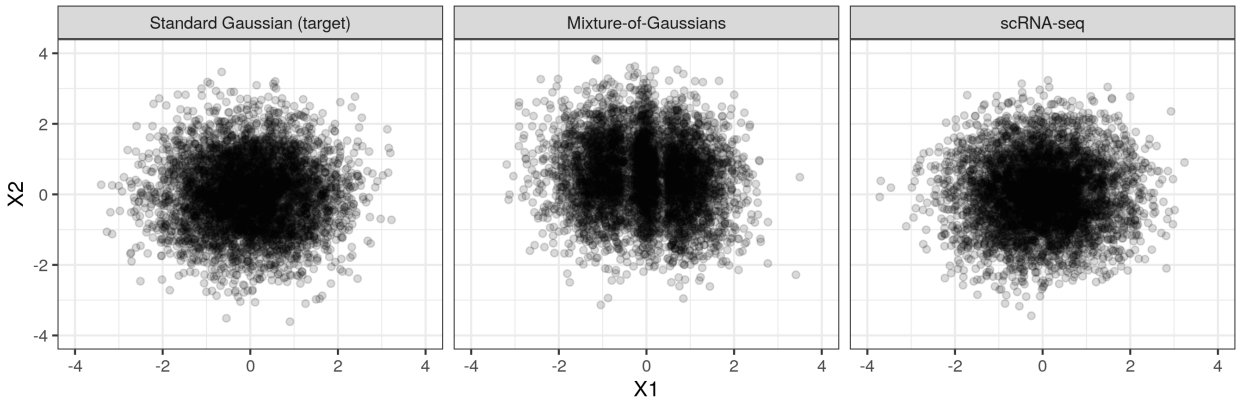


Figure D.7: Plot of features mapped to flow space by the learned normalizing flow within FLOWSELECT with $j = 1$ on the x-axis and $j = 2$ on the y-axis. Mapped features are shown for the mixture-of-Gaussians and scRNA-seq datasets, and they are compared to samples from a true standard Gaussian distribution.

BIBLIOGRAPHY

Agarwal, D., Wang, J., and Zhang, N. (2020). Data denoising and post-denoising corrections in single cell RNA sequencing. *Statistical Science*, 35(1):112–128.

Al-Rawahi, N. and Tryggvason, G. (2002). Numerical simulation of dendritic solidification with convection: Two-dimensional geometry. *Journal of Computational Physics*, 180(2):471–496.

Ambrogioni, L., Güçlü, U., Berezutskaya, J., Borne, E., Güçlütürk, Y., Hinne, M., Maris, E., and Gerven, M. (2019). Forward Amortized Inference for Likelihood-Free Variational Marginalization. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, pages 777–786. PMLR.

Andrieu, C., Doucet, A., and Holenstein, R. (2010). Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342.

Argo (2020). Argo float data and metadata from Global Data Assembly Centre (Argo GDAC).

Baker, N., Alexander, F., Bremer, T., Hagberg, A., Kevrekidis, Y., Najm, H., Parashar, M., Patra, A., Sethian, J., Wild, S., Willcox, K., and Lee, S. (2019). Workshop Report on Basic Research Needs for Scientific Machine Learning: Core Technologies for Artificial Intelligence. Technical report, USDOE Office of Science (SC), Washington, D.C. (United States).

Bates, S., Candès, E., Janson, L., and Wang, W. (2020). Metropolized knockoff sampling. *Journal of the American Statistical Association*, pages 1–15.

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B*, 57(1):289–300.

Benjamini, Y. and Yekutieli, D. (2001). The control of the false discovery rate in multiple testing under dependency. *The Annals of Statistics*, 29(4):1165–1188.

Beucler, T., Pritchard, M., Rasp, S., Ott, J., Baldi, P., and Gentine, P. (2021). Enforcing Analytic Constraints in Neural-Networks Emulating Physical Systems. *Physical Review Letters*, 126(9):098302.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

Bogachev, V. I., Kolesnikov, A. V., and Medvedev, K. V. (2005). Triangular transformations of measures. *Sbornik: Mathematics*, 196(3):309.

