Vector-Valued Control Variates

Zhuo Sun[1], Alessandro Barp[2,3], François-Xavier Briol[1,3]

[1]University College London, [2]University of Cambridge, [3]The Alan Turing Institute.

Abstract

Control variates are post-processing tools for Monte Carlo estimators which can lead to significant variance reduction. This approach usually requires a large number of samples, which can be prohibitive for applications where sampling from a posterior or evaluating the integrand is computationally expensive. Furthermore, there are many scenarios where we need to compute multiple related integrals simultaneously or sequentially, which can further exacerbate computational costs. In this paper, we propose *vector-valued control variates*, an extension of control variates which can be used to reduce the variance of multiple integrals *jointly*. This allows the transfer of information across integration tasks, and hence reduces the overall requirement for a large number of samples. We focus on control variates based on kernel interpolants and our novel construction is obtained through a generalised Stein identity and the development of novel matrix-valued Stein reproducing kernels. We demonstrate our methodology on a range of problems including multifidelity modelling and model evidence computation through thermodynamic integration.

1 Introduction

One of the main challenges for computational statistics is the approximation of integrals. Examples in Bayesian statistics include the computation of posterior moments, the model evidence (or marginal likelihood), and Bayes factors. In frequentist statistics, this is also often required when working with models containing latent variables. These issues have lead to the development of a range of Monte Carlo (MC) methods; see Green et al. (2015) for a review.

In this paper, we will focus on integration for functions defined on some Euclidean space. Let $f : \mathbb{R}^d \to \mathbb{R}$ denote some integrand of interest, and Π some probability distribution with Lebesgue density π. The integration task we consider can be expressed as estimating

$$\Pi[f] := \int_{\mathbb{R}^d} f(x)\pi(x)dx \tag{1}$$

using evaluations of the integrand at some points over the domain $\mathbb{R}^d$: $\{x_i, f(x_i)\}_{i=1}^n$. These evaluations are combined to create an estimate of $\Pi[f]$; for example, when realisations are independent and identically distributed (IID), a MC estimator can be constructed as follows: $\hat{\Pi}^{\mathrm{MC}}[f] = \frac{1}{n}\sum_{i=1}^n f(x_i)$. In that case, assuming that f is square-integrable with respect to Π (i.e. $\Pi[f^2] < \infty$), we can use the central limit theorem (CLT) to show that such estimators converge to $\Pi[f]$ as $n \to \infty$, and this convergence is then controlled by the asymptotic variance of the integrand. See also Jones (2004) for extensions to Markov chain Monte Carlo (MCMC) realisations.

The main insight behind the concept of *control variate* (CV) is that it is often possible to adapt the integration problem so as to require a MC estimator of $\Pi[f - g]$ for some $g : \mathbb{R}^d \to \mathbb{R}$ instead of a MC estimator of $\Pi[f]$. This is justified if $\Pi[g]$ is known in closed form, in which case we may use

$$\hat{\Pi}^{\mathrm{CV}}[f] := \hat{\Pi}^{\mathrm{MC}}[f - g] + \Pi[g]$$

Furthermore, if g is chosen appropriately, the variance of $f - g$ will be much smaller than that of f, and a smaller number of samples will be required for the estimator to attain a given level of accuracy. The reader is referred to Si et al. (2021); South et al. (2022) for two recent reviews of this literature.

Suppose now that $n = (n_1, \ldots, n_T) \in \mathbb{N}_+^T$ ($T \in \mathbb{N}_+$) is a multi-index. In this paper, we will focus on cases where we have not one integration problem, but a whole sequence of integrands $f_t : \mathbb{R}^d \to \mathbb{R}$ and distributions Π_t for which we would like to use evaluations $\{x_{tj}, f_t(x_{tj})\}_{j=1}^{n_t}$ to estimate

$$\Pi_t[f_t] := \int_{\mathbb{R}^d} f_t(x)\pi_t(x)dx \quad \text{for } t \in [T], \tag{2}$$

where for $T \in \mathbb{N}_+$, $[T]$ denotes the set $\{1, \ldots, T\}$. Of course, the integration problems in (2) could be tackled individually, and this is in fact the common approach in practice. However, the main insight in this paper is that if both the integrands and distributions are related, then we can construct a CV to *jointly reduce the variance* of estimators for these integration problems and hence obtain a more accurate approximation. We will call such a function a *vector-valued control variate* (vv-CV). In order to encode the relationship between integration tasks, we will propose a flexible class of CVs based on interpolation in *reproducing kernel Hilbert space of vector-valued functions* (vv-RKHS). More precisely, we generalise existing constructions of Stein reproducing kernels to derive novel vv-RKHSs with the property that each output has mean zero.

We note that very few methods exist to tackle multiple integrals simultaneously. One exception is the work of Xi et al. (2018), which also proposes an algorithm based on interpolation in vv-RKHSs. However, that work is limited to a small number of cases where the integral of the kernel can be computed in closed form, which is rarely possible in practice. In constrast, our vv-CVs are applicable to a wide range of problems so long as $\nabla_x \log \pi_t$ can be evaluated pointwise for all $t \in [T]$ (where $\nabla_x = (\partial/\partial x_1, \ldots, \partial/\partial x_d)^\top$). This latter condition will usually be satisfied in Bayesian statistics, and is a requirement for the implementation of most gradient-based MCMC algorithms.

The paper is structured as follows. In Section 2, we review existing CVs based on Stein's method. In Section 3, we propose novel families of vv-CVs, then in Section 4 we show how to find an optimal element from these families. Finally, in Section 5, we demonstrate the advantage of constructing CVs which share information across tasks on some problems in multifidelity modelling and model evidence computation through thermodynamic integration.

Notation All vectors $x \in \mathbb{R}^d$ are column vectors, $\|x\|_q = (\sum_{i=1}^d x_i^q)^{1/q}$ for $q \in \mathbb{N}$, and $\mathbf{1}_d = (1, \ldots, 1)^\top$ is a d-dimensional vector. For a multi-index $m \in \mathbb{N}_+^d$, we will write $|m| = \sum_{i=1}^d m_i$ for its total degree. For a matrix $M \in \mathbb{R}^{p\times q}$, M_{ij} denote the element at location (i, j), its Frobenius norm is given by $\|M\|_F^2 = \sum_{i=1}^p \sum_{j=1}^q M_{ij}^2$, its trace is denoted $\mathrm{Tr}(M) = \sum_{i=1}^m M_{ii}$ and its pseudoinverse will be written as $M^\dagger$. Furthermore, I_m denotes the m-dimensional identity matrix, and S_+^m denotes the set of symmetric positive definite matrices in $\mathbb{R}^{m\times m}$. We denote by C^j the set of functions whose j^{th} derivative exists and is continuous, and given a differentiable function g on $\mathbb{R}^{d_1} \times \mathbb{R}^{d_2}$, $\partial_x^r g(x, y)$ denotes its partial derivative in the r^{th}-coordinate of its first entry evaluated at (x, y).

2 Background: Control Variates for a Single Task

We now review existing CVs for a single integration problem. The usual approach requires splitting the dataset $\{x_i, f(x_i)\}_{i=1}^n$ into two sets $\{x_i, f(x_i)\}_{i=1}^m$ and $\{x_i, f(x_i)\}_{i=m+1}^{m+n}$. There are then three steps: (i) the construction a candidate set of functions with known mean; (ii) the selection of an optimal function from this set using the first dataset $\{x_i, f(x_i)\}_{i=1}^m$; (iii) the implementation of a MC estimator using the second dataset $\{x_i, f(x_i)\}_{i=m+1}^{m+n}$. These three steps will now be introduced in Section 2.1, 2.2 and 2.3 respectively.

2.1 Spaces of Functions with Known Mean

The first step consists of constructing a set of functions $\mathcal{G}$ which all integrate to a known value against Π. This can generally be challenging since Π may not be computationally tractable. For example, in the case of posterior distributions, Π will not be tractable in the sense that π cannot be evaluated since it contains an unknown normalisation constant. Without loss of generality, we will discuss the construction of functions which integrate to zero, but notice that we can obtain functions with mean equal to any constant $\beta \in \mathbb{R}$ by simply adding this constant β to a zero-mean function.

2.1.1 Zero-Mean Functions through Stein's Operators

One way of constructing zero-mean functions is to use Stein's method, an analytical tool from probability theory which has recently been shown to be a convenient computational tool (Anastasiou et al., 2021). The main ingredients of Stein's method are a function class and an operator acting on this class. More precisely, a *Stein class* of Π is a class of functions $\mathcal{U}$ associated to an operator $\mathcal{S}$, called *Stein operator*, such that a *Stein identity* holds:

$$\Pi[\mathcal{S}[u]] = 0 \qquad \forall u \in \mathcal{U}.$$

An obvious choice for the class of zero-mean functions $\mathcal{G}$ is to consider all functions of the form $g = \mathcal{S}[u]$ for $u \in \mathcal{U}$. To ensure such functions have finite variance, we will now assume that all functions in $\mathcal{G}$ are square-integrable with respect to Π. This can usually be guaranteed under weak regularity conditions on $\mathcal{U}$ and $\mathcal{S}$, and we will get back to this later in Theorem 3.2. Note also that $\mathcal{S}$ (and $\mathcal{U}$) depend implictly on Π, but we only make this explicit in our notation when it is helpful for clarity in which case we write $\mathcal{S}_\Pi$.

The most common choice of Stein operator is the *first-order Langevin Stein operator*, which applies to (sufficiently regular) *vector-valued functions* (vv-functions) $u : \mathbb{R}^d \to \mathbb{R}^d$:

$$\mathcal{L}'[u](x) := \nabla_x \cdot u(x) + u(x) \cdot \nabla_x \log \pi(x). \tag{3}$$

The Langevin Stein operator can also be adapted for (sufficiently regular) scalar-valued functions $u : \mathbb{R}^d \to \mathbb{R}$, in which case it is called the *second-order Langevin Stein operator*:

$$\mathcal{L}''[u](x) := \Delta_x u(x) + \nabla_x u(x) \cdot \nabla_x \log \pi(x), \tag{4}$$

where $\Delta_x = \nabla_x \cdot \nabla_x$. The main advantage of these operators is that they only require knowledge of Π through evaluations of $\nabla_x \log \pi$, which does not require any knowledge of normalisation constants of π. Indeed, let $\pi = \tilde{\pi}/C$ for some unknown $C \in \mathbb{R}$, then $\nabla_x \log \pi = \nabla_x \log \tilde{\pi}$. For more general Stein operators, we refer readers to the review of Anastasiou et al. (2021).

2.1.2 Parametric Spaces

For practical convenience, it is common to take $\mathcal{G}$ to be a parametric space and we will hence write it $\mathcal{G}_\Theta$, where Θ denotes the space of possible parameter values. Most existing CVs can be obtained by taking $g_\theta = \mathcal{S}[u_\theta]$ for some $u_\theta \in \mathcal{U}_\Theta$ where $\mathcal{U}_\Theta$ is a parametric class of functions. However, note that there might not be a unique u_θ leading to g_θ. For the remainder of the paper, θ will hence be a parameter indexing g_θ directly as opposed to an element of the Stein class.

A first example of parametric CVs are the polynomial-based CVs of Mira et al. (2013) (see also Assaraf and Caffarel (1999); Papamarkou et al. (2014); Oates et al. (2016); Belomestny et al. (2020)), in which case the Stein class is parametrised directly by coefficients of a polynomial. Let $\alpha = (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}^d$ denote a multi-index; polynomials of order l take the form $u_\theta(x) = \sum_{|\alpha| \leq l} \theta_\alpha x^\alpha = \sum_{|\alpha| \leq l} \theta_\alpha x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_d^{\alpha_d}$, where $\theta_\alpha \in \mathbb{R}\ \forall \alpha$ with $|\alpha| \leq t$, and can be combined with $\mathcal{L}''$. This approach is usually used only with first or second order polynomials as the number

of coefficients quickly becomes too large for higher-order polynomials, especially when d is large. However, recent work also studies sparse polynomials, see South et al. (2019).

A second example are kernel interpolants, and these will be the main focus of our paper. Technically, this class is nonparametric, but it is often convenient for implementation to fix the dataset size and parametrise it. Let $\mathcal{H}_k$ denote a reproducing kernel Hilbert space (RKHS) with reproducing kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ (Berlinet and Thomas-Agnan, 2011), so that k is symmetric ($k(x, y) = k(y, x)\ \forall x, y \in \mathbb{R}^d$) and positive definite ($\forall m \in \mathbb{N}_+$, $\sum_{i,j=1}^m c_i c_j k(x_i, x_j) \geq 0\ \forall c_1, \ldots, c_m \in \mathbb{R}$ and $\forall x_1, \ldots, x_m \in \mathbb{R}^d$). The kernel could be a squared-exponential kernel $k(x, y) = \exp(-\|x - y\|_2^2/2l^2)$, where $l > 0$ is known as the lengthscale, or a polynomial kernel $k(x, y) = (x^\top y + c)^l$ where $c \in \mathbb{R}$ and $l \in \mathbb{N}$ is the degree of the polynomial. Oates et al. (2017) noticed that, under mild regularity conditions, the image of $\mathcal{U} = \mathcal{H}_k^d := \mathcal{H}_k \times \ldots \times \mathcal{H}_k$ under $\mathcal{L}'$ is a RKHS with kernel

$$\begin{aligned} k_0(x, y) := \nabla_x \cdot \nabla_y k(x, y) + \nabla_x \log \pi(x) \cdot \nabla_y k(x, y) \\ + \nabla_y \log \pi(y) \cdot \nabla_x k(x, y) + (\nabla_x \log \pi(x) \cdot \nabla_y \log \pi(y)) k(x, y). \end{aligned} \tag{5}$$

Given m observations, it is known that the optimal interpolant will be of the form $g_\theta(x) = \sum_{i=1}^m \theta_i k_0(x, x_i)$ where $\theta_i \in \mathbb{R}$ for all $i \in [m]$. This therefore provides a natural parametrisation for practical implementation. Oates et al. (2017) called this class of CVs *control functionals* (CF); see also Briol et al. (2017); Oates et al. (2019); South et al. (2021) for more details. An optimal value for these parameters can be obtained either through optimisation, or in closed form; see Oates et al. (2017). The operator $\mathcal{L}''$ could also be used instead of $\mathcal{L}'$ (Barp et al., 2021a), or any other generator of measure-preserving diffusion (Gorham et al., 2019; Barp et al., 2021b, Sec. 5).

We remark that CFs are closely related to the Bayesian quadrature (BQ) algorithm (O'Hagan, 1991; Briol et al., 2019), which uses Gaussian process models of the integrand to estimate integrals. This connection provides a natural way of optimising kernel hyperparameters, for example through maximum likelihood estimation. We will return to this in the context of vv-CVs in Appendix C.

The last example is to use a space of neural networks; see Wan et al. (2019); Si et al. (2021). This approach can be advantageous due to the flexibility of the underlying function class, but is much more challenging to implement because selecting a CV from this class is a non-convex problem.

2.2 Selecting a Control Variate

In order to select CVs, we will pick the "best" element from $\mathcal{G}_\Theta$, where different definitions of "best" lead to different objectives. The most common objective and the subject of this paper is the variance of the integrand minus the CV:

$$J(\theta) = \mathbb{V}_\Pi[f - g_\theta] := \Pi[(f - g_\theta - \Pi[f])^2], \tag{6}$$

Alternative objectives include the variance of the randomised quasi-Monte Carlo or MCMC CLT; see e.g. Hickernell et al. (2005); Oates and Girolami (2016); Dellaportas and Kontoyiannis (2012); Mijatovic and Vogrinc (2018). Following the framework of empirical risk minimisation, the MC variance objective can be approximated with $\{x_j, f(x_j)\}_{j=1}^m$ as follows:

$$J_m(\theta, \beta) = \tfrac{1}{m} \textstyle\sum_{j=1}^m (f(x_j) - g_\theta(x_j) - \beta)^2 + \lambda \|g_\theta\|^2,$$

where $\beta \in \mathbb{R}$ is an additional parameter which tends to $\Pi[f]$ as $m \to \infty$, $\lambda \geq 0$ is a regularisation parameter, and $\|g_\theta\|$ can be any norm suitable for regularisation. For example, $\|g_\theta\| = \|\theta\|_2$ or $\|g_\theta\| = \|g_\theta\|_{\mathcal{H}_k}$ for some kernel k. Assuming that $\Theta \subseteq \mathbb{R}^p$, this objective can then be minimised through the solution to a linear system when $\theta \mapsto g_\theta$ is linear and $\theta \mapsto \|g_\theta\|^2$ is quadratic. In more general cases, it can be minimised using stochastic optimisation (Si et al., 2021). In that case, we initialise $\theta^{(0)}$ and $\beta^{(0)}$, then iteratively take gradient steps with minibatches of size $\tilde{m} << m$.

