Meta-learning Control Variates: Variance Reduction with Limited Data

Zhuo Sun[1,3] **Chris J. Oates**[2,3] **François-Xavier Briol**[1,3]

[1]Department of Statistical Science, University College London, London, UK
[2]School of Mathematics, Statistics & Physics, Newcastle University, UK
[3]The Alan Turing Institute, London, UK

Abstract

Control variates can be a powerful tool to reduce the variance of Monte Carlo estimators, but constructing effective control variates can be challenging when the number of samples is small. In this paper, we show that when a large number of related integrals need to be computed, it is possible to leverage the similarity between these integration tasks to improve performance even when the number of samples per task is very small. Our approach, called *meta learning CVs* (Meta-CVs), can be used for up to hundreds or thousands of tasks. Our empirical assessment indicates that Meta-CVs can lead to significant variance reduction in such settings, and our theoretical analysis establishes general conditions under which Meta-CVs can be successfully trained.

1 INTRODUCTION

Estimating integrals is a significant computational challenge encountered when performing uncertainty quantification in statistics and machine learning. In a Bayesian context, integrals arise in the estimation of posterior moments, marginalisation of hyperparameters, and the computation of predictive distributions. In frequentist statistics, it is often necessary to integrate out latent variables. In machine learning, integrals arise in gradient-based variational inference or reinforcement learning algorithms. These problems can usually be formulated as the task of computing

$$\mathbb{E}_\pi[f] := \int_{\mathcal{X}} f(x)\pi(x)\mathrm{d}x, \tag{1}$$

where $\mathcal{X} \subseteq \mathbb{R}^d$ is the domain of integration, $f : \mathcal{X} \to \mathbb{R}$ is an integrand, and $\pi : \mathcal{X} \to [0, \infty)$ is a probability density function. (For convenience, we will use π to denote both a density and the distribution associated to it.)

It is rare that such integrals can be exactly computed. This has led to the development of a range of approximation techniques, including both deterministic and randomised cubature rules. The focus of this paper is on *Monte Carlo* (MC) methods and their correlated extensions such as *Markov chain Monte Carlo* (MCMC), which make use of a finite collection of evaluations $f(x_i)$ at locations $\{x_i\}_{i=1}^N$ that are randomly sampled; see Green et al. [2015].

Since the variance of standard MC estimators can be large, control variates (CVs) are often also employed. The idea behind CVs is to approximate f using a suitable family of functions with known integral. Once an approximation g is identified, the CV estimator consist of the sum of $\mathbb{E}_\pi[g]$ and a MC (or MCMC) estimator for $\mathbb{E}_\pi[f - g]$. An effective control variate is one for which the difference $f - g$ has smaller MC variance than f (or *asymptotic* variance, in the case of MCMC). CVs have proved successful in a range of challenging tasks in statistical physics [Assaraf and Caffarel, 1999], Bayesian statistics [Dellaportas and Kontoyiannis, 2012, Mira et al., 2013, Oates et al., 2017, South et al., 2022c], gradient estimation in variational inference [Grathwohl et al., 2018, Shi et al., 2022] and MCMC [Baker et al., 2019], reinforcement learning [Liu et al., 2018, 2019], and computer graphics [Müller et al., 2020].

Unfortunately, construction of an effective CV usually requires a large number of samples N. This limits their usefulness in settings when either sampling from π or evaluating f is expensive, or when the computational budget is otherwise limited. High-dimensional settings also pose a challenge, since such functions are more difficult to approximate due to the curse of dimensionality. In the latter case, sparsity can be exploited for integrands with low *effective dimension* [South et al., 2022b, Leluc et al., 2021], but many integrands do not admit convenient structure that can be easily exploited.

This paper proposes a radically different solution, which *borrows strength* from multiple related integration tasks to aid in the construction of effective CVs. Our approach requires a setting where T integration tasks of the form in

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

(1) need to be tackled, and where the integrands $f_1, \ldots, f_T$ and densities $\pi_1, \ldots, \pi_T$ are different, but related. Related integration tasks arise in a broad range of settings, including multifidelity modelling [Peherstorfer et al., 2018, Li et al., 2023], sensitivity analysis [Demange-Chryst et al., 2022], policy gradient methods [Liu et al., 2018], and thermodynamic integration [Oates et al., 2016]. Further examples are considered in Section 5, including marginalisation of hyper-parameters in Bayesian inference (see the hierarchical Gaussian process example) and the computation of predictive distributions (see the Lotka–Volterra example). In all cases the integrands and densities are closely related, and sharing information across tasks can be expected to deliver a substantial performance improvement.

To date, the only CV method able to exploit related integration tasks is the *vector-valued CVs* of Sun et al. [2021a]. This algorithm learns the relationship between integrands through a multi-task learning approach in a vector-valued reproducing kernel Hilbert space. It has shown potential, but suffers from a prohibitive computational cost of $\mathcal{O}(T^6)$ and a significant memory cost of $\mathcal{O}(T^2)$. The largest experiment in Sun et al. [2021a], which focused on computation of the model evidence for a dynamical system, was for $T = 4$. This lack of scalability in T is a significant limitation; in many of the motivating examples mentioned above, it can be desirable to share information across hundreds or thousands of tasks. A key question is therefore: *"How can we construct CVs at scale, sharing information across a large number of tasks?"*

Our answer to this question is an algorithm we call *Meta-learning CVs* (Meta-CVs). As the name indicates, Meta-CVs are built on the meta-learning framework [Finn et al., 2017, 2018]. The benefits of this approach are three-fold: (i) the computational cost grows as $\mathcal{O}(T)$, making Meta-CVs feasible for large T, (ii) the effective number of parameters for a given task is constant in T, limiting significantly the memory cost, and (iii) the construction of the Meta-CV occurs offline, and a new CV can be computed at minimal computational cost whenever a new task arises. Before introducing Meta-CVs in Section 4, we first recall background on CV methods in Section 2 and highlighted relevant techniques from related fields in Section 3.

2 BACKGROUND

This section contains background information on general techniques used to construct CVs, which will be adapted to Meta-CVs in Section 4.

Control Variate Methods In the remainder, we will assume f is in $\mathcal{L}^2(\pi) = \{f : \mathcal{X} \to \mathbb{R} \text{ s.t. } \mathbb{E}_\pi[f^2] < \infty\}$, the space of π-square-integrable functions on $\mathcal{X}$. This assumption is necessary to ensure the variance of f, denoted $\mathbb{V}_\pi[f] := \mathbb{E}_\pi[f^2] - (\mathbb{E}_\pi[f])^2$, exists. The MC estimator of $\mathbb{E}_\pi[f]$ is $\hat{\mathbb{E}}_\pi^{\text{MC}}[f] = \frac{1}{N}\sum_{i=1}^N f(x_i)$, where $\{x_i\}_{i=1}^N$ are independent and identically distributed (IID) samples from π. Under the $\mathcal{L}^2(\pi)$ assumption, this estimator satisfies a central limit theorem:

$$\sqrt{N}\left(\hat{\mathbb{E}}_\pi^{\text{MC}}[f] - \mathbb{E}_\pi[f]\right) \to \mathcal{N}\left(0, \mathbb{V}_\pi[f]\right)$$

This result justifies the common use of $\mathbb{V}_\pi[f]$ as a proxy for the accuracy of the MC estimator; analogous results hold for MCMC [Dellaportas and Kontoyiannis, 2012, Belomestny et al., 2020, 2021, Alexopoulos et al., 2023] and (randomised) quasi-Monte Carlo [Hickernell et al., 2005], where the *asymptotic variance* and the *Hardy–Krause variation* serve as analogues of $\mathbb{V}_\pi[f]$. To limit scope, we focus on MC in the sequel.