Bolton, T. and Zanna, L. (2019). Applications of Deep Learning to Ocean Data Inference and Subgrid Parameterization. *Journal of Advances in Modeling Earth Systems*, 11(1):376–399.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Burden, A., Burden, R., and Faires, J. (2016). *Numerical Analysis*. CENGAGE Learning, tenth edition.

Candès, E., Fan, Y., Janson, L., and Lv, J. (2018). Panning for gold: 'model-X' knockoffs for high dimensional controlled variable selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80(3):551–577.

Cao, Y., Li, S., Wang, Z., Chang, F., Kong, J., Gai, J., and Zhao, T. (2017). Identification of major quantitative trait loci for seed oil content in soybeans by combining linkage and genome-wide association mapping. *Frontiers in Plant Science*, 8.

Cervone, D. and Pillai, N. S. (2015). Gaussian process regression with location errors. *arXiv:1506.08256 [math, stat]*.

Chamberlain, P. M., Talley, L. D., Mazloff, M. R., Riser, S. C., Speer, K., Gray, A. R., and Schwartzman, A. (2018). Observing the ice-covered Weddell Gyre with profiling floats: position uncertainties and correlation statistics. *Journal of Geophysical Research: Oceans*, 123(11):8383–8410.

Chang, Y.-S., Rosati, A. J., Zhang, S., and Harrison, M. J. (2009). Objective analysis of monthly temperature and salinity for the world ocean in the 21st century: Comparison with World Ocean Atlas and application to assimilation validation. *Journal of Geophysical Research*, 114(C2):C02014.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31.

Chen, S., Merriman, B., Osher, S., and Smereka, P. (1997). A simple level set method for solving stefan problems. *Journal of Computational Physics*, 135(1):8–29.

Chopin, N., Jacob, P. E., and Papaspiliopoulos, O. (2013). SMC^2: An efficient algorithm for sequential analysis of state space models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(3):397–426.

Cressie, N. A. C. (1993). *Statistics for Spatial Data*. Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics. J. Wiley, New York, rev. ed. edition.

Dai, R. and Barber, R. (2016). The knockoff filter for FDR control in group-sparse and multitask regression. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1851–1859. PMLR.

de Bézenac, E., Rangapuram, S. S., Benidis, K., Bohlke-Schneider, M., Kurle, R., Stella, L., Hasson, H., Gallinari, P., and Januschowski, T. (2020). Normalizing Kalman Filters for Multivariate Time Series Analysis. In *Advances in Neural Information Processing Systems*, volume 33, pages 2995–3007. Curran Associates, Inc.

Dey, A., Schlegel, D. J., Lang, D., Blum, R., Burleigh, K., Fan, X., Findlay, J. R., Finkbeiner, D., Herrera, D., Juneau, S., et al. (2019). Overview of the DESI legacy imaging surveys. *The Astronomical Journal*, 157(5):168.

Duan, J.-C. and Fulop, A. (2015). Density-tempered marginalized Sequential Monte Carlo samplers. *Journal of Business & Economic Statistics*, 33(2):192–202.

Edwards, C. (2022). Neural networks learn to speed up simulations. *Communications of the ACM*, 65(5):27–29.

Evans, L. (2010). *Partial Differential Equations*, volume 19 of *Graduate Studies in Mathematics*. American Mathematical Society, second edition.

Fan, J. and Gijbels, I. (1996). *Local Polynomial Modelling and Its Applications*, volume 66 of *Monographs on Statistics and Applied Probability*. Chapman & Hall, London.

Fan, Y., Demirkaya, E., Li, G., and Lv, J. (2020). RANK: Large-Scale Inference With Graphical Nonlinear Knockoffs. *Journal of the American Statistical Association*, 115(529):362–379.

Fetterer, F., Knowles, K., Meier, W. N., Savoie, M., and Windnagel, A. K. (2017). *Sea Ice Index: Sea Ice Concentration*.

