Meta-learning Control Variates: Variance Reduction with Limited Data

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 often be abstracted out to the task of computing

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

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

Such integrals can only be computed in closed-form on very rare occasions. This has led to the development of a range of approximate methods, including *Monte Carlo* (MC) methods, which make use of integrand evaluations $\{f(x_i)\}_{i=1}^N$; see Green et al. [2015] for a review. 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 a MC estimator of $\mathbb{E}_\pi[\tilde{f}]$ where $\tilde{f} = f - g + \mathbb{E}_\pi[g]$. This is a consistent estimator since $\mathbb{E}_\pi[\tilde{f}] = \mathbb{E}_\pi[f] - \cancel{\mathbb{E}_\pi[g]} + \cancel{\mathbb{E}_\pi[g]} = \mathbb{E}_\pi[f]$. However, it also has the advantage that $\tilde{f}$ usually has much smaller variance than f. As a result, 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] or Markov chain Monte Carlo (MCMC) [Baker et al., 2019], for reinforcement learning Liu et al. [2018, 2019] and in computer graphics [Müller et al., 2020].

Unfortunately, CVs perform poorly when only a small number of samples N is available. This occurs when sampling or evaluating the integrand is expensive, or when our computational budget is limited. The reason is that a small N provides limited data from which to construct an approximation g to f, which limits the effectiveness of the CV. It can be particularly problematic for high-dimensional settings, since such functions are even more difficult to approximate due to the curse of dimensionality. For example, Oates et al. [2019] shows that CVs perform poorly even for some applications in $d = 3$ when N is in the order of hundreds. This issue can be partly mitigated by exploiting sparsity South et al. [2022b], Leluc et al. [2021], but many integrands do not necessarily have sparsity which can be exploited.

We propose instead to consider a radically different solution. Our approach requires a setting where T integration tasks of the form in (1) need to be tackled, and where the integrands

Submitted to the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023). **To be used for reviewing only**.

$f_1, \ldots, f_T$ and densities $\pi_1, \ldots, \pi_T$ are different, but possibly closely related. Related integration tasks arise in a broad range of settings, including multifidelity modelling [Peherstorfer et al., 2018, Li et al., 2022], sensitivity analysis [Demange-Chryst et al., 2022], policy gradient methods [Liu et al., 2018] or thermodynamic integration [Oates et al., 2016]. Section 5 also considers marginalisation of hyperparameters in Bayesian inference (see the hierarchical Gaussian process example) and the computation of predictive distributions (see the Lotka-volterra example). A relatively natural idea in those cases is to construct CVs which share information across these tasks to improve performance. In particular, any samples from a given integration task could be used to improve the CVs constructed for all other tasks.

The only CV method currently able to do so is *vector-valued CVs* [Sun et al., 2021]. This algorithms learns the relationship between integrands through a multi-task learning approach in a vector-valued reproducing kernel Hilbert space. It has shown great potential, but suffers from a prohibitive computational cost when the number of tasks T is large (up to $\mathcal{O}(T^6)$). It also suffers from a significant memory cost in this setting, due to the need to store up to $\mathcal{O}(T^2)$ effective parameters. The largest experiment in Sun et al. [2021], which focuses on computation of the model evidence for a dynamical system, builds on the relationship between $T = 4$ tasks. This is despite the fact that this example has up to 62 related integrals but, unfortunately, a larger T would make the computation infeasible. This is a significant limitation; in many of the examples mentioned above, it may be desirable to share information for up to hundreds or thousands of tasks. A key question is therefore: *"How can we construct CVs sharing information across a large number of tasks in a computationally scalable manner?"*

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.

2 BACKGROUND

Before presenting Meta-CVs, we start by reviewing existing work on CVs on which these are built.

Variance Reduction for Monte Carlo 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. 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 estimators [Dellaportas and Kontoyiannis, 2012, Belomestny et al., 2020, 2021, Alexopoulos et al., 2023] and (randomised) quasi-Monte Carlo estimators [Hickernell et al., 2005]. Although we will not expand on those cases, which have a different asymptotic variance term, we note that $\mathbb{V}_\pi[f]$ is still a useful proxy for the error in those cases.

As previously mentioned, one strategy to improve such estimators is to identify a CV $g \in \mathcal{L}^2(\pi)$. This can then be used to construct the estimator

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

A similar central limit theorem holds for this estimator, but with $\mathbb{V}_\pi[f - g]$ instead of $\mathbb{V}_\pi[f]$. If g is selected appropriately, the CV estimator will therefore tend to have a much smaller error than the original MC estimator.