A potentially powerful strategy to improve MC estimators is to identify a function $g \in \mathcal{L}^2(\pi)$ for which $\mathbb{V}_\pi[f - g]$ is small, and for which the expectation $\mathbb{E}_\pi[g]$ can be exactly computed. From a practical perspective, the identification of a suitable g can be performed using a subset $\{x_i\}_{i=1}^m$ of all available samples (and corresponding integrand evaluations) for the integration task (as described below), and we denote the associated estimator as $\hat{g}_m$. The selected *control variate* $\hat{g}_m$ forms the basis of an improved estimator

$$\begin{aligned}\hat{\mathbb{E}}_\pi^{\text{CV}}[f] &:= \hat{\mathbb{E}}_\pi^{\text{MC}}[f - \hat{g}_m] + \mathbb{E}_\pi[\hat{g}_m] \\ &= \tfrac{1}{N-m}\textstyle\sum_{i=m+1}^N \left(f(x_i) - \hat{g}_m(x_i)\right) + \mathbb{E}_\pi[\hat{g}_m].\end{aligned} \tag{2}$$

Conditional on the training samples $\{x_i\}_{i=1}^m$, a central limit theorem holds for the CV estimator with $\mathbb{V}_\pi[f - \hat{g}_m]$ in place of $\mathbb{V}_\pi[f]$. If $\hat{g}_m$ is an accurate approximation to f, the CV estimator will therefore tend to have a smaller error than the original MC estimator. Refined analysis is possible when m and N jointly go to infinity and $\hat{g}_m$ converges to a limiting CV, but such asymptotic settings are not representative of the limited data scenarios that motivate this work.

In the remainder of this section, we detail various ways to estimate a CV from data.

Zero-Mean Functions A first challenge when selecting a CV is that we require a *known* mean $\mathbb{E}_\pi[g]$. Although *ad-hoc* approaches, such as Taylor expansions of f [Paisley et al., 2012, Wang et al., 2013], can be used when π is relatively simple, this is usually a challenge whenever π is a more complex density, such as can be encountered in a Bayesian inference task. One way forward is through Stein's method, and we will call any CV constructed in this way a *Stein-based CV*; see Anastasiou et al. [2023]. The main components of Stein's method are a function class $\mathcal{U}$, called a *Stein class*, and an operator $\mathcal{S}_\pi$ acting on $\mathcal{U}$, called a *Stein operator*, such that $g := S_\pi[u]$ satisfies $\mathbb{E}_\pi[g] = 0$ for any $u \in \mathcal{U}$. One such operator is the *Langevin–Stein operator*

$$\mathcal{S}_\pi[u](x) := u(x) \cdot \nabla \log \pi(x) + \nabla \cdot u(x)$$

acting on vector fields $u : \mathcal{X} \to \mathbb{R}^d$. From the divergence theorem, this operator satisfies $\mathbb{E}_\pi[\mathcal{S}_\pi[u]] = 0$ under standard tail conditions on the vector field u; see [Oates et al., 2019] for full detail. In addition, evaluation of this operator requires only pointwise evaluation of $\nabla \log \pi(x)$, which is possible even when π involves an unknown normalisation constant, i.e. $\pi = \tilde{\pi}/C$ where $\tilde{\pi}$ is known pointwise and $C > 0$ is an unknown constant. This is a significant advantage in the present setting since many applications, including problems where π is a Bayesian posterior distribution, fall into this category.

The first Stein-based CVs were proposed by Assaraf and Caffarel [1999], in which $\mathcal{U}$ was a finite-dimensional vector space of functions of the form $u = \nabla p$, with p polynomial of fixed degree; see also Mira et al. [2013], Papamarkou et al. [2014], Friel et al. [2014], South et al. [2022b]. For additional flexibility, Oates et al. [2017] proposed to take $\mathcal{U}$ to be a Cartesian product of reproducing kernel Hilbert spaces; see also Oates and Girolami [2016], Oates et al. [2019], Barp et al. [2022], calling this approach *control functionals (CFs)*. Since CFs are based on a non-parametric space of functions, they have the capability to approximate complex integrands, but will also have an effective number of parameters growing with N, leading to high memory and computational costs. It is on these types of CVs that vector-valued CVs are built [Sun et al., 2021a]. Alternatively, one may take $\mathcal{U}$ to be a (parametric) set of neural networks [Wan et al., 2019, Si et al., 2021, Ott et al., 2023], or even a combination of neural networks and the aforementioned spaces [South et al., 2022a, Si et al., 2021]. In this paper, we will focus on CVs constructed with neural networks, which are known as *neural control variates* (Neural-CVs). The rationale for this choice stems from the fact that neural networks are also able to approximate complex functions well, but have a fixed number of parameters, and thus a more manageable memory and computational cost.

Selecting Control Variates Once a family of CVs has been identified, we need to select from this family an effective CV for the integrand f of interest. We will limit ourselves to parametric families, and will aim to identify a good parameter value so that the variance of the CV estimator is minimised. Let $g(x;\gamma) = \gamma_0 + g_{\gamma_{1:p}}$ where $\gamma := \gamma_{0:p} \in \mathbb{R}^{p+1}$ consists of p parameters $\gamma_{1:p}$ determining the zero-mean Stein-based CV $g_{\gamma_{1:p}}$, and an additional parameter γ_0 that will be used to approximate $\mathbb{E}_\pi[f]$. Following the framework of empirical loss minimisation with samples $S = \{x_i, \nabla \log \pi(x_i), f(x_i)\}_{i=1}^m$, the parameter γ can be estimated by minimising

$$J_S(\gamma) := \frac{1}{m} \sum_{i=1}^m \left(f(x_i) - g(x_i;\gamma)\right)^2 .$$

The value of γ_0 minimising this objective is a consistent estimator for $\mathbb{E}_\pi[f]$ in the $m \to \infty$ limit. To avoid over-fitting when m is small, penalised objectives have also been proposed [Wan et al., 2019, Si et al., 2021], but determining the strength of the penalty can represent a very challenging task. To limit scope, we proceed to minimise the un-regularised objective in this work.

Conveniently, in the specific case of Neural-CVs, the backward propagation of gradients with respect to the parameters γ can be done end-to-end via automatic differentiation techniques implemented in modern deep learning frameworks such as PyTorch [Paszke et al., 2019] (which we use in our experiments). Unfortunately, Neural-CVs can require a large number of training samples to learn an accurate approximation to the integrand. As a result, Neural-CVs are not well-suited to solving single integration tasks when the total number of samples N is small. The contribution of this work seeks to leverage information from related integration tasks to directly address this weakness of Neural-CVs.

3 RELATED WORK

The idea of sharing information across integrations tasks has been explored in a range of settings, each building on a specific structure for the relationship between tasks. Unfortunately, as highlighted below, none of the main approaches can be used in the general setting of large T and arbitrary integrands and densities.

Multi-task Learning for Monte Carlo *Multi-output Bayesian quadrature* [Xi et al., 2018, Gessner et al., 2019] and vector-valued CVs [Sun et al., 2021a] are both approaches based on *multi-task learning*. These methods think of $f_1, \ldots, f_T$ as the output of a vector-valued function with a specific structure shared across outputs, and use this structure to construct an estimator. These approaches can perform very well when the algorithm is able to build on the relationship between tasks, but they also suffer from a computational cost between $\mathcal{O}(T^3)$ and $\mathcal{O}(T^6)$ where T is the number of tasks. These methods are therefore not applicable in settings with a large T.