Fraccaro, M., Kamronn, S., Paquet, U., and Winther, O. (2017). A Disentangled Recognition and Nonlinear Dynamics Model for Unsupervised Learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Friedman, J., Hastie, T., and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics (Oxford, England)*, 9(3):432–441.

Garnelo, M., Rosenbaum, D., Maddison, C. J., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D. J., and Eslami, S. M. A. (2018). Conditional Neural Processes. *arXiv:1807.01613 [cs, stat]*.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2015). *Bayesian Data Analysis*. Chapman and Hall/CRC, New York, third edition.

Genomics, x. (2017). Our 1.3 million single cell dataset is ready to download.

Gershman, S. J. and Goodman, N. (2014). Amortized Inference in Probabilistic Reasoning. *Proceedings of the Annual Meeting of the Cognitive Science Society, 36(36)*, page 7.

Gimenez, J. R., Ghorbani, A., and Zou, J. (2019). Knockoffs for the mass: New feature importance statistics with false discovery guarantees. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*.

Godsill, S. J., Doucet, A., and West, M. (2004). Monte Carlo smoothing for nonlinear time series. *Journal of the American Statistical Association*, 99(465):156–168.

Gordon, N. J., Salmond, D. J., and Smith, A. F. M. (1993). Novel approach to Nonlinear/Non-Gaussian Bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140(2):107–113.

Goswami, S., Bora, A., Yu, Y., and Karniadakis, G. E. (2022). Physics-informed deep neural operator networks. *arXiv preprint arXiv:2207.05748*.

Gray, A. R., Johnson, K. S., Bushinsky, S. M., Riser, S. C., Russell, J. L., Talley, L. D., Wanninkhof, R., Williams, N. L., and Sarmiento, J. L. (2018). Autonomous biogeochemical floats detect significant carbon dioxide outgassing in the high-latitude Southern Ocean. *Geophysical Research Letters*, 45(17):9049–9057.

Gray, A. R. and Riser, S. C. (2014). A global analysis of Sverdrup balance using absolute geostrophic velocities from Argo. *Journal of Physical Oceanography*, 44(4):1213–1229.

Guarniero, P., Johansen, A. M., and Lee, A. (2017). The Iterated Auxiliary Particle Filter. *Journal of the American Statistical Association*, 112(520):1636–1647.

Guinness, J. and Fuentes, M. (2016). Isotropic covariance functions on spheres: Some properties and modeling considerations. *Journal of Multivariate Analysis*, 143:143–152.

Gupta, G., Xiao, X., and Bogdan, P. (2021). Multiwavelet-based Operator Learning for Differential Equations. In *Advances in Neural Information Processing Systems*, volume 34.

Hansen, D., Maddix, D. C., Alizadeh, S., Gupta, G., and Mahoney, M. W. (2023). Learning Physical Models that Can Respect Conservation Laws. In *ICLR 2023 Workshop on Physics for Machine Learning*.

Hansen, D., Manzo, B., and Regier, J. (2022a). Normalizing Flows for Knockoff-free Controlled Feature Selection. In *Advances in Neural Information Processing Systems*.

Hansen, D., Mendoza, I., Liu, R., Pang, Z., Zhao, Z., Avestruz, C., and Regier, J. (2022b). Scalable Bayesian Inference for Detection and Deblending in Astronomical Images. *ICML 2022 Workshop on Machine Learning for Astrophysics*.

Hansen, D. and Yarger, D. (2022). A probabilistic model of ocean floats under ice. *arXiv preprint arXiv:2210.00118*.

Hastie, T., Tibshirani, R., and Friedman, J. (2013). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer New York.

Hinton, G., Dayan, P., Frey, B., and Neal, R. (1995). The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161.

Hosoda, S., Suga, T., Shikama, N., and Mizuno, K. (2009). Global surface layer salinity change detected by Argo and its implication for hydrological cycle intensification. *Journal of Oceanography*, 65(4):579–586.

Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. (2018). Neural autoregressive flows. In *International Conference on Machine Learning*.

Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 448–456. PMLR.

Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, volume 31.

Jagtap, A. D., Kharazmi, E., and Karniadakis, G. E. (2020). Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028.

