Journal of Machine Learning Research 25 (2024) 1-61 Submitted 11/23; Revised 7/24; Published 9/24

Boundary constrained Gaussian processes for robust physics-informed machine learning of linear partial differential equations

David Dalton DAVID.DALTON@GLASGOW.AC.UK
Alan Lazarus ALAN.LAZARUS@GLASGOW.AC.UK
Hao Gao HAO.GAO@GLASGOW.AC.UK
Dirk Husmeier DIRK.HUSMEIER@GLASGOW.AC.UK
School of Mathematics and Statistics
University of Glasgow
Glasgow G12 8QQ, UK

Editor: Jean-Philippe Vert

Abstract

We introduce a framework for designing boundary constrained Gaussian process (BCGP) priors for exact enforcement of linear boundary conditions, and apply it to the machine learning of (initial) boundary value problems involving linear partial differential equations (PDEs). In contrast to existing work, we illustrate how to design boundary constrained mean and kernel functions for all classes of boundary conditions typically used in PDE modelling, namely Dirichlet, Neumann, Robin and mixed conditions. Importantly, this is done in a manner which allows for both forward and inverse problems to be naturally accommodated. We prove that the BCGP kernel has a universal representational capacity under Dirichlet conditions, and establish a formal equivalence between BCGPs and boundary-constrained neural networks (BCNNs) of infinite width. Finally, extensive numerical experiments are performed involving several linear PDEs, the results of which demonstrate the effectiveness and robustness of BCGP inference in the presence of sparse, noisy data.

Keywords: Physics-informed machine learning, Gaussian processes, partial differential equations, boundary-value problems, inverse problems

1. Introduction

Physics-informed machine learning (PIML) is a rapidly developing field which integrates data-based machine learning approaches with physics-based mathematical methods (Karniadakis et al., 2021). A PIML model of a physical system leverages observational data with known physical principles, which can include, for example, boundary constraints, conservation laws and partial differential equations (PDEs). Physics-informed approaches can offer more robust and interpretable predictions than purely data-based approaches, in addition to delivering insights and inference about the system of interest that would not be possible without accounting for domain-specific information. Consequently, PIML has become one of the most topical research areas in computational physics and machine learning, with

©2024 David Dalton, Alan Lazarus, Hao Gao and Dirk Husmeier.

License: CC-BY 4.0, see https://creativecommons.org/licenses/by/4.0/. Attribution requirements are provided at http://jmlr.org/papers/v25/23-1508.html.

applications in a wide range of disciplines. This includes quantum chemistry (Pun et al., 2019), solid mechanics (Nguyen-Thanh et al., 2020), fluid dynamics (Cai et al., 2021), soft-tissue mechanics (Dalton et al., 2023), and climate modelling (Lütjens et al., 2021), to give some examples.

Gaussian process regression (GPR) is one machine learning framework that has found application in the context of PIML (Raissi et al., 2017). GPR can be especially effective for data that is limited or expensive to obtain, and it offers well-calibrated predictive uncertainty estimates, which may be essential for scientific applications. In this work, we will consider the application of GPR to physical systems subject to linear boundary and linear PDE constraints. GPR is particularly effective in this case, as it allows for seamless integration of observational data with linear PDE information, which enables efficient joint inference of any unknown PDE parameters together with the solution function itself. In this work, our objective is to expand upon the existing literature and design a GPR approach which also seamlessly integrates known boundary conditions into the inferential framework.

1.1 Related Work

The closure of GPs under linear operators has been long understood (Adler, 2010), and was recently formalised by Pförtner et al. (2024). Early work leveraged this property of GPs to build models to incorporate derivative information (Solak et al., 2002), learn latent forces (Alvarez et al., 2013), and model ordinary and partial differential equations (Graepel, 2003; Melkumyan, 2012; Dondelinger et al., 2013). Of specific relevance to the material presented here is the physics-informed Gaussian process (PIGP) inference framework introduced in the seminal work of Raissi et al. (2017), in which it was outlined how noise levels, any unknown PDE parameters and the function of interest itself could be jointly inferred using GPR.

A recent thread of research in the GPR literature has focused on the design of models that exactly satisfy known boundary conditions *a-priori*, which we refer to as *boundary-constrained* methods. In the context of data-driven surrogate modelling, approaches have been proposed which design non-stationary mean and covariance functions so that boundary conditions on the value of the unknown function itself are satisfied (Tan, 2016; Li and Tan, 2022). Other approaches include the design of a bespoke kernel in terms of hyperbolic sine and cosine functions so that Dirichlet conditions are satisfied (Ding et al., 2019), and the use of kernels derived from variational harmonic features which can be used to enforce Neumann and mixed boundary conditions in addition to Dirichlet conditions (Solin and Kok, 2019; Solin and Särkkä, 2020; Gulian et al., 2022). GPs can also be designed to satisfy other types of constraints (Swiler et al., 2020), including, for example, monotonicity (Riihimäki and Vehtari, 2010), bound (Jensen et al., 2013) and general linear inequality constraints (Da Veiga and Marrel, 2012), as well as certain linear PDEs (Harkonen et al., 2023) and boundary-value problems (BVPs) (Lange-Hegermann, 2021).

Physics-informed neural networks (PINNs) are an alternative paradigm for the machine learning of PDE systems (Raissi et al., 2019). While PINNs are considered to be a more "data-hungry" method than PIGPs, they enjoy the particular advantage of not being limited to the modelling of linear PDEs. Contemporaneous with the introduction of the above boundary-constrained methods in the PIGP literature has been the development of methods

for imposing the same types of constraints in PINNs. As we discuss in Section 4.2, these two threads of research are, in fact, highly analogous. Early work in PINNs used simple distance functions to enforce homogeneous Dirichlet conditions (Nguyen-Thanh et al., 2020), while more recent work has introduced techniques to handle more complex domains and more varied boundary conditions (Sheng and Yang, 2021; Sukumar and Srivastava, 2022; Liu et al., 2022).

1.2 Contributions

In this work, we introduce a framework for explicitly enforcing linear boundary conditions into the structure of a GP prior. This framework goes beyond existing approaches for Dirichlet boundary conditions (Tan, 2016; Li and Tan, 2022) to also consider more general Robin, Neumann and mixed boundary conditions. Importantly, time dependence, forward problems, and inverse problems can all be accommodated. We prove that the boundary-constrained kernel structure under Dirichlet conditions has a universal approximation property, and also introduce a theoretical link between the boundary-constrained PINN and PIGP literatures in the limit of an infinite wide network. We then conduct extensive numerical experiments involving multiple linear PDE systems, the results of which demonstrate that explicitly enforcing boundary conditions allows for more robust inference to be performed, when compared to methods which ignore boundary conditions (Raissi et al., 2017) or which introduce boundary information using a penalty approach (Zhang et al., 2022).

The paper is laid out as follows. Firstly, Section 2 presents background material on linear PDEs, Gaussian processes, distance functions, and solution structures for BVPs, before Section 3 describes our new framework for constructing BCGPs. Section 4 presents theoretical results, Section 5 details the numerical experiments we conducted to evaluate our BCGP framework, while Section 6 concludes.

2. Preliminaries

2.1 Definitions and Notation

In the below, Ω denotes a non-empty, connected, open bounded subset of $\mathbb{R}^D$, with boundary $\partial\Omega$ and closure (i.e. $\Omega \cup \partial\Omega$) the compact set $\bar{\Omega}$. In Section 2.3, $\mathcal{X}$ refers to a more general set on which a kernel can be defined. Finally, the space of all continuous functions on $\mathcal{X}$ is denoted by $\mathcal{C}(\mathcal{X})$.

2.2 Linear Partial Differential Equations

A linear partial differential equation (PDE) takes the form

$$\mathcal{L}^{\boldsymbol{\theta}}_{\boldsymbol{x},t}[u](\boldsymbol{x},t) = f(\boldsymbol{x},t),\ \boldsymbol{x} \in \Omega,\ t > 0, \tag{1}$$

in which $u : \bar{\Omega} \times [0,\infty) \to \mathbb{R}$ is the unknown solution function describing the behaviour of the system, $f : \Omega \times [0,\infty) \to \mathbb{R}$ is called the forcing term and is typically known, and $\mathcal{L}^{\boldsymbol{\theta}}_{\boldsymbol{x},t}$ is a linear differential operator (see Definition 17), parameterised by $\boldsymbol{\theta}$. Systems are often *spatio-temporal*, depending on both spatial location $\boldsymbol{x}$ and time t. For notational simplicity, however, for the remainder of Section 2 we do not make the temporal input explicit.

2.2.1 Boundary Conditions

Solutions to PDEs are not uniquely defined. To make a problem well posed, *boundary conditions* can be imposed, which specify the behaviour of the system on the boundary of the spatial domain, $\partial\Omega$. In this work, we will primarily consider *Dirichlet*, *Robin* and *mixed* boundary conditions. Periodic boundary conditions can be treated similarly to Dirichlet conditions, which we illustrate using a numerical example in Section 5.4.2. Other forms of boundary conditions can also be accommodated if they are linear in the solution field—for further discussion, see Section 2.5.4.

Dirichlet conditions specify the form of the solution function itself on the boundary as

$$\textbf{Dirichlet:} \quad u(\boldsymbol{x}) = b(\boldsymbol{x}), \ \boldsymbol{x} \in \partial\Omega, \tag{2}$$

where b is a known function. Robin conditions specify the form of a linear combination of the solution and its directional derivative outward of the boundary as

$$\textbf{Robin:} \quad \boldsymbol{n}(\boldsymbol{x}) \cdot \nabla u(\boldsymbol{x}) + a(\boldsymbol{x})u(\boldsymbol{x}) = h(\boldsymbol{x}), \ \boldsymbol{x} \in \partial\Omega, \tag{3}$$

where a and h are known functions and $\boldsymbol{n}(\boldsymbol{x})$ is the outward normal vector at $\boldsymbol{x} \in \partial\Omega$. Note that *Neumann* boundary conditions are recovered from Eq. (3) by setting $a = 0$.

Remark 1 *Robin boundary conditions can be defined such that Dirichlet conditions are also recovered as a special case. However, constructing solution spaces to explicitly satisfy Dirichlet conditions (see Eq. 26) is much simpler than for Robin/Neumann conditions (see Eq. 27), and for this reason, we keep the definitions separate here.*

Finally, if $\partial\Omega$ is decomposed into disjoint segments $\partial\Omega_1$ and $\partial\Omega_2$, mixed boundary conditions can be specified by imposing Dirichlet boundary conditions on the first segment and Robin boundary conditions on the second as

$$\textbf{Mixed:} \quad \begin{cases} u(\boldsymbol{x}) = b(\boldsymbol{x}), & \boldsymbol{x} \in \partial\Omega_1, \\ \boldsymbol{n}(\boldsymbol{x}) \cdot \nabla u(\boldsymbol{x}) + a(\boldsymbol{x})u(\boldsymbol{x}) = h(\boldsymbol{x}), & \boldsymbol{x} \in \partial\Omega_2. \end{cases} \tag{4}$$

Remark 2 *For notational simplicity in Sections 2.5 and 3, we assume that the functions a, b and h in Eqs. (2)-(4) are known in the interior of the domain Ω in addition to the boundary $\partial\Omega$. If this is not the case, then interpolation can be used—see Remark 10 for more details.*

When considered together, a PDE and a set of boundary conditions are called a *boundary value problem* (BVP). If a PDE is time-dependent, an *initial boundary value problem* (IBVP) can be defined, whereby an additional constraint is placed on the initial condition of the system. Initial conditions can be treated analogously to boundary conditions, which we demonstrate using several experiments involving IBVPs in Section 5.

2.3 Gaussian process regression

Regression refers to the process of using a finite set of possibly noise corrupted data to perform inference about an unknown function of interest, which we denote $u : \mathcal{X} \to \mathbb{R}$. Specifically, suppose a data set of input-output pairs $(\boldsymbol{x}_u^{(i)}, y_u^{(i)})$ has been observed for $i = 1, \ldots, N_u$, with the following noise model assumed for each observation:

$$y_u^{(i)} = u(\boldsymbol{x}_u^{(i)}) + \epsilon_u^{(i)} \text{ with } \boldsymbol{x}_u^{(i)} \in \mathcal{X} \text{ and } \epsilon_u^{(i)} \sim \mathcal{N}(0, \sigma_u^2). \tag{5}$$

The goal of regression is to use these observations $\boldsymbol{y}_u = (y_u^{(1)}, \ldots, y_u^{(N_u)})^\top$ to learn the underlying function, allowing its output $\boldsymbol{u}_* = (u_*^{(1)}, \ldots, u_*^{(N_*)})^\top$ to be predicted at any test points of interest $\boldsymbol{x}_*^{(1)}, \ldots, \boldsymbol{x}_*^{(N_*)}$. In this section, we discuss the essential concepts of a non-parametric Bayesian method for performing regression based on *Gaussian processes* (GPs). For further details, we direct the reader to Rasmussen and Williams (2006), while Kanagawa et al. (2018) offers a more technical introduction.

Central to the formulation of GPs are *Mercer kernels* (Kanagawa et al., 2018, Definition 2.1), which we henceforth refer to as simply *kernels*.

Definition 3 (Mercer kernel) *Let $\mathcal{X}$ be a non-empty set. A symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a Mercer kernel if, for any $N \in \mathbb{N}, c^{(1)}, \ldots, c^{(N)} \subset \mathbb{R}$ and $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)} \subset \mathcal{X}$, we have*

$$\sum_{i=1}^{N} \sum_{j=1}^{N} c^{(i)} c^{(j)} k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) \geq 0. \tag{6}$$

This property of kernels is referred to as *positive-definiteness*. When the input space is Euclidean, the prototypical example of a kernel is the *squared-exponential*.