2.3 The Control Variate Estimator

Once a function $g_{\hat{\theta}}$ has been selected, we can construct an MC estimator on $f - g_{\hat{\theta}}$. This can be done using the remainder of the data $\{x_i, f(x_i)\}_{i=m+1}^{m+n}$:

$$\hat{\Pi}^{\text{CV}}[f] = \hat{\Pi}^{\text{MC}}[f - g_{\hat{\theta}}] + \Pi[g_{\hat{\theta}}] = \frac{1}{n}\sum_{i=m+1}^{m+n}(f(x_i) - g_{\hat{\theta}}(x_i)) + \Pi[g_{\hat{\theta}}]. \tag{7}$$

The question of how to select m is of practical importance for the quality of the estimator. When m is small, most of the function evaluations are used for the MC estimator of $f - g_{\hat{\theta}}$, whereas when m is large, most of the evaluations are used to construct the CV. In some cases, it might be preferable to use the entire dataset to find a good CV and estimate $\Pi[f]$ directly using $\Pi[g_{\hat{\theta}}]$. This will lead to a biased estimator (for finite m), but this estimator will be much more accurate than when splitting the dataset if the problem of functional approximation can be solved at a faster rate than the MC convergence rate.

3 Constructing Vector-Valued Control Variates

In this paper, we extend CVs to the case where multiple related integration problems are tackled simultaneously or sequentially. This will be done through a multi-task learning approach (Micchelli and Pontil, 2005; Evgeniou et al., 2005), where each integral corresponds to a task and the relationship between tasks will be modelled explicitly. This will allow us to share information across integration problems, and hence improve accuracy when the number of integrand evaluations is limited.

3.1 Constructing Vector-Valued Functions using Stein's Method

For the remainder of this paper, we consider integrands corresponding to the outputs of a vector-valued function $f : \mathbb{R}^d \to \mathbb{R}^T$ so that $f(x) = (f_1(x), \ldots, f_T(x))^\top$. Our objective is to approximate the vector $\Pi[f] = (\Pi_1[f_1], \ldots, \Pi_T[f_T])^\top$, and shall assume that f_t is square-integrable with respect to Π_t $\forall t \in [T]$. Formally, Π is known as a *vector probability distribution*. To approximate $\Pi[f]$, we will construct a class of zero-mean vv-functions through Stein's method. This can be done by considering a Stein class $\mathcal{U}$ whose image $\mathcal{G}$ under the Stein operator $\mathcal{S}^{\text{vv}} : \mathcal{U} \to \mathcal{G}$ is a class of $\mathbb{R}^T$-valued functions on $\mathbb{R}^d$. We will need a generalised form of Stein identity for vv-functions:

$$\Pi_t[g_t] = \Pi_t[(\mathcal{S}^{\text{vv}}[u])_t] = 0 \qquad \forall u \in \mathcal{U} \text{ and } \forall t \in [T]. \tag{8}$$

In other words, each output of the vv-function should integrate to zero against the corresponding probability distribution. Of course, the ordering of the sequence of integrands and distributions matters here, as we do not guarantee that $\Pi_t[g_{t'}] = 0$ for $t \neq t'$.

The property above can be obtained by constructing an operator $\mathcal{S}^{\text{vv}}$ through a sequence of Stein operators $\mathcal{S}^{\text{sv}}_{\Pi_t}$ for $t \in [T]$ whose images are scalar-valued functions integrating to zero under Π_t. These can then be applied in an element-wise fashion as follows

$$g = \mathcal{S}^{\text{vv}}[u] = (\mathcal{S}^{\text{sv}}_{\Pi_1}[u_1], \ldots, \mathcal{S}^{\text{sv}}_{\Pi_T}[u_T])^\top. \tag{9}$$

Once again, $\mathcal{G}$ can be parametrised and we will denote it $\mathcal{G}_\Theta$. We can then use an objective based on the variances of all these integrals to select an optimal element:

$$J^{\text{vv}}(\theta) = \|\mathbb{V}_\Pi[f - g_\theta]\|^2 = \|\Pi[(f - g_\theta - \Pi[f])^2]\|^2, \tag{10}$$

where $g_\theta \in \mathcal{G}_\Theta$. In the above $\mathbb{V}_\Pi$ should be thought of as applying $\mathbb{V}_{\Pi_t}$, the variance under Π_t, to the t^{th} element of the vv-function. The norm could be any norm on $\mathbb{R}^T$, but we will usually make use of the 1-norm so as to interpret this objective as the sum of variances on each integrand. For this objective to make sense, we require $(g_\theta)_t$ to be squared-integrable with respect to Π_t $\forall t \in [T]$.

Let $m = (m_1, \ldots, m_T) \in \mathbb{N}_+^T$. Once again, the objective can be approximated via MC estimates based on the dataset $\mathcal{D} = \{\{x_{1j}, f_1(x_{1j})\}_{j=1}^{m_1}, \ldots, \{x_{Tj}, f_T(x_{Tj})\}_{j=1}^{m_T}\}$ following the framework of empirical risk minimisation. For example, when the norm above is a 1-norm, the objective is simply the sum of individual variances:

$$L_m^{\text{vv}}(\theta, \beta) := J_m^{\text{vv}}(\theta, \beta) + \lambda \|g_\theta\|^2 = \sum_{t=1}^T \frac{1}{m_t} \sum_{j=1}^{m_t} (f_t(x_{tj}) - (g_\theta(x_{tj}))_t - \beta_t)^2 + \lambda \|g_\theta\|^2, \qquad (11)$$

where $\lambda \geq 0$ and now $\beta = (\beta_1, \ldots, \beta_T) \in \mathbb{R}^T$. Once again, the second term is used to regularise J_m^{vv}, but the norm acts on vv-functions. For $\mathcal{U}$, we could take a class of polynomials, kernels or neural networks, but will usually require flexible classes of vv-functions in order to encode relationships between tasks. Assuming that each Π_t has a C^1 and strictly positive density π_t with respect to the Lebesgue measure, we will also be able to use the Langevin Stein operators $\mathcal{L}'$ and $\mathcal{L}''$, which only require access to the score functions for each distribution $\Pi_1, \ldots, \Pi_T$. For this purpose, we now define $l : \mathbb{R}^d \to \mathbb{R}^{T \times d}$ to be the matrix-valued function with entries $l_{ij}(x) = \partial^j \log \pi_i(x)$.

3.2 Kernel-based Vector-Valued Control Variates

The main choice of Stein class $\mathcal{U}$ studied in this paper is vv-RKHSs (Carmeli et al., 2006, 2010; Álvarez et al., 2012). This choice is particularly convenient as it allows us to build on the rich literature in statistical learning theory which considers kernels encoding relationships between tasks, usually in the context of regression or classification. A main contribution of this section will be the design of novel Stein reproducing kernels specifically for numerical integration, a task not commonly tackled in statistical learning theory.

A vv-RKHS $\mathcal{H}_K$ is a Hilbert space of functions mapping from $\mathbb{R}^d$ to $\mathbb{R}^T$ with an associated *matrix-valued reproducing kernel* (mv-kernel) $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{T \times T}$ which is symmetric ($K(x, y) = K(y, x)^\top$ $\forall x, y \in \mathbb{R}^d$) and positive definite ($\forall m \in \mathbb{N}_+$, $\sum_{i,j=1}^m c_i^\top K(x_i, x_j) c_j \geq 0$ for all $c_1, \ldots, c_m \in \mathbb{R}^T$ and $x_1, \ldots, x_m \in \mathbb{R}^d$). Any vv-RKHS satisfies a reproducing property, so that $\forall f \in \mathcal{H}_K$, $f(x)^\top c = \langle f, K(\cdot, x)c \rangle_{\mathcal{H}_K}$ and $K(\cdot, x)c \in \mathcal{H}_K$ $\forall x \in \mathbb{R}^d$ and $\forall c \in \mathbb{R}^T$. The reproducing kernels discussed in Section 2 are a special case which can be recovered when $T = 1$. We say that K is $C^{r,r}(\mathbb{R}^d \times \mathbb{R}^d)$ provided that $\partial_x^\alpha \partial_y^\alpha K(x, y)$ exists and is continuous for all multi-indices $\alpha = (\alpha_1, \ldots, \alpha_d)$ with $\alpha_1 + \cdots + \alpha_d \leq r$, and that K is bounded with bounded derivatives if there exists $C \in \mathbb{R}$ for which $\|K(x, y)\|_F \leq C$ and $\|\partial_x^r \partial_y^r K(x, y)\|_F \leq C$ for all $x, y \in \mathbb{R}^d$.

To construct vv-CVs, one natural approach is to construct a mv-kernel K_0 using another mv-kernel K and an operator $\mathcal{S}^{\text{vv}}$. Then, assuming we have access to such a kernel K_0 and to data $\mathcal{D}$, a natural generalisation of the scalar valued case is:

$$g_\theta(x) = \sum_{t=1}^T \sum_{j=1}^{m_t} \theta_{tj}^\top K_0(x, x_{tj}), \qquad \text{where } \theta_{tj} \in \mathbb{R}^T \text{ for all } t \in [T], j \in [m_t]. \qquad (12)$$

The remainder of this section will hence focus on how to obtain K_0. Our construction is based on the CFs of Oates et al. (2017, 2019), which use a scalar-valued kernel k_0 obtained through a tensor product structure $\mathcal{U} = \mathcal{H}_k^d$. The initial motivation for introducing vv-functions in this setting was that $\mathcal{L}'$ requires inputs which are vv-functions. This approach can be thought of as introducing a dummy dimension for convenience, and is in no way related to the multitask setting in this paper. We now illustrate the natural extension of k_0 to a mv-kernel.

Theorem 3.1. *Consider $\mathcal{H}_K$ which is a vv-RKHS with mv-kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{T \times T}$, and suppose that $K \in C^{1,1}(\mathbb{R}^d \times \mathbb{R}^d)$. Furthermore, for suitably regular functions $u_t : \mathbb{R}^d \to \mathbb{R}^d$ for $t \in [T]$, let $u = (u_1, \ldots, u_T)$ and*

$$\mathcal{S}^{vv}[u] = (\mathcal{L}'_{\Pi_1}[u_1], \ldots, \mathcal{L}'_{\Pi_T}[u_T])^\top.$$

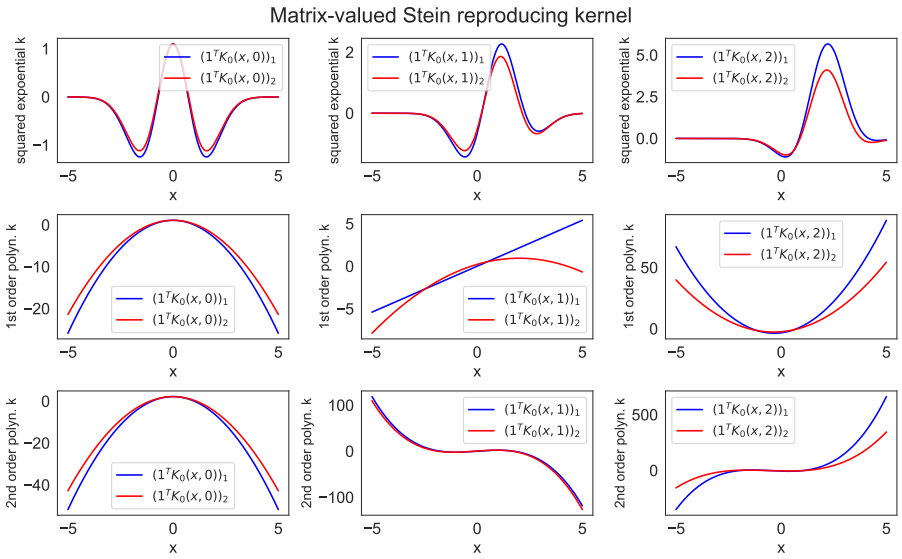

Figure 1: Illustration of a separable mv-kernel K_0 for $T = 2$ through projections with $\mathbf{1} = (1, 1)$. Here, $\Pi_1 = \mathcal{N}(0, 1)$, $\Pi_2 = \mathcal{N}(0, 1.25)$, $B_{11} = B_{22} = 1$ and $B_{12} = B_{21} = 0.1$. The first row corresponds to taking k to be a squared-exponential kernel, whereas the second and third row correspond to taking a polynomial kernel $k(x, y) = (x^\top y + 1)^l$ with $l = 1$ and $l = 2$ respectively.

Then, the image of $\mathcal{H}_K^d = \mathcal{H}_K \times \ldots \times \mathcal{H}_K$ *under* $\mathcal{S}^{vv}$ *is a vv-RKHS with kernel* $K_0 : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{T \times T}$*:*

$$(K_0(x,y))_{tt'} = \sum_{r=1}^d \partial_x^r \partial_y^r K(x,y)_{tt'} + l_{t'r}(y)\partial_x^r K(x,y)_{tt'} + l_{tr}(x)\partial_y^r K(x,y)_{tt'} + l_{tr}(x) l_{t'r}(y) K(x,y)_{tt'} \qquad \forall t, t' \in [T].$$

The proof of this theorem is in Appendix A.1. Notice that $(K_0(x, y))_{tt'}$ depends only on the score function of Π_t and $\Pi_{t'}$, and so we (once again) do not require knowledge of normalisation constants of the corresponding densities. However, in order to evaluate $K_0(x, y)$ for some $x, y \in \mathbb{R}^d$, we will require knowledge of $\nabla_x \log \pi_t(x)$ and $\nabla_y \log \pi_t(y)$ for all $t \in [T]$. Finally, another interesting point is that $(K_0(x, y))_{tt'}$ is a sv-kernel when $t = t'$, but this is not the case for $t \neq t'$ since it is not symmetric in that case.

In order to use elements of this new RKHS as vv-CVs, we will require that the objective in (10) is well-defined. This can be guaranteed when the elements of the RKHS are square-integrable, and the theorem below, proved in Appendix A.2, provides sufficient conditions for this to hold.

Theorem 3.2. *Suppose that* K *is bounded with bounded derivatives, and* $\Pi_t[\|\nabla_x \log \pi_t\|_2^2] < \infty$ *for all* $t \in [T]$*. Then, for any* $g \in \mathcal{H}_{K_0}$*,* g_t *is square-integrable with respect to* Π_t *for all* $t \in [T]$*.*

Now the mv-kernel in Theorem 3.1 takes a very general form as it has minimal requirements on K or $\Pi_1, \ldots, \Pi_T$. This is convenient as it can be applied in a wide range of settings, but this generality comes at the cost of computational complexity. We will now study several special cases which will often be sufficient for applications.

Special Case I: Separable kernel K For simplicity, the literature on multi-task learning often focuses on the case of *separable kernels*. We say a mv-kernel K is separable if it can be written as $K(x, y) = Bk(x, y)$, where k is a scalar valued kernel and $B \in S_+^T$. The advantage of this formulation is that it decouples the model for individual outputs (as given by k) from the model

of their relationship, as given by the components of the matrix B, which can be thought of as a covariance matrix for tasks. As we will see in Section 4.3, this can be particularly advantageous for selecting the hyperparameters of vv-CVs. Using such a kernel K, the kernel K_0 in Theorem 3.1 becomes:

$$(K_0(x,y))_{tt'} = B_{tt'} \sum_{r=1}^{d} \partial_x^r \partial_y^r k(x,y) + l_{t'r}(y)\partial_x^r k(x,y) \\ + l_{tr}(x)\partial_y^r k(x,y) + l_{tr}(x) l_{t'r}(y) k(x,y) \qquad t,t' \in [T]. \tag{13}$$

Note that although K is separable, this is not the case for K_0. This is because t and t' enter the expression not only through indices of the matrix B, but also through indices of the function l. The expression in (13) is interesting because it reduces the choice of K to the choice of a matrix B and a kernel k, and this matrix B has a natural interpretation.