We highlight that the index in the sum above runs from $m+1$ to N, instead of 1 to N. This is because it is common to split our samples into two datasets: one (with m samples) used to select a CV, and the other (with n samples) to compute the estimator $\hat{\mathbb{E}}_\pi^{\text{CV}}[f]$, so that $N = m + n$. In the remainder of this section, we detail the various ways in which the first m samples can be used to select a CV.

Zero-Mean Functions A first challenge when selecting a CV g is that we require it to have 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 a Bayesian posterior.

One way forward is through Stein's method, and we will call any CV constructed through this method a Stein-based CV; see Anastasiou et al. [2023] for a review. The main components of Stein's method are a function class $\mathcal{U}$, called Stein class, and an operator $\mathcal{S}_\pi$ acting on $\mathcal{U}$, called 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*:

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

where $\nabla_x := (\partial/\partial x_1, \ldots, \partial/\partial x_d)^\top$ and $\nabla_x \cdot u(x) := \sum_{j=1}^d \partial u_j(x)/\partial x_j$. This operator satisfies $\mathbb{E}_\pi[\mathcal{S}_\pi[u]] = 0$ under standard conditions on $\mathcal{U}$; see [Oates et al., 2019] for full details. In addition, this operator only requires pointwise evaluations of $\nabla_x \log \pi(x)$, which can be done even when π involves an unknown normalisation constant, i.e.

$\pi = \tilde{\pi}/C$ where $\tilde{\pi}$ is known point-wise and $C > 0$ is unknown. This is a significant advantage 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 polynomials with 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 reproducing kernel Hilbert space; 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., 2021]. Alternatively, one may take $\mathcal{U}$ to be a (parametric) set of neural networks [Wan et al., 2019, Si et al., 2021], 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 fixed memory and computational cost.

Selecting Control Variates Once a family of CVs has been identified, we still need to select a good 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 $g(x;\gamma) = g_{\gamma_{1:p}} + \gamma_0$ where $\gamma := \gamma_{0:p} \in \Gamma \subseteq \mathbb{R}^{p+1}$ consists of p parameters $\gamma_{1:p}$ for the zero-mean Stein-based function, and an additional parameter γ_0 approximating $\mathbb{E}_\pi[f]$. Following the framework of empirical loss minimisation with samples $S = \{x_i, \nabla \log \pi(x_i), f(x_i)\}_{i=1}^m$, this can be estimated through:

$$\begin{aligned} J_S(\gamma) &:= \tfrac{1}{m}\textstyle\sum_{i=1}^m \left(f(x_i) - g(x_i;\gamma)\right)^2 \\ &\approx \mathbb{E}_{X\sim\pi}[(f(X) - g(X;\gamma) - \mathbb{E}_\pi[f - g(\cdot;\gamma)])^2] \\ &= \mathbb{V}_\pi[f - g(\cdot,\gamma)] \end{aligned}$$

Note that as $m \to \infty$, the value of γ_0 minimising this objective is a consistent estimator for the integral $\mathbb{E}_\pi[f]$. To avoid over-fitting on the data when m is small, it is also common to use a penalised objective $J_m^\lambda(\gamma) = \frac{1}{m}\sum_{i=1}^m \left(f(x_i) - g(x_i;\gamma)\right)^2 + \lambda\|\gamma_{t,1:p}\|_2^2$ where $\lambda > 0$ [Wan et al., 2019, Si et al., 2021].

Conveniently, in the specific case of Neural-CVs, the backward propagation of gradients with respect to these parameters can be done end-to-end, and challenging terms such as $\nabla_x \cdot u_{\gamma_{1:p}}(x)$ can be automatically computed 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 Integration *Multi-output Bayesian quadrature* [Xi et al., 2018, Gessner et al., 2019] and vector-valued CVs [Sun et al., 2021] 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 paper.

Monte Carlo Methods for Parametric and Conditional Expectations Parametric expectation or conditional expectation methods [Krumscheid and Nobile, 2018, Alfonsi et al., 2022] consider the task of approximating $\mathbb{E}_{X\sim\pi}[f(X,y)]$ 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. To our knowledge, the only approach which considers specifically the problem of multiple related integrals is the work of Demange-Chryst et al. [2022] (and briefly Section 3 of He and Owen [2014]), where the authors seek an importance distribution π' which performs well across a range of integration tasks. However, identifying such an importance distribution will usually not be possible when T is large.