Example 1 (Squared-exponential kernel) *Let $\mathcal{X} \subset \mathbb{R}^D$. Given $\tau, \ell > 0$, a squared-exponential kernel k_{SE} is defined as*

$$k_{SE}(\boldsymbol{x}, \boldsymbol{x}'; \tau, \ell) \triangleq \tau^2 \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2\ell^2} \right), \quad \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}. \tag{7}$$

A kernel will, in general, depend on a set of hyperparameters, which we denote as $\boldsymbol{\xi}$. With the squared exponential kernel, we have $\boldsymbol{\xi} = (\tau, \ell)$, where τ is called the amplitude and ℓ the lengthscale. For ease of notation we usually leave this dependence implicit, i.e. writing $k(\boldsymbol{x}, \boldsymbol{x}')$ in place of $k(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\xi})$.

We are now ready to define a GP (Kanagawa et al., 2018, Definition 2.2), where kernels allow for the covariance between random function outputs to be specified, and are hence often called covariance functions in this context.

Definition 4 (Gaussian process) *Let $\mathcal{X}$ be a non-empty set, $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a Mercer kernel and $m : \mathcal{X} \to \mathbb{R}$ any function. Then a random function $u : \mathcal{X} \to \mathbb{R}$ is called a Gaussian process (GP) with mean m and covariance k, which we denote*

$$u(\boldsymbol{x}) \sim \mathcal{GP}\left(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')\right), \tag{8}$$

if, for any finite collection of inputs $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)} \in \mathcal{X}$*, the distribution of the corresponding outputs* $\boldsymbol{u} = (u(\boldsymbol{x}^{(1)}), u(\boldsymbol{x}^{(2)}), ..., u(\boldsymbol{x}^{(N)}))^\top$ *is Gaussian, that is* $p(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{m}, \mathbf{K})$*, where the mean vector* $\boldsymbol{m}$ *and covariance matrix* $\mathbf{K}$ *are found by evaluating the mean and covariance functions from Eq.* (8) *as follows:*

$$\mathbb{E}\Big(u(\boldsymbol{x}^{(i)})\Big) = m^{(i)} = m(\boldsymbol{x}^{(i)}), \tag{9}$$

$$\mathrm{Cov}\Big(u(\boldsymbol{x}^{(i)}), u(\boldsymbol{x}^{(j)})\Big) = K^{(i,j)} = k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}). \tag{10}$$

Performing *Gaussian process regression* (GPR) begins with the assumption that the unknown function of interest u follows a GP with specified mean and covariance functions, which we denote m_u and k_{uu} respectively. This assumption, along with the independent and identically distributed Gaussian observation noise model (see Eq. 5), implies that the training observations $\boldsymbol{y}_u \in \mathbb{R}^{N_u \times 1}$ and unknown test outputs $\boldsymbol{u}_* \in \mathbb{R}^{N_* \times 1}$ have the following joint Gaussian distribution:

$$\begin{bmatrix} \boldsymbol{y}_u \\ \boldsymbol{u}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{m}_u \\ \boldsymbol{m}_* \end{bmatrix}, \begin{bmatrix} \mathbf{K}_{uu} + \sigma_u^2 \mathbf{I}_{N_u} & \mathbf{K}_{u*} \\ \mathbf{K}_{*u} & \mathbf{K}_{**} \end{bmatrix} \right). \tag{11}$$

Here, $\boldsymbol{m}_u \in \mathbb{R}^{N_u \times 1}$ and $\boldsymbol{m}_* \in \mathbb{R}^{N_* \times 1}$ are found using the mean function m_u as in Eq. (9), while $\mathbf{K}_{uu} \in \mathbb{R}^{N_u \times N_u}$ and $\mathbf{K}_{**} \in \mathbb{R}^{N_* \times N_*}$ are found using the covariance function k_{uu} as in Eq. (10). $\mathbf{I}_{N_u} \in \mathbb{R}^{N_u \times N_u}$ is the identity matrix, while the off diagonal terms are found as $K_{u*}^{(j,i)} = K_{*u}^{(i,j)} = k_{uu}(\boldsymbol{x}_*^{(i)}, \boldsymbol{x}_u^{(j)})$.

Given this joint distribution, the regression task reduces to evaluating $p(\boldsymbol{u}_* \,|\, \boldsymbol{y}_u)$, the conditional distribution of the unknown test data given the observed data. The properties of the multivariate Gaussian are such that this distribution is again Gaussian (Murphy, 2023, Section 2.3.1.5).

Proposition 5 (Posterior Predictive Gaussian Distribution) *Let* $\boldsymbol{u}_*$ *and* $\boldsymbol{y}_u$ *be jointly Gaussian distributed as in Eq. (11). Then* $p(\boldsymbol{u}_* \,|\, \boldsymbol{y}_u) = \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$ *with*

$$\begin{aligned} \boldsymbol{\mu}_* &= \boldsymbol{m}_* + \mathbf{K}_{u*}^\top \left(\mathbf{K}_{uu} + \sigma_u^2 \mathbf{I}_{N_u}\right)^{-1} (\boldsymbol{y}_u - \boldsymbol{m}_u), \\ \boldsymbol{\Sigma}_* &= \mathbf{K}_{**} - \mathbf{K}_{u*}^\top \left(\mathbf{K}_{uu} + \sigma_u^2 \mathbf{I}_{N_u}\right)^{-1} \mathbf{K}_{u*}. \end{aligned} \tag{12}$$

Carrying out GPR requires then the mean and covariance functions to be specified. A common choice in practice is to use a zero-mean function and a squared exponential kernel. Various approaches can then be used to fit any tunable mean/kernel parameters, including Markov-chain Monte-Carlo, variational inference and empirical Bayesian methods respectively.

2.3.1 Linear Partial Differential Equation Constraints

In this work, we consider processes for which linear PDE information is available. Specifically, suppose that, in addition to observations $\boldsymbol{y}_u \in \mathbb{R}^{N_u \times 1}$ in u-space (see Eq. 5), observations in $\boldsymbol{y}_f \in \mathbb{R}^{N_f \times 1}$ in f-space (see Eq. 1) are available with observation model

$$y_f^{(i)} = f(\boldsymbol{x}_f^{(i)}) + \epsilon_f^{(i)} \text{ with } \epsilon_f^{(i)} \sim \mathcal{N}(0, \sigma_f^2), \tag{13}$$

for all $i = 1, \ldots, N_f$. We seek to incorporate $\boldsymbol{y}_u$ and $\boldsymbol{y}_f$ into a joint inference framework using GPs. To do so, we employ the GPR algorithm introduced by Raissi et al. (2017), which is generalised in this section to allow for non-zero mean function. As with usual GPR, the algorithm begins by assuming that $u(\boldsymbol{x})$ follows a GP, i.e.

$$u(\boldsymbol{x}) \sim \mathcal{GP}\left(m_u(\boldsymbol{x}), k_{uu}(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\xi})\right). \tag{14}$$

For clarity, we use subscripts in this section to denote the spaces in which observations are made, so that $m_g(\boldsymbol{x}) = \mathbb{E}\left(g(\boldsymbol{x})\right)$ and $k_{gh}(\boldsymbol{x}, \boldsymbol{x}') = \mathrm{Cov}(g(\boldsymbol{x}), h(\boldsymbol{x}'))$, for $g, h \in \{u, f\}$. Additionally, we make explicit the dependence of the kernel on any hyperparameters $\boldsymbol{\xi}$ (we assume the mean function has no trainable parameters).

The key insight required here is that GPs are closed under linear operations (Pförtner et al., 2024, Corollary 2.). This means that our GP prior assumption on $u(\boldsymbol{x})$ in Eq. (14) implies that

$$\mathcal{L}_{\boldsymbol{x}}^{\boldsymbol{\theta}}[u](\boldsymbol{x}) = f(\boldsymbol{x}) \sim \mathcal{GP}\left(m_f(\boldsymbol{x}; \boldsymbol{\theta}), k_{ff}(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\xi}, \boldsymbol{\theta})\right). \tag{15}$$

Furthermore, we have the following fundamental relationship between the mean and covariance functions of the two processes (Raissi et al., 2017):

$$\begin{aligned} m_f(\boldsymbol{x}; \boldsymbol{\theta}) &= \mathcal{L}_{\boldsymbol{x}}^{\boldsymbol{\theta}} m_u(\boldsymbol{x}), \\ k_{ff}(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\xi}, \boldsymbol{\theta}) &= \mathcal{L}_{\boldsymbol{x}}^{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{x}'}^{\boldsymbol{\theta}} k_{uu}\left(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\xi}\right). \end{aligned} \tag{16}$$

Similarly, the cross-covariance between the observations of the two processes are found as

$$\begin{aligned} k_{uf}(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\xi}, \boldsymbol{\theta}) &= \mathcal{L}_{\boldsymbol{x}'}^{\boldsymbol{\theta}} k_{uu}(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\xi}), \\ k_{fu}(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\xi}, \boldsymbol{\theta}) &= \mathcal{L}_{\boldsymbol{x}}^{\boldsymbol{\theta}} k_{uu}(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\xi}) \end{aligned} \tag{17}$$

Note that we are assuming here that the chosen kernel k_{uu} is sufficiently smooth for the PDE operator to be applied, an issue which is discussed in Section 2.2 of Adler (2010).

The above properties of GPs along with our Gaussian noise assumptions for observations $\boldsymbol{y}_u$ in u-space and observations $\boldsymbol{y}_f$ in f-space imply the following joint distribution for the observed data.

Proposition 6 (Joint distribution of $\boldsymbol{y}_u$ and $\boldsymbol{y}_f$) *Let u follow a Gaussian process as in Eq. (14), with $\mathcal{L}_{\boldsymbol{x}}^{\boldsymbol{\theta}}[u] = f$. Assume $\boldsymbol{y}_u \in \mathbb{R}^{N_u \times 1}$ has been observed with noise model given in Eq. (5), and $\boldsymbol{y}_f \in \mathbb{R}^{N_f \times 1}$ has been observed with noise model in Eq. (13). Then, $\boldsymbol{y}_u$ and $\boldsymbol{y}_f$ follow a joint normal distribution $p(\boldsymbol{y}_u, \boldsymbol{y}_f; \boldsymbol{\xi}, \boldsymbol{\theta}, \sigma_u^2, \sigma_f^2)$ of the form*

$$\begin{bmatrix} \boldsymbol{y}_u \\ \boldsymbol{y}_f \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{m}_u \\ \boldsymbol{m}_f \end{bmatrix}, \mathbf{K}_{\boldsymbol{yy}}\right), \quad \textit{with} \quad \mathbf{K}_{\boldsymbol{yy}} = \begin{bmatrix} \mathbf{K}_{uu} + \sigma_u^2 \mathbf{I}_{N_u} & \mathbf{K}_{uf} \\ \mathbf{K}_{fu} & \mathbf{K}_{ff} + \sigma_f^2 \mathbf{I}_{N_f} \end{bmatrix}, \tag{18}$$

where $m_g^{(i)} = m(\boldsymbol{x}_g^{(i)})$ and $K_{gh}^{(i,j)} = k_{gh}(\boldsymbol{x}_g^{(i)}, \boldsymbol{x}_h^{(i)})$ for $g, h \in \{u, f\}$, while $\mathbf{I}_N$ is the identity matrix of dimension N, for $N \in \{N_u, N_f\}$.

A major contribution of Raissi et al. (2017) was to recognise that the parameters $\boldsymbol{\theta}$ of the linear PDE operator are turned into hyperparameters of m_f, k_{ff}, k_{uf} and k_{fu} (note how these functions depend on $\boldsymbol{\theta}$ in Eqs. 16-17). Therefore, if these parameters are unknown,

they can be inferred in the same manner as the original kernel parameters $\boldsymbol{\xi}$. Raissi et al. (2017) perform inference by maximisation of $p(\boldsymbol{y}_u, \boldsymbol{y}_f; \boldsymbol{\xi}, \boldsymbol{\theta}, \sigma_u^2, \sigma_f^2)$ as follows:

$$\{\hat{\boldsymbol{\xi}}, \hat{\boldsymbol{\theta}}, \hat{\sigma}_u^2, \hat{\sigma}_f^2\} = \underset{\boldsymbol{\xi}, \boldsymbol{\theta}, \sigma_u^2, \sigma_f^2}{\operatorname{argmax}} \log p(\boldsymbol{y}_u, \boldsymbol{y}_f; \boldsymbol{\xi}, \boldsymbol{\theta}, \sigma_u^2, \sigma_f^2), \tag{19}$$

where the log is taken for reasons of numerical stability.