An illustration of such K_0 is provided in Figure 1 for the case $T = 2$. As observed, the choice of kernel k can have significant impacts on K_0. Moreover, K_0 possesses a well-known property of Stein kernels: even when k is translation-invariant (see the top row) this may not be the case for K_0. This is due to the fact that K_0 depends on l. Finally, we can also observe that the two outputs of $1^\top K_0(x,y)$ are always closely related, a property which will be key when it comes to vv-CVs.

Special Case II: Separable kernel K with a single target distribution A further simplification of the kernel is possible when both K is separable and all distributions are the same: $\Pi_1 = \cdots = \Pi_T \equiv \Pi$. In this case K_0 itself becomes a separable kernel of the form:

$$(K_0(x,y))_{tt'} = B_{tt'} k_0(x,y) \qquad \forall t,t' \in [T]. \tag{14}$$

where k_0 is given in (5). The scalar case can then be recovered by taking $T = 1$ and $B = 1$.

Before concluding this section, we remark on the connections between our work and the multi-output BQ algorithm first introduced by Xi et al. (2018) (and further studied by Karvonen et al. (2019); Gessner et al. (2020)). In the same way as the kernel k_0 introduced for kernel-based CVs in Oates et al. (2017) can be used in a BQ framework, our mv-kernels K_0 could be used in a multi-output BQ framework. However, the original multi-output BQ work is limited to a single target distribution Π which has to be relatively simple (such as a Gaussian or uniform) as otherwise the method becomes computationally intractable. Our novel kernel allows for multiple distributions and only requires evaluations of score functions $\nabla_x \log \pi_t(x)$, which greatly extends the class of problems the method can tackle.

3.3 Alternative Constructions

Other choices of Stein operators for vv-functions can be made, and a more general construction is presented in Appendix B. This general formulation is able to recover the Stein reproducing kernels of Barp et al. (2021a). Whilst this construction is strictly more general, it is also more complex and leads to higher computational costs, which is why we focused setting described in Theorem 3.1.

Although kernels are a natural way of constructing functions for multi-task problems, it is also possible to generalise constructions based on other parametric families such as polynomials or neural networks. We will not explore this avenue in detail in the present paper, but now provide brief comments on how such generalisations could be obtained.

Firstly, u_θ could be based on any additive model such as a polynomial or wavelet expansion. In that case, it is straightforward to construct vv-CVs with a separable structure as follows:

$$(u_\theta(x))_t = \sum_i \sum_{t'=1}^{T} B_{tt'} \theta_i \phi_i(x), \quad (g_\theta(x))_t = \sum_i \sum_{t'=1}^{T} B_{tt'} \theta_i \mathcal{S}^{\mathrm{sv}}_{\Pi_t}[\phi_i(x)] \quad \forall t \in [T], \tag{15}$$

where $B \in S_+^T$ and $\phi_i : \mathbb{R}^d \to \mathbb{R}$ is a (sufficiently regular) basis function. In particular, taking the basis functions to be of the form x^α for $\alpha \in \mathbb{N}^d$ recovers the polynomial-based CVs of Mira et al.

(2013). We also note that any model of this form leads to a quadratic MC variance objective, whose solution can be obtained in closed form under mild regularity conditions on the basis functions.

Secondly, we could use non-linear models for u_θ. In that case, one approach would be to use a separable structure of the form:

$$(u_\theta(x))_t = \sum_{t'=1}^T B_{tt'}\phi_\theta(x), \quad (g_\theta(x))_t = \sum_{t'=1}^T B_{tt'}\mathcal{S}^{\text{sv}}_{\Pi_t}[\phi_\theta(x)] \quad \forall t \in [T]. \tag{16}$$

where $\phi_\theta(x)$ is a non-linear function of the parameters θ. The above is a generalisation of the neural networks-based CVs of Wan et al. (2019); Si et al. (2021) whenever ϕ_θ is a neural network. Unfortunately the MC variance objective will usually be non-convex in those cases, and we therefore have no guarantees of recovering the optimal parameter value when using most numerical optimisers.

4 Selecting a Vector-Valued Control Variate

In the previous section, we introduced our novel RKHS of zero-mean vv-functions. We will now derive a closed-form expression for the optimal parameters of these kernel-based vv-CVs in Section 4.1. However, the closed-form solution will be computationally intractable in most applications and we hence propose to use stochastic optimisation to minimise the objective in (11). First, in Section 4.2, we propose a strategy for selecting θ and β which assumes that a good choice of B is known a-priori. Then, in Section 4.3 we extend our proposed approach to the case where B is unknown and needs to be inferred from data. Finally, Section 4.4 discusses the computational complexity of our approaches.

4.1 Closed-form Solutions

We start with a theorem which provides a result akin to the RKHS representer theorem, but which focuses specifically on the function which minimises the objective in (11). This theorem shows that there exists a unique parameter minimising the variance objective. See Appendix A.3 for the proof.

Theorem 4.1. *Given $\mathcal{D} = \{\{x_{1j}, f_1(x_{1j})\}_{j=1}^{m_1}, \ldots, \{x_{Tj}, f_T(x_{Tj})\}_{j=1}^{m_T}\}$, the function which minimises the objective in* (11) *where $\|g_\theta\| := \|g_\theta\|_{\mathcal{H}_{K_0}}$ and $\beta \in \mathbb{R}^T$ is of the form:*

$$g_\theta(x) = \sum_{t=1}^T \sum_{j=1}^{m_t} \theta_{tj}^\top K_0(x, x_{tj}), \qquad \theta_{tj} \in \mathbb{R}^T \ \forall t \in [T], j \in [m_t],$$

with optimal parameter θ^* given by the solution of this convex linear system of equations:

$$\sum_{t'=1}^T \sum_{j'=1}^{m_{t'}} \Big(\sum_{t=1}^T \tfrac{1}{m_t} \sum_{j=1}^{m_t} K_0(x_{t''j''}, x_{tj})_{\cdot t} K_0(x_{tj}, x_{t'j'})_{t\cdot} + \lambda K_0(x_{t''j''}, x_{t'j'}) \Big) \theta^*_{t'j'}$$
$$= \sum_t^T \tfrac{1}{m_t} \sum_{j=1}^{m_t} K_0(x_{t''j''}, x_{tj})_{\cdot t}(f_t(x_{tj}) - \beta_t), \qquad \forall t'' \in [T], j'' \in [m_T].$$

When K_0 is strictly positive definite, then the system is strictly convex and θ^* is unique.

Theorem 4.1 guarantees that there is a unique minimiser of the objective in (11) in our vv-RKHS, and that this minimiser takes the form in (12) for some value of θ. It also guarantees that L_m^{vv} is a convex objective. The form of this system of equations simplifies significantly when the points at which $f_1, \ldots, f_T$ are evaluated are the same, or when $\Pi_1 = \cdots = \Pi_T$, however the applications considered in Section 5 require this general form.

Theorem 4.1 assumes that β is known and fixed, which may not be the case in practice. However, given a fixed g_θ the objective (11) is a quadratic in β with the optimal value

$$\beta_t^* = \tfrac{1}{m_t} \sum_{j=1}^{m_t} f_t(x_{tj}) - (g_\theta(x_{tj}))_t.$$

The above naturally leads to the use of block coordinate descent approaches. This could be either directly implemented using the closed-form solutions, or through the use of numerical optimisers such as the stochastic optimisation approaches we will now present. The next two sections highlight how this can be implemented for special case I and II.

Algorithm 1: Stochastic optimisation for vv-CVs with known task relationship

Input: $\mathcal{D}$, $\tilde{m}$, L, λ, $\beta^{(0)}$ and $\theta^{(0)}$.

for *iterations l from 1 to L* **do**

  Select a mini-batch $\mathcal{D}_{\tilde{m}}$ of size $\tilde{m}$.

  $\left(\theta^{(l)}, \beta^{(l)}\right) \leftarrow \text{Update}_{\theta,\beta}(\theta^{(l-1)}, \beta^{(l-1)}, B; \mathcal{D}_{\tilde{m}})$.

Output: Return $\theta^{(L)}$, $\beta^{(L)}$.

4.2 Stochastic Optimisation with a Known Task Relationship

Our first approach will assume that $B \in S_+^T$ is known. The proposed algorithm is a stochastic optimisation algorithm which we run for L time steps; pseudo-code is provided in Algorithm 1. We propose to initialise the algorithm at $\theta^{(0)} = (0, \ldots, 0) \in \mathbb{R}^p$, since this is equivalent to having $g_\theta(x) = 0$ (i.e. having no CV) for both the kernel- and polynomial-based vv-CVs. We also suggest initialising the parameter $\beta \in \mathbb{R}^T$ with any estimate of $\Pi[f]$. This is a natural initialisation since we expect β_t to equal $\Pi[f_t]$ for all $t \in [T]$ when $m_1, \ldots, m_T \to \infty$. For example, when the data consists of IID realisations from $\Pi_1, \ldots, \Pi_T$, a natural initialisation point is $\beta^{(0)} = (\hat{\Pi}_1^{\text{MC}}[f_1], \ldots, \hat{\Pi}_T^{\text{MC}}[f_T])^\top$.

For each iteration, we take mini-batches of size $\tilde{m} \in \mathbb{N}_+$ where $|\tilde{m}| \leq |m|$. Note here that $\tilde{m} = (\tilde{m}_1, \ldots, \tilde{m}_T)^\top$ is a multi-index giving the size of the mini-batch for each of the T datasets. This formulation allows for the use of different datasets across integrands, but also different mini-batch sizes for each task (which may be useful if the datasets are of different size for each integrand). An epoch consists of having gone through all data points for all T tasks, and we randomly shuffle the indices for mini-batches after each epoch. As default, we propose to take $\tilde{m}_t \propto m_t / (\sum_{t=1}^T m_t)$ for all $t \in [T]$. This choice guarantees that the number of samples for each integrand in the mini-batches is proportional to the proportion of samples for that integrand in the full dataset.

Once a mini-batch has been selected, we update our current estimate of the parameters θ and β using steps based on the gradient of our loss function: $\nabla_{(\theta,\beta)} L_{\tilde{m}}^{\text{vv}}(\theta, \beta)$. The pseudo-code in Algorithm 1 presents this abstractly as a function $\text{Update}_{\theta,\beta}(\theta, \beta, B; \mathcal{D})$, which takes in the current estimates of the parameters, the value of B and the dataset (or minibatch) used for the update; this is because different choices of vv-CVs might benefit from different updates. For example, pre-conditioners for the gradients could be used when readily available, or when these can be estimated from data. In Section 5, we will exclusively be using the Adam optimiser (Kingma and Ba, 2015), a first order method with estimates of lower-order moments.

For the penalisation term, it would be natural to take the RKHS norm $\|g_\theta\| = \|g_\theta\|_{\mathcal{H}_{K_0}}$ since this would lead to the objective used in Theorem 4.1. However, this can be impractical from a computational viewpoint since this norm depends on kernel evaluations for all of the training points. For this reason, we follow the recommendation of Si et al. (2021) and use instead the Euclidean norm: $\|g_\theta\| = \|\theta\|_2$. This still leads to a convex objective since the objective remains quadratic in θ.

Algorithm 1 is a natural approach to minimising our objective since our kernel-based vv-CVs are linear in θ and L_m^{vv} is convex in (θ, β). Many stochastic optimisation methods, such as stochastic gradient descent, will hence converge to a global minimum under regularity conditions (Bottou et al., 2018). However, note that Algorithm 1 naturally applies to other vv-CVs whether linear or not.

4.3 Stochastic Optimisation with an Unknown Task Relationship

We now extend our approach to account for simultaneously estimating β, θ, but also the matrix B. In this setting, we will use the same objective as in (11), but penalised by the norm of B:

$$\bar{L}_m^{\text{vv}}(\theta, \beta, B) = J_m^{\text{vv}}(\theta, \beta, B) + \lambda \|g_\theta\|^2 + \|B\|^2, \tag{17}$$

Algorithm 2: Block-coordinate descent for vv-CVs with unknown task relationship

Input: $\mathcal{D}, \tilde{m}, L, \lambda, \beta^{(0)}, \theta^{(0)}$ and $B^{(0)}$.

for *iterations* l *from* 1 *to* L **do**

  Select a mini-batch $\mathcal{D}_{\tilde{m}}$ of size $\tilde{m}$.

  $\left(\theta^{(l)}, \beta^{(l)}\right) \leftarrow \text{Update}_{\theta,\beta}(\theta^{(l-1)}, \beta^{(l-1)}, B^{(l-1)}; \mathcal{D}_{\tilde{m}})$.

  $B^{(l)} \leftarrow \text{Update}_{B}\left(\theta^{(l)}, \beta^{(l)}, B^{(l-1)}; \mathcal{D}_{\tilde{m}}\right)$.

Output: Return $\theta^{(L)}$, $\beta^{(L)}$ and $B^{(L)}$.

We now use $J_m^{\text{vv}}(\theta, \beta, B)$ to denote $J_m^{\text{vv}}(\theta, \beta)$ in order to emphasise the dependence on B. A second regularisation parameter is unnecessary as this would be equivalent to rescaling k. The objective in (17) is a natural extension to (11) and can be straightforwardly minimised through stochastic optimisation; see Algorithm 2. This algorithm combines gradient steps in (θ, β) and gradient steps in B. The latter is again presented abstractly as function $\text{Update}_B(\theta, \beta, B; \mathcal{D})$ and will be based on the gradient of the objective function with respect to elements of the matrix B: $\nabla_B \bar{L}_{\tilde{m}}^{\text{vv}}(\theta, \beta, B)$. The initial values $\theta^{(0)}$ and $\beta^{(0)}$ can be chosen according to the recommendations in the previous section, and we suggest initialising $B^{(0)} = I_T$ as this corresponds to having no relationship across tasks. To ensure B is stricly positive definite, we can take $B = LL^\top$, where L is a lower triangle matrix with diagonal elements forced to be greater than zero via an exponential transformation.

Algorithm 2 is straightforward to implement, but we cannot provide any guarantees of convergence to a global minimum due to its lack of convexity. Even in the case where g_θ is linear such as for kernel-based vv-CVs, the objective will be convex in (θ, β) or in B, but not jointly in all three inputs.

4.4 Computational Complexity

Although vv-CVs can be beneficial from an accuracy viewpoint, they also come with an increased computational cost, especially when T is large. The decision as to whether they should be used will therefore depend on the computational budget available. In particular, when integrand or score function evaluations are computationally expensive, the higher cost of using vv-CVs may be negligible in comparison. Table 1 provides the computational complexity of all approaches considered in the paper. We notice the impact of T, the number of tasks. In the case of existing kernel-based CVs, the dependence is $\mathcal{O}(T)$, but when using vv-CVs, this becomes between $\mathcal{O}(T^5)$ and $\mathcal{O}(T^6)$.

The table also highlights the difference in computational complexity between obtaining closed form solutions of θ and β by solving the linear system of equations in Theorem 4.1, and using the stochastic-optimisation approaches in Section 4.2 and Section 4.3. Here, the difference is mainly in terms of powers of m_t and $\tilde{m}_t$. As we can see the exact solutions usually come with an $\mathcal{O}(m_t^3)$ cost, whereas the stochastic optimisation approach is associated with a $\mathcal{O}(m_t^2 \tilde{m}_t L)$ cost. In this case, whenever $\tilde{m}_t L$ is small relative to m_t, this will lead to computational gains.

5 Experimental Results

We now illustrate our method on a range of problems with related integration tasks, including multi-fidelity models and computation of the model evidence for dynamical systems through thermodynamic integration. To ensure a fair comparison across methods, we will always use the same learning rate, decay coefficient, batch size, and initial values for each example. Since we are interested in gains obtained from the CVs, we will always fix $n = 0$. A description of how to set kernel hyperparameters is also provided in Appendix C.