4 METHODOLOGY

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

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

We assume that T_{train} of these tasks are available a-priori, and $T_{\text{test}} = T - T_{\text{train}}$ of those tasks are unseen initially and will need to be tackled at a later stage. We 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

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

where $N_t \in \mathbb{N}^+$ is relatively small. In addition, we will assume that these tasks are related in the sense that $\mathcal{T}_1, \ldots, \mathcal{T}_T$ are independent realisations from a distribution over tasks denoted ρ, called the *environment*. The set-up above allows us to frame our challenge 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] 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 a model 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 this framework to the construction of CVs, which leads to *meta-learning CVs*. This leads to a two-step approach: the first step, highlighted in Algorithm 1, consists of learning a *Meta-CV* from the first T_{train} tasks, whilst the second step, highlighted in Algorithm 2, consists of adapting this Meta-CV to obtain *task-specific CVs*. This can be achieved for any of the T_{train} tasks used to train the Meta-CV, or for any of the T_{test} unseen tasks.

Before describing these algorithms, we note that for each task $\mathcal{T}_t := \{f_t, \pi_t\}$, we will need to 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. Although the exact proportion of points assigned to each dataset could theoretically have an impact on the algorithm, N_t will be small, so both n_t and m_t will also be small and we do not expect this impact to be very significant in practice. As a result and for simplicity, all of our experiments will consider $m_t = n_t = N_t/2$. Note that these datasets naturally correspond to the concepts of *support set* and *query set* in gradient-based meta-learning [Finn et al., 2017, 2019].

Constructing the Meta-CV Our goal is to construct a CV, called Meta-CV, which can be adapted to a new task $\mathcal{T}_t$ from the environment ρ in a small number of steps (denoted L) of some gradient-based optimisation method. We will follow the approach in Section 2 and use a Neural-CV together with an objective taking the form of the MC variance.

To allow for flexibility in the choice of optimiser, a single step of an arbitrary gradient-based optimiser will be denoted abstractly by $\textsc{Update}(\gamma, \nabla_\gamma J(\gamma); \alpha)$, where $\gamma \in \Gamma$ is the current parameter value, $\nabla_\gamma J(\gamma)$ is the gradient of some arbitrary objective $J : \Gamma \to \mathbb{R}$, and α represents parameters of the optimiser. $\textsc{Update}_L(\gamma, \nabla_\gamma J(\gamma); \alpha)$ will then denote L such updates starting from the input γ. Examples of optimisers include gradient descent, Adam [Kingma and Ba, 2015], or more flexible alternatives [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 our *idealised Meta-CV* as the CV whose parameters given by

$$\gamma_{\text{meta}} := \arg\min_\gamma \mathbb{E}_{t\sim\rho}\left[J_t\left(\textsc{Update}_L\left(\gamma, \nabla_\gamma J_t(\gamma); \alpha\right)\right)\right], \tag{3}$$

where $J_t(\gamma) = \mathbb{V}_{\pi_t}[f_t - g(\cdot;\gamma)]$ is the variance of the CV estimator for the tth task. 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 twice per task $\mathcal{T}_t$. To prevent overfitting, 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.

Task-Specific CVs Once a meta-parameter $\hat{\gamma}_{\text{meta}}$ has been identified and assuming we are presented with a new task $\mathcal{T}_t$, we only need to adapt the meta-parameter through a

Algorithm 1: Learning a Meta-CV

Input: Training tasks $\mathcal{T}_1, \ldots, \mathcal{T}_{T_{\text{train}}}$, initial parameters γ_0, UPDATE rule, # update steps L, optimiser parameters α and η, mini-batch size B, # meta-iterations E_{tr}.

for *e from 1 to* E_{tr} **do**
  Sample a mini-batch of size B from $\mathcal{T}_1, \ldots, \mathcal{T}_{T_{\text{train}}}$.
  for *each task index* $t = 1, \ldots, B$ *in the minibatch* **do**
    Initialize $\gamma_0^t \leftarrow \gamma_{e-1}$.
    for *j from 1 to L* **do**
      $\gamma_j^t \leftarrow \text{UPDATE}(\gamma_{j-1}^t, \nabla_{\gamma_{j-1}^t} J_{S_t}(\gamma_{j-1}^t); \alpha)$.
  $\gamma_e \leftarrow \text{UPDATE}(\gamma_{e-1}, \frac{1}{B}\sum_{i=1}^{B} \nabla_{\gamma_{e-1}} J_{Q_t}(\gamma_L^t); \eta_e)$.