Jekel, C. F., Sterbentz, D. M., Aubry, S., Choi, Y., White, D. A., and Belof, J. L. (2022). Using conservation laws to infer deep learning model accuracy of Richtmyer-meshkov instabilities. *arXiv preprint arXiv:2208.11477*.

Jordon, J., Yoon, J., and van der Schaar, M. (2019). KnockoffGAN: Generating knockoffs for feature selection using generative adversarial networks. In *International Conference on Learning Representations*.

Katzfuss, M. and Guinness, J. (2021). A general framework for Vecchia approximations of Gaussian processes. *Statistical Science*, 36(1):124–141.

Katzfuss, M., Jurek, M., Zilber, D., and Gong, W. (2020). *GPvecchia: Scalable Gaussian-process Approximations*.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., and Teh, Y. W. (2019). Attentive Neural Processes. *arXiv:1901.05761 [cs, stat]*.

King, A. A., Ionides, E. L., Pascual, M., and Bouma, M. J. (2008). Inapparent infections and cholera dynamics. *Nature*, 454(7206):877–880.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational Bayes. In *International Conference on Learning Representations (ICLR)*.

Kingma, D. P. and Welling, M. (2019). An Introduction to Variational Autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392.

Klatt, O., Boebel, O., and Fahrbach, E. (2007). A profiling float's sense of ice. *Journal of Atmospheric and Oceanic Technology*, 24(7):1301–1308.

Kobyzev, I., Prince, S. J. D., and Brubaker, M. A. (2020). Normalizing Flows: An Introduction and Review of Current Methods. *arXiv:1908.09257 [cs, stat]*.

Krishnapriyan, A. S., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W. (2021). Characterizing possible failure modes in physics-informed neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 26548–26560.

Krishnapriyan, A. S., Queiruga, A. F., Erichson, N. B., and Mahoney, M. W. (2022). Learning continuous models for continuous physics. *arXiv preprint arXiv:2202.08494*.

Kuusela, M. and Stein, M. L. (2018). Locally stationary spatio-temporal interpolation of Argo profiling float data. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2220):20180400.

Le, T. A., Kosiorek, A. R., Siddharth, N., Teh, Y. W., and Wood, F. (2020). Revisiting Reweighted Wake-Sleep for Models with Stochastic Control Flow. In *Uncertainty in Artificial Intelligence*, pages 1039–1049. PMLR.

LeVeque, R. J. (1990). *Numerical Methods for Conservation Laws*. Lectures in Mathematics ETH Zürich. Birkhäuser Verlag.

LeVeque, R. J. (2002). *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press.

LeVeque, R. J. (2007). *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-state and Time-Dependent Problems*. SIAM.

Li, C.-Y., Garimella, S. V., and James E. Simpson (2003). FIXED-GRID FRONT-TRACKING ALGORITHM FOR SOLIDIFICATION PROBLEMS, PART I: METHOD AND VALIDATION. *Numerical Heat Transfer, Part B: Fundamentals*, 43(2):117–141.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. (2020). Neural Operator: Graph Kernel Network for Partial Differential Equations. *arXiv preprint arXiv:2003.03485*.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. (2021a). Fourier Neural Operator for Parametric Partial Differential Equations. In *International Conference on Learning Representations*, arXiv Preprint arXiv:2010.08895.

Li, Z., Zheng, H., Kovachki, N. B., Jin, D., Chen, H., Liu, B., Azizzadenesheli, K., and Anandkumar, A. (2021b). Physics-informed neural operator for learning partial differential equations. *arXiv preprint arXiv:2111.03794*.

Lin, M., Chen, R., and Liu, J. S. (2013). Lookahead Strategies for Sequential Monte Carlo. *Statistical Science*, 28(1):69–94.

Lipnikov, K., Manzini, G., Moulton, J. D., and Shashkov, M. (2016). The mimetic finite difference method for elliptic and parabolic problems with a staggered discretization of diffusion coefficient. *Journal of Computational Physics*, 305:111–126.