When viewed as a function of the tunable parameters given fixed observations as in Eq. (19) above, $p(\boldsymbol{y}_u, \boldsymbol{y}_f; \boldsymbol{\xi}, \boldsymbol{\theta}, \sigma_u^2, \sigma_f^2)$ is called the *marginal likelihood* or *evidence* of the observed data. Maximisation of the log marginal likelihood is a common method for training GPs, as this quantity balances a trade-off between model fit and complexity (Murphy, 2023, Section 3.8.1). This can be seen by writing the objective function out in explicit form (Murphy, 2023, Section 18.3.5), where we suppress dependence on the parameters $\boldsymbol{\xi}, \boldsymbol{\theta}, \sigma_u^2, \sigma_f^2$ and ignore any added constants for notational convenience:

$$\log p(\boldsymbol{y}_u, \boldsymbol{y}_f) = -\overbrace{\frac{1}{2}\begin{bmatrix} \boldsymbol{y}_u - \boldsymbol{m}_u \\ \boldsymbol{y}_f - \boldsymbol{m}_f \end{bmatrix}^\top \mathbf{K}_{\boldsymbol{yy}}^{-1} \begin{bmatrix} \boldsymbol{y}_u - \boldsymbol{m}_u \\ \boldsymbol{y}_f - \boldsymbol{m}_f \end{bmatrix}}^{\text{Data-Fit}} - \overbrace{\frac{1}{2}\log|\mathbf{K}_{\boldsymbol{yy}}|}^{\text{Regularisation}}. \tag{20}$$

The first term above is the squared distance between the observed and predicted values under the Mahalanobis metric (Murphy, 2023, Section 2.3.1). This is a *data-fit* term, as it favours models which better fit the observations. The second term is the log determinant of $\mathbf{K}_{\boldsymbol{yy}}$, which is a measure of model complexity, since smoother functions will yield smaller determinants. Therefore, this is a *regularisation* term, as it favours more simple models. By balancing fit and complexity, the marginal likelihood can enable effective model training even in the low-data regime. This is discussed further in the context of a numerical experiment in Section 5.2.1.

Once the GP has been trained, prediction at any new test points of interest follows almost the same formulas as presented in Eq. (12). The only difference is that any terms involving the training data now have a block structure to account for the fact that observations are available in two different spaces. For instance, $\mathbf{K}_{u*}^\top$ in Eq. (12) is replaced with a matrix of the form $[\mathbf{K}_{u*}^\top \ \mathbf{K}_{f*}^\top]$.

2.4 Approximate Distance Functions

Given a compact set $\bar{\Omega} \subset \mathbb{R}^D$ of the form specified in Section 2.1, there exists a continuous distance function $d : \bar{\Omega} \to \mathbb{R}$ given by

$$d(\boldsymbol{x}) \equiv \inf_{\boldsymbol{x}' \in \partial\Omega} \|\boldsymbol{x} - \boldsymbol{x}'\|_2, \tag{21}$$

which gives the shortest distance from any point $\boldsymbol{x}$ to the boundary $\partial\Omega$ (Shapiro, 2007, Section 2.2). Distance functions have wide applicability in modelling and engineering problems, including, for instance, computer-aided design (Frisken and Perry, 2006), robotics (Gilbert and Johnson, 1985), computer vision (Perera et al., 2015) and medical imaging (Felipe et al., 2009). In this work, we make use of (approximate) distance functions to construct GPs which lie with the solution space of BVPs. Intuitively, a distance function

will collapse the variance of the GP to zero at a boundary where the solution function (or a linear transformation of it) is prescribed—for more details, see Sections 2.5 and 3 below.

There are, however, two immediate drawbacks of using the exact distance function d from Eq. (21) in this context. Firstly, the precise functional form of d may not be known for complex geometries, and evaluating a numerical approximation can be computationally expensive (Sukumar and Srivastava, 2022). Secondly, d will not be differentiable at points equidistant from more than one point at the boundary (see $x = 0.5$ in Figure 1 a for example). For these reasons, in practical applications the exact distance function is often replaced by a smoothed *approximate distance function* (ADF), which we denote as ϕ in this work. Like the exact distance function, an ADF will equal zero for points on the boundary $\partial\Omega$ and be positive otherwise, i.e.

$$\textbf{ADF:} \quad \begin{cases} \phi(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \partial\Omega, \\ \phi(\boldsymbol{x}) > 0, & \boldsymbol{x} \in \Omega, \end{cases} \tag{22}$$

while also satisfying any desired smoothness properties.

As is detailed in Section 2.5 below, ADFs are sufficient for constructing solution spaces which exactly satisfy Dirichlet boundary conditions. For Robin conditions, however, an additional normalisation condition is required, which states that the outward normal derivative of the ADF must be exactly minus one on the boundary. This can be expressed as

$$\textbf{Normalised ADF:} \quad \nabla\phi(\boldsymbol{p}) = -\boldsymbol{n}(\boldsymbol{p}) \tag{23}$$

for all regular points $\boldsymbol{p} \in \partial\Omega$, i.e. points where the unit outward normal vector $\boldsymbol{n}(\boldsymbol{p})$ is well-defined. Any ADF with a non-vanishing gradient near $\partial\Omega$ can be normalised (Biswas and Shapiro, 2004, Eq. A.1).

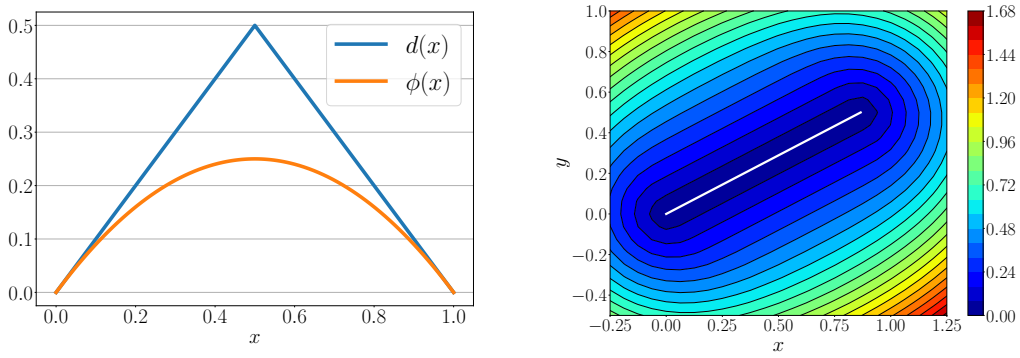
Remark 7 *The concept of a normalised ADF can be extended to higher orders (Biswas and Shapiro, 2004, Eq. 1). This is not necessary in the present work, however, as we only consider boundary conditions which are first order in the solution function u.*

Example 2 *If $\bar{\Omega} = [0, 1]$, then the exact distance function has the form $d(x) = \mathrm{Min}(x, 1 - x)$. An ADF which is normalised is given by $\phi(x) = x(1 - x)$. Both of these functions are plotted in Figure 1 (a)—note that d is not differentiable at $x = 0.5$, while ϕ is differentiable everywhere.*

2.4.1 Methods for constructing ADFs

As shown in Example 2, it is simple to design an ADF for a 1D interval. This construction easily extends to higher dimensions if the domain has a grid-like shape—see for instance Figure 2. For domains of more general shape, however, constructing an ADF is non-trivial. Here, we briefly summarise some construction techniques.

Interpolation methods are one way in which ADFs can be defined for domains of arbitrary shape. For example, Freytag et al. (2006) used B-splines fitted to a point cloud of sampled distance values to construct an ADF, while Sheng and Yang (2021) used radial basis function interpolation to construct individual distance functions between pairs of points



(a) d and ϕ for $\bar{\Omega} = [0, 1]$ (b) ADF to line segment

Figure 1: Panel (a) shows exact distance function d and ADF ϕ for $\bar{\Omega} = [0, 1]$. Panel (b) shows the density plot of an ADF to a line segment (the white line) in $\mathbb{R}^2$.

on the boundary, before joining them into a single ADF using multiplication (Sheng and Yang, 2021, Eq. 10). Particular advantages of these methods include their ease of implementation and computational efficiency. However, as noted in (Biswas and Shapiro, 2004), it can be difficult to normalise the ADFs found by interpolation because the gradient of the ADF may have flat spots near $\partial\Omega$.

Level set methods (Osher et al., 2004) are an alternative approach for constructing ADFs, in which the domain $\bar{\Omega}$ is implicitly represented by a level set function—the boundary is represented by the zero level of the function, and the interior as those points with positive level. The main drawback of this approach is that the level set function is defined implicitly, and its evaluation requires the repeated solution of a Hamilton Jacobi PDE (Nilsson, 2009, Eq. 4.7).

ADFs can also be constructed using the theory of mean-value interpolation, the objective of which is to extend a function defined on the boundary of a spatial domain to its interior (Dyken and Floater, 2009). An exact formula for performing this interpolation in general dimensions is given in (Bruvoll and Floater, 2010, Eq. 1). It turns out that the reciprocal of the weight function in this formula will be an ADF for the domain. Furthermore, this ADF can be normalised (Bruvoll and Floater, 2010, Theorem 5) and will not have a flat gradient in the interior of the domain (Bruvoll and Floater, 2010, Section 4). A disadvantage of this approach is that the ADF is only defined implicitly via an integral equation that needs to be solved, and furthermore, the ADF can have derivative discontinuities where neighbouring boundary segments meet.

2.4.2 Constructing ADFs for segmented boundaries

In practice, the boundary of the domain $\partial\Omega$ may be represented in terms of M disjoint segments as $\partial\Omega = \cup_{i=1}^{M} \partial\Omega_i$. In this case, the theory of R-functions (reviewed in detail by Shapiro, 2007) can be used to construct a global ADF for $\partial\Omega$ when individual ADFs ϕ_i are known for each segment. Briefly, an R-function is a real-valued function whose sign is

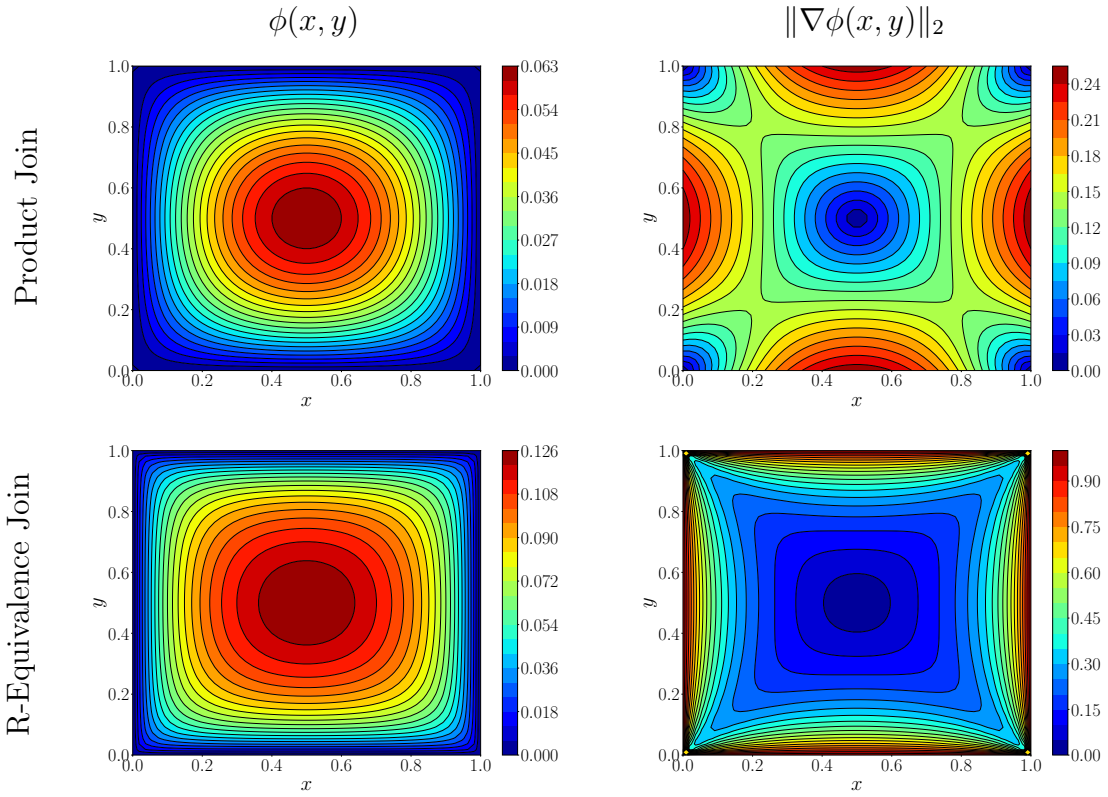


Figure 2: Comparison of ADFs for domain $\bar{\Omega} = [0, 1]^2$ obtained using by applying the product (Eq. 24) and R-equivalance (Eq. 25) joins respectively to the functions $\phi_1(x, y) = x$, $\phi_2(x, y) = 1 - x$, $\phi_3(x, y) = y$ and $\phi_4(x, y) = 1 - y$. The left column shows the ADFs and the right column the norm of the gradient of the ADFs.

determined solely by the sign of its inputs, which can be used to compose the individual ADFs into a single global ADF (Biswas and Shapiro, 2004, Section 2.2).

To illustrate this idea, we introduce two examples. Firstly, consider the simple grid domain $\bar{\Omega} = [0, 1]^2$, whose boundary can be decomposed into $M = 4$ segments in which $\partial\Omega_1$ is where $x = 0$, $\partial\Omega_2$ is where $x = 1$, $\partial\Omega_3$ is where $y = 0$ and $\partial\Omega_4$ is where $y = 1$. For the second example, consider the intricate polygon in Figure 3 (a), which is a map of County Laois in Ireland. Here, the boundary is represented by $M = 503$ individual line segments.

Given a set of M segments, an ADF for the entire boundary can be constructed in two stages. The initial stage requires that an individual ADF ϕ_i is defined for each boundary segment $\partial\Omega_i$. For the grid domain example, these can be specified simply as $\phi_1(x, y) = x$, $\phi_2(x, y) = 1 - x$, $\phi_3(x, y) = y$ and $\phi_4(x, y) = 1 - y$. For the polygon in Figure 3 (a), we use the trimming approach described in (Biswas and Shapiro, 2004, Section 2.1) to define an ADF for each individual line segment in the boundary, the precise details of which are provided in Appendix A.4 (see Eq. 72 specifically). For illustration, we provide a heatmap

in Figure 1 (b) of the distance field generated by this approach for the line segment that joins the points $(0, 0)$ and $(0.8, 0.5)$.