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

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_{train} tasks in the training set, or to one of the T_{test} unseen tasks. Once such a task-specific CV is identified, we can simply use the CV estimator in Equation (2) to estimate the corresponding integral $\mathbb{E}_\pi[f_t]$. 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.

To understand why these task-specific CVs perform well, we highlight that the task-specific CV for any task $\mathcal{T}_t$ derived through our approach is constructed using $\sum_{t'=1}^{T_{\text{train}}} N_{t'}$ samples when $\mathcal{T}_t$ is a training task, or $\sum_{t'=1}^{T_{\text{train}}} N_{t'} + N_t$ samples when it is an unseen task. When T_{train} is large and N_t is small for all tasks, this means that the task-specific CV is based on a large number of samples from related tasks which complement the very few samples we have access to for that particular task. It is exactly this sharing of information across tasks which will allow us to obtain significant variance reduction in the experiments in our next Section.

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 from 1 to L* **do**
  $\gamma_j \leftarrow \text{UPDATE}(\gamma_{j-1}, \nabla J_{S_t}(\gamma_{j-1}); \alpha)$.
Return the task-specific parameter $\hat{\gamma}_t := \gamma_L$ for task $\mathcal{T}_t$.

Computational Complexity Before studying the performance of our proposed Meta-CV in experiments, we briefly pause to comment on the computational complexity of the approach. Suppose the total number of tasks for which we want task-specific CVs is T, with $N_1 = \ldots = N_T$ and $m_1 = \ldots = m_T$ both taking small values for all tasks.

Under these assumptions and assuming a Meta-CV has been pre-computed, the computational complexity of Neural-CVs is $\mathcal{O}(TL_{\text{NCV}})$ when L_{NCV} gradient-based steps are used per task, while the complexity for Meta-CVs is $\mathcal{O}(TL_{\text{MCV}})$ when L_{MCV} gradient-based steps are used in the update for deriving each task-specific CV. Since the number of gradient steps for Neural-CVs is normally large (in the hundreds or thousands) and the number of gradient steps for MCV is relatively small (most often $L_{\text{MCV}} = 1$ in our experiments), we expect task-specific CVs to be much faster to compute than Neural-CVs on any unseen task. This comes of course in addition of the advantage that it builds on information from all tasks in the training set, which is essential when the number of samples per task is small.

Of course the comparison above depends on having pre-computed the Meta-CV. If we are instead in a setting with T tasks on which we must both compute the Meta-CV and task-specific CVs, then the scaling in N_t and T is the same for both Neural-CVs and our approach, but the scaling in p, the number of neural network parameters, is at least $\mathcal{O}(p^2)$ 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

We will now evaluate the performance of the proposed Meta-CVs on 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 $N_1 = \ldots = N_T$ and where the meta-learning updates consist of Adam steps [Kingma and Ba, 2015]. We note that none of the methods discussed in Section 3 can be applied to the problems in this section due to the large value of T, and we therefore only compare to methods which do not build on task relationships: MC, Neural-CVs and CFs. The code to reproduce our results is available at: [temporarily withheld for anonymity].

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]. We therefore start by considering integrands of the form $f_t(x; a_t) = \cos(2\pi a_{t,1} + \sum_{i=1}^{d} a_{t,i+1} x_i)$ with parameters $a_t \in \mathbb{R}^{d+1}$. This is a convenient synthetic problem since the integral of f_t with respect to the uniform distribution on $\mathcal{X} = [0,1]^d$ can be evaluated in closed-form for any a_t, and we are therefore able to assess the performance of different methods. Note that a_t controls the difficulty of the tth integration task and ρ is a distribution over the values of a_t consisting of independent uniforms; see Appendix B.1 for full detail.