Multilevel and Multi-fidelity Integration *Multilevel Monte Carlo* [Giles, 2015] and related methods are applicable in the specific case where $f_1, \ldots, f_T$ are all approximations of some function f with varying levels of accuracy. Although their cost is usually $\mathcal{O}(T)$, these methods are mostly used for problems with small T and where the computational cost of function evaluation varies per integrand. In particular, they are commonly used with a large N for cheaper but less accurate integrands, and a small N for expensive but accurate integrands. This setting is therefore different from that considered in the present work.

Monte Carlo Methods for Parametric and Conditional Expectations *Parametric expectation* or *conditional expectation* methods [Longstaff and Schwartz, 2001, Krumscheid and Nobile, 2018] consider the task of approximat-

ing $\mathbb{E}_{X\sim\pi}[f(X,y)]$ or $\mathbb{E}_{X\sim\pi(\cdot|Y=y)}[f(X)]$ uniformly over y in some interval. These methods can be applied when T is large, but they usually rely on a specific structure of the problem: smoothness of these quantities as y varies. The methodological development in our work does not rely on smoothness assumptions of this kind.

Importance Sampling Importance sampling is commonly used to tackle an integration task with respect to π when samples from a related distribution π' are available. It works by weighting samples according to the ratio π/π', and is applicable to multiple tasks with a $\mathcal{O}(T)$ cost. However, the challenge is that π' needs to be chosen carefully in order for the estimator to have low variance. The problem of multiple related integrals was considered by [Glynn and Igelhart, 1989, Section 8] and Madras and Piccioni [1999], Demange-Chryst et al. [2022], where the authors seek an importance distribution π' which performs well across a range of tasks. However, identifying such an importance distribution will usually not be possible when T is large.

4 METHODOLOGY

We now set out the details of our proposed Meta-CVs.

Problem Set-up Consider a finite (but possibly large) number, T, of integration tasks

$$\mathbb{E}_{\pi_1}[f_1], \ldots, \mathbb{E}_{\pi_T}[f_T]$$

and denote by $\mathcal{T}_t := \{f_t, \pi_t\}$ the components of the tth task, consisting of a density $\pi_t : \mathcal{X} \to [0,\infty)$ and an integrand $f_t \in \mathcal{L}^2(\pi_t)$. For each task, we assume we have access to data of the form

$$D_t = \{x_i, \nabla \log \pi_t(x_i), f_t(x_i)\}_{i=1}^{N_t},$$

where $N_t \in \mathbb{N}^+$ is relatively small. In addition, we will assume that these tasks are related. Informally, we may suppose that $\mathcal{T}_1, \ldots, \mathcal{T}_T$ are independent realisations from a distribution over tasks arising from an *environment*, but we do not attempt to make this notion formal. This set-up allows us to frame Meta-CVs in the framework of gradient-based meta-learning.

Meta-learning CVs *Gradient-based meta learning* [Finn et al., 2017, 2019, Grant et al., 2018, Yoon et al., 2018, Sun et al., 2021b] was first proposed in the context of *model-agnostic meta-learning* [Finn et al., 2017, 2019]. It was originally designed for "learning-to-learn" in a supervised-learning context, with a specific focus on regression and image classification. The focus of this approach is on the ability to rapidly adapt to new tasks. This is achieved by identifying a meta-model, which acts as an initial model which can be quickly adapted to a new task by taking a few steps of some gradient-based optimiser on its parameters.

In this paper we adapt gradient-based meta learning to the construction of CVs. This leads to a two-step approach: The first step, highlighted in Algorithm 1, consists of learning a *Meta-CV*, a CV that performs "reasonably well for most tasks". The second step, highlighted in Algorithm 2, consists of fine-tuning this Meta-CV to each specific task, using a few additional steps of stochastic optimisation on a task-specific objective function, to obtain a *task-specific* CV.

Before describing these algorithms, for each task $\mathcal{T}_t := \{f_t, \pi_t\}$, we split the samples into two disjoint sets $D_t = S_t \cup Q_t$, so that

$$S_t := \{x_j, \nabla \log \pi_t(x_j), f_t(x_j)\}_{j=1}^{m_t}$$
$$Q_t := \{x_j, \nabla \log \pi_t(x_j), f_t(x_j)\}_{j=m_t+1}^{N_t}.$$

The roles of these two datasets will differ depending on whether the task is used for training the Meta-CV, or for deriving a task-specific CV, and we will return to this point below. For simplicity, all of our experiments will consider $m_t = N_t/2$. Note that these datasets correspond to the concepts of the *support set* and the *query set* in the terminology of gradient-based meta learning [Finn et al., 2017, 2019].

Constructing the Meta-CV The first step in our method is to construct a Meta-CV; this will later be fine-tuned into a task-specific CV. Here we will follow the approach in Section 2 and use a flexibly-parametrised Neural-CV.

To decouple the choice of optimisation method from the general construction of a Meta-CV, L steps of an arbitrary gradient-based optimiser will be denoted $\textsc{Update}_L(\gamma, \nabla_\gamma J(\gamma); \alpha)$, where $\gamma \in \mathbb{R}^{p+1}$ is the initial parameter value, $\nabla_\gamma J(\gamma)$ is the gradient of an objective $J : \mathbb{R}^{p+1} \to \mathbb{R}$, and α represents parameters of the optimisation method. Popular optimisers include gradient descent and Adam [Kingma and Ba, 2015], but more flexible alternatives also exist [Andrychowicz et al., 2016, Grefenstette et al., 2019]. For example, the update corresponding to L-step gradient descent starting at γ_0 consists of $\gamma_j := \gamma_{j-1} - \alpha \nabla J(\gamma_{j-1})$ for $j = 1, \ldots, L$. Using this notation, we can represent an *idealised Meta-CV* as a CV whose parameters satisfy

$$\gamma_{\text{meta}} \in \arg\min_{\gamma\in\mathbb{R}^{p+1}} \mathbb{E}_t \left[J_t \left(\textsc{Update}_L \left(\gamma, \nabla_\gamma J_t(\gamma); \alpha \right) \right) \right], \tag{3}$$

where $\mathbb{E}_t$ denotes expectation with respect to a uniformly sampled task index $t \in \{1, \ldots, T\}$. This objective is challenging to approximate since it requires solving nested optimisation problems. We therefore follow the approach in Finn et al. [2017] and use a gradient-based bi-level optimisation scheme described in Algorithm 1. This requires estimating the gradient of the loss J_t in both the inner and outer level. To prevent over-fitting, we do this using two independent datasets: S_t and Q_t. We will call the output of Algorithm 1, denoted $\hat{\gamma}_{\text{meta}}$, our *meta-parameter*, and $g(\cdot; \hat{\gamma}_{\text{meta}})$ will be called the Meta-CV.

Algorithm 1: Learning a Meta-CV

Input: Training tasks $\mathcal{T}_1, \ldots, \mathcal{T}_T$, initial parameter γ_0, UPDATE rule, # update steps L, optimiser parameters α and $\eta_1, \ldots, \eta_{I_{\text{tr}}}$, mini-batch size B, # meta-iterations I_{tr}.

for $i = 1, \ldots, I_{tr}$ **do**
  Sample $t_1, \ldots, t_B$ uniformly from $\{1, \ldots, T\}$.
  for $t \in \{t_1, \ldots, t_B\}$ **do**
    Initialize $\gamma_0^t \leftarrow \gamma_{i-1}$.
    for $j = 1, \ldots, L$ **do**
      $\gamma_j^t \leftarrow \text{UPDATE}(\gamma_{j-1}^t, \nabla_{\gamma_{j-1}^t} J_{S_t}(\gamma_{j-1}^t); \alpha)$.
  $\gamma_i \leftarrow \text{UPDATE}(\gamma_{i-1}, \frac{1}{B}\sum_{b=1}^{B} \nabla_{\gamma_{i-1}} J_{Q_{t_b}}(\gamma_L^{t_b}); \eta_i)$.