Method	CV	vv-CV
Exact solution	$\mathcal{O}((dm_t^2 + m_t^3)T)$	$\mathcal{O}(dm_t^2T^5 + m_t^3T^6)$
Stochastic optimisation	$\mathcal{O}(dm_t^2\tilde{m}_tLT)$	$O(dm_t^2\tilde{m}_tLT^5)$

Table 1: Computational complexity of kernel-based CVs and vv-CVs as a function of $d, m, \tilde{m}, L$ and T. We assume that m_t is the same $\forall t \in [T]$ up to additive or multiplicative constants (and similarly for all $\tilde{m}_t$ with $t \in [T]$ in the stochastic optimisation algorithms). The cost of stochastic optimisation algorithms is assumed to only scale with the cost of stochastic estimates of the gradient of J^{vv}.

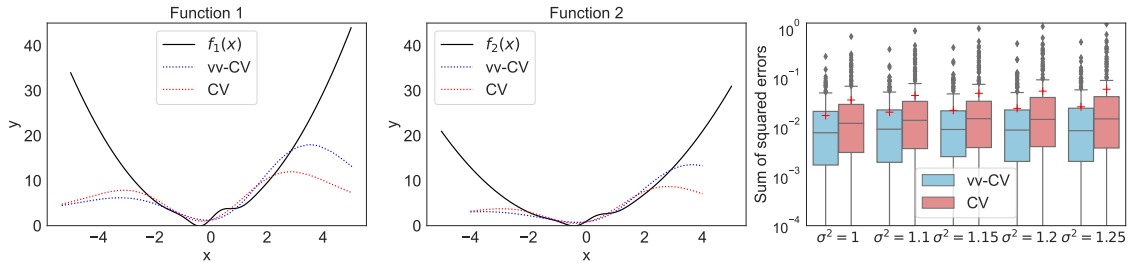

Figure 2: *Numerical integration of problem from South et al. (2022). Left and center:* Illustration of f_1 and f_2, as well as the corresponding kernel-based CVs and vv-CVs obtained through stochastic optimisation when $\Pi_1 = \mathcal{N}(0, 1)$ and $\Pi_2 = \mathcal{N}(0, 1.25)$. *Right:* Sum of the squared errors in estimating $\Pi_1[f_1]$ and $\Pi_2[f_2]$. Here, $\Pi_1 = \mathcal{N}(0, 1)$ whilst $\Pi_2 = \mathcal{N}(0, \sigma^2)$ where $\sigma^2 \in \{1, 1.1, 1.15, 1.2, 1.25\}$.

5.1 Synthetic Example

We start with a synthetic example selected from South et al. (2022) (denoted f_2), and to make the problem fit into our framework we introduced another similar integrand (denoted f_1):

$$f_1(x) = 1.5 + x + 1.5x^2 + 1.75\sin(\pi x)\exp(-x^2), \qquad f_2(x) = 1 + x + x^2 + \sin(\pi x)\exp(-x^2).$$

For this problem, we trained all CVs through stochastic optimisation and use $m = (50, 50)$ MC samples. This synthetic example was originally used by South et al. (2022) to show one of the drawbacks of kernel-based CVs, namely that the fitted CV will usually tend to β in parts of the domain where we do not have any function evaluations. This phenomenon can be observed on the red lines in Figure 2 (left and center) which gives a CV based on a squared-exponential kernel. This behaviour is clearly one of the biggest drawbacks of existing kernel-based approaches. However, the blue curve, representing a kernel-based vv-CV with separable kernel where B was inferred through optimisation, partially overcomes this issue by using evaluations of both integrands, hence clearly demonstrating potential advantages of sharing function values across integration tasks.

The right-most plot in Figure 2 presents several box plots for the sum of squared errors for each integration problem calculated over 100 repetitions of the experiment. The different box plots show the impact of the difference in Π_1 and Π_2. As we observed, vv-CVs always outperform CVs, although this difference in performance is more stark when Π_2 has a larger tail than Π_1. This reinforces the previous point, since a more disperse Π_2 means that the second integrand will be evaluated more often at more extreme areas of the domain, which will help obtain a better vv-CV by improving the fit at the tails of the distribution.

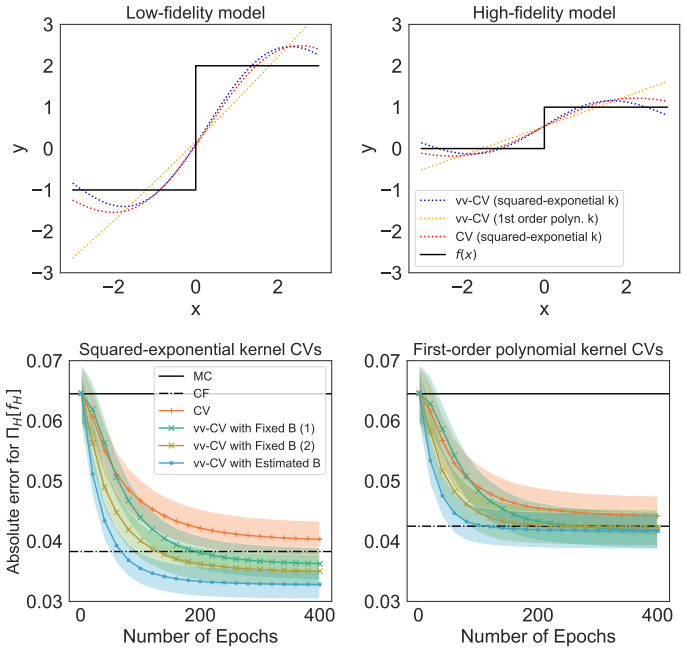

Figure 3: *Numerical integration of univariate discontinuous multifidelity model. Left:* performance of CVs based on a squared-exponential kernel as a number of epochs of the optimisation algorithm. The lines provide the mean over 100 repetitions of the experiment, whereas the shaded areas provide one standard deviation above and below the mean. *Right:* same experiment for a polynomial kernel.

5.2 Multi-fidelity Modelling

Many problems in the engineering and physical sciences can be tackled with multiple models of a single system of interest. These different models are often associated with varying computational costs and levels of accuracy, and their combination to solve a task is usually referred to as multi-fidelity modelling; see Peherstorfer et al. (2018) for a review. In this section, we will focus on the case with $T = 2$ models. We will consider a high-fidelity model f_H and a low-fidelity model f_L, and will attempt to estimate the integral of f_H with our vv-CVs and using function evaluations from both the high- and low-fidelity models. For clarity, we will now denote the function $f = (f_L, f_H)$ and the vector-probability distribution $\Pi = (\Pi_L, \Pi_H)$.

We note that this is a special case of the problem considered in our paper since we use evaluations of multiple functions but are only interested in $\Pi_H[f_H]$ (whereas $\Pi_L[f_L]$ is not of interest). The most common approach to tackling this type of problem is multi-level MC (Giles, 2015), and in the context of unnormalised densities, multi-level MCMC can be used (Dodwell et al., 2019). However, since vv-CVs are post-processing tools which can be used with any MC method, we will restrict ourselves to reducing the variance of simple MC estimators.

Univariate step function Our first experiment is a common synthetic problem in the multi-fidelity literature; see for example Xi et al. (2018). The low-fidelity function is $f_{\mathrm{L}}(x) = 2$ if $x \geq 0$ and -1 otherwise. The high-fidelity function is $f_{\mathrm{H}}(x) = 1$ if $x \geq 0$ and 0 otherwise. Although this problem is only one-dimensional, it is nonetheless not straightforward for CVs since the integrands are discontinuous but existing CVs are all continuous. The integral is over the real line and taken against $\Pi = \mathcal{N}(0, 1)$, and we fix the sample sizes to $m = (m_{\mathrm{L}}, m_{\mathrm{H}}) = (40, 40)$.

Results with a squared-exponential and 1st order polynomial kernel k can be found in Figure 3. The upper plots clearly show that the approximations are not of very high-quality, but the lower plots show that all CVs can still lead to an order of two gain in accuracy over MC methods.

We also observe that vv-CVs can lead to further gains over existing CVs by leveraging evaluations of f_L. For both kernels, we provide three different versions of the vv-CVs with separable structure to highlight the impact of the matrix B. The first two cases use Algorithm 1 with a fix value of B. In the first instance, $B_{11} = B_{22} = 0.1, B_{12} = B_{21} = 0.01$, whereas in the second instance $B_{11} = B_{22} = 0.5, B_{12} = B_{21} = 0.01$. The third case is based on estimating B through Algorithm 2. Clearly, B can have a significant impact on the performance of the vv-CV, and estimating a good value from data can provide further gains. The choice of k is also significant; all CVs based on the squared-exponential kernel significantly outperform the CVs based on a 1st order polynomial kernel.

Multivariate Model of Water Flow We now move on to a much more challenging task based on multi-fidelity models of the flow of water through a borehole drilled from the ground surface through aquifers (Xiong et al., 2013). The two functions have $d = 8$ inputs $x = (r_w, r, T_u, T_l, H_u, H_l, L, K_w)$ representing a range of parameters influencing the geometry and hydraulic conductivity of the borehole, as well as transmissivity of the aquifer. The two functions are given by

$$f_L(x) = \frac{5T_u(H_u - H_l)}{\log\left(\frac{r}{r_w}\right)\left(1.5 + \frac{2LT_u}{\log\left(\frac{r}{r_w}\right)r_w^2 K_w} + \frac{T_u}{T_l}\right)} \quad \text{and} \quad f_H(x) = \frac{2\pi T_u(H_u - H_l)}{\log\left(\frac{r}{r_w}\right)\left(1 + \frac{2LT_u}{\log\left(\frac{r}{r_w}\right)r_w^2 K_w} + \frac{T_u}{T_l}\right)}.$$

They provides estimates of water flow through a borehole in m^3 per year. Prior distributions have been elicited from scientists over input parameters to account for uncertainties about their exact value (see Appendix D.3 for further details). One quantity of interest here is the expected water flow under these prior distributions. We hence have $\Pi_1 = \Pi_2$. Here, f_H is not more expensive than f_L and so the two-fidelity approach is not essential for this example. However, this example is popular in the multifidelity modelling literature (Xiong et al., 2013; Kandasamy et al., 2016; Park et al., 2017) since it is representative of the difficulty of problems commonly tackled in this field.

Results of our simulation study are presented in Table 2. We compare a standard MC estimator with a kernel-based CV fitted with a closed form solution (denoted CF) and two kernel-based vv-CVs corresponding to special case II in Section 3. The first with $B_{11} = B_{22} = 5e-4$ and $B_{12} = B_{21} = 5e-5$, and the second with B estimated using Algorithm 2. The kernel used was a tensor product of squared-exponential kernels with a separate lengthscale for each dimension. Clearly, vv-CVs significantly outperform MC in the large majority of cases, and estimating B can lead to significant gains over using a fixed B. The worst performance for vv-CVs with estimated B is when values of m are the lowest. This is because m is not large enough to learn a good value of B.

5.3 Computation of the Model Evidence for Dynamical Systems

We now consider Bayesian inference for non-linear differential equations such as dynamical systems, which can be particularly challenging due to the need to compute the model evidence. This is usually a computationally expensive task since sampling from the posterior repeatedly requires the use of a numerical solver for differential equations which needs to be used at a fine resolution.

In Calderhead and Girolami (2009), the authors propose to use *thermodynamic integration* (TI) (Friel and Pettitt, 2008) to tackle this problem, and Oates et al. (2016, 2017) later showed that CVs can lead to significant gains in accuracy in this context. Let Θ denote our parameter space. TI introduces a path from the prior $p(\theta)$ to the posterior $p(\theta|y)$, where y and θ represent the observations and the unknown parameters respectively. This is accomplished by the power posterior $p(\theta|y,t) \propto p(y|\theta)^t p(\theta)$, where $t \in [0,1]$ is called the inverse temperature. When $t = 0$, $p(\theta|y,t) = p(\theta)$, whereas when $t = 1$, $p(\theta|y,t) = p(\theta|y)$. The standard TI formula for the model evidence has a simple

m	vv-CV- Estimated B	vv-CV-Fixed B	CF	MC
$(10, 10)$	3.722 (0.273)	**1.943** (0.150)	2.236 (0.159)	6.418 (0.435)
$(20, 20)$	**1.290** (0.096)	1.352 (0.103)	1.960 (0.101)	4.314 (0.313)
$(50, 50)$	**1.044** (0.064)	1.766 (0.120)	1.761 (0.070)	2.629 (0.172)
$(100, 100)$	**1.074** (0.064)	1.647 (0.137)	1.712 (0.045)	1.827 (0.152)
$(150, 150)$	**0.854** (0.047)	1.302 (0.094)	1.671 (0.039)	1.423 (0.102)

Table 2: *Expected values of the flow of water through a borehole under an expert-elicited prior distribution and using the high-fidelity model.* The numbers provided give the mean absolute integration error for 100 repetition of the task of estimating $\Pi_H[f_H]$, and the number in bracket provide the sample standard deviation. To provide the absolute error, the true value of the integral (72.8904) is estimated by a MC estimator with 5×10^5 samples.

form which can be approximated using second-order quadrature over a discretized temperature ladder $0 = t_1 \leq \cdots \leq t_w = 1$ Friel et al. (2014). It takes the following form

$$\log p(y) = \int_0^1 \left[\int_\Theta \log p(y|\theta) p(\theta|y,t) d\theta\right] dt \approx \sum_{i=1}^w \frac{t_{i+1}-t_i}{2}(\mu_{i+1}+\mu_i) - \frac{(t_{i+1}-t_i)^2}{12}(v_{i+1}-v_i),$$

where μ_i is the mean and v_i the variance of $\log p(y|\theta)$ under $\pi_i = p(\theta|y,t_i)$. To estimate $\{\mu_i, v_i\}_{i=1}^w$, we need to sample from all power posteriors on the ladder, then use a MCMC estimator which can be enhanced through CVs. This gives $T = 2w$ integrals which are *related*: w integrals to compute means and w integrals to compute variances, each against different power posteriors. As we will see, this relationship between integration tasks will allow vv-CVs to provide significant gains in accuracy.

Our experiments will focus on the *van der Poll oscillator*, which is a second order oscillator $x : \mathbb{R}_+ \times \Theta \to \mathbb{R}$ (where here $\theta \in \mathbb{R}$) given by the solution of $d^2x/ds^2 - \theta(1-x^2)dx/ds + x = 0$, where s represents the time index. For this experiment, we will follow the exact setup of Oates et al. (2017)and transform the equation into a system of first order equations:

$$\frac{dx_1}{ds} = x_2, \qquad \frac{dx_2}{ds} = \theta(1-x_1^2)x_2 - x_1,$$

which can be tackled with ODE solvers. Our data will consist of noisy observation of x_1 (the first component of that system) given by $y(s) = \mathcal{N}(x_1(s;\theta), \sigma^2)$ with $\sigma = 0.1$ at each point $s \in \{0, 1, \ldots, 10\}$; see the left-most plot of Figure 4 for an illustration. We will take a ladder of size $w = 31$ with $t_i = ((i-1)/30)^5$ for $i \in \{1, \ldots, 31\}$. This gives a total of $T = 62$ integrals will need to be computed simultaneously, which is likely to be too computationally expensive. As a result, we group integrands in groups of 4 means or 4 variances (except one group of 3 for mean and variance), and use vv-CVs for each group separately. To sample from the power-posterior, we use population MC with the manifold Metropolis-adjusted Langevin algorithm (Girolami and Calderhead, 2011). Due to the high computational cost of using ODE solvers, our samples will be limited to less than 100 per integrand and this number will be the same for each integration task.

Our results are presented in Figure 4, and the boxplots present 20 repetitions of the experiment. The kernel parameters were taken to be identical to those in Oates et al. (2017). As observed in the centre plot, kernel-based CVs provide relatively accurate estimates of the model evidence. As the sample size increases, we notice less variability in these estimates, but the central 50% of the runs are contained in an interval which excludes the true value. In comparison, the right-most plot shows that kernel-based vv-CVs can provide significant further reduction in variance. The distribution of estimates is also much more concentrated and centered around the true value.