In Figure 1, we first consider the case $d = 2$ and $T_{\text{train}} =$

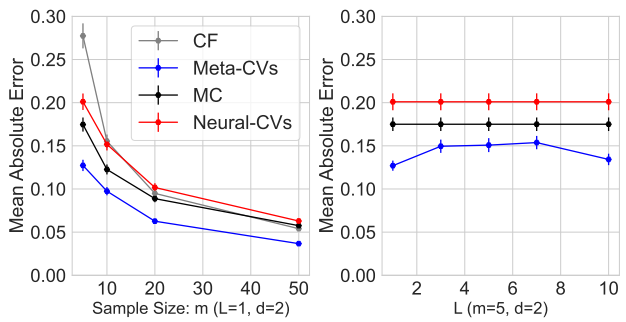

Figure 1: Mean absolute error (with 95% confidence intervals) for $T_{\text{test}} = 1000$ oscillatory functions. *Left*: Increasing sample size $m_t = n_t = m$ in $d = 2$; *Right*: Increasing number of inner gradient steps L.

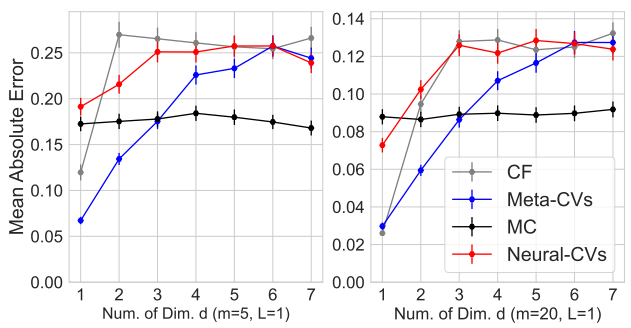

Figure 2: Mean absolute errors (with 95% Confidence Intervals) for $T = 1000$ oscillatory functions for increasing d.

10000. 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 $T_{\text{test}} = 1000$ new unseen tasks sampled from ρ. This can be explained by the fact that Meta-CVs is the only method which can transfer information across tasks. 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 $d < 6$.

We conclude with a brief discussion of computational cost. The cost of computing independent Neural-CVs on all tasks is around 2 minutes (when $T_{\text{train}} = 20000$, $m = 5$ and $d = 1$). In contrast, the offline time for training the Meta-CV is around 5 minutes (with $L = 1$), but the online time taken for deriving a task-specific CV for a new task is approximately 6 seconds. This demonstrates that our Meta-CV can be adapted to new tasks in real time.

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]:

$$\frac{d}{ds}\left(c(s)\frac{du}{ds}\right) = -50x^2 \qquad \text{for } 0 < s < 1,$$

with boundary $u(0) = u(1) = 0$, $c(s) = 1 + as$. The integrand of interest is $f_t(x) = \int_0^1 u(s, x; a_t)ds$ 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 $\pi_1 = \ldots = \pi_T$ are densities of a $\mathcal{N}(0, 1)$. We use a finite difference approximation of f_t described in [Giles, 2015]; see Appendix B.2 for details. This is a relatively simple example, but it is representative of a broader class of challenging problems where numerical methods are needed to approximate integrands, leading to a large cost per integrand evaluation and therefore limited N_t.

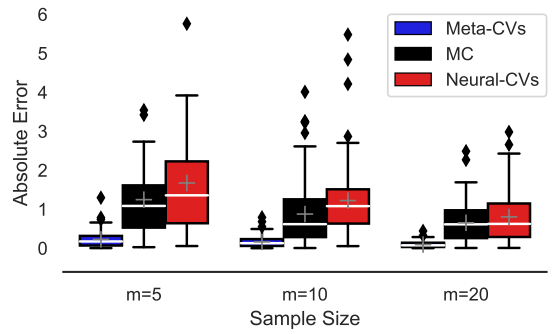

Figure 3: Absolute error for $T_{\text{test}} = 100$ unseen tasks from the boundary value ODE problem with $m_t = n_t = m$.

The results are presented in Figure 3. We compare the performance of Meta-CVs with MC and Neural-CVs on $T_{\text{test}} = 100$ unseen tasks (grey crosses are mean absolute 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 equations-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, and given by

$$\frac{du_1}{ds} = x_1u_1 - x_2u_1u_2, \quad \frac{du_2}{ds} = x_3u_1u_2 - x_4u_2,$$

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 some noisy observations of $u = (u_1, u_2)$ at time points $\{s_1, \ldots s_q\}$ observed with log-normal noise with variance x_7 and x_8 respectively. We will follow the approach of Si et al. [2021], Riabiz et al. [2022] perform