Liu, M., Katsevich, E., Janson, L., and Ramdas, A. (2020). Fast and Powerful Conditional Randomization Testing via Distillation. *arXiv:2006.03980*.

Liu, R., McAuliffe, J. D., and Regier, J. (2021). Variational Inference for Deblending Crowded Starfields. *arXiv:2102.02409 [astro-ph, stat]*.

Liu, Y., Wang, D., He, F., Wang, J., Joshi, T., and Xu, D. (2019). Phenotype prediction and genome-wide association study using deep convolutional neural network of soybean. *Frontiers in Genetics*, 10.

Liu, Y. and Zheng, C. (2018). Auto-Encoding Knockoff Generator for FDR Controlled Variable Selection. *arXiv:1809.10765*.

Lopes, H. F. and Tsay, R. S. (2011). Particle filters and Bayesian inference in financial econometrics. *Journal of Forecasting*, 30(1):168–209.

Louizos, C., Shi, X., Schutte, K., and Welling, M. (2019). The Functional Neural Process. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. (2021). Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. 3:218–229.

Lupton, R. H., Ivezic, Z., et al. (2005). SDSS image processing II: The photo pipelines. Technical report, Princeton University.

Lyman, J. M. and Johnson, G. C. (2014). Estimating global ocean heat content changes in the upper 1800 m since 1950 and the influence of climatology choice. *Journal of Climate*, 27(5):1945–1957.

Maddix, D. C., Sampaio, L., and Gerritsen, M. (2018a). Numerical artifacts in the discontinuous Generalized Porous Medium Equation: How to avoid spurious temporal oscillations. *Journal of Computational Physics*, 368:277–298.

Maddix, D. C., Sampaio, L., and Gerritsen, M. (2018b). Numerical artifacts in the Generalized Porous Medium Equation: Why harmonic averaging itself is not to blame. *Journal of Computational Physics*, 361:280–298.

Mao, Z., Jagtap, A. D., and Karniadakis, G. E. (2020). Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789.

Meng, C., Song, Y., Song, J., and Ermon, S. (2020). Gaussianization flows. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*.

Müller, E. H. (2022). Exact conservation laws for neural network integrators of dynamical systems. *arXiv preprint arxiv:2209.11661*.

Naesseth, C. A., Lindsten, F., and Schön, T. B. (2019). Elements of Sequential Monte Carlo. *Foundations and Trends® in Machine Learning*, 12(3):307–392.

Négiar, G., Mahoney, M. W., and Krishnapriyan, A. S. (2022). Learning differentiable solvers for systems with hard constraints. *arXiv preprint arXiv:0902.0885*.

Onken, D. and Ruthotto, L. (2020). Discretize-optimize vs. Optimize-discretize for time-series regression and continuous normalizing flows. *arXiv preprint arXiv:2005.13420*.

Osher, S. and Sethian, J. A. (1988). Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49.

Ott, K., Katiyar, P., Hennig, P., and Tiemann, M. (2021). ResNet after all: Neural ODEs and their numerical solution. In *International Conference on Learning Representations*.

Owen, A. (2013). Importance Sampling. In *Monte Carlo Theory, Methods and Examples*.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64.

Papamakarios, G., Pavlakou, T., and Murray, I. (2017). Masked Autoregressive Flow for Density Estimation.

Petersen, K. B., Pedersen, M. S., et al. (2008). The matrix cookbook. *Technical University of Denmark*, 7(15):510.

Pitt, M. K. (2002). Smooth Particle Filters for Likelihood Evaluation and Maximisation. The Warwick Economics Research Paper Series (TWERPS), University of Warwick, Department of Economics.

Pitt, M. K., Silva, R. d. S., Giordani, P., and Kohn, R. (2012). On some properties of Markov chain Monte Carlo simulation methods based on the particle filter. *Journal of Econometrics*, 171(2):134–151.

Raissi, M., Perdikaris, P., and Karniadakis, G. (2019). 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.

Rasmussen, C. and Williams, C. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Reeve, K., Boebel, O., Kanzow, T., Strass, V., Rohardt, G., and Fahrbach, E. (2016). A gridded data set of upper-ocean hydrographic properties in the Weddell Gyre obtained by objective mapping of Argo float measurements. *Earth System Science Data*, 8:15–40.