Output: The meta-parameter $\hat{\gamma}_{\text{meta}} := \gamma_{I_{\text{tr}}}$.

Task-Specific CVs Once a meta-parameter $\hat{\gamma}_{\text{meta}}$ has been identified, for each task $\mathcal{T}_t$ we only need to adapt the meta-parameter through a few optimisation steps to obtain a task-specific parameter, $\hat{\gamma}_t$, and hence a corresponding task-specific CV $g(\cdot; \hat{\gamma}_t)$. This can be done by using Algorithm 2, and can be applied either to one of the T tasks in the training set, or indeed to an as yet unseen task. Once such a task-specific CV is identified, we can simply use the CV estimator in Equation (2) to estimate the corresponding integral $\mathbb{E}_{\pi_t}[f_t]$. Note that we once again use two datasets per task, but their role differs from that in Algorithm 1: S_t will be used for selecting the task-specific CV $g(\cdot; \hat{\gamma}_t)$ through Algorithm 2, whilst Q_t will be used to evaluate the CV estimator in (2).

To understand how these task-specific CVs *borrow strength*, we highlight that the task-specific CV are constructed using $\sum_{t=1}^{T} N_t$ samples in total. Thus, when T is large and N_t is small, our task-specific CV may be based on a much larger number of samples compared to any CV constructed solely using data on a single task. The closeness of the relationship between tasks of course determines the value of including these additional data into the training procedure for a CV; this will be experimentally assessed in Section 5.

Algorithm 2: Task-specific CVs from the Meta-CV

Input: Integration task $\mathcal{T}_t$, meta-parameter $\hat{\gamma}_{\text{meta}}$, UPDATE rule, # update steps L, optimiser parameters α.

Initialize $\gamma_0 \leftarrow \hat{\gamma}_{\text{meta}}$.
for $j = 1, \ldots, L$ **do**
  $\gamma_j \leftarrow \text{UPDATE}(\gamma_{j-1}, \nabla J_{S_t}(\gamma_{j-1}); \alpha)$.

Output: Task-specific parameter $\hat{\gamma}_t := \gamma_L$.

Computational Complexity To discuss the complexity of our method, suppose first that the parameter $\hat{\gamma}_{\text{meta}}$ of the Meta-CV has already been computed. The additional computational complexity of training all task-specific Neural-CVs is then $\mathcal{O}(TL)$, where L is the number of optimisation steps used to fine-tune the CV to each specific task. Ordinarily a large number of optimisation steps are required to learn parameters of a neural network, but due to meta-learning we expect the number L of steps required to fine-tune task-specific CVs to be very small (indeed, we take $L = 1$ in most of our experiments). This is because as L grows, the task-specific CV is less and less dependent on the meta-CV; see also Antoniou et al. [2019]. In addition, taking L to be small means fine-tuning a Meta-CV can be orders of magnitude faster compared to training Neural-CVs independently for each task.

Of course, we also need to consider the complexity of training the Meta-CV. This can require a large number L of optimisation steps in general - a point we assess experimentally in Section 5 and theoretically in Section 6 - but this number is broadly comparable to that required to train a Neural CV to a single task. However, the scaling in p, the number of neural network parameters, is at least $\mathcal{O}(p^2)$ [Fallah et al., 2020] for our approach (due to second-order derivatives in Algorithm 1) against $\mathcal{O}(p)$ for Neural-CVs. This will be a challenge for our method when p is large, and we will return to this issue in the conclusion of the paper.

5 EXPERIMENTAL ASSESSMENT

The performance of the proposed Meta-CV method will now be experimentally assessed, using a range of problems of increasing complexity where N_t is small and T is large. For simplicity, we will limit ourselves to the setting where the N_t are equal and where the Adam optimiser is used. The existing methods discussed in Section 3 *cannot be applied to the problems in this section* due to the large value of T and associated prohibitive computational cost, and we therefore only compare to methods which do not borrow strength between tasks: MC, Neural-CVs and CFs. The code to reproduce our results is available at: `https://github.com/jz-fun/Meta_Control_Variates`.

A Synthetic Example Trigonometric functions are common benchmarks for meta-learning [Finn et al., 2017, Grant et al., 2018] and CVs [Oates et al., 2017, 2019]. Consider integrands of the form

$$f_t(x; a_t) = \cos\left(2\pi a_{t,1} + \sum_{i=1}^{d} a_{t,i+1} x_i\right),$$

with parameters $a_t \in \mathbb{R}^{d+1}$, and let π_t be the uniform distribution on $\mathcal{X} = [0, 1]^d$. The integrals $\mathbb{E}_{\pi_t}[f_t]$ can then be explicitly computed and serve as a ground truth for the purpose of assessment. Note that a_t controls the difficulty of the tth integration task. To generate related tasks, we sample the a_t from a distribution ρ consisting of independent uniforms; see Appendix B.1 for full detail.

Figure 1 considers the case $d = 2$, where we train the Meta-

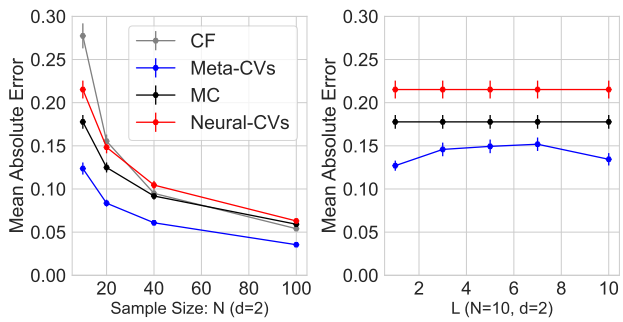

Figure 1: Mean absolute error (with 95% confidence intervals) for $T_{\text{test}} = 1,000$ oscillatory functions (with $N_t = N$ and $m_t = n_t = N/2$ for all t). *Left*: Increasing sample size N_t when $d = 2$ (Meta-CVs with $L = 1$); *Right*: Increasing number of inner gradient steps L of Meta-CVs.

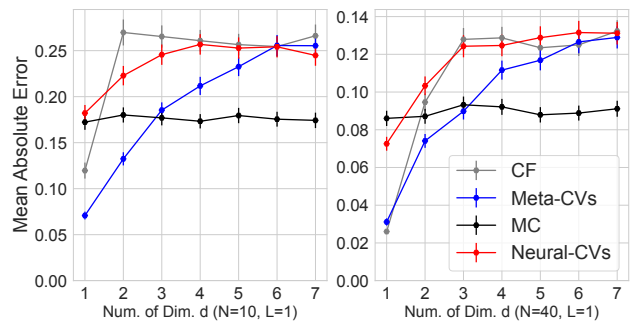

Figure 2: Mean absolute error (with 95% confidence intervals) for $T_{\text{test}} = 1,000$ oscillatory functions for increasing dimension d (with $N_t = N$ and $m_t = n_t = N/2$ for all t).