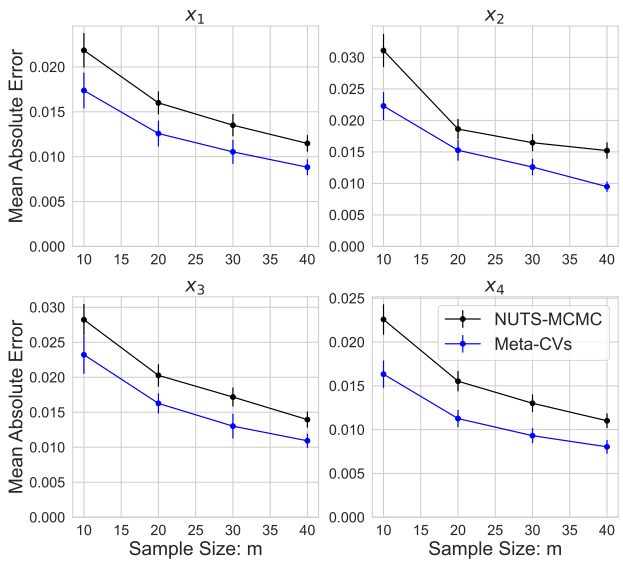

Figure 4: Mean absolute errors (with 95% confidence intervals) over 40 sub-populations for varying $m_t = n_t = m$.

Bayesian inference for all parameters x of this model, so that $d = 8$. 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 samples N_t available.

Suppose we have a meta-population of preys and predators - this could for example happen if we have data on different species or data from different regions. In this case, ρ is a distribution of sub-populations. From the point of view of integration, we can consider posterior moments for each sub-populations, in which case $f_1(x) = \ldots = f_T(x) = x_i$ for $i \in \{1, \ldots, 8\}$, but $\pi_1, \ldots, \pi_T$ are (different but) closely related posterior distributions. 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 more details.

Results are presented in Figure 4. 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 achieve a lower mean absolute error than MC for 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 some unknown function ν from observations $y_i = \nu(z_i) + \epsilon_i$ at states z_i for $i = 1, \ldots, q$, where ϵ_i are IID zero-mean Gaussian random variables with standard deviation $\sigma > 0$ assumed known. We posit a zero-mean Gaussian process prior 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 n observations $y_{1:q} = (y_1, \ldots, y_q)^\top$ at some states $z_1, \ldots, z_q$, we will usually be interested in predicting the response y^* at an unseen state z^*, marginalising out any uncertainty in the posterior over the hyperparameters x.

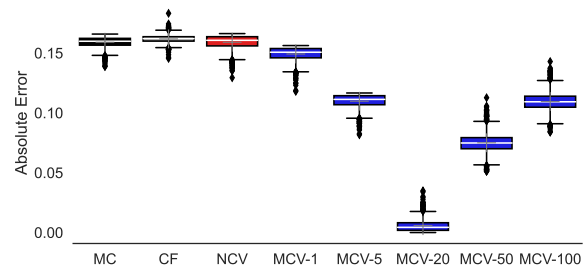

Figure 5: Effect of L: Absolute errors over $T_{\text{test}} = 1000$ unseen states of the Sarcos anthropomorphic robot arm (*CF*: Control functionals; *NCV*: Neural-CVs; *MCV-L*: Meta-CVs with L inner steps).

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 the GP posterior mean $f(x) = \mathbb{E}[Y^*|y_{1:q}, x] = K_{z^*,q}(x)(K_{q,q}(x) + \sigma^2 I_q)^{-1} y_{1:q}$ 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 hundreds. However, it is also common to want to compute this quantity for several points $z_1^*, \ldots, z_T^*$ simultaneously, leading to closely related integrands $f_1, \ldots, f_T$ whose relationship could potentially be leveraged (here the distribution is the same for all integrals).

We demonstrate this through a hierarchical Gaussian process estimating the inverse dynamics of a 7 degrees-of-freedom Sarcos anthropomorphic robot arm, based on a subset of the dataset described in [Rasmussen and Williams, 2006]. This consists of estimating a map ν from a 21-dimensional input space. Our dataset is divided into two parts. The first part is used for obtaining the posterior (which is approximated through variational inference) and consists of $q = 1000$ data points. The second part includes 4449 data points, half of which are used to construct the Meta-CV and the other half is used for studying the performance of various estimators. See Appendix B.4 for full experimental details.

The results are presented in Figure 5. We compare the performance of Meta-CVs with MC, CFs and Neural-CVs on

$T_{\text{test}} = 1000$ unseen tasks, where $N_t = 4$ for each task. Because we do not have access to the exact value of these integrals, the error is estimated by replacing this value with y_t^*. 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 $L > 1$.