Reeve, K. A., Boebel, O., Strass, V., Kanzow, T., and Gerdes, R. (2019). Horizontal circulation and volume transports in the Weddell Gyre derived from Argo float data. *Progress in Oceanography*, 175:263–283.

Regier, J., Miller, A. C., Schlegel, D., Adams, R. P., McAuliffe, J. D., and Prabhat (2019). Approximate inference for constructing astronomical catalogs from images. *The Annals of Applied Statistics*, 13(3):1884–1926.

Richter-Powell, J., Lipman, Y., and Chen, R. T. Q. (2022). Neural conservation laws: A divergence-free perspective. *arXiv preprint arXiv:2210.01741*.

Riser, S. C., Swift, D., and Drucker, R. (2018). Profiling floats in SOCCOM: technical capabilities for studying the Southern Ocean. *Journal of Geophysical Research: Oceans*, 123(6):4055–4073.

Roemmich, D. and Gilson, J. (2009). The 2004–2008 mean and annual cycle of temperature, salinity, and steric height in the global ocean from the Argo Program. *Progress in Oceanography*, 82(2):81–100.

Romano, Y., Sesia, M., and Emmanuel Candès (2020). Deep knockoffs. *Journal of the American Statistical Association*, 115(532):1861–1872.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-Resolution Image Synthesis With Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695.

Rowe, B. T., Jarvis, M., Mandelbaum, R., Bernstein, G. M., Bosch, J., Simet, M., Meyers, J. E., Kacprzak, T., Nakajima, R., Zuntz, J., et al. (2015). GALSIM: The modular galaxy image simulation toolkit. *Astronomy and Computing*, 10:121–150.

Rubin, D. B., editor (1987). *Multiple Imputation for Nonresponse in Surveys*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., Hoboken, NJ, USA.

Saad, N., Gupta, G., Alizadeh, S., and Maddix, D. (2022). Guiding continuous operator learning through Physics-based boundary constraints. *arXiv preprint arXiv:2212.07477*.

Salimans, T. and Kingma, D. P. (2016). Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Sanchez, J., Mendoza, I., Kirkby, D. P., and Burchat, P. R. (2021). Effects of overlapping sources on cosmic shear estimation: Statistical sensitivity and pixel-noise bias. *Journal of Cosmology and Astroparticle Physics*, 2021(07):043.

Sargsyan, S. (2016). Dimensionality hyper-reduction and machine learning for dynamical systems with varying parameters. In *Ph.D. Thesis, University of Washington*.

Sethian, J. A. and Strain, J. (1992). Crystal growth and dendritic solidification. *Journal of Computational Physics*, 98(2):231–253.

Shephard, N. (1996). Statistical aspects of ARCH and stochastic volatility. In *Time Series Models*. Chapman and Hall/CRC.

Smith, A. F. M. and Roberts, G. O. (1993). Bayesian Computation Via the Gibbs Sampler and Related Markov Chain Monte Carlo Methods. *Journal of the Royal Statistical Society: Series B (Methodological)*, 55(1):3–23.

Sonah, H., ODonoughue, L., Cober, E., Rajcan, I., and Belzile, F. (2014). Identification of loci governing eight agronomic traits using a GBS-GWAS approach and validation by QTL mapping in soya bean. *Plant Biotechnology Journal*, 13(2):211–221.

Song, Q., Yan, L., Quigley, C., Jordan, B. D., Fickus, E., Schroeder, S., Song, B.-H., Charles An, Y.-Q., Hyten, D., Nelson, R., Rainey, K., Beavis, W. D., Specht, J., Diers, B., and Cregan, P. (2017). Genetic characterization of the soybean nested association mapping population. *The Plant Genome*, 10(2).

Sturm, P. O. and Wexler, A. S. (2022). Conservation laws in a neural network architecture: Enforcing the atom balance of a Julia-based photochemical model (v0.2.0). *Geoscientific Model Development*, 15:3417–3431.