The second stage is then to join the individual ADFs ϕ_i together into a single ADF ϕ for all of $\partial\Omega$ using an R-function. A simple joining method is the product join:

$$\textbf{Product Join: } \phi(\boldsymbol{x}) = \prod_{i=1}^{M} \phi_i(\boldsymbol{x}). \tag{24}$$

Note that this is an R-function because the sign of ϕ is solely determined by the sign of the individual ADFs ϕ_i. We find this join is sufficient under Dirichlet boundary conditions for simple-grid like domains. However, the ADF it defines is, in general, not normalised (Biswas and Shapiro, 2004, Section 2.2.1) and therefore not appropriate for Robin conditions (see Section 2.5). To illustrate this point, we used the product join to construct an ADF for the boundary of the domain $[0, 1]^2$, using the ADFs $\phi_1, \cdots, \phi_4$ given above. The top panel of Figure 2 shows the distance and normed gradient values over the domain for this ADF. Although the individual ADFs are normalised, the joined ADF is not. Furthermore, the product join can also be numerically unstable to evaluate if M is large.

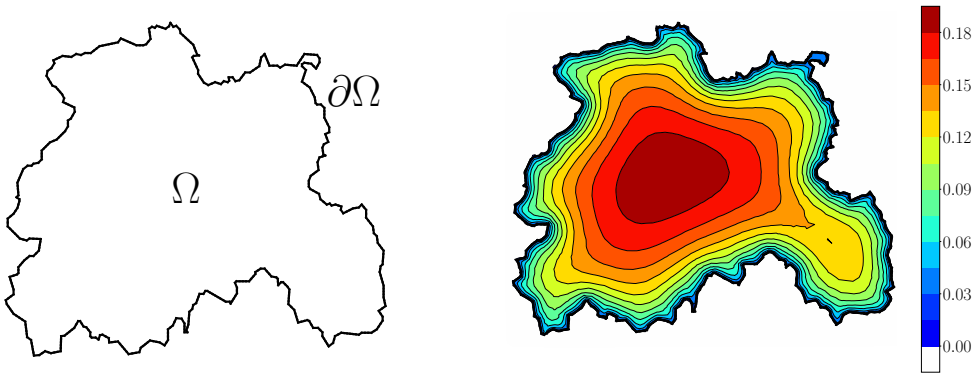
An alternative joining function then which can preserve normalisation at all regular points of the boundary (Biswas and Shapiro, 2004) is the R-equivalence join:

$$\textbf{R-Equivalence Join: } \phi(\boldsymbol{x}) = \frac{1}{\sum_{i=1}^{M} \phi_i(\boldsymbol{x})^{-1}}. \tag{25}$$

We use this join for BVPs involving Robin/Neumann boundary conditions and for domains with many boundary segments M. The bottom row of Figure 2 distance and normed gradient values over the example domain $[0, 1]^2$ for an ADF found using the R-equivalence join. Normalisation is preserved at all regular points of the boundary where $\|\nabla\phi(x, y)\|_2 = 1$. Note that normalisation fails at non-regular points, which in this case are the corner points where two boundaries meet, such as $(0, 0)$. A more complex example is provided in Figure 3 (b), which displays a heat map of an ADF constructed using Eq. (25) for the polygon on the left side of the figure.

2.5 The R-Function Method in Boundary Value Problems

The R-function method (RFM) is an approach for designing a flexible solution structure $\tilde{u}$ using R-functions that satisfies exactly a set of (mixed) boundary conditions (Rvachev et al., 2000). RFM is applicable to problems involving mixed inhomogeneous boundary conditions on domains that can be irregularly shaped (Rvachev et al., 2001), heterogeneous (Tsukanov and Shapiro, 2005) and time varying (Shapiro and Tsukanov, 1999a). It has been applied to physical problems involving, for instance, fluid dynamics (Artiukh et al., 2021; Sidorov and Artyukh, 2014), heat transfer (Kolyada et al., 2019; Basarab et al., 2020) and solid mechanics (Kosta and Tsukanov, 2014; Kurpa et al., 2013). For any given BVP, the form that the RFM structure will take depends on the shape of the geometry and the boundary conditions, but not the PDE itself. In the following, we present the structural forms in the case of Dirichlet, Robin, and mixed boundary conditions respectively.



(a) Polygonised map (b) ADF for $\bar{\Omega}$

Figure 3: Panel (a) shows a polygonised map of County Laois in Ireland, the boundary of which is represented using 503 individual line segments. Panel (b) shows an ADF for the map, found by joining individual ADFs for each boundary segment using the R-equivalence join from Eq. (25).

2.5.1 Dirichlet Boundary Conditions

Given Dirichlet boundary conditions (see Eq. 2), consider the following solution structure

$$\tilde{u}(\boldsymbol{x}) = b(\boldsymbol{x}) + \phi(\boldsymbol{x})\hat{u}(\boldsymbol{x}), \tag{26}$$

where ϕ is an ADF (see Eq. 22) for the boundary $\partial\Omega$, and $\hat{u}$ is an undetermined trial function. The ADF will collapse the contribution of any continuous $\hat{u}$ to zero on $\partial\Omega$, ensuring the Dirichlet condition is exactly satisfied. This type of solution structure dates back to the pioneering work of Kantorovich et al. (1960), and has recently begun to be deployed in the context of physics-informed machine learning (Berg and Nyström, 2018; Sheng and Yang, 2021).

2.5.2 Robin Boundary Conditions

An extension of Eq. (26) to allow more varied linear boundary conditions to be satisfied can be derived using a generalised form of the Taylor series expansion (Shapiro, 2007, Section 7). Here, we simply state the solution structure for Robin boundary conditions (see Eq. 3) to be the following ansatz

$$\tilde{u}(\boldsymbol{x}) = (1 + \phi(\boldsymbol{x})a(\boldsymbol{x}))\,\hat{u}_1(\boldsymbol{x}) - \phi(\boldsymbol{x})\nabla\phi(\boldsymbol{x}) \cdot \nabla\hat{u}_1(\boldsymbol{x}) - \phi(\boldsymbol{x})h(\boldsymbol{x}) + \phi(\boldsymbol{x})^2\hat{u}_2(\boldsymbol{x}), \tag{27}$$

where $\hat{u}_1$ and $\hat{u}_2$ are trial functions, a and h are as given in Eq. (3), and ϕ is a *normalised* ADF (see Eq. 23) for the boundary $\partial\Omega$. In this case, it is not immediately obvious that the boundary conditions are exactly satisfied by $\tilde{u}$. We therefore introduce the following proposition, which is proved in Appendix C.1.

Proposition 8 *If $\hat{u}_1, \hat{u}_2, a$ and h are smooth functions and ϕ is a normalised ADF for the boundary $\partial\Omega$, then the RFM solution structure $\tilde{u}$ from Eq. (27) exactly satisfies the Robin boundary conditions from Eq. (3) everywhere on $\partial\Omega$.*

2.5.3 Mixed Boundary Conditions

A solution for the mixed boundary conditions in Eq. (4) can be derived by constructing one structure for the Dirichlet boundary $\partial\Omega_1$ and another for the Robin boundary $\partial\Omega_2$, before adding them together. Appendix C.2 presents step-by-step details of this derivation, which yields the expression below, where ϕ_1 is an ADF for $\partial\Omega_1$, ϕ_2 is a normalised ADF for $\partial\Omega_2$ and the joined ADF ϕ is also normalised for $\partial\Omega_2$.

$$\begin{aligned}\tilde{u}(\boldsymbol{x}) = &\left(\phi_1(\boldsymbol{x}) + \phi(\boldsymbol{x})\left[a(\boldsymbol{x})\phi_1(\boldsymbol{x}) - \nabla\phi_2(\boldsymbol{x}) \cdot \nabla\phi_1(\boldsymbol{x})\right]\right)\hat{u}_1(\boldsymbol{x}) \\ &- \phi(\boldsymbol{x})\phi_1(\boldsymbol{x})\nabla\phi_2(\boldsymbol{x}) \cdot \nabla\hat{u}_1(\boldsymbol{x}) + \phi(\boldsymbol{x})\phi_2(\boldsymbol{x})\hat{u}_2(\boldsymbol{x}) \\ &+ \phi(\boldsymbol{x})\left[a(\boldsymbol{x})b(\boldsymbol{x}) - h(\boldsymbol{x}) - \nabla\phi_2(\boldsymbol{x}) \cdot \nabla b(\boldsymbol{x})\right] + b(\boldsymbol{x}).\end{aligned} \tag{28}$$

2.5.4 Other Boundary Conditions

RFM solution structures can be constructed for a more varied class of boundary conditions than considered above. In some cases, a bespoke structure can be derived as in Appendix C.2—we direct the reader to (Rvachev and Sheiko, 1995) for several examples. Alternatively, for mixed BVPs with more than two boundary segments, separate solution structures can be defined for each segment before being joined into a single global solution structure using *transfinite interpolation* (Rvachev et al., 2001).

3. Boundary Constrained Gaussian Processes

Modelling problems in which boundary conditions are known are common in scientific applications. In some cases, they can be explicitly controlled or accurately measured (Maillet, 2019), while in other cases, physical reasoning can be used to deduce what values the system must take on its boundary (Li and Tan, 2022). Consequently, there has recently been an interest in the construction of *boundary constrained Gaussian process* (BCGP) models for scientific modelling, which we define as follows.

Definition 9 (Boundary-Constrained Gaussian Process) *Given a set of linear boundary conditions (e.g. Eq. 4), we define a boundary constrained Gaussian process (BCGP) to be a GP whose mean and covariance functions have been designed so that the boundary conditions are exactly satisfied a-priori of any observation data being observed.*

The mean and covariance functions of a BCGP are denoted $\tilde{m}$ and $\tilde{k}$ for the remainder of the paper. We have deliberately not presented a constructive definition of a BCGP, as several construction procedures have already been proposed (Tan, 2016; Ding et al., 2019; Solin and Kok, 2019). In the following, we present a new approach for designing BCGPs based on the RFM from Section 2.5.

3.1 R-Function Method Approach to Constructing BCGPs

Existing work involving RFM has typically specified the undetermined trial functions using, for instance, polynomials or splines. Here, we instead propose placing zero mean GP priors on these trial functions. Note that each RFM solution structure $\tilde{u}$ presented in Section 2.5 (see Eqs. 26-28) involves an *affine* transformation of the trial functions, i.e. a linear transformation plus an offset term. By Corollary 2 of Pförtner et al. (2024), therefore, each $\tilde{u}$ will also be a GP, with mean function equal to the offset term and kernel found by plugging the linear transformation into Eq. (16) (see Eqs. 29-32 below for details). Furthermore, this GP will be constrained to satisfy the given boundary conditions, since the RFM solution structure is valid for any choice of continuous trial functions. Therefore, this procedure constitutes a new way to construct BCGPs.

For the Dirichlet solution structure from Eq. (26), the GP prior on $\hat{u}$ induces the below BCGP for Eq. (2):

$$\textbf{Dirichlet BCGP:} \quad \begin{cases} \tilde{m}(\boldsymbol{x}) = b(\boldsymbol{x}), \\ \tilde{k}(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})\phi(\boldsymbol{x}')k(\boldsymbol{x}, \boldsymbol{x}'). \end{cases} \tag{29}$$

This form of BCGP is equivalent to the construction introduced by Graepel (2003).

Remark 10 *In practice, the boundary function b in Eq. (29) may not be known on the interior of the domain (see, for instance, Section 5.3.1). In this case, interpolation can be used to construct a function that matches b on the boundary and is smooth on the interior. In Appendix B.1, we present an algorithm for doing so if the domain has a grid-like shape, while for domains of a more complex shape, mean value interpolation (Floater and Patrizi, 2020) or neural network approximations (Sheng and Yang, 2021) could be applied.*

For the Robin solution structure from Eq. (27), we place a $\mathcal{GP}(0, k_1)$ prior on $\hat{u}_1$ and a $\mathcal{GP}(0, k_2)$ prior on $\hat{u}_2$. This induces the following form of BCGP for Eq. (3):

$$\textbf{Robin BCGP:} \quad \begin{cases} \tilde{m}(\boldsymbol{x}) = -\phi(\boldsymbol{x})h(\boldsymbol{x}), \\ \tilde{k}(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{B}_{\boldsymbol{x}}^{r_1}\mathcal{B}_{\boldsymbol{x}'}^{r_1}k_1(\boldsymbol{x}, \boldsymbol{x}') + \mathcal{B}_{\boldsymbol{x}}^{r_2}\mathcal{B}_{\boldsymbol{x}'}^{r_2}k_2(\boldsymbol{x}, \boldsymbol{x}'), \end{cases} \tag{30}$$

where the linear operators $\mathcal{B}_{\boldsymbol{x}}^{r_1}$ and $\mathcal{B}_{\boldsymbol{x}}^{r_2}$ are given by

$$\begin{aligned} \mathcal{B}_{\boldsymbol{x}}^{r_1}[\cdot](\boldsymbol{x}) &= (1 + \phi(\boldsymbol{x})a(\boldsymbol{x}))\,[\cdot](\boldsymbol{x}) - \phi(\boldsymbol{x})\nabla\phi(\boldsymbol{x}) \cdot \nabla[\cdot](\boldsymbol{x}), \\ \mathcal{B}_{\boldsymbol{x}}^{r_2}[\cdot](\boldsymbol{x}) &= \phi(\boldsymbol{x})^2[\cdot](\boldsymbol{x}). \end{aligned} \tag{31}$$