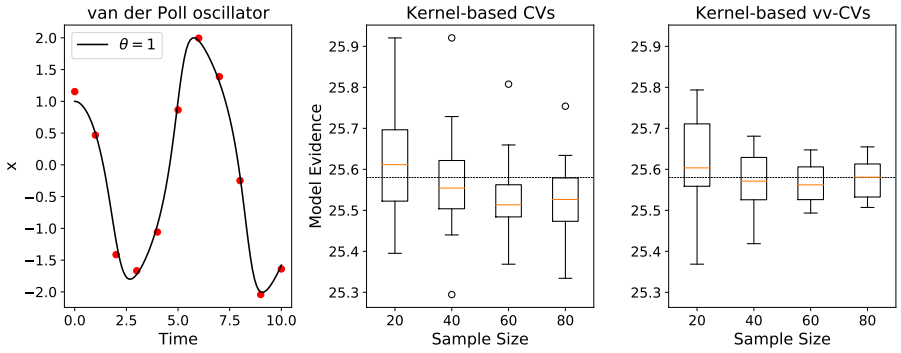

Figure 4: *Model evidence computation through thermodynamic integration. Left:* Illustration of the van der Poll oscillator model (black line) and corresponding observations (red dots) used to obtain a posterior distribution. *Center:* Estimates of the model evidence as a function of the number of posterior samples for kernel-based CVs. The box-plots were created by repeating the experiment 20 times and the black line gives an estimate of the truth obtained from Oates et al. (2017) and equals 25.58. *Right:* Same experiment repeated with kernel-based vv-CVs.

6 Conclusion

This paper considered variance reduction techniques that share information across related integration problems. The proposed solution, *vector-valued control variates*, was shown to lead to significant variance reduction for problems including multi-fidelity modelling and Bayesian evidence computation for dynamical systems. Our approach is, to the best of our knowledge, the first algorithm able to perform multi-task learning for numerical integration by using only evaluation of the score functions of the corresponding target distributions. It is also the first algorithm which can simultaneously learn the relationship between integrands and provide estimates of the corresponding integrals.

There are a number of possible ways forward. Firstly, the approach could be specialised for a range of other problems in Bayesian statistics outside of model evidence computations. A first example is when performing a sensitivity analysis for the choice of prior. In that case, we might run MCMC chains for several choices of priors, and would then compare the corresponding posterior expectations of interest which differ only up due to the priors. Clearly, these integrals will be closely related and gains may be obtained by sharing information across tasks. A second example would be in settings where our data becomes available in a sequential manner, and we would like to update our estimates of posterior expectations whenever new data is available. This setting leads to a sequence of integration tasks where information from the earlier tasks can inform future tasks. These are of course just two examples of possible uses, and we expect that many others could be found.

On the methodology side, this paper proposed an approach for constructing vv-CVs and showed this can lead to significant gains in accuracy. However, further work will be needed to make the method more computationally practical and efficient. One particular line of research warranting further investigation is how special cases of our matrix-valued Stein kernels in Theorem 3.1 can be selected to reduce the computational cost whilst still producing a rich class of vv-CVs. Another line of research which warrants further work is the question of when transferring information across tasks will lead to sufficient gains in accuracy to warrant the additional computational cost.

Finally, we remark that a number of algorithms have been developed using statistical divergences called Stein discrepancies, which are derived from Stein reproducing kernels (Anastasiou et al., 2021). Most relevant to Bayesian computation are sampling methods such as Stein variational gradient descent (Liu and Wang, 2016; Wang et al., 2019) and Stein points (Chen et al., 2018). One of the

main contributions of this paper has been the development of matrix-valued Stein kernels which give zero mean vv-functions against vector-probability distributions. This new tool could hence lead to novel (and more flexible) Stein discrepancies, and hence novel statistical methods. In the context of sampling, this could lead to methods which approximate multiple distributions simultaneously.

Acknowledgements

The authors would like to thank Chris Oates for helpful discussions and for sharing some of his code for the thermodynamic integration example. ZS was supported under the EPSRC grant [EP/R513143/1]. AB was supported by the Department of Engineering at the University of Cambridge, and this material is based upon work supported by, or in part by, the U.S. Army Research Laboratory and the U. S. Army Research Office, and by the U.K. Ministry of Defence and the U.K. Engineering and Physical Sciences Research Council (EPSRC) under grant number [EP/R018413/2]. FXB was supported by the Lloyd's Register Foundation Programme on Data-Centric Engineering and The Alan Turing Institute under the EPSRC grant [EP/N510129/1], and through an Amazon Research Award on "Transfer Learning for Numerical Integration in Expensive Machine Learning Systems".

References

Álvarez, M. A., Rosasco, L., and Lawrence, N. D. (2012). Kernels for vector-valued functions: A review. *Foundations and Trends in Machine Learning*, 4(3):195–266.

Anastasiou, A., Barp, A., Briol, F.-X., Ebner, B., Gaunt, R. E., Ghaderinezhad, F., Gorham, J., Gretton, A., Ley, C., Liu, Q., Mackey, L., Oates, C. J., Reinert, G., and Swan, Y. (2021). Stein's method meets Statistics: A review of some recent developments. *arXiv:2105.03481*.

Assaraf, R. and Caffarel, M. (1999). Zero-variance principle for Monte Carlo algorithms. *Physical Review Letters*, 83(23):4682.

Barp, A., Oates, C. J., Porcu, E., and Girolami, M. (2021a). A Riemannian–Stein kernel method. *Bernoulli (to appear), arXiv:1810.04946*.

Barp, A., Takao, S., Betancourt, M., Arnaudon, A., and Girolami, M. (2021b). A unifying and canonical description of measure-preserving diffusions. *arXiv preprint arXiv:2105.02845*.

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

Berlinet, A. and Thomas-Agnan, C. (2011). *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Springer Science & Business Media.

Bottou, L., Curtis, F. E., and Nocedal, J. (2018). Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311.

Briol, F.-X., Oates, C. J., Cockayne, J., Chen, W. Y., and Girolami, M. (2017). On the sampling problem for kernel quadrature. In *Proceedings of the International Conference on Machine Learning*, pages 586–595.

Briol, F.-X., Oates, C. J., Girolami, M., Osborne, M. A., Sejdinovic, D., et al. (2019). Probabilistic integration: A role in statistical computation? *Statistical Science*, 34(1):1–22.

Calderhead, B. and Girolami, M. (2009). Estimating Bayes factors via thermodynamic integration and population MCMC. *Computational Statistics and Data Analysis*, 53(12):4028–4045.

Carmeli, C., De Vito, E., and Toigo, A. (2006). Vector valued reproducing kernel Hilbert spaces of integrable functions and Mercer theorem. *Analysis and Applications*, 4(4):377–408.

Carmeli, C., De Vito, E., Toigo, A., and Umanita, V. (2010). Vector valued reproducing kernel Hilbert spaces and universality. *Analysis and Applications*, 8(1):19–61.

Chen, W. Y., Mackey, L., Gorham, J., Briol, F.-X., and Oates, C. J. (2018). Stein points. In *Proceedings of the International Conference on Machine Learning, PMLR 80*, pages 843–852.

Ciliberto, C., Mroueh, Y., Poggio, T., and Rosasco, L. (2015). Convex learning of multiple tasks and their structure. In *International Conference on Machine Learning*, pages 1548–1557.

Dellaportas, P. and Kontoyiannis, I. (2012). Control variates for estimation based on reversible Markov chain Monte Carlo samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 74(1):133–161.

Dinuzzo, F., Ong, C. S., Gehler, P. V., and Pillonetto, G. (2011). Learning output kernels with block coordinate descent. In *International Conference on Machine Learning*.

Dodwell, T. J., Ketelsen, C., Scheichl, R., and Teckentrup, A. L. (2019). Multilevel Markov chain Monte Carlo. *SIAM Review*, 61(3):509–545.

Evgeniou, T., Micchelli, C. A., and Pontil, M. (2005). Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637.

Friel, N., Hurn, M., and Wyse, J. (2014). Improving power posterior estimation of statistical evidence. *Statistics and Computing*, 24(5):709–723.

Friel, N. and Pettitt, A. N. (2008). Marginal likelihood estimation via power posteriors. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(3):589–607.

Gessner, A., Gonzalez, J., and Mahsereci, M. (2020). Active multi-information source Bayesian quadrature. In *Uncertainty in Artificial Intelligence*, pages 712–721.

Giles, M. B. (2015). Multilevel Monte Carlo methods. *Acta Numerica*, 24:259–328.

Girolami, M. and Calderhead, B. (2011). Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 73(2):123–214.

Gorham, J., Duncan, A. B., Vollmer, S. J., and Mackey, L. (2019). Measuring sample quality with diffusions. *Annals of Applied Probability*, 29(5):2884–2928.

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

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

Jones, G. L. (2004). On the Markov chain central limit theorem. *Probability Surveys*, 1:299–320.

Kandasamy, K., Dasarathy, G., Oliva, J., Schneider, J., and Poczos, B. (2016). Gaussian process optimisation with multi-fidelity evaluations. In *Neural Information Processing Systems*, pages 992–1000.

Karvonen, T., Särkkä, S., and Oates, C. J. (2019). Symmetry exploits for Bayesian cubature methods. *Statistics and Computing*, 29:1231–1248.

Kingma, D. P. and Ba, J. L. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Liu, Q. and Wang, D. (2016). Stein variational gradient descent: A general purpose Bayesian inference algorithm. In *Advances in Neural Information Processing Systems*, pages 2378–2386.

Micchelli, C. A. and Pontil, M. (2005). On learning vector-valued functions. *Neural Computation*, 17(1):177–204.

Micheli, M. and Glaunes, J. A. (2014). Matrix-valued kernels for shape deformation analysis. *Geometry, Imaging and Computing*, 1(1):57–139.

Mijatovic, A. and Vogrinc, J. (2018). On the Poisson equation for Metropolis-Hastings chains. *Bernoulli*, 24(3):2401–2428.

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

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

Oates, C. J. and Girolami, M. (2016). Control functionals for quasi-Monte Carlo integration. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 51, pages 56–65.

Oates, C. J., Girolami, M., and Chopin, N. (2017). Control functionals for Monte Carlo integration. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(3):695–718.

Oates, C. J., Papamarkou, T., and Girolami, M. (2016). The controlled thermodynamic integral for Bayesian model comparison. *Journal of the American Statistical Association*, 111(514):634–645.

O'Hagan, A. (1991). Bayes–Hermite quadrature. *Journal of Statistical Planning and Inference*, 29:245–260.

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

Park, C., Haftka, R. T., and Kim, N. H. (2017). Remarks on multi-fidelity surrogates. *Structural and Multidisciplinary Optimization*, 55(3):1029–1050.

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

Si, S., Oates, C. J., Duncan, A. B., Carin, L., and Briol, F.-X. (2021). 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*.

South, L. F., Karvonen, T., Nemeth, C., Girolami, M., and Oates, C. J. (2021). Semi-exact control functionals from Sard's method. *Biometrika (to appear)*.

South, L. F., Oates, C. J., Mira, A., and Drovandi, C. (2019). Regularised zero-variance control variates for high-dimensional variance reduction. *arXiv:1811.05073*.

South, L. F., Riabiz, M., Teymur, O., and Oates, C. J. (2022). Post-Processing of MCMC. *Annual Review of Statistics and Its Application*.

Steinwart, I. and Christmann, A. (2008). *Support vector machines*. Springer Science & Business Media.

Wan, R., Zhong, M., Xiong, H., and Zhu, Z. (2019). Neural control variates for variance reduction. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, page 533–547.

Wang, D., Tang, Z., Bajaj, C., and Liu, Q. (2019). Stein variational gradient descent with matrix-valued kernels. *Advances in Neural Information Processing Systems*, 32:7834.

Xi, X., Briol, F.-X., and Girolami, M. (2018). Bayesian quadrature for multiple related integrals. In *International Conference on Machine Learning*, pages 5373–5382.

Xiong, S., Qian, P. Z., and Wu, C. J. (2013). Sequential design and analysis of high-accuracy and low-accuracy computer codes. *Technometrics*, 55(1):37–46.

Appendix

We now complements the main text with additional details. First, in Appendix A, we provide the proofs of all the theoretical results in the main text. Then, in Appendix B, we present alternative constructions of vv-CVs. These constructions include a more general form of kernel-based vv-CVs as well as some polynomial-based CVs. Finally, in Appendix C and Appendix D, we provide additional details on our implementation of the vv-CVs and provide additional numerical experiments.

A Proofs of the Main Theoretical Results

In this section, we recall the proofs of the theoretical results in the main text. The derivation of the mv-kernel from Theorem 3.1 can be found in Appendix A.1. The proof that kernel-based vv-CVs are square-integrable (Theorem 3.2) is in Appendix A.2. Finally, the proof of the existence of the optimal parameters as the solution to a linear system (Theorem 4.1) is given in Appendix A.3.

A.1 Proof of Theorem 3.1

We will show K_0 is a kernel and derive its matrix components by constructing an appropriate feature map. The first order Stein operator maps matrix-valued functions $u = (u_1, u_2 \ldots, u_T) : \mathbb{R}^d \to \mathbb{R}^{d \times T}$ to the vv-function $\mathcal{S}^{\text{vv}}[u] : \mathbb{R}^d \to \mathbb{R}^T$ given by

$$\mathcal{S}^{\text{vv}}[u] = \left(\mathcal{L}'_{\Pi_1}[u_1], \ldots, \mathcal{L}'_{\Pi_T}[u_T]\right)^\top$$
$$\text{where} \quad \mathcal{L}'_{\Pi_t}[u_t](x) = \nabla_x \cdot u_t(x) + \nabla_x \log \pi_t(x) \cdot u_t(x) \quad \forall t \in [T].$$

Since $K \in C^{1,1}(\mathbb{R}^d \times \mathbb{R}^d)$, we can use (Steinwart and Christmann, 2008, Corollary 4.36) to conclude that $\mathcal{H}_K$ is a vector-valued RKHS of continuously differentiable functions from $\mathbb{R}^d$ to $\mathbb{R}^T$, hence the tensor product $\mathcal{H}_K^d$ consists of suitable functions $u \in \mathcal{H}_K^d$, with components $u^i = (u_1^i, \dots, u_T^i) \in \mathcal{H}_K$ for $i \in [d]$. Now recall that (see for example Theorem 2.11 (Micheli and Glaunes, 2014)):

$$\left\langle \partial_x^j K(\cdot, x) e_t, u^i \right\rangle_{\mathcal{H}_K} = e_t \cdot \partial^j u^i(x) = \partial^j u_t^i(x) \equiv \frac{\partial u_t^i}{\partial x^j}(x) \quad \forall t \in [T]$$

where $e_t \in \mathbb{R}^T$ is a vector of zeros with value 1 in the t^{th} component. Then, writing $K_x^{e_t} \equiv K(\cdot, x)e_t$,

$$\begin{aligned}
\mathcal{S}^{\text{vv}}[u](x) &= \textstyle\sum_{t=1}^T \sum_{r=1}^d (\partial_x^r u_t^r(x) + \partial_x^r \log \pi_t \, u_t^r(x)) e_t \\
&= \textstyle\sum_{t=1}^T \sum_{r=1}^d \langle \partial_x^r K_x^{e_t} + l_{tr}(x) K_x^{e_t}, u^r \rangle_{\mathcal{H}_K} \, e_t \\
&= \textstyle\sum_{t=1}^T \langle \partial_x^\bullet K_x^{e_t} + l_{t\bullet}(x) K_x^{e_t}, u \rangle_{\mathcal{H}_K^d} \, e_t,
\end{aligned}$$

where $l_{tr}(x) = \partial_x^r \log \pi_t(x)$, and $\partial_x^\bullet K_x^{e_t}$ and $l_{t\bullet}(x)$ denote respectively the tuples $(\partial_x^1 K_x^{e_t}, \dots, \partial_x^d K_x^{e_t}) \in \mathcal{H}_K^d$ and $(l_{t1}(x), \dots, l_{td}(x)) \in \mathbb{R}^d$.