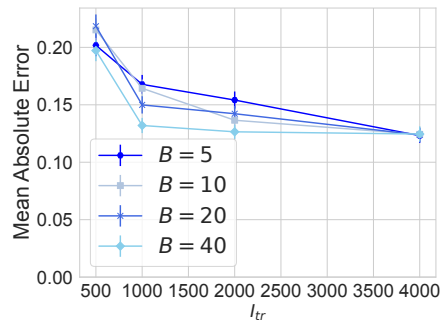

Figure 3: Mean absolute error (with 95% confidence intervals) of Meta-CVs for $T_{\text{test}} = 1,000$ 2-dimensional oscillatory functions for increasing B and I_{tr} (with $L = 1$, $N_t = N$ and $m_t = n_t = N/2 = 5$ for all t).

CVs on $T = 20,000$ tasks in total. To challenge Meta-CVs, all methods were assessed in terms of their performance evaluated on an *additional* $T_{\text{test}} = 1,000$ tasks, not available during training of the Meta-CV. On the left panel, we consider the performance of CVs as we increase the sample size N_t per task. Regardless of the sample size considered, Meta-CVs outperform MC, CFs, and Neural-CVs in terms of mean absolute error over new unseen tasks. This can be explained by the fact that Meta-CVs is the only method which can transfer information across tasks, able to exploit the large training dataset. In this example, Neural-CVs and CF perform even worse than MC when N_t is small, highlighting the challenge of using CVs in these settings. In the right panel of Figure 1, we investigate the effect of the number of gradient-based updates L, which shows the robustness of Meta-CVs to L. We investigate the performance of these methods as d increases in Figure 2. Clearly, all CVs suffer from a curse of dimensionality, but Meta-CVs do improve on the other CVs for $d < 6$. Alleviating this curse of dimensionality could be an important direction for future research in CVs. We also investigate the effect of B and I_{tr} in Figure 3 by comparing the resulting performance of Meta-CVs on 1000 2-dimensional unseen test tasks. It is found empirically that a larger value of B helps to achieve the optimal performance faster; and a large value of I_{tr} results in improvement in performance as expected.

We conclude with a brief discussion of computational cost. The cost of computing independent Neural-CVs on all unseen test tasks is around 2 minutes. In contrast, the offline time for training the Meta-CV is around 7 minutes (with $L = 1$ when $B = 5$, $I_{\text{tr}} = 4000$, $N_t = 10$ and $d = 1$), but the online time taken for deriving task-specific CVs for all the same 1000 unseen test tasks is approximately 6 seconds in total. This demonstrates that our Meta-CV can be rapidly adapted to new tasks.

Uncertainty Quantification for Boundary Value ODEs Our second example considers the computation of expectations of functionals of physical models represented through differential equations. The expectations are taken with respect to expert-specified distributions over parameters of these models, with the aim of performing uncertainty quantification. We consider a boundary-value ODE with unknown forcing closely resembling that of Giles [2015]:

$$\tfrac{\mathrm{d}}{\mathrm{d}s}\left(c(s)\tfrac{\mathrm{d}u}{\mathrm{d}s}\right) = -50x^2, \qquad 0 < s < 1,$$

with boundary $u(0) = u(1) = 0$, $c(s) = 1 + as$. The integrand of interest is $f_t(x) = \int_0^1 u(s, x; a_t)\mathrm{d}s$ where a_t are draws from $\rho = \mathsf{Unif}(0, 1)$, and the integral of interest is $\mathbb{E}_{X\sim\pi_t}[f_t(X)]$ where each $\pi_t = \mathcal{N}(0, 1)$. We use a finite difference approximation of f_t described in Giles [2015]; see Appendix B.2 for detail. This is a relatively simple example, but it is representative of a broader class of challenging problems where improved numerical methods are needed to approximate integrals due to a large cost per integrand evaluation and therefore limited N_t.

The results are presented in Figure 4. We compare the performance of Meta-CVs with MC and Neural-CVs on $T_{\text{test}} = 100$ unseen tasks (grey crosses are mean absolute

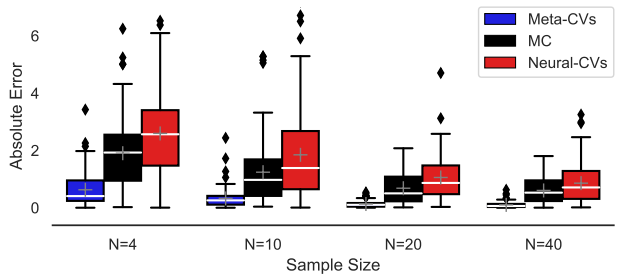

Figure 4: Absolute error for $T_{\text{test}} = 100$ (with $N_t = N$ and $m_t = n_t = N/2$ for all t.) unseen tasks from the boundary value ODE problem.

errors; white horizontal lines are medians). For this example, Meta-CVs outperform Neural-CVs and MC consistently in all cases, highlighting once again the benefits of sharing information across a large number of tasks when N_t is small.

Bayesian Inference for the Lotka–Volterra System Our next example also considers uncertainty quantification for differential equation-based models, but this time in a fully Bayesian framework. In particular, we consider a parametric ODE system, the Lotka–Volterra model [Lotka, 1927], commonly used in ecology and epidemiology, given by

$$\frac{\mathrm{d}u_1}{\mathrm{d}s} = x_1 u_1 - x_2 u_1 u_2, \quad \frac{\mathrm{d}u_2}{\mathrm{d}s} = x_3 u_1 u_2 - x_4 u_2,$$

where $u_1(s)$ and $u_2(s)$ are the numbers of preys and predators at time s, and $u_1(0) = x_5$ and $u_2(0) = x_6$. Suppose we have access to observations of $u = (u_1, u_2)$ at time points $\{s_1, \dots s_q\}$, corrupted with independent log-normal noise with variances x_7 and x_8 respectively. A 'task' here corresponds to computing the posterior expectation of model parameters x for a given dataset; different datasets, which could for example correspond to different animal species, or to different geographical regions, determine the posterior distribution π_t of interest. Bayesian inference on this type of ecological [Bolker, 2008] and epidemiological [Brauer, 2017] models is challenging due to the high cost of MCMC sampling, significantly limiting the number of effectively independent samples N_t. In our experiments, we use the dataset from Hewitt [1921] on snowshoe hares (preys) and Canadian lynxes (predators). We sub-sample the whole dataset to mimic the process of sampling sub-populations and our goal is to learn a Meta-CV which can be quickly adapted to new sub-populations observed in the future; see Appendix B.3 for full detail.

Results are presented in Figure 5. We compare Meta-CV to MCMC (a *No-U-Turn Sampler* (NUTS) implemented in Stan [Carpenter et al., 2017]). As previously discussed, CVs perform poorly in high-dimensions when N_t is small. This is exactly what we observe: Neural-CVs performs between $5-$ and $12-$times worse than MCMC and is therefore not included in the figure. In contrast, Meta-CVs is able to

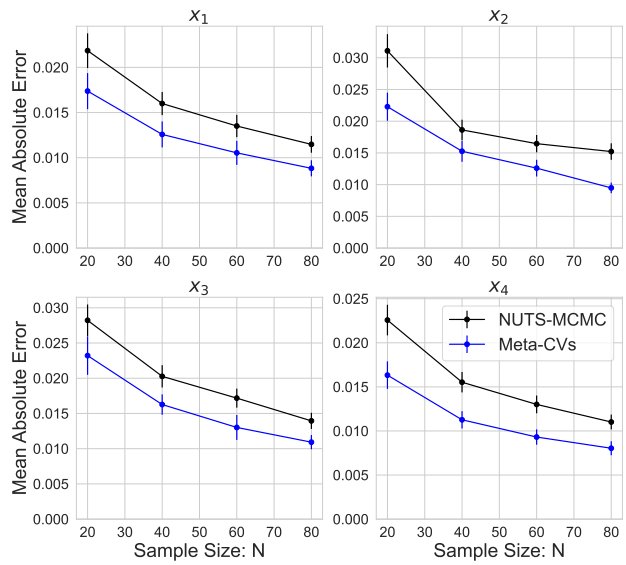

Figure 5: Mean absolute errors (with 95% confidence intervals) over 40 sub-populations for varying N_t. Here, $N_t = N$ and $m_t = n_t = N/2$ for all t.

achieve a lower mean absolute error than MCMC for the values of N_t considered, demonstrating the clear advantage of sharing information across tasks for higher-dimensional problems.