6 THEORETICAL ANALYSIS

The empirical results of the previous section demonstrate the advantage of leveraging the relationship between integration tasks to improve the performance of CVs, even for tasks that have not been used during the process of learning the Meta-CVs. This section will now focus on obtaining theoretical insight that can be used to guide the implementation of gradient-based optimisation within Meta-CVs.

Our analysis here focuses on the training of Meta-CVs and is restricted to the performance on the same set of training tasks $I = \{\mathcal{T}_1, \ldots, \mathcal{T}_{T_{\text{train}}}\}$, rather than unseen new tasks. In particular, we consider the datasets $D_1, \ldots, D_{T_{\text{train}}}$ to be given and fixed. In this case ρ is a discrete measure on I. Our objective for learning a Meta-CV is then given by

$$\arg\min_\gamma \mathbb{E}_{t\sim\rho}\left[\mathcal{J}_t(\gamma)\right], \tag{4}$$

where $\mathcal{J}_t(\gamma) := J_{Q_t}\left(\textsc{Update}_L\left(\gamma, \nabla_\gamma J_{S_t}(\gamma); \alpha\right)\right)$, and the UPDATE rule is gradient descent with L steps and learning rate α. To proceed, we make the following assumptions:

Assumption 1. *For each $t \in I$ 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 $t \in I$ 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 (3), for any $\epsilon > 0$.

Theorem 1. *Let γ be the output of Algorithm 1 with gradient descent steps using the meta-step-size η_t and batch size B proposed in Theorem 9 and Corollary 10 of Ji et al. [2022]. Then, under Assumptions 1, 2:*

$$\mathbb{E}[\|\mathbb{E}_{t\sim\rho}[\nabla\mathcal{J}_t(\gamma)]\|_2] = \mathcal{O}\left(\sqrt{\tfrac{1}{E_{tr}} + \tfrac{1}{B}}\right),$$

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

Thus, if we take $B \geq C_B\epsilon^{-2}$ with C_B a large constant, then, with at most $E_{\text{tr}} = \mathcal{O}(1/\epsilon^2)$ meta iterations, the Meta-CV learnt by Algorithm 1 can attain a solution γ_ϵ such that $\mathbb{E}[\|\mathbb{E}_{t\sim\rho}[\nabla\mathcal{J}_t(\gamma_\epsilon)]\|] = \mathcal{O}(\epsilon)$. The requirements on the step-size and batch size are inherited from Ji et al. [2022] and spelled out in Appendix A.1. The following corollary shows that, under strong convexity, γ_ϵ is close to a minimiser of the empirical variance computed on Q_t for any task $\mathcal{T}_t \in I$.

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

$$\mathbb{E}[\mathbb{E}_{t\sim\rho}[\|\gamma_\epsilon - \gamma_t^*\|_2]] \leq \tfrac{C_1}{\mu}\epsilon + \tfrac{C_2}{\mu},$$

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.

Proofs are presented in Appendix A. These results justify the use of Algorithm 1 to train the Meta-CV and task-specific CV, guaranteeing the convergence to a global minimiser of the training objective under the conditions of Corollary 1.1. In particular, they 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

We introduced Meta-CVs, an extension of existing CVs which brings the advantages of the meta-learning framework to the construction of CVs for variance reduction of MC estimators. 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 large neural networks. For example, it is $\mathcal{O}(p^2)$ when $L = 1$. This prevents us from using very large networks, which is likely to be essential for complex integrands. We therefore hope to study first-order or Hessian-free meta-learning algorithms [Fallah et al., 2020] in future work, in order to further reduce the cost to $\mathcal{O}(p)$.

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.

References

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

A. Alfonsi, B. Lapeyre, and J. Lelong. How many inner simulations to compute conditional expectations with least-square Monte Carlo? *arXiv:2209.04153*, 2022.

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.

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.

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*. PMLR, 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.

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.

H. Y. He and A. B. Owen. Optimal mixture weights in multiple importance sampling. *arXiv:1411.3954*, 2014.

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. *arXiv:2210.08329*, 2022.

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.

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

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.

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.

M. Riabiz, W. Chen, J. Cockayne, and P. Swietach. Optimal thinning of MCMC output. *J. R. Stat. Soc. Series B*, 2022.

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*, 2021.

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.