We have thus obtained a feature map, i.e., a map $\gamma : \mathbb{R}^d \to \mathcal{B}(\mathcal{H}_K^d, \mathbb{R}^T)$, where $\mathcal{B}(\mathcal{H}_K^d, \mathbb{R}^T)$ denotes the space of bounded linear maps from $\mathcal{H}_K^d$ to $\mathbb{R}^T$, via the relation

$$\gamma(x)[u] = \mathcal{S}^{\text{vv}}[u](x),$$

with adjoint $\gamma(x)^* = \sum_{t=1}^T (\partial_x^\bullet K_x^{e_t} + l_{t\bullet}(x) K_x^{e_t}) e_t$. Recall the adjoint map $\gamma(y)^* \in \mathcal{B}(\mathbb{R}^T, \mathcal{H}_K^d)$ to $\gamma(y)$, is defined for any $a \in \mathbb{R}^T, u \in \mathcal{H}_K^d$ by the relation

$$\langle \gamma(y)^*[a], u \rangle_{\mathcal{H}_K^d} = \gamma(y)[u] \cdot a.$$

In particular, by Proposition 1 of Carmeli et al. (2010) we have that

$$K_0(x, y) \equiv \gamma(x) \circ \gamma(y)^* \in \mathbb{R}^{T \times T}$$

will then be the kernel associated to the "feature operator" (that is, a surjective partial isometry whose image is $\mathcal{H}_{K_0}$) $\mathcal{S}^{\text{vv}} : \mathcal{H}_K^d \to \mathcal{H}_{K_0}$. Subbing in the expressions for the feature map and its adjoint derived above, and using the equalities

$$\begin{aligned}
&\left\langle \partial_x^s K(\cdot, x) e_t, \partial_y^r K(\cdot, y) e_{t'} \right\rangle_{\mathcal{H}_K} = e_t \cdot \partial_x^s \partial_y^r K(x, y) e_{t'} = (\partial_x^s \partial_y^r K(x, y))_{tt'} \qquad \forall t, t' \in [T] \\
&\text{and} \qquad \left\langle \partial_x^{ss} K(\cdot, x) e_t, \partial_y^r K(\cdot, y) e_{t'} \right\rangle_{\mathcal{H}_K} = (\partial_x^{ss} \partial_y^r K(x, y))_{tt'} \qquad \forall t, t' \in [T],
\end{aligned}$$

which hold for any $C^{1,1}(\mathbb{R}^d \times \mathbb{R}^d)$ mv-kernel (Micheli and Glaunes, 2014), we obtain the following expression for the components of K_0

$$\begin{aligned}
(K_0(x, y))_{tt'} = &\textstyle\sum_{r=1}^d (\partial_x^r \partial_y^r K(x, y))_{tt'} + l_{t'r}(y)(\partial_x^r K(x, y))_{tt'} \\
&+ l_{tr}(x) \partial_y^r (K(x, y))_{tt'} + l_{tr}(x) l_{t'r}(y) (K(x, y))_{tt'}.
\end{aligned}$$

In particular for separable kernels (i.e. $K(x, y) = Bk(x, y)$) we have

$$(K_0(x, y))_{tt'} = B_{tt'} \textstyle\sum_{r=1}^d \partial_x^r \partial_y^r k(x, y) + l_{t'r}(y) \partial_x^r k(x, y) + l_{tr}(x) \partial_y^r k(x, y) + l_{tr}(x) l_{t'r}(y) k(x, y).$$

A.2 Proof of Theorem 3.2

Recall that if a scalar kernel k satisfies $\int_{\mathbb{R}^d} k(x, x) \mathrm{d}\mu(x) < \infty$, then its RKHS consists of square μ-integrable functions (for any finite measure μ) (Steinwart and Christmann, 2008, Theorem 4.26).

If $g \in \mathcal{H}_{K_0}$ then g_t belongs to the RKHS with scalar-valued kernel (this follows from (Carmeli et al., 2010, Prop. 1), using as feature operator the dot product with respect to e_t, where e_t is defined in appendix A.1)

$$\begin{aligned}(K_0(x,y))_{tt} = &\textstyle\sum_{r=1}^d (\partial_x^r \partial_y^r K(x,y))_{tt} + l_{tr}(y)\partial_x^r (K(x,y))_{tt} \\ &+ l_{tr}(x)\partial_y^r (K(x,y))_{tt} + l_{tr}(x) l_{tr}(y)(K(x,y))_{tt} \qquad \forall t \in [T].\end{aligned}$$

In particular since K is bounded with bounded derivatives, and

$$\Pi_t\left[|l_{tr}|\right] + \Pi_t\left[|l_{tr}|^2\right] \leq \sqrt{\Pi_t\left[|l_{tr}|^2\right]} + \Pi_t\left[|l_{tr}|^2\right] \qquad \forall t \in [T], r \in [d]$$

then $\int_{\mathbb{R}^d} (K_0(x,x))_{tt} \mathrm{d}\Pi_t(x) < \infty$ if $\|\nabla_x \log \pi_t(x)\|_2$ is square integrable with respect to Π_t, and the result follows.

A.3 Proof of Theorem 4.1

Proof. We want to find

$$\begin{aligned}&\arg\min_{g \in \mathcal{H}_{K_0}} L_m^{\mathrm{vv}}(g, \beta) \\ \text{where} \qquad &L_m^{\mathrm{vv}}(g,\beta) := \textstyle\sum_{t=1}^T \frac{1}{m_t} \sum_{j=1}^{m_t} (f_t(x_{tj}) - g_t(x_{tj}) - \beta_t)^2 + \lambda \|g\|^2_{\mathcal{H}_{K_0}}.\end{aligned}$$

Note that the objective is the same as that in (11), with the only difference being that the first input is now a function as opposed to the parameter value parameterising this function. We will abuse notation by using the same mathematical expression for both objectives.

By Ciliberto et al. (2015, Section 2.1), any solution of the minimization problem has the form $\hat{g}(\cdot) \equiv \sum_{t'=1}^T \sum_{j'=1}^{m_{t'}} K_0(\cdot, x_{t'j'})\theta_{t'j'}$. Subbing this solution into $L_m^{\mathrm{vv}}(g,\beta)$ yields

$$\begin{aligned}L_m^{\mathrm{vv}}(\hat{g},\beta) &= \textstyle\sum_{t=1}^T \frac{1}{m_t} \sum_{j=1}^{m_t} (f_t(x_{tj}) - (\sum_{t'=1}^T \sum_{j'=1}^{m_{t'}} K_0(x_{tj}, x_{t'j'})_t \theta_{t'j'} - \beta_t)^2 \\ &\quad + \lambda \textstyle\sum_{t',t''=1}^T \sum_{j'=1}^{m_{t'}} \sum_{j''=1}^{m_{t''}} \theta_{t'j'}^T K_0(x_{t'j'}, x_{t''j''}) \theta_{t''j''} \\ &= \lambda \textstyle\sum_{t',t''=1}^T \sum_{j'=1}^{m_{t'}} \sum_{j''=1}^{m_{t''}} \theta_{t'j'}^T K_0(x_{t'j'}, x_{t''j''}) \theta_{t''j''} \\ &\quad + \textstyle\sum_{t=1}^T \frac{1}{m_t} \sum_{j=1}^{m_t} \Big(y_{tj}^2 + (\sum_{t'=1}^T \sum_{j'=1}^{m_{t'}} K_0(x_{tj}, x_{t'j'})_t \theta_{t'j'})^2 \\ &\qquad\qquad - 2 \textstyle\sum_{t'=1}^T \sum_{j'=1}^{m_{t'}} y_{tj} K_0(x_{tj}, x_{t'j'})_t \theta_{t'j'} \Big),\end{aligned}$$

where $y_{tj} \equiv f_t(x_{tj}) - \beta_t$. The problem thus becomes a minimization problem over the coefficients θ,

$$\begin{aligned}\arg\min_{\theta \in \mathbb{R}^{|\mathcal{D}|}} &\lambda \textstyle\sum_{t',t''=1}^T \sum_{j'=1}^{m_{t'}} \sum_{j''=1}^{m_{t''}} \theta_{t'j'}^T K_0(x_{t'j'}, x_{t''j''}) \theta_{t''j''} \\ &- 2 \textstyle\sum_{t,t'=1}^T \frac{1}{m_t} \sum_{j=1}^{m_t} \sum_{j'=1}^{m_{t'}} \theta_{t'j'}^T K_0(x_{t'j'}, x_{tj})_{\cdot t} y_{tj} + \sum_{t=1}^T \frac{1}{m_t} \sum_{j=1}^{m_t} y_{tj}^2 \\ &+ \textstyle\sum_{t,t',t''=1}^T \sum_{j=1}^{m_t} \sum_{j'=1}^{m_{t'}} \sum_{j''=1}^{m_{t''}} \theta_{t'j'}^T K_0(x_{t'j'}, x_{tj})_{\cdot t} \frac{1}{m_t} K_0(x_{tj}, x_{t''j''})_{t\cdot} \theta_{t''j''}.\end{aligned}$$

Since the quadratic terms are semi-positive definite, the resulting objective is a convex function of θ, thus, by differentiating it, we obtain that the solution θ is the solution to

$$\begin{aligned}\textstyle\sum_{t'=1}^T \sum_{j'=1}^{m_{t'}} \Big(\sum_{t=1}^T \frac{1}{m_t} \sum_{j=1}^{m_t} K_0(x_{t''j''}, x_{tj})_{\cdot t} K_0(x_{tj}, x_{t'j'})_{t\cdot} + \lambda K_0(x_{t''j''}, x_{t'j'}) \Big) \theta_{t'j'} \\ = \textstyle\sum_t^T \frac{1}{m_t} \sum_{j=1}^{m_t} K_0(x_{t''j''}, x_{tj})_{\cdot t} (f_t(x_{tj}) - \beta_t), \qquad \forall t'' \in [T], j'' \in [m_T].\end{aligned}$$

□

B Alternative Constructions

In this second appendix, we will now complement Section 3.3 by covering alternative constructions to those presented in the main text. First, in Appendix B.1, we present kernel-based vv-CVs based on the second order Langevin Stein operator. Then, in Appendix B.2 we point out how these constructions can lead to polynomial-based vv-CVs.

B.1 Kernel-based vv-CVs from Second-Order Langevin Stein Operators

In this section we will consider the second-order Langevin Stein operator which acts on scalar-valued functions. The following theorem provides a characterisation of the class of vv-functions obtained when applying this operator to functions in a vv-RKHS.

Theorem B.1. *Consider $\mathcal{H}_K$ which is a vv-RKHS with mv-kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{T\times T}$, and suppose that $K \in C^{2,2}(\mathbb{R}^d \times \mathbb{R}^d)$. Furthermore, for suitably regular vv-functions $u = (u_1, \dots, u_T) : \mathbb{R}^d \to \mathbb{R}^T$ define the differential operator*

$$\mathcal{S}^{vv}[u] = (\mathcal{L}''_{\Pi_1}[u_1], \dots, \mathcal{L}''_{\Pi_T}[u_T])^\top.$$

Then, the image of $\mathcal{H}_K$ under $\mathcal{S}^{vv}$ is a vv-RKHS with reproducing kernel $K_0 : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{T\times T}$:

$$\begin{aligned}(K_0(x,y))_{tt'} = \sum_{r,s=1}^d \partial_x^{ss}\partial_y^{rr}(K(x,y))_{tt'} + l_{t'r}(y)\partial_x^{ss}\partial_y^{r}(K(x,y))_{tt'} \\ + l_{ts}(x)\partial_x^{s}\partial_y^{rr}(K(x,y))_{tt'} + l_{ts}(x)l_{t'r}(y)\partial_x^{s}\partial_y^{r}(K(x,y))_{tt'} \qquad \forall t,t' \in [T].\end{aligned}$$

We note that this theorem is very similar to Theorem 3.1, and recovers the kernel of Barp et al. (2021a) when $T = 1$. However, one particular disadvantage of this construction from a computational viewpoint is that it requires higher-order derivatives of the kernel K. It also requires the evaluation of a double sum, which significantly increases computational cost relative to our construction in the main text. For this reason, we did not explore this construction in more details.

Proof. We proceed as for the proof of Theorem 3.1 and shall derive a feature map for K_0. Recall that $g = \mathcal{S}^{\text{vv}}[u] = (\mathcal{L}''_{\Pi_1}[u_1], \dots, \mathcal{L}''_{\Pi_T}[u_T])^\top$, where $\mathcal{L}''_{\Pi_i}$ is the second-order Stein operator, which maps scalar functions to scalar functions. Here u belongs to a RKHS of $\mathbb{R}^T$-valued functions with matrix kernel K. From the differentiability assumption on K, we have $\mathcal{H}_K \subset C^2$, i.e., it is a space of twice continuously differentiable functions. Note that (here $\partial^{jj} = \partial^j\partial^j = \frac{\partial^2}{\partial x_j \partial x_j}$)

$$\left\langle \partial_x^{jj} K(\cdot,x)e_t, u\right\rangle_{\mathcal{H}_K} = \partial^{jj}u_t(x) \equiv \tfrac{\partial^2 u_t}{\partial x_j \partial x_j}(x) \qquad \forall t \in [T],$$

where e_t is the t^{th} standard basis vector of $\mathbb{R}^T$ as before. Thus

$$\begin{aligned}\mathcal{L}''_{\Pi_t}[u_t](x) &= \Delta_x u_t(x) + \nabla_x \log \pi_t(x) \cdot \nabla_x u_t(x) \\ &= \textstyle\sum_{s=1}^d \partial^{ss}u_t(x) + \sum_{s=1}^d l_{ts}(x)\partial^s u_t(x) \\ &= \textstyle\sum_{s=1}^d \langle \partial_x^{ss}K(\cdot,x)e_t, u\rangle_{\mathcal{H}_K} + \sum_{s=1}^d \langle l_{ts}(x)\partial_x^s K(\cdot,x)e_t, u\rangle_{\mathcal{H}_K} \\ &= \textstyle\sum_{s=1}^d \langle \partial_x^{ss}K(\cdot,x)e_t + l_{ts}(x)\partial_x^s K(\cdot,x)e_t, u\rangle_{\mathcal{H}_K} \qquad \forall t \in [T].\end{aligned}$$

Hence

$$\mathcal{S}^{\text{vv}}[u](x) = \begin{pmatrix} \left\langle \sum_{s=1}^d \partial_x^{ss}K(\cdot,x)e_1 + l_{1s}(x)\partial_x^s K(\cdot,x)e_1, u\right\rangle_{\mathcal{H}_K} \\ \vdots \\ \left\langle \sum_{s=1}^d \partial_x^{ss}K(\cdot,x)e_T + l_{Ts}(x)\partial_x^s K(\cdot,x)e_T, u\right\rangle_{\mathcal{H}_K} \end{pmatrix} \in \mathbb{R}^T.$$

Note that for each $x \in \mathbb{R}^d$, each component of the above is a bounded linear operator $\mathcal{H}_K \to \mathbb{R}$ (i.e., the map $u \mapsto (\mathcal{S}^{\text{vv}}(u)(x))_s \in \mathbb{R}$ to the s-component is a bounded linear operator), then we have obtained a a feature map, i.e., a map $\gamma : \mathbb{R}^d \to \mathcal{B}(\mathcal{H}_K, \mathbb{R}^T)$, where $\mathcal{B}(\mathcal{H}_K, \mathbb{R}^T)$ denotes the space of bounded linear maps from $\mathcal{H}_K$ to $\mathbb{R}^T$. Specifically

$$\gamma(x) \equiv \mathcal{S}^{\text{vv}}[\cdot](x) \in \mathcal{B}(\mathcal{H}_K, \mathbb{R}^T).$$

In particular, as before

$$K_0(x, y) \equiv \gamma(x) \circ \gamma(y)^* \in \mathcal{B}(\mathbb{R}^T, \mathbb{R}^T)$$

will thus be the kernel associated to the "feature operator" $\mathcal{S}^{\text{vv}} : \mathcal{H}_K \to \mathcal{H}_{K_0}$. Recall that $\gamma(y)^* \in \mathcal{B}(\mathbb{R}^T, \mathcal{H}_K)$ is the adjoint map to $\gamma(y)$, i.e., it satisfies for any $a \in \mathbb{R}^T, u \in \mathcal{H}_K$:

$$\langle \gamma(y)^*[a], u \rangle_{\mathcal{H}_K} = \gamma(y)[u] \cdot a.$$

From this we obtain

$$\gamma(y)^* : a \mapsto \textstyle\sum_{r=1}^d \sum_{t=1}^T a_t \left(\partial_y^{rr} K(\cdot, y) e_t + l_{tr}(y) \partial_y^r K(\cdot, y) e_t \right) \in \mathcal{H}_K.$$