Subramanian, S., Kirby, R. M., Mahoney, M. W., and Gholami, A. (2022). Adaptive self-supervision algorithms for physics-informed neural networks. *arXiv preprint arXiv:2207.04084*.

Sudarshan, M., Tansey, W., and Ranganath, R. (2020). Deep direct likelihood knockoffs. In *Advances in Neural Information Processing Systems*, volume 33.

Talley, L. D., Pickard, G. L., Emery, W. J., and Swift, J. H. (2011). Chapter 7 - Dynamical processes for descriptive ocean circulation. In *Descriptive Physical Oceanography (Sixth Edition)*, pages 187–221. Academic Press, Boston.

Tansey, W., Veitch, V., Zhang, H., Rabadan, R., and Blei, D. M. (2021). The holdout randomization test for feature selection in black box models. *Journal of Computational and Graphical Statistics*.

Tezaur, I. K., Fike, J. A., Carlberg, K. T., Barone, M. F., Maddix, D., Mussoni, E. E., and Balajewicz, M. (2017). Advanced fluid reduced order models for compressible flow. *Sandia National Laboratories Report, Sand No. 2017-10335*.

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society: Series B*, 58:267–288.

Tran, D., Vafa, K., Agrawal, K. K., Dinh, L., and Poole, B. (2019). Discrete Flows: Invertible Generative Models of Discrete Data. *arXiv:1905.10347 [cs, stat]*.

Turner, S. (2018). Qqman: An R package for visualizing GWAS results using Q-Q and Manhattan plots. *The Journal of Open Source Software*.

Vahdat, A. and Kautz, J. (2020). NVAE: A Deep Hierarchical Variational Autoencoder. *Advances in Neural Information Processing Systems*, 33:19667–19679.

van der Meer, J., Kraaijevanger, J., Möller, M., and Jansen, J. (2016). Temporal oscillations in the simulation of foam enhanced oil recovery. *ECMOR XV - 15th European Conference on the Mathematics of Oil Recovery*, pages 1–20.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Vázquez, J. (2007). *The Porous Medium Equation: Mathematical Theory*. The Clarendon Press, Oxford University Press, Oxford.

Verdy, A. and Mazloff, M. R. (2017). A data assimilating model for estimating Southern Ocean biogeochemistry. *Journal of Geophysical Research: Oceans*, 122(9):6968–6988.

Vernet, M., Geibert, W., Hoppema, M., Brown, P. J., Haas, C., Hellmer, H. H., Jokat, W., Jullion, L., Mazloff, M., Bakker, D. C. E., Brearley, J. A., Croot, P., Hattermann, T., Hauck, J., Hillenbrand, C.-D., Hoppe, C. J. M., Huhn, O., Koch, B. P., and Lechtenfeld, O. J. ... Verdy, A. (2019). The Weddell Gyre, Southern Ocean: present knowledge and future challenges. *Reviews of Geophysics*, 57(3):623–708.

Wang, S., Yu, X., and Perdikaris, P. (2022). When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449(110768).

Wong, A. P. S., Wijffels, S. E., Riser, S. C., Pouliquen, S., Hosoda, S., Roemmich, D., Gilson, J., Johnson, G. C., Martini, K., Murphy, D. J., Scanderbeg, M., Bhaskar, T. V. S. U., Buck, J. J. H., Merceur, F., Carval, T., Maze, G., Cabanes, C., André, X., and Poffa, N. ... Park, H.-M. (2020). Argo data 1999–2019: Two million temperature-salinity profiles and subsurface velocity observations from a global array of profiling floats. *Frontiers in Marine Science*, 7.

Xavier, A., Muir, W., and Rainey, K. (2016). Impact of imputation methods on the amount of genetic variation captured by a single-nucleotide polymorphism panel in soybeans. *BMC Bioinformatics*, 17(1).

Zanna, L. and Bolton, T. (2020). Data-Driven Equation Discovery of Ocean Mesoscale Closures. *Geophysical Research Letters*, 47(17).

Zhang, C., Bütepage, J., Kjellström, H., and Mandt, S. (2018). Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026.