In the same manner, a BCGP for Eq. (4) can be constructed from the mixed solution structure from Eq. (28):

$$\textbf{Mixed BCGP:} \quad \begin{cases} \tilde{m}(\boldsymbol{x}) = \phi(\boldsymbol{x})\left[a(\boldsymbol{x})b(\boldsymbol{x}) - h(\boldsymbol{x}) - \nabla\phi_2(\boldsymbol{x}) \cdot \nabla b(\boldsymbol{x})\right] + b(\boldsymbol{x}), \\ \tilde{k}(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{B}_{\boldsymbol{x}}^{m_1}\mathcal{B}_{\boldsymbol{x}'}^{m_1}k_1(\boldsymbol{x}, \boldsymbol{x}') + \mathcal{B}_{\boldsymbol{x}}^{m_2}\mathcal{B}_{\boldsymbol{x}'}^{m_2}k_2(\boldsymbol{x}, \boldsymbol{x}'), \end{cases} \tag{32}$$

where the linear operators $\mathcal{B}_{\boldsymbol{x}}^{m_1}$ and $\mathcal{B}_{\boldsymbol{x}}^{m_2}$ are given by

$$\begin{aligned} \mathcal{B}_{\boldsymbol{x}}^{m_1}[\cdot](\boldsymbol{x}) &= (\phi_1(\boldsymbol{x}) + \phi(\boldsymbol{x})\left[a(\boldsymbol{x})\phi_1(\boldsymbol{x}) - \nabla\phi_2(\boldsymbol{x}) \cdot \nabla\phi_1(\boldsymbol{x})\right])\,[\cdot](\boldsymbol{x}) \\ &\quad - \phi(\boldsymbol{x})\phi_1(\boldsymbol{x})\nabla\phi_2(\boldsymbol{x}) \cdot \nabla[\cdot](\boldsymbol{x}), \\ \mathcal{B}_{\boldsymbol{x}}^{m_2}[\cdot](\boldsymbol{x}) &= \phi(\boldsymbol{x})\phi_2(\boldsymbol{x})[\cdot](\boldsymbol{x}). \end{aligned} \tag{33}$$

3.1.1 Initial Conditions

As discussed in Section 2.2.1, for spatio-temporal processes $u(\boldsymbol{x}, t)$, the initial condition of the system at $t = 0$ can be specified in addition to its behaviour at the spatial boundary $\partial\Omega$. A BCGP for Dirichlet initial conditions (where $u(\boldsymbol{x}, 0)$ is known) can be designed in the same form as Eq. (29)—see Section 5.5 for examples.

Cauchy initial conditions are also commonly used, whereby the position $\mathcal{I}(\boldsymbol{x})$ and time-derivative $v(\boldsymbol{x})$ of the system are specified at $t = 0$, i.e.

$$\textbf{Cauchy:} \quad \begin{cases} u(\boldsymbol{x}, 0) = \mathcal{I}(\boldsymbol{x}), & \boldsymbol{x} \in \bar{\Omega}, \\ \partial_t u(\boldsymbol{x}, 0) = v(\boldsymbol{x}), & \boldsymbol{x} \in \bar{\Omega}. \end{cases} \tag{34}$$

A BCGP for Cauchy initial conditions can be designed by a slight adjustment of the case of Dirichlet boundary conditions—for two examples, see Sections 3.2.4 and 5.4.1 respectively.

3.2 One Dimensional Examples

To provide examples of the construction of BCGPs using the RFM, we consider different types of boundary conditions in the one-dimensional case where $\bar{\Omega} = [0, 1]$. We denote the two boundary segments as $\partial\Omega_1 = \{0\}$ and $\partial\Omega_2 = \{1\}$, and set their corresponding normalised ADFs to be $\phi_1(x) = x$ and $\phi_2(x) = 1 - x$, which we join using the Eq. (24) as $\phi(x) = \phi_1(x)\phi_2(x)$ because in this simple one dimensional case the product join does generate a normalised ADF (see Eq. 23).

3.2.1 Dirichlet Boundary Conditions

Consider the Dirichlet conditions $u(0) = -2.5$ and $u(1) = 0$. We can enforce these boundary values using a Dirichlet BCGP as in Eq. (29) with $b(x) = 2.5x - 2.5$, and ϕ as stated above. Five samples from this BCGP and its derivative process are shown in the top row of Figure 4. In the left panel, it is clear that the boundary conditions are satisfied by each sample. In the right panel, it is apparent that the derivative process takes random values across the domain, i.e. imposing a constraint on the function itself at the boundary has not introduced any artificial constraints on the derivative values.

3.2.2 Neumann Boundary Conditions

Next, consider the Neumann conditions $\partial_x u(0) = -2.5$ and $\partial_x u(1) = 2.5$. A Robin BCGP as in Eq. (30) can be used to enforce these conditions, with $a = 0$ and $h = -2.5$. The second row of Figure 4 displays five samples from this BCGP and its derivative process. In the left panel, each sample takes a random value across $\bar{\Omega}$, whereas it is apparent in the right panel that the samples are constrained to satisfy the Neumann conditions on $\partial\Omega$.

3.2.3 Mixed Dirichlet and Robin Boundary Conditions

Now consider mixed boundary conditions, with a Dirichlet left boundary ($u(0) = -2.5$) and a Robin right boundary ($u(1) = -\partial_x u(1)$). In this instance, a mixed BCGP can be defined as in Eq. (32), with $b = -2.5$, $a = 1$ and $h = 0$. Function and derivative samples from this BCGP are displayed in the third row of Figure 4. The Dirichlet condition at $x = 0$ is clearly

satisfied by each sample, while by comparing the function and derivative values at $x = 1$, it is apparent that the Robin condition is also satisfied.

3.2.4 Cauchy Initial Condition

For this example, we denote the scalar input as time t. Suppose that the Cauchy initial condition (see Eq. 34) $u(0) = -7.5$ and $\partial_t u(0) = 5$ is known. A BCGP for this problem can then be specified as

$$\begin{cases} \tilde{m}(t) = -7.5 + 5t, \\ \tilde{k}(t, t') = t(t')^2 k(t, t'). \end{cases} \tag{35}$$

This BCGP can be derived by two adjustments to the formula given in Eq. (29) for Dirichlet boundary conditions. Firstly, an additional term $(5t)$ is included in the mean function $\tilde{m}$ to account for the known derivative value. Secondly, the "distance" to the initial time (given by t itself) is squared in the boundary constrained kernel $\tilde{k}$, which ensures that $\tilde{k}$ makes no contribution to the derivative value at $t = 0$. The bottom row of Figure 4 shows samples from this BCGP, from which it is clear that the initial condition at $t = 0$ is satisfied in each case.

3.3 Comparison with Spectral Expansion Approach

Solin and Kok (2019) and Solin and Särkkä (2020) developed an approach for constructing BCGPs under homogeneous Dirichlet boundary conditions (i.e. $b(\boldsymbol{x}) = 0$ in Eq. 2) using spectral analysis. The boundary-constrained kernel they derived takes the form

$$\tilde{k}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{j=1}^{m} \psi_j(\boldsymbol{x}) \psi_j(\boldsymbol{x}') s(\lambda_j), \tag{36}$$

in which s is the spectral decomposition of a specified stationary and isotropic kernel function, while $\{\lambda_j\}_{j=1}^m$ are the eigenvalues (ranked in descending order) and $\{\psi_j\}_{j=1}^m$ are the associated eigenfunctions of the Dirichlet Laplacian BVP on the domain Ω, i.e.

$$\begin{cases} -\nabla^2 \psi_j(\boldsymbol{x}) = \lambda_j^2 \psi_j(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ \psi_j(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \partial\Omega. \end{cases} \tag{37}$$

For domains of a simple form, such as circles or rectangles, this BVP can be solved in closed form (Swiler et al., 2020), while for more complex domains, efficient numerical solvers have been proposed (Song et al., 2017). An extension of this method was later presented by Gulian et al. (2022), in which the eigenfunctions/eigenvalues of more general elliptical operators were considered under Robin boundary conditions.

Despite their distinct design techniques, the spectral expansion and RFM approaches yield similar forms of boundary constrained kernels under Dirichlet conditions. In both cases, functions that equal zero on the boundary $\partial\Omega$ (the eigenfunctions $\{\psi_j\}_{j=1}^m$ in Eq. 36 and the ADF ϕ in Eq. 29) in turn collapse to zero the variance of the GP on $\partial\Omega$. The key difference is that the eigenfunctions $\{\psi_j\}_{j=2}^m$ will not be strictly positive on the interior of the domain Ω, and therefore will not be ADFs (see Eq. 22). Under certain mild conditions, it can be shown however that the principal eigenfunction ψ_1 will be strictly positive in Ω

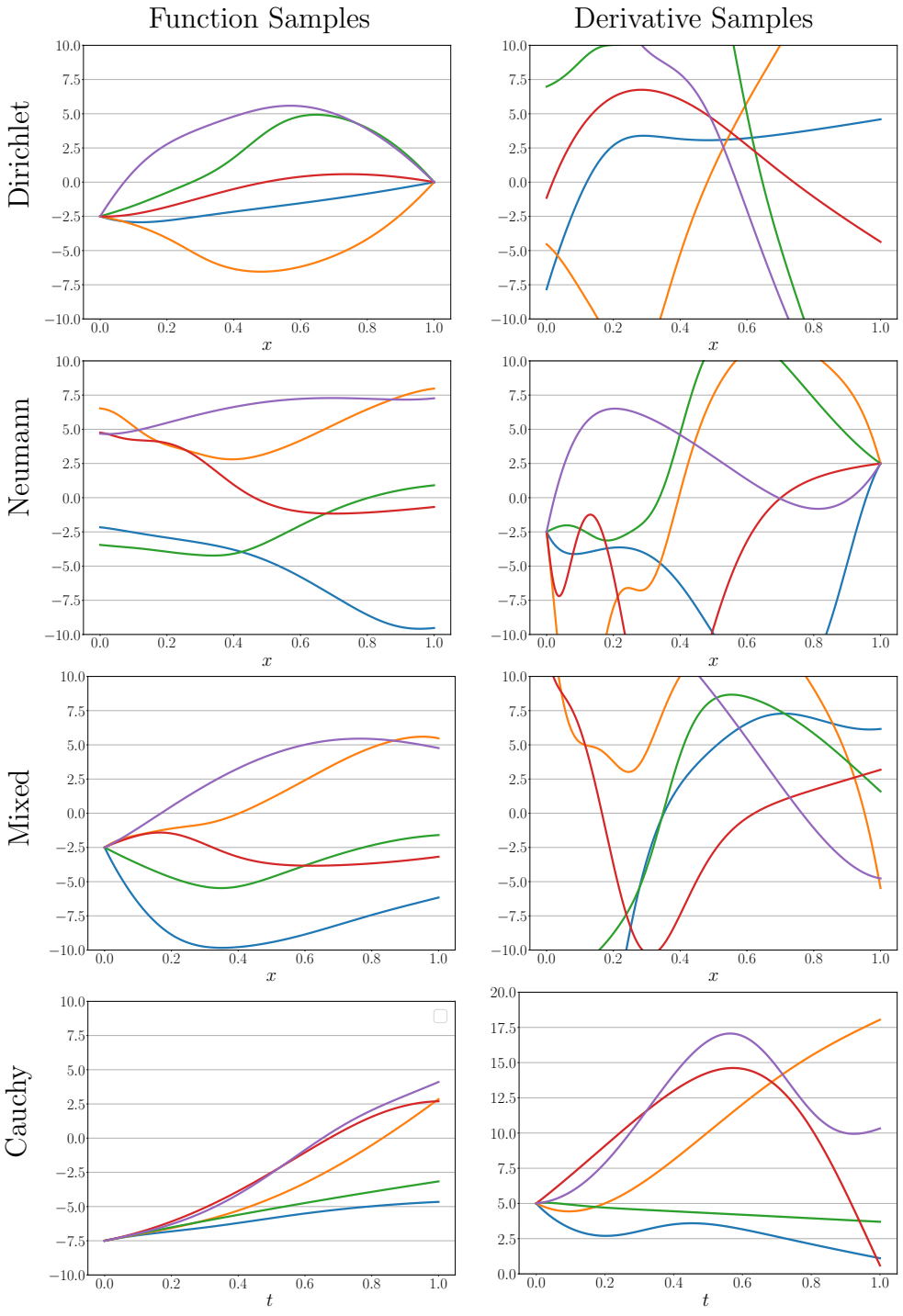


Figure 4: Function and derivative samples from the Dirichlet, Neumann, mixed and Cauchy BCGPs detailed in Section 3.2.

(Berestycki et al., 1994, Theorem 2.1-i). Therefore, ψ_1 will be an ADF, and could be used to design a BCGP kernel using the RFM method as in Eq. (29).

4. Theoretical Results

4.1 Universality of the Boundary Constrained Kernel

Remark 11 *In the following, we present a concise and non-technical description of reproducing kernel Hilbert spaces—for a more complete overview of this material, see Appendix A.2.*

In Section 2.3, we introduced a kernel as a positive definite function appropriate for use as a covariance function in the specification of a GP. Kernels also have a deep and one-to-one correspondence with Hilbert spaces of functions which are "well-behaved" in the sense that norm convergence implies pointwise convergence (Steinward and Christmann, 2008, Equation 4.7). Such function spaces are called *reproducing kernel Hilbert spaces* (RKHSs). The correspondence is one-to-one because, for each kernel k, there exists a unique RKHS $\mathcal{H}_k$ for which k "reproduces" the functions in the space, and conversely, each RKHS has a unique reproducing kernel[1].