Marginalization in Hierarchical Gaussian Processes Marginalisation of hyper-parameters is a common problem in Bayesian statistics. We consider a canonical example for hierarchical Gaussian process regression [Rasmussen and Williams, 2006], which was tackled with CVs by Oates et al. [2017]. The problem consists of recovering an unknown function ν describing a 7 degrees-of-freedom Sarcos anthropomorphic robot arm, from a 21-dimensional input space, based on a subset of the dataset described in Rasmussen and Williams [2006]. Data consist of observations $y_i = \nu(z_i) + \epsilon_i$ at inputs z_i for $i = 1, \dots, q$, where ϵ_i are IID zero-mean Gaussian random variables with known standard deviation $\sigma > 0$. A zero-mean Gaussian process prior is placed on ν, with covariance function $k_x(z, z') = x_1 \exp(-\|z - z'\|_2^2 / 2x_2^2)$, as well as priors on the hyper-parameters $x = (x_1, x_2)$. Given observations $y_{1:q} = (y_1, \dots, y_q)^\top$, we consider the 'task' of predicting the response $\nu(x^*)$ at an unseen state z^*, marginalising out any posterior uncertainty associated with the hyper-parameters x of the Gaussian process model. This can be achieved through the Bayesian posterior predictive mean $\mathbb{E}[Y^*|y_{1:q}] = \mathbb{E}_{X \sim \pi(\cdot|y_{1:q})}[\mathbb{E}[Y^*|y_{1:q}, X]]$. This is an integral of

$$\begin{aligned} f(x) &= \mathbb{E}[Y^*|y_{1:q}, x] \\ &= K_{z^*,q}(x)(K_{q,q}(x) + \sigma^2 I_q)^{-1} y_{1:q} \end{aligned}$$

against the posterior on hyperparameters $\pi(x|y_{1:q})$, where $(K_{q,q}(x))_{i,j} = k_x(z_i, z_j)$ and $(K_{z^*,q}(x))_j = k_x(z^*, z_j)$ for $i, j \in \{1, \ldots, q\}$. The integrand is therefore an expensive function: $\mathcal{O}(q^3)$ operations are needed per evaluation, which will be significant when q is beyond a few hundred. However, it is also common to want to compute this quantity for several new inputs $z_1^*, \ldots, z_T^*$, leading to closely related integrands $f_1, \ldots, f_T$ whose relationship could potentially be leveraged.

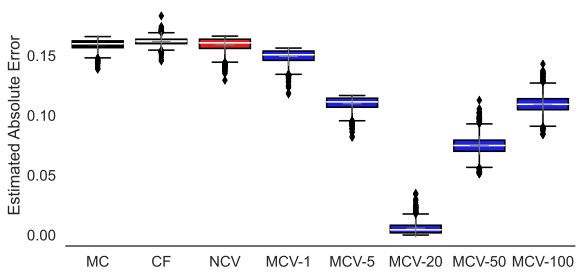

Figure 6: Effect of L: Estimated absolute errors over $T_{\text{test}} = 1,000$ unseen states of the Sarcos anthropomorphic robot arm (*CF*: Control functionals; *NCV*: Neural-CVs; *MCV-L*: Meta-CVs with L inner steps).

Our dataset is divided into two parts. The first part is used to obtain the posterior on Gaussian process hyperparameters (which is approximated through variational inference) and consists of $q = 1,000$ data points. The second part includes $4,449$ data points, half of which are used to construct the Meta-CV and the other half is used to define a held-out test set of tasks for assessment. See Appendix B.4 for full experimental detail.

The results are presented in Figure 6. We compare the performance of Meta-CVs with MC, CFs and Neural-CVs on $T_{\text{test}} = 1,000$ unseen tasks, where $N_t = 4$ for each task. Although we do not have access to the exact value of these integrals, the value y_t^* is an unbiased estimator, and this enables integration error to be unbiasedly estimated. We find that Meta-CVs are once again able to outperform competitors, but interestingly the performance improves significantly when the number of inner gradient steps $L > 1$. Meanwhile, it is found empirically that there is a trade-off between remaining close to the Meta-CV, and specialising each CV to a specific task; see [Antoniou et al., 2019] for a detailed discussion. In general, we would recommend to split the training set into a training set and a validation set, and choose the optimal value of L (and other hyperparameters) on the validation set. This is known as a "meta-validation" process in the meta-learning literature.

6 THEORETICAL ANALYSIS

The empirical results of the previous section demonstrate the advantage of leveraging the relationship between a large number of integration tasks. This section will focus on obtaining theoretical insight to guide the implementation of gradient-based optimisation within Meta-CVs.

Our analysis focuses on strategies for training of Meta-CVs. Recall that the (global) objective for learning a Meta-CV is

$$\arg\min_{\gamma \in \mathbb{R}^{p+1}} \; \mathbb{E}_t \left[\mathcal{J}_t(\gamma) \right], \qquad (4)$$
$$\mathcal{J}_t(\gamma) := J_{Q_t} \left(\textsc{Update}_L \left(\gamma, \nabla_\gamma J_{S_t}(\gamma) ; \alpha \right) \right),$$

where in what follows $\textsc{Update}_L$ is gradient descent with L steps and inner-step size α. To proceed, we make the following assumptions:

Assumption 1. *For each t and $x \in D_t$, $\gamma \mapsto g(x;\gamma)$ and $\gamma \mapsto \nabla_\gamma g(x;\gamma)$ are bounded and Lipschitz.*

Assumption 2. *For each t and $x \in D_t$, $\gamma \mapsto \nabla_\gamma g(x;\gamma) \nabla_\gamma g(x;\gamma)^\top - \nabla_\gamma^2 g(x;\gamma)$ is bounded and Lipschitz.*

Assumption 1 can in principle be satisfied by the Stein-based CVs introduced in Section 2, since it concerns the behaviour of $g(x;\gamma)$ and $\nabla_\gamma g(x;\gamma)$ as γ, rather than x, is varied (recall that, as a function of x, Stein-based CVs are usually unbounded). For Assumption 2, we note that $\nabla_\gamma g(x;\gamma) \nabla_\gamma g(x;\gamma)^\top$ is a popular low-rank approximation to the Hessian $\nabla_\gamma^2 g(x;\gamma)$, so Assumption 2 explicitly requires this low-rank approximation to be reasonably good.

The following theorem, which builds on the work of Ji et al. [2022], establishes conditions under which Algorithm 1 can find an ϵ-first order stationary point of the meta-learning objective function (4), for any $\epsilon > 0$.