From $K_0(x, y)a = \gamma(x) \circ \gamma(y)^*[a]$ for all $a \in \mathbb{R}^T$ and the above expressions we can finally calculate K_0. We have

$$K_0(x, y)a = \mathcal{S}^{\text{vv}}[\gamma(y)^* a](x) = \begin{pmatrix} \left\langle \sum_{s=1}^d \partial_x^{ss} K(\cdot, x) e_1 + l_{1s}(x) \partial_x^s K(\cdot, x) e_1, \gamma(y)^* a \right\rangle_{\mathcal{H}_K} \\ \vdots \\ \left\langle \sum_{s=1}^d \partial_x^{ss} K(\cdot, x) e_T + l_{Ts}(x) \partial_x^s K(\cdot, x) e_T, \gamma(y)^* a \right\rangle_{\mathcal{H}_K} \end{pmatrix}.$$

We obtain that $K_0(x, y)a$ is a vector with components:

$$\begin{aligned}(K_0(x, y)a)_t = \textstyle\sum_{r,s=1}^d \sum_{t'=1}^T a_{t'} \Big(& (\partial_x^{ss} \partial_y^{rr} K(x, y))_{tt'} + l_{t'r}(y) (\partial_x^{ss} \partial_y^r K(x, y))_{tt'} \\ & + l_{ts}(x) (\partial_x^s \partial_y^{rr} K(x, y))_{tt'} + l_{ts}(x) l_{t'r}(y) (\partial_x^s \partial_y^r K(x, y))_{tt'} \Big) \quad \forall t \in [T].\end{aligned}$$

Thus the components of $K_0(x, y) \in \mathbb{R}^{T \times T}$ are

$$\begin{aligned}(K_0(x, y))_{tt'} = & \textstyle\sum_{r,s=1}^d (\partial_x^{ss} \partial_y^{rr} K(x, y))_{tt'} + l_{t'r}(y) (\partial_x^{ss} \partial_y^r K(x, y))_{tt'} \\ & + l_{ts}(x) (\partial_x^s \partial_y^{rr} K(x, y))_{tt'} + l_{ts}(x) l_{t'r}(y) (\partial_x^s \partial_y^r K(x, y))_{tt'} \qquad \forall t, t' \in [T].\end{aligned}$$

□

Analogously to the mv-kernel in Theorem 3.1, there are several cases of practical interest. The first is when $K(x, y) = Bk(x, y)$ is a separable kernel, in which case:

$$\begin{aligned}(K_0(x, y))_{tt'} = B_{tt'} & \textstyle\sum_{r,s=1}^d \partial_x^{ss} \partial_y^{rr} k(x, y) + l_{t'r}(y) \partial_x^{ss} \partial_y^r k(x, y) \\ & + l_{ts}(x) \partial_x^s \partial_y^{rr} k(x, y) + l_{ts}(x) l_{t'r}(y) \partial_x^s \partial_y^r k(x, y) \qquad \forall t, t' \in [T].\end{aligned}$$

The second is when K is separable and $\Pi_1 = \ldots = \Pi_T$, in which case $l_r(x) := l_{1r}(x) = \ldots = l_{Tr}(x)$ $\forall r \in [d]$ and:

$$\begin{aligned}(K_0(x, y))_{tt'} = B_{tt'} & \textstyle\sum_{r,s=1}^d \partial_y^{ss} \partial_x^{rr} k(x, y) + l_r(x)\ \partial_y^{ss} \partial_x^r k(x, y) \\ & + l_s(y)\ \partial_y^s \partial_x^{rr} k(x, y) + l_s(y) l_r(x) \partial_y^s \partial_x^r k(x, y) \qquad \forall t, t' \in [T].\end{aligned}$$

B.2 Polynomial vv-CVs

In Section 3.3, we discussed a construction for vv-CVs based on polynomials which recovers the work of Mira et al. (2013). However, it is also possible to obtain polynomial-based vv-CVs directly through our kernel constructions in Theorem 3.1 and Appendix B.1. In particular, one option would be to take $K(x,y) = Bk(x,y)$ where $B \in S_+^T$ and $k(x,y) = (x^\top y + c)^l$ where $c \in \mathbb{R}$ and $l \in \mathbb{N}$. Firstly, using the first-order Langevin Stein operator and setting $l = 1$, we obtain:

$$(K_0(x,y))_{tt'} = B_{tt'} \sum_{r=1}^d \left[1 + l_{t'r}(y)y_r + l_{tr}(x)x_r + l_{tr}(x)l_{t'r}(y)\left(x^\top y + c\right)\right] \qquad \forall t \in [T].$$

Similarly when $l = 2$, we get:

$$\begin{aligned}(K_0(x,y))_{tt'} = B_{tt'} \sum_{r=1}^d \Big[&2x_r y_r + 2\left(x^\top y + c\right) + 2y_r l_{t'r}(y)\left(x^\top y + c\right) \\ &+ 2x_r l_{tr}(x)\left(x^\top y + c\right) + l_{tr}(x)l_{t'r}\left(x^\top y + c\right)^2\Big] \qquad \forall t \in [T].\end{aligned}$$

These two choices were considered in the experiments in Section 5. An alternative would be to consider this same kernel, but using the construction based on second-order Langevin Stein operators. Again, taking $l = 1$, we obtain:

$$(K_0(x,y))_{tt'} = \sum_{r=1}^d l_{tr}(x)l_{t'r}(y)B_{tt'} \qquad \forall t \in [T].$$

Similarly, when $l = 2$, we get:

$$\begin{aligned}(K_0(x,y))_{tt'} = B_{tt'}\Big[&4\left(d + \sum_{r=1}^d l_{t'r}(y)y_r + l_{tr}(x)x_r\right) \\ &+ 2\left(\sum_{r=1}^d l_{tr}(x)l_{t'r}(y)\left(x^\top y + c\right) + \sum_{r,s=1}^d l_{ts}(x)l_{t'r}(y)x_r y_s\right)\Big] \qquad \forall t \in [T].\end{aligned}$$

C Implementation Details

In this appendix, we focus on implementation details which may be helpful for implementing the algorithms in the main text. Firstly, in Appendix C.1 we derive the derivatives of several common kernels; this is essential for the implementation of Stein reproducing kernels. Then, in Appendix C.2, we provide details on how to select hyperparameters. Finally, in Appendix C.3, we discuss how to turn the problem of estimating B from data into a sequence of convex optimisation problems.

C.1 Kernels and Their Derivatives

We now provide details of all the kernels used in the paper, as well as expressions for their derivatives.

Polynomial Kernel The polynomial kernel $k_l(x,y) = (x^\top y + c)^l$ with constant $c \in \mathbb{R}$ and power $l \in \mathbb{N}$ has derivatives given by

$$\begin{aligned}\nabla_x k_l(x,y) &= l(x^\top y + c)^{l-1} y, \quad \nabla_y k_l(x,y) = l(x^\top y + c)^{l-1} x, \\ \nabla_x \cdot \nabla_y k_l(x,y) &= \sum_{j=1}^d \frac{\partial^2}{\partial x_j \partial y_j} k_l(x,y) = \sum_{j=1}^d \frac{\partial}{\partial x_j}\left[l(x^\top y + c)^{l-1} x_j\right] \\ &= \sum_{j=1}^d l(l-1)(x^\top y + c)^{l-2} y_j x_j + l(x^\top y + c)^{l-1} \\ &= l(l-1)(x^\top y + c)^{l-2} x^\top y + dl(x^\top y + c)^{l-1}.\end{aligned}$$

Squared-Exponential Kernel The squared-exponential kernel (sometimes called Gaussian kernel) $k(x,y) = \exp(-\frac{\|x-y\|_2^2}{2\lambda})$ with lengthscale $\lambda > 0$ has derivatives given by

$$\begin{aligned}
\nabla_x k(x,y) &= -\tfrac{(x-y)}{\lambda} k(x,y), \qquad \nabla_y k(x,y) = \tfrac{(x-y)}{\lambda} k(x,y), \\
\nabla_x \cdot \nabla_y k(x,y) &= \textstyle\sum_{j=1}^d \frac{\partial^2}{\partial y_j \partial x_j} k(x,y) = \sum_{j=1}^d \frac{\partial}{\partial y_j} \left[-\frac{(x_j - y_j)}{\lambda} k(x,y) \right] \\
&= \textstyle\sum_{j=1}^d \left[\frac{1}{\lambda} - \frac{(x_j-y_j)^2}{\lambda^2} \right] k(x,y) = \left[\frac{d}{\lambda} - \frac{(x-y)^\top (x-y)}{\lambda^2} \right] k(x,y).
\end{aligned}$$

Preconditioned Squared-Exponential Kernel Following Oates et al. (2017), we also considered a preconditioned squared-exponential kernel:

$$k(x,y) = \tfrac{1}{(1+\alpha\|x\|_2^2)(1+\alpha\|y\|_2^2)} \exp\left(-\tfrac{\|x-y\|_2^2}{2\lambda^2} \right).$$

with lengthscale $\lambda > 0$ and preconditioner parameter $\alpha > 0$. This kernel has derivatives given by:

$$\begin{aligned}
\nabla_x k(x,y) &= \left[\tfrac{-2\alpha x}{1+\alpha\|x\|_2^2} - \tfrac{(x-y)}{\lambda^2} \right] k(x,y), \quad \nabla_y k(x,y) = \left[\tfrac{-2\alpha y}{1+\alpha\|y\|_2^2} + \tfrac{(x-y)}{\lambda^2} \right] k(x,y), \\
\nabla_x \cdot \nabla_y k(x,y) &= \textstyle\sum_{j=1}^d \frac{\partial^2}{\partial x_j \partial y_j} k(x,y) = \sum_{j=1}^d \frac{\partial}{\partial y_j} \left[\left(\frac{-2\alpha x_j}{1+\alpha\|x\|_2^2} - \frac{(x_j-y_j)}{\lambda^2} \right) k(x,y) \right] \\
&= \textstyle\sum_{j=1}^d \left(\frac{1}{\lambda^2} k(x,y) + \left[\frac{-2\alpha x_j}{1+\alpha\|x\|_2^2} - \frac{(x_j-y_j)}{\lambda^2} \right] \frac{\partial}{\partial y_j} k(x,y) \right) \\
&= \textstyle\sum_{j=1}^d \left(\frac{1}{\lambda^2} k(x,y) + \left[\frac{-2\alpha x_j}{1+\alpha\|x\|_2^2} - \frac{(x_j-y_j)}{\lambda^2} \right] \left[\frac{-2\alpha y_j}{1+\alpha\|y\|_2^2} + \frac{(x_j-y_j)}{\lambda^2} \right] k(x,y) \right) \\
&= k(x,y) \left[\tfrac{4\alpha^2 x^\top y}{(1+\alpha\|x\|_2^2)(1+\alpha\|y\|_2^2)} + \tfrac{2\alpha(x-y)^\top y}{\lambda^2(1+\alpha\|y\|_2^2)} - \tfrac{2\alpha(x-y)^\top x}{\lambda^2(1+\alpha\|x\|_2^2)} + \tfrac{d}{\lambda^2} - \tfrac{(x-y)^\top (x-y)}{\lambda^4} \right].
\end{aligned}$$

Product of Kernels Finally, some of our examples will also use products of well-known kernels. Consider the kernel $k(x,y) = \prod_{j=1}^d k_j(x_j, y_j)$. The derivatives of this kernel can be expressed in terms of the components of the product and their derivates as follows:

$$\begin{aligned}
\nabla_x k(x,y) &= \left(\tfrac{\partial k_1(x_1,y_1)}{\partial x_1} \textstyle\prod_{j\neq 1} k_j(x_j,y_j), \ldots, \tfrac{\partial k_d(x_d,y_d)}{\partial x_d} \prod_{j\neq d} k_j(x_j,y_j) \right)^\top \\
\nabla_y k(x,y) &= \left(\tfrac{\partial k_1(x_1,y_1)}{\partial y_1} \textstyle\prod_{j\neq 1} k_j(x_j,y_j), \ldots, \tfrac{\partial k_d(x_d,y_d)}{\partial y_d} \prod_{j\neq d} k_j(x_j,y_j) \right)^\top \\
\nabla_y \cdot \nabla_x k(x,y) &= \textstyle\sum_{j=1}^d \frac{\partial^2}{\partial x_j \partial y_j} k(x,y) = \sum_{j=1}^d \frac{\partial}{\partial y_j} \left(\frac{\partial k_j(x_j,y_j)}{\partial x_j} \prod_{i\neq j} k_i(x_i,y_i) \right) \\
&= \textstyle\sum_{j=1}^d \left[\frac{\partial^2 k_j(x_j,y_j)}{\partial y_j \partial x_j} \prod_{i\neq j} k_i(x_i,y_i) \right].
\end{aligned}$$

C.2 Hyper-parameters Selection

Most kernels (whether scalar- or matrix-valued) will have hyperparameters which we will have to select. For example, the squared-exponential kernel will often have a lengthscale or amplitude parameter, and these will have a significant impact on the performance.

We propose to select kernel hyperparameters through a marginal likelihood objective by noticing the equivalence between the optimal vv-CV based on the objective in (11) and the posterior mean of a zero-mean Gaussian process model with covariance matrix $K_0(x,y)$; see Oates et al. (2017) for a discussion in the sv-CV case. Unfortunately, computing the marginal likelihood in the general case can be prohibitively expensive due to the need to take inverses of large kernel matrices; the exact issue we were attempting to avoid through the use of the stochastic optimisation approaches. For simplicity, we instead maximise the marginal likelihood corresponding to $B = I_T$:

$$\nu^* := \arg\max_\nu -\tfrac{1}{2} \textstyle\sum_{t=1}^T \left(\sum_{j,j'=1}^{m_t} f_t(x_{tj}) (K_{\Pi_t}(\nu) + \lambda I_{m_t})^{-1}_{jj'} f_t(x_{tj'}) + \log\det[K_{\Pi_t}(\nu) + \lambda I_{m_t}] \right).$$

where $K_{\Pi_t}(\nu)$ is a matrix with entries $K_{\Pi_t}(\nu)_{ij} = k_{\Pi_t}(x_{ti}, x_{tj}; \nu)$ where k_{Π_t} is a Stein reproducing kernel of the form in (5) specialised to Π_t which has hyperparameters given by some vector ν. This form is not optimal when $B \neq I_T$, but we found that it tend to perform well in our numerical experiments. The regularisation parameter λ can also be selected through the marginal likelihood. However, in practice we are in an interpolation setting and therefore choose λ as small as possible whilst still being large enough to guarantee numerically stable computation of the matrix inverses above.

C.3 Convex Optimisation for Estimating B

As discussed in Section 4.3, estimating the matrix B for a separable kernel from data leads to a non-convex optimisation problem. Thankfully, we can approximate the optimum using a sequence of convex problems by extending the work of Dinuzzo et al. (2011); Ciliberto et al. (2015) together with Theorem 4.1 above. For this, we will require that the kernel K_0 is separable, and shall thus restrict ourselves to the case where we have a single target distribution (i.e. special case II).

Theorem C.1. *Suppose that $\Pi_t = \Pi$ for $t \in [T]$ and $K(x, y) = Bk(x, y)$ so that $K_0(x, y) = Bk_0(x, y)$ where k_0 is defined in (5). Then the following objective is convex in (θ, β, B) for any value of $\delta > 0$:*

$$\bar{L}^{vv}_{m,\delta}(\theta, \beta, B) = J^{vv}_m(\theta, \beta, I_T) + \lambda \sum_{t,t'=1}^{T} \sum_{j=1}^{m_t} \sum_{j'=1}^{m_{t'}} \mathrm{Tr}\left[B^\dagger \left(k_0(x_{tj}, x_{t'j'})\theta_{tj}\theta_{t'j'}^\top + \delta^2 I_T\right)\right] + \|B\|^2,$$

and for each β and any sequence $\delta_\ell \to 0$, the associated sequence of minimisers (θ_ℓ, B_ℓ) converges to (θ_*, B_*) s.t., $(\theta_* B_*^\dagger, B_*)$ minimises the objective in (17).