Intuitively, the RKHS $\mathcal{H}_k$ of a given kernel k can be thought of as the space of all posterior means of a zero-mean GP with covariance function equal to k. In the general regression case, it would clearly be desirable if a kernel allowed for any continuous function to be approximated to arbitrary accuracy. Kernels for which this property is true are called *universal kernels* (Steinward and Christmann, 2008, Definition 4.52).

Definition 12 (Universal Kernel) *A continuous kernel k defined on $\mathcal{X} \times \mathcal{X}$ with $\mathcal{X}$ a compact metric space is called universal if its associated RKHS $\mathcal{H}_k$ is dense in $\mathcal{C}(\mathcal{X})$ with respect to the metric induced by the supremum norm. That is, for every $u \in \mathcal{C}(\mathcal{X})$ and for all $\varepsilon > 0$, there exists $\hat{u} \in \mathcal{H}_k$ such that $\|u - \hat{u}\|_\infty \leq \varepsilon$.*

It can be shown that the squared exponential kernel k_{SE} from Eq. (7) is universal on any compact subset of $\mathbb{R}^D$ (Micchelli et al., 2006). By contrast, a boundary constrained kernel $\tilde{k}$ of the form described in Section 3.1 for Dirichlet boundary conditions will not be universal because all functions in its associated RKHS $\mathcal{H}_{\tilde{k}}$ will be constrained to equal zero at the domain boundary, $\partial\Omega$. This behaviour of $\tilde{k}$ is desirable when the value of the solution function is prescribed on $\partial\Omega$. However, the function is unknown on the interior of its domain, Ω. Therefore, it would also be desirable if $\tilde{k}$ maintains universal approximation within $\bar{\Omega}$, for all continuous functions which satisfy the boundary conditions.

We examine this problem in the particular case of homogeneous Dirichlet conditions, i.e. we consider the function space

$$\mathcal{H}_{bc} \triangleq \{u \in \mathcal{C}(\bar{\Omega}) : u(\boldsymbol{x}) = 0 \ \forall\, \boldsymbol{x} \in \partial\Omega\}, \tag{38}$$

where $\bar{\Omega}$ is a compact set of the form described in Section 2.1. An appropriate boundary-constrained kernel $\tilde{k}$ for this problem is given in Eq. (29). We then have the following result, which is proved in Appendix C.3.

1. See Theorem 19 and related discussion in Appendix A.2 for precise details.

Theorem 13 (Universal Boundary Constrained Kernel) *Let $\tilde{k}$ be a boundary-constrained kernel of the form stated in Eq. (29), in which k is any universal kernel on $\bar{\Omega} \times \bar{\Omega}$ and ϕ any continuous ADF for the boundary $\partial\Omega$ (see Eq. 22). Then, the RKHS $\mathcal{H}_{\tilde{k}}$ is dense in $\mathcal{H}_{bc}$ from Eq. (38) with respect to the metric induced by the supremum norm. That is, for every $u \in \mathcal{H}_{bc}$ and for all $\varepsilon > 0$, there exists $\tilde{u} \in \mathcal{H}_{\tilde{k}}$ such that $\|u - \tilde{u}\|_{\infty} \leq \varepsilon$.*

4.2 Connection to Neural Networks

As discussed in Section 1.1, the problem of directly enforcing boundary conditions has also been explored in the context of neural networks. Specifically, a *boundary-constrained neural network* (BCNN) can be defined analogously to a BCGP (Definition 9), in which the output of a neural network is transformed so that a given set of boundary conditions are exactly satisfied. To our knowledge, no one has yet introduced a formal link between the BCNN and BCGP literatures. Notice, however, that BCNNs can be constructed in almost the same manner as BCGPs, by replacing the trial functions in an RFM solution structure (see Section 2.5) with neural networks (Sukumar and Srivastava, 2022), instead of using GPs as in Section 3.1. Consider, for instance, the case of Dirichlet boundary conditions (see Eq. 2). In this case, a BCNN can be constructed following Eq. (26) to give :

$$\textbf{Dirichlet BCNN: } \tilde{u}_{nn}(\boldsymbol{x}) = \tilde{m}(\boldsymbol{x}) + \phi(\boldsymbol{x})\hat{u}_{nn}(\boldsymbol{x}), \tag{39}$$

in which $\hat{u}_{nn}$ is a neural network. This type of BCNN has been widely used recently in the PINN literature (Berg and Nyström, 2018; Sheng and Yang, 2021; Sukumar and Srivastava, 2022; Anagnostopoulos et al., 2024).

Remark 14 *Similar arguments to those presented in Appendix C.3 can be used to show that the BCNN structure in Eq. (39) is a universal function approximator within the boundary constrained function space $\mathcal{H}_{bc}$ in Eq. (38)*[2]*.*

A further equivalence between the two approaches can be derived by considering the limiting case of an infinitely wide BCNN. Recall that, in the infinite width limit and under certain regularity conditions, a single layer neural network converges to a GP. Specifically, we have the following result from Chapter 2 of Neal (1996), which is reviewed succinctly in Section 18.7.1 of Murphy (2023).

Theorem 15 (Convergence of neural network to GP in infinite width limit) *Consider a single layer neural network of width H, i.e. a model of the form*

$$\hat{u}_{nn}(\boldsymbol{x}) = b^{(1)} + \sum_{j=1}^{H} w_j^{(1)} h_j(\boldsymbol{x}), \quad h_j(\boldsymbol{x}) = \varphi\left(b_j^{(0)} + \boldsymbol{x}^{\top}\boldsymbol{w}_j^{(0)}\right) \tag{40}$$

where H is the width of the hidden layer and φ is a non-linear activation function taken to be bounded (such as the tanh function). Assuming Gaussian priors on the parameters of the form

$$b^{(1)} \sim \mathcal{N}\left(0, \sigma_{b^{(1)}}\right), w_j^{(1)} \sim \mathcal{N}\left(0, \sqrt{\frac{\omega}{H}}\right), b_j^{(0)} \sim \mathcal{N}\left(0, \sigma_{b^{(0)}}\right), \boldsymbol{w}_j^{(0)} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\boldsymbol{w}^{(0)}}), \tag{41}$$

2. This relies on the well-known universal approximation theorem for neural networks (Cybenko, 1989).

then, in the limit as $H \to \infty$, we have

$$\hat{u}_{nn}(\boldsymbol{x}) \to \mathcal{GP}(0, k_{nn}(\boldsymbol{x}, \boldsymbol{x}')), \tag{42}$$

where the form of the limiting kernel k_{nn} depends on the choice of activation function φ.

We now introduce a corresponding theorem for the limiting form of a single layer BCNN.

Theorem 16 (Convergence of BCNN to BCGP in infinite width limit) *Let $\tilde{u}_{nn}$ be a BCNN of the form of Eq. (39) and $\hat{u}_{nn}$ a single layer network of width H, with prior distributions of the form of Eq. (41) assumed on its parameters. Then, in the limit as $H \to \infty$, we have*

$$\tilde{u}_{nn}(\boldsymbol{x}) \to \mathcal{GP}(\tilde{m}(\boldsymbol{x}), \phi(\boldsymbol{x})\phi(\boldsymbol{x}')k_{nn}(\boldsymbol{x}, \boldsymbol{x}')), \tag{43}$$

where k_{nn} is the same kernel found in Eq. (42).

This is proved in Appendix C.4—note also that the result immediately generalises to Robin and mixed boundary conditions.

5. Numerical Experiments

We conducted a range of numerical experiments to evaluate the effectiveness of BCGPs for the machine learning of linear PDEs, the results of which are presented and discussed in Sections 5.2-5.7. Firstly, however, Section 5.1 below provides the implementation details for these experiments.

5.1 Experimental Details

The numerical experiments we performed are summarised in Table 1, the final column of which contains a GitHub link to the Jupyter notebooks where they were performed. The experiments involved forward and inverse problems for five different linear PDEs under Dirichlet, Cauchy, Neumann, Robin and periodic boundary/initial conditions. For inverse problems, we introduced white noise to the observation data, with standard deviation (σ_u in Eq. 5) typically set to a fraction of the standard deviation in the true function u.

Experiment	D	N_u	N_f	Inverse	Initial	Boundary	Code
Poisson-BVP-1	1	0	10	No	NA	Mixed	Link 1
Poisson-BVP-2	2	0	1000	No	NA	Dirichlet	Link 2
Poisson-BVP-3	2	25	25	Yes	NA	Mixed	Link 3
Poisson-BVP-4	2	25	25	Yes	NA	Dirichlet	Link 4
Heat-IBVP-1	1	25	25	Yes	Dirichlet	Dirichlet	Link 5
Heat-IBVP-2	2	50	50	Yes	Dirichlet	Dirichlet	Link 6
Wave-IBVP-1	1	25	25	Yes	Cauchy	Neumann	Link 7
Wave-IBVP-2	1	25	25	Yes	Cauchy	Periodic	Link 8
Adv-Diff-IBVP	1	150	150	Yes	Dirichlet	Dirichlet	Link 9
Helmholtz-BVP	2	10	10	Yes	NA	Mixed	Link 10

Table 1: Summary of the ten (I)BVPs considered in Section 5.

5.1.1 Gaussian process Specifications

For each experiment, we constructed a BCGP to exactly satisfy the given boundary conditions by applying Eqs. (29), (30) and (32) (adjusted slightly in the presence of initial conditions or periodic boundary conditions). For ease of readability, we present the exact functional form of the boundary-constrained mean and kernel for each (I)BVP in Appendix B.2 instead of the main text.

To benchmark BCGP performance, we used two alternative GP models:

- **UCGP:** Unconstrained Gaussian Process—here no boundary information is accounted for in the modelling problem. This is the approach used in the seminal paper by Raissi et al. (2017).
- **PCGP:** Penalty Constrained Gaussian Process—here a set of N_b pseudo-observations on $\partial\Omega$ are used to account for the boundary conditions. This is similar to a penalty or soft enforcement approach, as used by Zhang et al. (2022).

For all models, we used as base kernel the squared-exponential with separate length scales for each dimension, i.e.

$$k(\boldsymbol{x}, \boldsymbol{x}') = \tau^2 \exp\left(-\frac{1}{2}\sum_{i=1}^{D}\frac{(x_i - x_i')^2}{\ell_i^2}\right), \tag{44}$$

in which the kernel hyperparameters are $\boldsymbol{\xi} = (\tau, \ell_1, \ldots, \ell_D)$. This is called an automatic relevance determination (ARD) kernel, as it allows the most important input dimensions to be identified automatically during the training procedure (Rasmussen and Williams, 2006, Section 5.1). We used this kernel because it is infinitely differentiable, and therefore appropriate for each BVP we considered. If more fine-grained control over kernel differentiability is desired, a tensor product of Matérn kernels could be used (Pförtner et al., 2024, Example B.1).

As discussed in Section 2.3.1, we trained all GP models by optimisation of the marginal likelihood (see Eq. 19). Note that this objective function is not convex with respect to $\boldsymbol{\xi}$, and therefore we considered multiple random restarts when training.

5.1.2 Neural Network Specifications

BCNNs (see Section 4.2) were also used for performance benchmarking. We denote a BCNN as $\tilde{u}_{nn}$, and its weights and biases collectively as $\boldsymbol{\omega}$. Given observations $\boldsymbol{y}_u \in \mathbb{R}^{N_u \times 1}$ in u-space (see Eq. 5) and observations $\boldsymbol{y}_f \in \mathbb{R}^{N_f \times 1}$ in f-space (see Eq. 13), training was performed by solving the following optimisation problem.

$$\{\hat{\boldsymbol{\omega}}, \hat{\boldsymbol{\theta}}\} = \underset{\boldsymbol{\omega},\boldsymbol{\theta}}{\operatorname{argmin}} \left[L_u(\boldsymbol{\omega}) + L_f(\boldsymbol{\omega}, \boldsymbol{\theta})\right], \tag{45}$$

in which

$$L_u(\boldsymbol{\omega}) = \frac{1}{N_u}\sum_{i=1}^{N_u}\left(y_u^{(i)} - \tilde{u}_{nn}(\boldsymbol{x}_u^{(i)}; \boldsymbol{\omega})\right)^2, \tag{46}$$

$$L_f(\boldsymbol{\omega}, \boldsymbol{\theta}) = \frac{1}{N_f}\sum_{i=1}^{N_f}\left(y_f^{(i)} - \mathcal{L}_{\boldsymbol{x}}^{\boldsymbol{\theta}}[\tilde{u}_{nn}](\boldsymbol{x}_f^{(i)}; \boldsymbol{\omega})\right)^2. \tag{47}$$

Again, we use multiple random restarts when solving this optimisation problem because the objective function is non-convex with respect to $\boldsymbol{\omega}$. Typically with PINNs, a boundary loss term is also included in the objective function; however, this is not necessary for a BCNN $\tilde{u}_{nn}$ because its output is constrained to satisfy the boundary conditions exactly.

Unless otherwise stated, for all experiments, we used a network architecture of four hidden layers each of width 10, which we found to give strong accuracy across a range of modelling problems.

5.1.3 Comparison of different modelling approaches

The modelling approaches detailed above have different strengths and weaknesses, and the choice of which to deploy in practice will depend on the specifics of the problem of interest. PIGPs are particularly suited to inference in systems governed by linear PDEs. As detailed in Section 2.3.1, any unknown PDE parameters $\boldsymbol{\theta}$ become hyperparameters of the PIGP kernel, allowing for efficient inference of $\boldsymbol{\theta}$ from observational data. The number of parameters to be inferred when using a PIGP (that is, the kernel, noise, and PDE parameters) will be much lower than is the case with a PINN, where typically thousands of weights and biases will need to be trained. PIGPs also give full uncertainty quantification through the analytically tractable posterior predictive distribution (Eq. 12). The advantage of PINNs is that they are significantly more computationally efficient than PIGPs, and can naturally handle non-linear PDEs. As for the handling of boundary conditions, unconstrained approaches (i.e. UCGPs) are appropriate for applications where boundary conditions are unknown. When boundary conditions are known, they can either be softly enforced using a penalty approach or explicitly enforced with a BCGP/BCNN. One problem with PCGPs is that computational resources are "wasted" accounting for the boundary conditions, whereas with a BCGP, the conditions are enforced without using observational data. However, the mean and kernel of a BCGP are more complex than a standard GP, which leads to increased expense in performing just-in-time compilation of code.