Theorem 1. *Let $\hat{\gamma}_{\text{meta}}$ be the output of Algorithm 1 with gradient descent steps, using the meta-step-sizes $\eta_1, \ldots, \eta_{I_{tr}}$, the inner-step size α and batch size B proposed in Theorem 9 and Corollary 10 of Ji et al. [2022]. Then, under Assumptions 1, 2:*

$$\mathbb{E}[\|\mathbb{E}_t[\nabla \mathcal{J}_t(\hat{\gamma}_{\text{meta}})]\|_2] = \mathcal{O}\left(\sqrt{\tfrac{1}{I_{tr}} + \tfrac{1}{B}} \right),$$

where the outer expectation is with respect to sampling of the mini-batches of tasks in Algorithm 1.

The proof is contained in Appendix A.2. If we take $B \geq C_B \epsilon^{-2}$ with C_B a large constant, the theorem shows that, with at most $I_{\text{tr}} = \mathcal{O}(1/\epsilon^2)$ meta iterations, the output $\hat{\gamma}_\epsilon$ of Algorithm 1 satisfies $\mathbb{E}[\|\mathbb{E}_t[\nabla \mathcal{J}_t(\hat{\gamma}_\epsilon)]\|] = \mathcal{O}(\epsilon)$. The requirements on the step-sizes and batch size are inherited from Ji et al. [2022], are spelled out in Appendix A.1, and provide guiding insight into the practical side of training of Meta-CVs, e.g. theoretically optimal meta-step-sizes

$\eta_1, \ldots, \eta_{I_{\text{tr}}}$ and inner-step size α. For Neural CVs it is difficult to go beyond Theorem 1, since for one thing there will not be a unique γ_{meta} in general. However, for simpler CVs, such as those based on polynomial regression [Assaraf and Caffarel, 1999, Mira et al., 2013, Papamarkou et al., 2014, Friel et al., 2014, South et al., 2022b], it is reasonable to assume a unique γ_{meta} and convexity of the Meta-CV objective around this point. In these scenarios, the following corollary shows that $\hat{\gamma}_\epsilon$ is typically close to the minimiser of the task-specific objective functional.

Corollary 1.1. *Under the setting of Theorem 1, further suppose that there exists $\mu > 0$ such that for all t and all γ, $\nabla^2 J_{Q_t}(\gamma) \succeq \mu I_{p+1}$ where I_{p+1} is an identity matrix of size $p+1$. Then there exist constants $C_1, C_2 > 0$ such that*

$$\mathbb{E}[\mathbb{E}_t[\|\hat{\gamma}_\epsilon - \gamma_t^*\|_2]] \leq \frac{C_1}{\mu}\epsilon + \frac{C_2}{\mu},$$

where γ_t^* is the (unique) minimiser of $\gamma \mapsto J_{Q_t}(\gamma)$, and here again the outer expectation is with respect to sampling of the mini-batches of tasks in Algorithm 1.

The proof is contained in Appendix A.3. These results justify the use of Algorithm 1 to train the Meta-CV and task-specific CVs. In particular, they provide insight into step size selection, and establish explicit conditions on the form of CV $g(x;\gamma)$ that can be successfully trained using the methodology that we have proposed.

7 CONCLUSION

This paper introduced Meta-CVs, an extension of existing CV methods that brings meta-learning to bear on MC and MCMC. More precisely, our method can achieve significant variance reduction when the number of samples per integration task is small, but a large number T of similar tasks are available. In addition, most of the computational cost is an offline cost for identifying a Meta-CV, and CVs for new integration tasks can be identified with minimal additional computational cost.

Although our algorithm is scalable in T and N_t, the computational cost for training the Meta-CV can still be significant when dealing with flexible CVs, such as Neural CVs. For example, computational complexity scales as $\mathcal{O}(p^2)$ in the number of parameters p in the CV. This prevents us from using very large neural networks, which could limit performance on more challenging integration tasks. First-order or Hessian-free meta-learning algorithms [Fallah et al., 2020] are therefore a promising direction for future work.

Alternatively, online meta-learning algorithms [Finn et al., 2019] could be adapted to CVs. These could be particularly powerful for cases where integration tasks arrive sequentially and the Meta-CV cannot be computed offline. Examples includes application areas where sequential importance sampling and sequential MC-type algorithms [Doucet et al., 2000, 2001] are currently being used, such as in the context of state-space models.

Finally, it is also possible to further extend our theoretical analysis of Meta-CVs. The current convergence rate in I_{tr} of Meta-CVs aligns with [Fallah et al., 2020, Ji et al., 2022]. Future work could extend the theoretical analysis of Meta-CVs from a information-theoretic aspect [Chen et al., 2021] or towards a faster rate [Riou et al., 2023] with additional conditions.

Acknowledgements

The authors would like to thank Kaiyu Li for sharing some of her code for the boundary value ODE example. ZS was supported under the EPSRC grant [EP/R513143/1] and The Alan Turing Institute's Enrichment Scheme. CJO and FXB were supported by the Lloyd's Register Foundation Programme on Data-Centric Engineering and The Alan Turing Institute under the EPSRC grant [EP/N510129/1]. CJO was supported by the EPSRC grant [EP/W019590/1].

References

A. Alexopoulos, P. Dellaportas, and M. K. Titsias. Variance reduction for Metropolis–Hastings samplers. *Stat. Comput.*, 33(6), 2023.

A. Anastasiou, A. Barp, F-X. Briol, R. E. Ebner, B.and Gaunt, F. Ghaderinezhad, J. Gorham, A. Gretton, C. Ley, Q. Liu, et al. Stein's method meets computational statistics: a review of some recent developments. *Stat. Sci.*, 38(1):120–139, 2023.

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas. Learning to learn by gradient descent by gradient descent. *NeurIPS*, 2016.

A. Antoniou, H. Edwards, and A. Storkey. How to train your maml. In *ICLR*, 2019.

R. Assaraf and M. Caffarel. Zero-variance principle for Monte Carlo algorithms. *Phys. Rev. Lett.*, 83(23):4682, 1999.

J. Baker, P. Fearnhead, E. B. Fox, and C. Nemeth. Control variates for stochastic gradient MCMC. *Stat. Comput.*, 29:599–615, 2019.

A. Barp, C. J. Oates, E. Porcu, and M. Girolami. A Riemannian–Stein kernel method. *Bernoulli*, 28(4):2181–2208, 2022.

D. Belomestny, L. Iosipoi, E. Moulines, A. Naumov, and S. Samsonov. Variance reduction for Markov chains with application to MCMC. *Stat. Comput.*, 30:973–997, 2020.

D. Belomestny, L. Iosipoi, E. Moulines, Al. Naumov, and S. Samsonov. Variance reduction for dependent sequences with applications to stochastic gradient MCMC. *SIAM-ASA J. Uncertain.*, 9(1):507–535, 2021.

B. M. Bolker. Ecological models and data in R. In *Ecological Models and Data in R*. Princeton University Press, 2008.

F. Brauer. Mathematical epidemiology: Past, present, and future. *Infect. Dis. Model.*, 2(2):113–127, 2017.

B. Carpenter, A. Gelman, M. D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell. Stan: A probabilistic programming language. *J. Stat. Softw.*, 76(1), 2017.

Q. Chen, C. Shui, and M. Marchand. Generalization bounds for meta-learning: An information-theoretic analysis. *NeurIPS*, 34:25878–25890, 2021.

P. Dellaportas and I Kontoyiannis. Control variates for estimation based on reversible Markov chain Monte Carlo samplers. *J. R. Stat. Soc. Series B*, 74(1):133–161, 2012.

J. Demange-Chryst, F. Bachoc, and J. Morio. Efficient estimation of multiple expectations with the same sample by adaptive importance sampling and control variates. *arXiv:2212.00568*, 2022.