Proof. Since the kernel K_0 is separable, the objective (17) may be written in the form of Ciliberto et al. (2015, Problem ($\mathcal{Q}$)). Has shown therein, $\sum_{t,t'=1}^{T} \sum_{j=1}^{m_t} \sum_{j'=1}^{m_{t'}} \mathrm{Tr}\left[B^\dagger \left(k_0(x_{tj}, x_{t'j'})\theta_{tj}\theta_{t'j'}^\top\right)\right]$ is jointly convex in B and θ, and since the first term in $L^{\mathrm{vv}}_{m,\delta}(\theta, \beta, B)$ is convex in β and θ jointly, $L^{\mathrm{vv}}_{m,\delta}(\theta, \beta, B)$ is jointly convex in (θ, B, β). Moreover, by Theorem 3.1 & 3.3 in (Ciliberto et al., 2015), when $\delta \to 0$, (θ, B) converges in Frobenius norm to (θ_*, B_*), where $(\theta_* B_*^\dagger, B_*)$ a minimiser of (17), where $B_*^\dagger$ denotes the pseudoinverse of B_*.

□

This theorem could therefore be used to construct an approach based on convex optimisation algorithms which are used iteratively for a decreasing sequence of penalisation parameters in order to converge to an optimum approaching the global optimum. However, this approach is limited to the case where all distributions are identical, and is hence not as widely applicable as Algorithm 2.

D Additional Results for the Experimental Study

This last Appendix provides additional details on our numerical experiments from Section 5.

D.1 Experimental Details of the Illustration Example

The experiment was replicated 100 times for all methods. The exact details of the implementation are as follows.

- CV
  - Sample size: 50.
  - Hyper-parameter tuning: batch size 5; learning rate 0.05; total number of epochs 30.
  - Base kernel: squared exponential kernel

- Optimisation: $\lambda = 0.001$; batch size is 5; learning rate is 0.001; total number of epochs 400.

- vv-CV (estimated B)
  - Sample size: $(50, 50)$ from (Π_1, Π_2) for (f_1, f_2).
  - Hyper-parameter tuning: batch size 5 (10 in total for (f_1, f_2)); learning rate 0.05; total number of epochs 30.
  - Base kernel: squared exponential kernel
  - Optimisation: $B^{(0)}$ is initialized at the identity matrix I_2. $\lambda = 0.001$; batch size is 5 (10 in total for (f_1, f_2)); learning rate is 0.001; total number of epochs 400.

D.2 Experimental Details of the Multifidelity Univariate Step Functions

The experiment was replicated 100 times for all methods. Details of their implementation is given below:

Squared-exponential kernel

- CV
  - Sample size: 40.
  - Base kernel: squared exponential kernel.
  - Hyper-parameter tuning: batch size is 10; learning rate 0.02; total number of epochs 15.
  - Optimisation: $\lambda = 1e-5$; batch size is 10; learning rate is $3e-4$; total number of epochs 400.

- vvCV (estimated B/fixed B)
  - Sample size: $(40, 40)$ from $(\mathcal{N}(0,1), \mathcal{N}(0,1))$ for (f_L, f_H).
  - Hyper-parameter tuning: batch size is 5 (10 in total for (f_L, f_H)); learning rate 0.02; total number of epochs 15.
  - Base kernel: squared exponential kernel.
  - Optimisation: When B is fixed, we set $B_{11} = B_{22} = 0.5, B_{12} = B_{21} = 0.01$; otherwise, $B^{(0)}$ is initialized at the identity matrix I_2. $\lambda = 1e-5$; batch size is 5 (10 in total for (f_L, f_H)); learning rate is $3e-4$; total number of epochs 400.

First-order polynomial kernel

- CV
  - Sample size: 40.
  - Base kernel: first order polynomial kernel.
  - Optimisation: $\lambda = 1e-5$; batch size is 10; learning rate is $3e-4$; total number of epochs 400.

- vv-CV (estimating B/fixed B)
  - Sample size: $(40, 40)$ from $(\mathcal{N}(0,1), \mathcal{N}(0,1))$ for (f_L, f_H).
  - Base kernel: first order polynomial kernel.
  - Optimisation: When B is fixed, we set $B_{11} = B_{22} = 0.5, B_{12} = B_{21} = 0.01$; otherwise, $B^{(0)}$ is initialized at the identity matrix I_2. $\lambda = 1e-5$; batch size is 5 (10 in total for (f_L, f_H)); learning rate is $3e-4$; total number of epochs 400.

Random variable	Distributions	Random variable	Distributions
r_w	$\mathcal{N}(0.1, 0.0161812^2)$	r	$\mathcal{N}(100, 0.01)$
T_u	$\mathcal{N}(89335, 20)$	T_l	$\mathcal{N}(89.55, 1)$
H_u	$\mathcal{N}(1050, 1)$	H_l	$\mathcal{N}(760, 1)$
L	$\mathcal{N}(1400, 10)$	K_w	$\mathcal{N}(10950, 30)$

Table 3: *Prior Distributions for the inputs of the Borehole function.*

D.3 Experimental Details of the Multifidelity Modelling of Waterflow

In this section, we provide details on the Borehole example from the main text, and provide complementary experiments. The distributions with respect to which the integral is taken is an eight-dimensional Gaussian with independent marginals provided in Table 3.

D.3.1 Experiment in the main text: Balanced vv-CVs

The number of replications is 100 for all methods. Details of their implementation is given below:

- Base kernel: Instead of using $k(x, x') = \exp(-\|x - x'\|_2^2/2\nu)$ with $l > 0$ which implicitly assumes that the length-scales are identical in all directions, we now allow that each dimension can have its own length-scale. That is,

  $$k(x, x') := \prod_{j=1}^d k_j(x_j, x'_j) \qquad \text{where} \qquad k_j(x_j, x'_j) = \exp\left(-\frac{(x_j - x'_j)_2^2}{2\nu_j}\right).$$

  Each of the components has its own length-scale $\nu_j > 0$ to be determined.

- Since $\pi(x) = \prod_{j=1}^d \pi_j(x_j)$, the score function is $\nabla_x \log \pi(x) = \left(\frac{\partial \log \pi_1(x)}{\partial x_1}, \ldots, \frac{\partial \log \pi_d(x)}{\partial x_d}\right)^\top$.

- Hyper-parameter tuning: batch size 5 (10 in total for (f_L, f_H)); learning rate of tuning 0.05; epochs of tuning 20.

- Optimisation (estimated B/pre-fixing B): When B is fixed, we set $B_{11} = B_{22} = 5e-4, B_{12} = B_{21} = 5e-5$; otherwise, $B^{(0)}$ is initialized at $1e-5 \times I_2$. $\lambda = 1e-5$; batch size 5 (10 in total for (f_L, f_H)); learning rate for the cases when sample sizes are $(10, 20, 50, 100, 150)$ are $(0.09, 0.06, 0.012, 0.0035, 0.002)$, respectively.

D.3.2 Additional Experiments: Unbalanced vv-CVs

In Figure 5, we present the results of vv-CVs when the sample sizes are unbalanced; that is, we have a different number of samples for the low-fidelity and high-fidelity models. The exact setup is given below, and we replicated the experiment 100 times.

- Sample size: m_H is fixed to be 20, while $m_L \in \{20, 40, 60\}$.

- Base kernel: product of squared exponential kernels. $k(x, x') := \prod_{j=1}^d k_j(x_j, x'_j)$, where each $k_j(x_j, x'_j) = \exp(-(x_j - x'_j)_2^2/2\nu_j)$ has its own length-scale $\nu_j > 0$ to be determined.

- Hyperparameter tuning: batch size of tuning 5 (10 in total for (f_L, f_H)); learning rate of tuning is 0.05; epochs of tuning is 20.

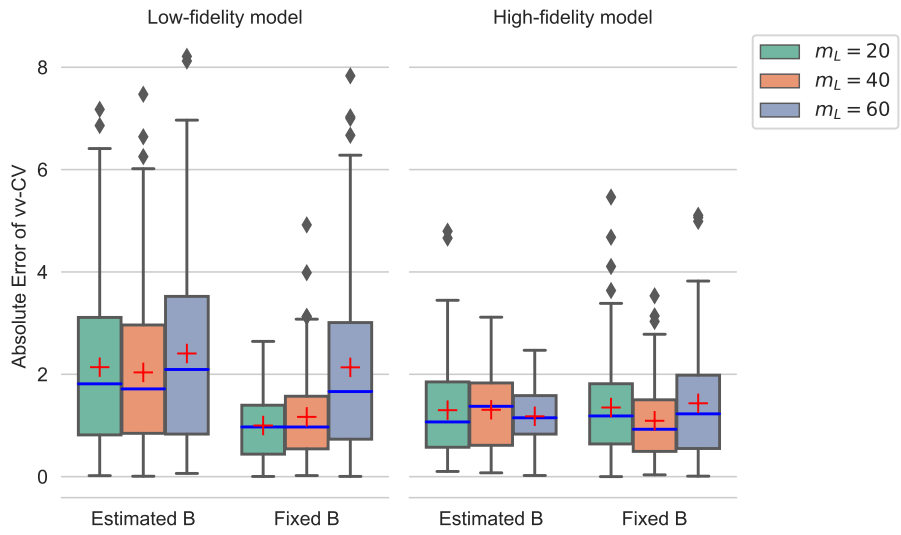

Figure 5: Performance of vv-CVs with unbalanced sample sizes. Here we fix $m_H = 20$, and changing m_L to be $20, 40, 60$. Each experiment is repeated 100 times.

- Optimisation (estimated B/pre-fixing B): When B is fixed, we set $B_{11} = B_{22} = 5e-4, B_{12} = B_{21} = 5e-5$; otherwise, $B^{(0)}$ is initialized at $1e-5 \times I_2$. $\lambda = 1e-5$; learning rate is $(0.06, 0.04, 0.02)$ when $m_L \in \{20, 40, 60\}$, respectively.

Interestingly, we notice that not much is gained when increasing the number of samples for the low-fidelity model. In fact, in the case of a fixed B, the performance tends to decrease with a larger m_L. This is likely due to the "negative transfer" phenomenon which is well-known in machine learning. This phenomenon can occur when two tasks are not similar enough to provide any gains in accuracy. In this case, there is clearly no advantage in using a larger m_L since this increases computational cost and does not provide any gains in accuracy.

D.4 Experimental Details of the Computation of the Model Evidence through Thermodynamic Integration

To implement our vv-CVs, we need to derive the corresponding score functions. For a power posterior, the score function is of the form:

$$\nabla_\theta \log p(\theta|y,t) = t\nabla_\theta \log p(y|\theta) + \nabla_\theta \log p(\theta)$$

where $\nabla_\theta \log p(\theta)$ is the score function corresponding to the prior. In our case, the prior is a log-normal distribution $\log\theta \sim \mathcal{N}(\mu, \sigma^2)$ (where $\sigma = 0.25$), and its score function is given by:

$$\nabla_\theta \log p(\theta) = -\tfrac{1}{\theta} - \tfrac{\log\theta - \mu}{x\sigma^2}.$$

The score functions for all temperatures are plotted in Figure 6; as observed, temperatures consecutive score functions are very similar to one another.

In order to keep the computational cost manageable, we split the $T = 62$ integration problems into groups of closely related problems. In particular, we jointly estimate the means in terms of 4 consecutive temperatures on the ladder (group 1 is $\mu_1, \mu_2, \mu_3, \mu_4$, group 2 is $\mu_5, \mu_6, \mu_7, \mu_8$, etc...). Since 31 is not divisible by 4, our last group consists of three means $\mu_{29}, \mu_{30}, \mu_{31}$. Then, the same approach is taken to create groups of 4 (or 3 for the last group) variances.

The number of replications was 20 for each method. Details are given below:

- CV
  - Base kernel: Preconditioned squared-exponential kernel (Oates et al., 2017).
  - Hyperparameter tuning: we use the values $(0.1, 3)$ in (Oates et al., 2017).
  - Optimisation: $\lambda = 1e-3$; batch size is 5; total number of epochs is 400.
- vv-CV(estimated B)
  - Base kernel: Preconditioned squared-exponential kernel (Oates et al., 2017).
  - Hyperparameter tuning: we use the values $(0.1, 3)$ in (Oates et al., 2017).
  - Optimisation: $\lambda = 1e-3$; batch size is 5; learning rate is 0.01; number of epochs is 400.

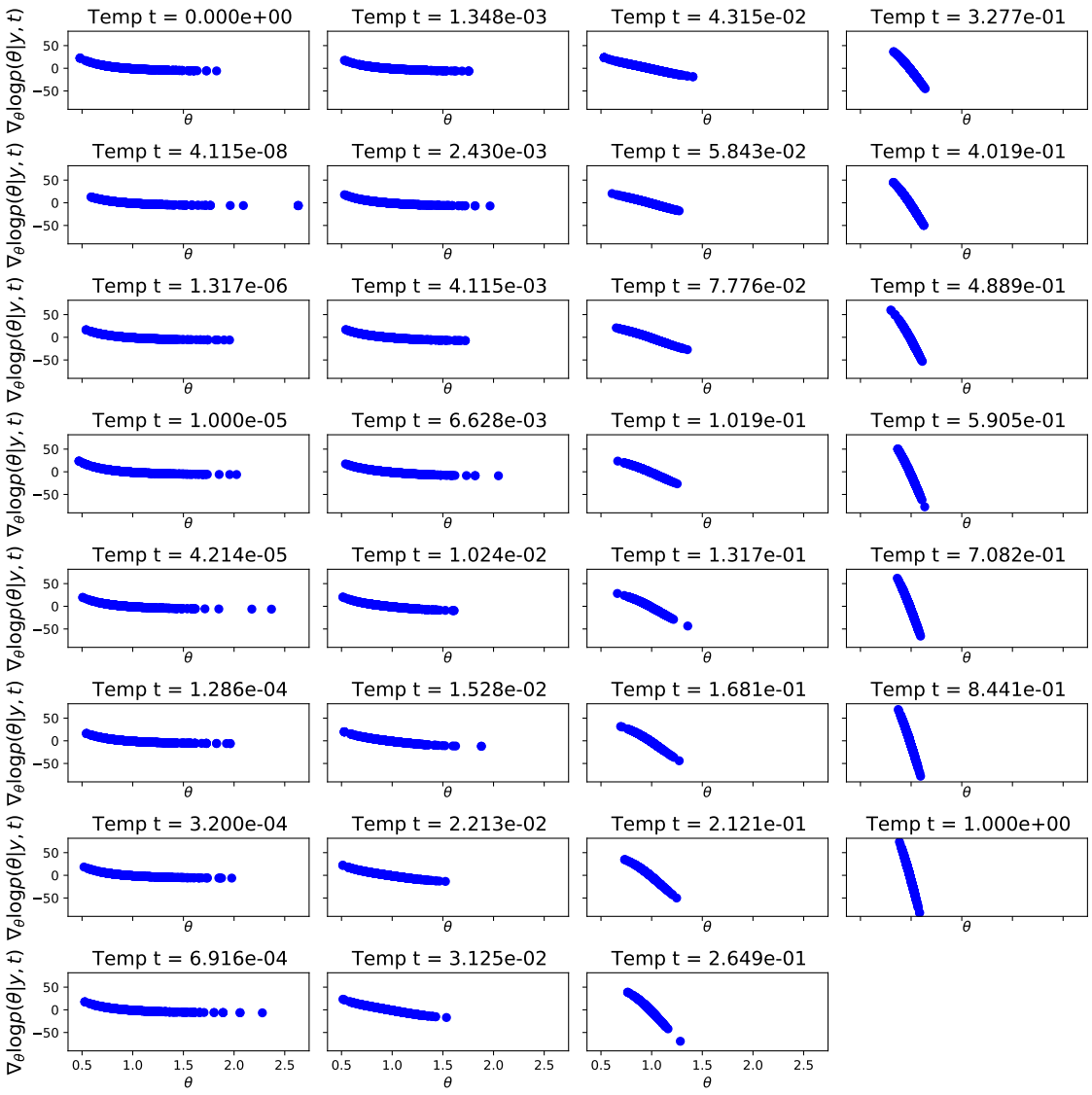

Figure 6: Score functions corresponding to the power posteriors at different temperatures on the temperature ladder.