5.1.4 Evaluation Metrics

The results of the PIML models were assessed using the following error metrics

$$err_{\boldsymbol{u}} = \text{Mean}(|u - \hat{u}_{ml}|), \quad err_{\boldsymbol{\theta}} = \|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}\|_2, \tag{48}$$

where $\hat{\boldsymbol{\theta}}$ is the point estimate of the PDE parameters found during training and $\hat{u}_{ml}$ the best prediction for the true solution[3]. The Mean$(\cdot)$ operator in $err_{\boldsymbol{u}}$ was taken over a large grid of test points in the specified domain.

5.1.5 Implementation Details

The experiments were implemented in Python using the JAX library (Bradbury et al., 2018). Adam (Kingma and Ba, 2017) was used for training with an exponentially decaying learning rate. For each PDE considered, the differential operator was implemented in JAX code, which allowed for m_f, k_{ff}, k_{uf} and k_{fu} (see Section 2.3.1 for details) to be found using the automatic differentiation system provided by JAX, i.e. no hand derivations were

3. For GP models, $\hat{u}_{ml}$ was set to the posterior mean.

required. All experiments were performed on an NVIDIA RTX A6000 GPU. Code and data are available at `https://github.com/dodaltuin/jax-pigp/blob/main/examples/BCGPs`.

5.2 Poisson Equation

Our first set of numerical experiments involved Poisson's equation, the prototypical elliptic PDE that has wide utility in physics. Given two spatial dimensions x and y, the equation takes the form

$$\mathcal{L}^{\theta}_{x,y}u(x,y) \triangleq -\theta\left(\frac{\partial^2}{\partial x^2}u(x,y) + \frac{\partial^2}{\partial y^2}u(x,y)\right) = f(x,y). \tag{49}$$

We applied the equation to four different sets of boundary conditions, specifying $\theta = 1$ in each case and $f = 4$ unless otherwise stated. We used these experiments to compare the performance of BCGPs with BCNNs (see Section 4.2)—comparisons with PCGPs and UCGPs (see Section 5.1.1) were conducted for all other PDEs considered.

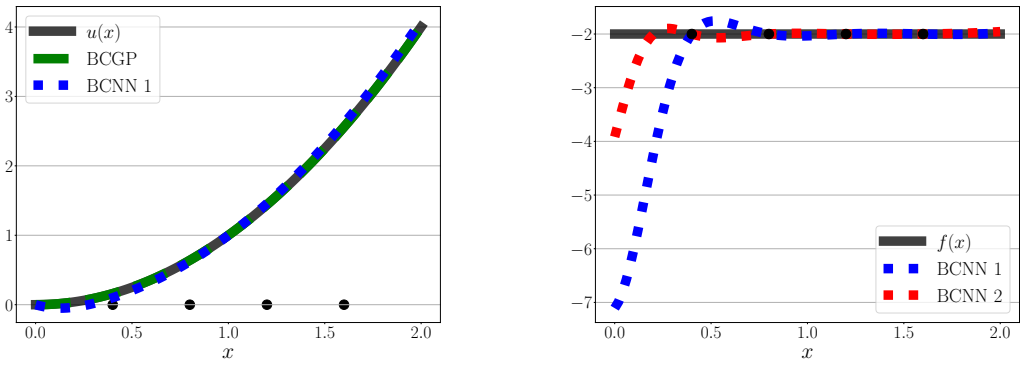
5.2.1 Poisson-BVP-1

Our first experiment was modelled on an example from the DeepXDE documentation (Lu et al., 2021), by applying the Poisson equation on the spatial domain $\Omega = (0, 2)$ with $f(x) = 2$, subject to the mixed Dirichlet and Robin boundary conditions in Eq. (50). This BVP yields true solution $u(x) = x^2$.

$$\textbf{Poisson-BVP-1:}\ \begin{cases} u(x) = 0, & x \in \partial\Omega_1 = \{0\}, \\ \partial_x u(x) = u(x), & x \in \partial\Omega_2 = \{2\}. \end{cases} \tag{50}$$

We used this simple example for exposition of the boundary-constrained framework for modelling BVPs, both for neural networks (BCNNs) and Gaussian processes (BCGPs). Because this problem involved mixed boundary conditions, a BCGP could be specified using Eq. (32), and a BCNN specified using Eq. (28)—see Appendix B.2.1 for further details. To compare the two models, we considered the forward problem of learning u given $N_f = 4$ collocation points spread evenly in Ω. BCGP training was performed to maximise the marginal likelihood of the collocation points (see Eq. 19), while BCNN training was performed by minimising the loss term L_{pinn} (see Eq. 45), with $L_{pinn} = L_f$ in this case because no observations in u-space were available.

Figure 5 (a) presents the predictions of both models after training. The BCGP recovered the true solution virtually exactly, with $err_{\boldsymbol{u}} = 5.4 \times 10^{-11}$, while the BCNN obtained an $err_{\boldsymbol{u}}$ value of 2.0×10^{-2}. BCNN results were also sensitive to the initialisation of the weights and biases $\boldsymbol{\omega}$ at the beginning of training. This is illustrated in Figure 5 (b), which displays $\mathcal{L}^{\boldsymbol{\theta}}_{\boldsymbol{x}}[\tilde{u}_{nn}]$ under two different random restarts. Both BCNNs have been trained to almost perfectly interpolate the collocation points, but BCNN 2 exhibits less oscillation near $x = 0$ and consequently achieves smaller prediction error ($err_{\boldsymbol{u}} = 3.2 \times 10^{-3}$). This illustrates a problem with the PINN training scheme from Eq. (45) in the presence of sparse data, as it is unable to distinguish between models which give the same data fit. A regularisation term could be incorporated into the loss function, which would favour BCNN 2 in this example because it is the simpler function. However, this requires both the form and the strength of the regularisation to be specified, which would likely require manual tuning. Note the



(a) u-space results

(b) BCNN f-space results

Figure 5: Illustration of results for Poisson-BVP-1. The black circles in panel (a) indicate the location of the collocation points.

contrast to the GP objective function, which *automatically* incorporates both a data fit and regularisation term (see Eq. 20). This balance of model fit and complexity (called the Bayesian Occam's razor) is what allows the GP to learn the underlying function u even with only $N_f = 4$ collocation points. With more training data, the performance of the BCNN improves considerably; however, approximately $N_f = 50$ collocation points are required before the BCNN matches the accuracy of the BCGP trained on $N_f = 4$ points.

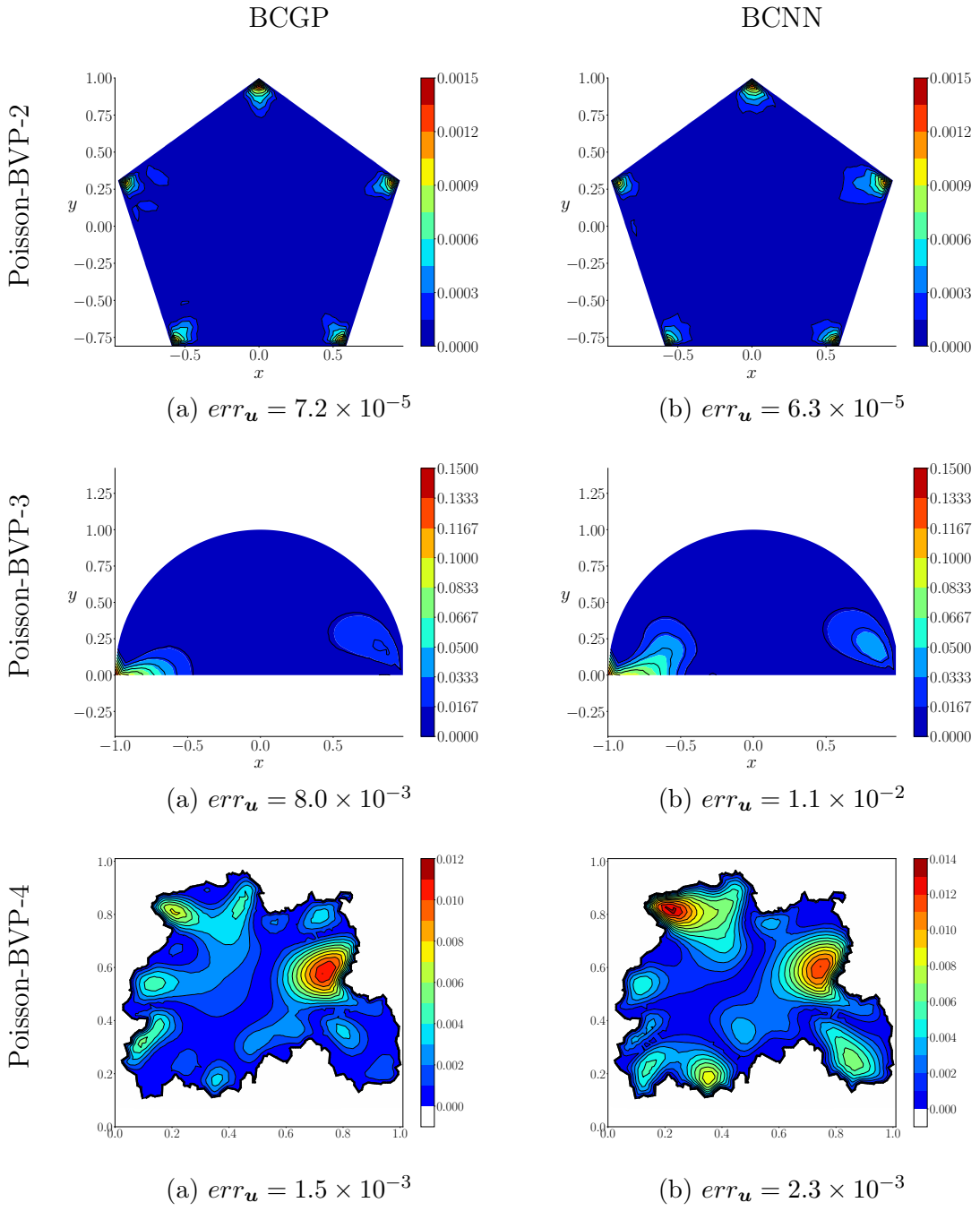
5.2.2 Poisson-BVP-2

We subsequently applied the Poisson equation to a pentagon-shaped domain Ω embedded within $[-1, 1]^2$ (see the top row of Figure 6), subject to the following Dirichlet boundary conditions:

$$\textbf{Poisson-BVP-2:} \quad u(x, y) = 0, \ (x, y) \in \partial\Omega. \tag{51}$$

The solution to this BVP is not available in closed form, and therefore we used a numerical approximation found using the finite-element method (FEM) (Zienkiewicz et al., 2005), implemented in Python using the FEniCS library (Alnæs et al., 2015).

A BCGP was specified using Eq. (29), with $\tilde{m} = 0$ and ADF ϕ constructed in the manner detailed in Section 2.4.2 for the polygon in Figure 3 (a). A BCNN was defined using Eq. (39) with the same choice of $\tilde{m}$ and ϕ. We used this example to compare the performance of each model on the forward problem of learning the displacement field given $N_f = 1000$ collocation points. The BCNN performed slightly better in this task, attaining an $err_{\boldsymbol{u}}$ value of 6.3×10^{-5} versus 7.2×10^{-5} for the BCGP. Both models incurred highest prediction errors near the corner points where the solution field turns sharply, as can be seen from the density plots of prediction error in the top row of Figure 6.



(a) $err_{\boldsymbol{u}} = 7.2 \times 10^{-5}$ (b) $err_{\boldsymbol{u}} = 6.3 \times 10^{-5}$

(a) $err_{\boldsymbol{u}} = 8.0 \times 10^{-3}$ (b) $err_{\boldsymbol{u}} = 1.1 \times 10^{-2}$

(a) $err_{\boldsymbol{u}} = 1.5 \times 10^{-3}$ (b) $err_{\boldsymbol{u}} = 2.3 \times 10^{-3}$

Figure 6: Density plots of prediction error for BCGP (left) and BCNN (right) for Poisson-BVP-2, Poisson-BVP-3 and Poisson-BVP-4.

5.2.3 Poisson-BVP-3

Our next experiment involved a half-circle domain, which is displayed in the second row of Figure 6. Mixed boundary conditions were considered, with a Dirichlet condition applied to the curved segment of the boundary $\partial\Omega_1$, and a Neumann condition applied to the flat bottom segment $\partial\Omega_2$:

$$\textbf{Poisson-BVP-3:} \begin{cases} u(x,y) = 0, & (x,y) \in \partial\Omega_1, \\ \boldsymbol{n}(x,y) \cdot \nabla u(x,y) = 0, & (x,y) \in \partial\Omega_2. \end{cases} \tag{52}$$

This BVP can be solved in closed form to yield

$$u(x,y) = 1 - x^2 - y^2. \tag{53}$$

We used this BVP to compare BCGP and BCNN performance on the inverse problem of learning θ given $N_u = 25$ observations of u (subject to 2.5% Gaussian noise) and $N_f = 25$ collocation points. A mixed BCGP was specified using Eq. (32), and the BCNN was constructed analogously—see Appendix B.2.2 for further details. In this case, the BCGP achieved both a better parameter estimate ($err_{\boldsymbol{\theta}} = 2.2 \times 10^{-2}$ versus 3.0×10^{-2}), and was more accurate in recovering the solution function ($err_{\boldsymbol{u}} = 8.0 \times 10^{-3}$ versus 1.1×10^{-2}). The second row of Figure 6 displays density plots of prediction error—each model incurred highest error at the bottom left region of the domain, near where the Dirichlet boundary joins the Neumann boundary.