A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Stat. Comput.*, 10:197–208, 2000.

A. Doucet, N. De Freitas, and N. J. Gordon. *Sequential Monte Carlo methods in practice*, volume 1. Springer, 2001.

A. Fallah, A. Mokhtari, and A. Ozdaglar. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In *AISTATS*. PMLR, 2020.

C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

C. Finn, K. Xu, and S. Levine. Probabilistic model-agnostic meta-learning. In *NeurIPS*, 2018.

C. Finn, A. Rajeswaran, S. Kakade, and S. Levine. Online meta-learning. In *ICML*, 2019.

N. Friel, A. Mira, and C. J. Oates. Exploiting multi-core architectures for reduced-variance estimation with intractable likelihoods. *Bayesian Anal.*, 11(1):215–245, 2014.

A. Gessner, J. Gonzalez, and M. Mahsereci. Active multi-information source Bayesian quadrature. In *UAI*, 2019.

M. Giles. Multilevel Monte Carlo methods. *Acta Numer.*, 24:259–328, 2015.

P. Glynn and D. Igelhart. Importance sampling for stochastic simulations. *Management Science*, 35(1367-1392), 1989.

E. Grant, C. Finn, S. Levine, T. Darrell, and T. Griffiths. Recasting gradient-based meta-learning as hierarchical Bayes. In *ICML*, 2018.

W. Grathwohl, D. Choi, Y. Wu, G. Roeder, and D. Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *ICLR*, 2018.

P. Green, K. Latuszyski, M. Pereyra, and C. Robert. Bayesian computation: a summary of the current state, and samples backwards and forwards. *Stat. Comput.*, 25: 835–862, 2015.

E. Grefenstette, B. Amos, D. Yarats, P. Htut, A. Molchanov, F. Meier, D. Kiela, K. Cho, and S. Chintala. Generalized inner loop meta-learning. *arXiv:1910.01727*, 2019.

C. Hewitt. *The conservation of the wild life of Canada*. New York: C. Scribner, 1921.

F.J. Hickernell, C. Lemieux, and A. B. Owen. Control variates for quasi-Monte Carlo. *Stat. Sci.*, 20(1):1–31, 2005.

K. Ji, J. Yang, and Y. Liang. Theoretical convergence of multi-step model-agnostic meta-learning. *J. Mach. Learn. Res.*, 23:29–1, 2022.

D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

S. Krumscheid and F. Nobile. Multilevel monte carlo approximation of functions. *SIAM-ASA J. Uncertain.*, 6(3): 1256–1293, 2018.

R. Leluc, F. Portier, and J. Segers. Control variate selection for monte carlo integration. *Stat. Comput.*, 31(4):1–27, 2021.

K. Li, D. Giles, T. Karvonen, S. Guillas, and F-X. Briol. Multilevel Bayesian quadrature. In *AISTATS*, pages 1845–1868, 2023.

H. Liu, Y. Feng, Y. Mao, D. Zhou, J. Peng, and Q. Liu. Action-dependent control variates for policy optimization via stein's identity. In *ICLR*, 2018.

H. Liu, R. Socher, and C. Xiong. Taming maml: Efficient unbiased meta-reinforcement learning. In *ICML*. PMLR, 2019.

F. A. Longstaff and E. S. Schwartz. Valuing american options by simulation: A simple least-squares approach. *Rev. Financ. Stud.*, 14(1):113–147, 2001.

A. Lotka. Fluctuations in the abundance of a species considered mathematically. *Nature*, 119(2983):12–12, 1927.

N. Madras and M. Piccioni. Importance sampling for families of distributions. *Ann. Appl. Probab.*, 9(4):1202–1225, 1999.

A. Mira, R. Solgi, and D. Imparato. Zero variance Markov chain Monte Carlo for Bayesian estimators. *Stat. Comput.*, 23(5):653–662, 2013.

T. Müller, F. Rousselle, J. Novák, and A. Keller. Neural control variates. *ACM Trans. Graph.*, 39(6):1–19, 2020.

C. J Oates and M. Girolami. Control functionals for quasi-Monte Carlo integration. In *AISTATS*, 2016.

C. J. Oates, T. Papamarkou, and M. Girolami. The controlled thermodynamic integral for Bayesian model comparison. *J. Am. Stat. Assoc.*, 111(514):634–645, 2016.

C. J. Oates, M. Girolami, and N. Chopin. Control functionals for Monte Carlo integration. *J. R. Stat. Soc. Series B*, 79(3):695–718, 2017.

C. J. Oates, J. Cockayne, F-X. Briol, and M. Girolami. Convergence rates for a class of estimators based on Stein's method. *Bernoulli*, 25(2):1141–1159, 2019.

K. Ott, M. Tiemann, P. Hennig, and F-X. Briol. Bayesian numerical integration with neural networks. *arXiv:2305.13248*, 2023.

J. Paisley, D. M. Blei, and M. I. Jordan. Variational bayesian inference with stochastic search. In *ICML*, 2012.

T. Papamarkou, A. Mira, and M. Girolami. Zero variance differential geometric Markov chain Monte Carlo algorithms. *Bayesian Anal.*, 9(1):97–128, 2014.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 32, 2019.

B. Peherstorfer, K. Willcox, and M. Gunzburger. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *SIAM Review*, 60(3):550–591, 2018.

C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*, volume 1. Springer, 2006.

C. Riou, P. Alquier, and B-E. Chérief-Abdellatif. Bayes meets bernstein at the meta level: an analysis of fast rates in meta-learning with pac-bayes. *arXiv preprint arXiv:2302.11709*, 2023.

J. Shi, Y. Zhou, J. Hwang, M. K. Titsias, and L. Mackey. Gradient estimation with discrete Stein operators. In *NeurIPS*, 2022.

S. Si, C. J. Oates, A. B. Duncan, L. Carin, and F-X. Briol. Scalable control variates for Monte Carlo methods via stochastic optimization. *Proceedings of the 14th Conference on Monte Carlo and Quasi-Monte Carlo Methods. arXiv:2006.07487*, 2021.

L. F. South, T. Karvonen, C. Nemeth, M. Girolami, and C. J. Oates. Semi-exact control functionals from Sard's method. *Biometrika*, 2022a.

L. F. South, C. J. Oates, A. Mira, and C. Drovandi. Regularized zero-variance control variates. *Bayesian Anal.*, 1(1): 1–24, 2022b.

L. F. South, M. Riabiz, O. Teymur, and C. J. Oates. Post-Processing of MCMC. *Annu. Rev. Stat. Appl.*, 2022c.

Z. Sun, A. Barp, and F-X. Briol. Vector-Valued Control Variates. *arXiv:2109.08944, to appear at ICML 2023*, 2021a.

Z. Sun, J. Wu, X. Li, W. Yang, and J-H. Xue. Amortized Bayesian Prototype Meta-learning: A new probabilistic meta-learning approach to few-shot image classification. In *AISTATS*, pages 1414–1422. PMLR, 2021b.

R. Wan, M. Zhong, H. Xiong, and Z. Zhu. Neural control variates for variance reduction. *ECML PKDD*, page 533–547, 2019.

C. Wang, X. Chen, A. J. Smola, and E. P. Xing. Variance reduction for stochastic gradient optimization. *NeurIPS*, 2013.

X. Xi, F-X. Briol, and M. Girolami. Bayesian quadrature for multiple related integrals. In *ICML*, 2018.

J. Yoon, T. Kim, O. Dia, S. Kim, Y. Bengio, and S. Ahn. Bayesian model-agnostic meta-learning. In *NeurIPS*, 2018.