5.2.4 Poisson-BVP-4

In our final experiment involving the Poisson equation, we set the domain Ω to be the interior of the intricate polygon from Figure 3 (a), and applied homogeneous Dirichlet conditions of the form presented in Eq. (51) to its boundary $\partial\Omega$. For this problem, no explicit solution is available, so we instead treated a BCNN trained on 1000 collocation points as the true solution. We specified a BCGP using Eq. (29) and a BCNN using Eq. (39), in both cases setting $\tilde{m} = 0$ and ADF ϕ equal to that displayed in Figure 3 (b). As in Section 5.2.3, we considered the problem of learning both u and θ given $N_u = 25$ noisy solution space observations and $N_f = 25$ collocation points. From this data, the BCGP achieved lower $err_{\boldsymbol{u}}$ value (1.5×10^{-3} versus 2.3×10^{-3}) but higher $err_{\boldsymbol{\theta}}$ value (3.7×10^{-2} versus 1.1×10^{-2}). The bottom row of Figure 6 shows density plots of prediction error, which illustrate that the BCNN incurred slightly higher errors at the top left of the domain.

5.3 Heat Equation

We next analysed the heat equation, which describes the diffusion of heat in an isotropic medium. Over two spatial dimensions x and y, the equation takes the form

$$\mathcal{L}^{\theta}_{x,y,t} u(x,y) \triangleq \frac{\partial}{\partial t} u(x,y,t) - \theta \left(\frac{\partial^2}{\partial x^2} u(x,y,t) - \frac{\partial^2}{\partial y^2} u(x,y,t) \right) = f(x,y,t) = 0. \tag{54}$$

We performed experiments which involved the inverse problem of learning the thermal diffusivity parameter $\theta = 1$, using two IBVPs introduced by Zhang et al. (2022).

5.3.1 Heat-IBVP-1

We first applied the heat equation over the spatial domain $\Omega = \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$ and time interval $(0, 1]$, subject to the following Dirichlet initial/boundary conditions:

$$\textbf{Heat-IBVP-1:} \quad \begin{cases} u(x,0) = \sin(x), & x \in \bar{\Omega}, \\ u(x,t) = -\exp(-t), & x \in \partial\Omega_1,\ t > 0, \\ u(x,t) = \exp(-t), & x \in \partial\Omega_2,\ t > 0, \end{cases} \tag{55}$$

in which $\partial\Omega_1 = \{-\pi/2\}$ and $\partial\Omega_2 = \{\pi/2\}$. The separation of variables technique can be used to show the solution to this problem has the form stated in Eq. (56) below, which is plotted in Figure 7 (a).

$$u(x,t) = \sin(x)\exp(-t). \tag{56}$$

We used a BCGP to jointly learn θ and u given a data set of $N_u = 25$ observations in solution space corrupted with 2.5% Gaussian noise, and $N_f = 25$ collocation points of $f = 0$. Mean and kernel functions of the form given in Eq. (29) were used to construct the BCGP, as the IBVP only contained Dirichlet conditions—see Appendix B.2.3 for further details. The BCGP learned a parameter estimate of $\hat{\theta} = 1.000$, while its posterior predictive mean incurred a prediction error of $err_{\boldsymbol{u}} = 4.9 \times 10^{-5}$. This prediction is displayed in Figure 7 (c)—visually, it provides a perfect match to the true function.

To benchmark these results, we re-performed the experiment using PCGPs. Recall from Section 5.1.1 that a PCGP incorporates boundary condition information by conditioning on N_b pseudo-observations from the boundary. To account for the heteroscedasticity between the noisy observations $\boldsymbol{y}_u$ in the interior of the domain and the noise-free boundary observations, we introduced a small noise variance σ_n^2 for the boundary points. We furthermore let σ_n^2 be a trainable hyperparameter, which we found yielded improved performance. To explore how the number of boundary enforcement points (whose locations were chosen by random sampling) affected PCGP results, inference was performed for a range of values of $N_b < 150$. Figure 8 (a) displays these results as a line plot of $err_{\boldsymbol{\theta}}$ against N_b, which indicates that $err_{\boldsymbol{\theta}}$ fell with higher values of N_b to closely match (but never exceed) the BCGP result. The posterior means of the two models for $N_u = N_f = 25$ are presented in panels (c) and (e) respectively of Figure 7. Visually, there is strong agreement between the two results; however, the BCGP obtained an $err_{\boldsymbol{u}}$ value approximately half that of the UCGP.

5.3.2 Heat-IBVP-2

The spatial domain $\Omega = (0, \pi)^2$ and time interval $(0, 0.5]$ was specified for our next experiment, with the below Dirichlet initial/boundary conditions applied.

$$\textbf{Heat-IBVP-2:} \quad \begin{cases} u(x,y,0) = \sin(x)\sin(y), & (x,y) \in \bar{\Omega}, \\ u(x,y,t) = 0, & (x,y) \in \partial\Omega,\ t > 0. \end{cases} \tag{57}$$

The separation of variables technique can be used to show that the exact solution to this problem has the form given in Eq. (58), which is displayed in Figure 7 (b) for $t = 0.5$.

$$u(x,y,t) = \sin(x)\sin(y)\exp(-2t). \tag{58}$$

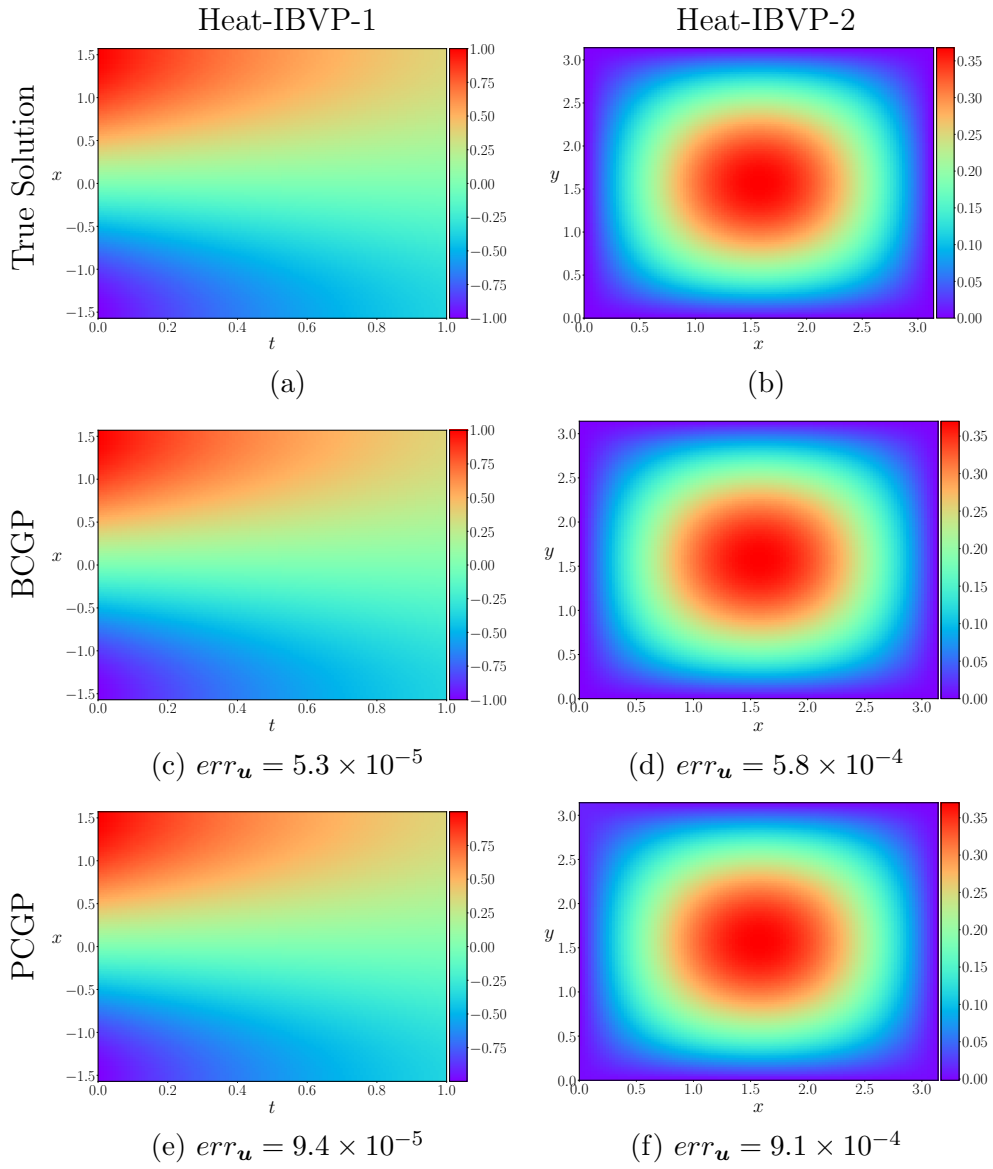
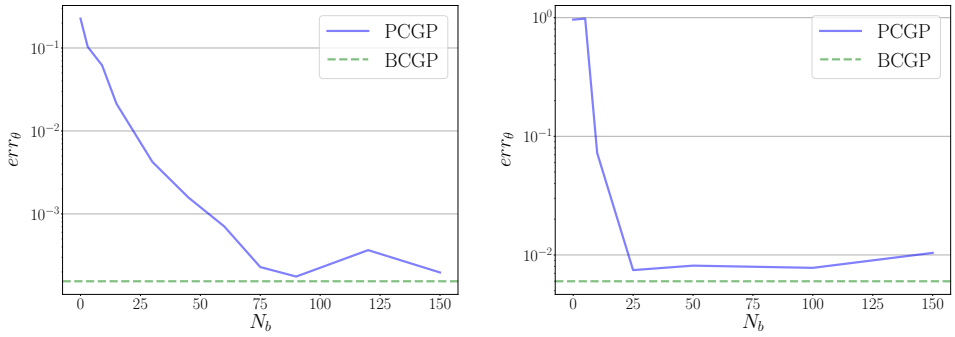


Figure 7: Plots of true and predicted results for Heat-IBVP-1 and Heat-IBVP-2.

As in Section 5.3.1, a Dirichlet BCGP for this problem was specified following Eq. (29)—see Appendix B.2.4 for details. To perform inference, we generated a data set of $N_u = 50$ noise-corrupted function observations and $N_f = 50$ collocation points. More training points were used here due to the increased dimensionality of the problem. From this data, the BCGP learned an estimate of $\hat{\theta} = 0.994$, with $err_{\boldsymbol{u}} = 5.8 \times 10^{-4}$. Figure 7 (d) shows the posterior mean prediction over $\bar{\Omega}$ at time $t = 0.5$, from which it is clear that the true function was accurately recovered. PCGPs with different values of N_b were again used as benchmarking, the results of which can be seen in Figure 8 (b). In this case, high $err_{\boldsymbol{\theta}}$ errors were incurred



(a) Heat-IBVP-1 (b) Heat-IBVP-2

Figure 8: Effect on parameter inference accuracy of number of penalty observation points N_b for PCGP for Heat-IBVP-1 and Heat-IBVP-2.

for $N_b < 25$, however this rapidly switched to yield significantly more accurate results for $N_b > 25$.

We remark here that when training a PCGP for this problem with a low number of boundary enforcement points (N_b), we found that the optimiser consistently converged to a result which yielded a posterior mean approximately equal to zero across the domain, in which case all variation in the data was attributed to noise. This is a well-known problem in physics informed training in the presence of sparse data (Leiteritz and Pflüger, 2021). We call this the *trivial solution*, as the zero function trivially satisfies the Heat PDE in Eq. (54). As N_b was increased, a greater proportion of the random restarts began to (approximately) learn the true solution. However, even for values of N_b up to 1000, we continued to encounter the problem. Furthermore, this was not a local optima—in fact, the value of the objective function (see Eq. 19) for restarts where the posterior mean was close to zero was consistently higher than for restarts which recovered the true solution accurately. For the PCGP results presented in Figure 8 (b), we manually filtered all restarts and removed any optimisation results which found the trivial solution. If this was not performed, the PCGP results would have been significantly worse. We believe this was a particular problem in this example because we applied homogeneous boundary conditions (i.e. $u = 0$ on $\partial\Omega$), and the introduction of pseudo-observations of $u = 0$ is unlikely to help the optimiser avoid the trivial solution. By contrast, the initial condition in Eq. (57) is non-zero. We therefore tried biasing the sampling of the boundary/initial pseudo-observations so that more initial points were chosen compared to boundary points. This did make the trivial solution less of a problem, but prediction errors were higher for those optimisations which escaped the trivial optimum. This was in stark contrast to the BCGP, which consistently found the correct solution under different random restarts without any hand-tuning.

5.3.3 Effect of Observation Noise

In Sections 5.3.1 and 5.3.2 above, 2.5% Gaussian observation noise was applied to the function space observations $\boldsymbol{y}_u$ (measured as a percentage of the standard deviation of the underlying signal). To explore how results change as this noise level is varied, we re-

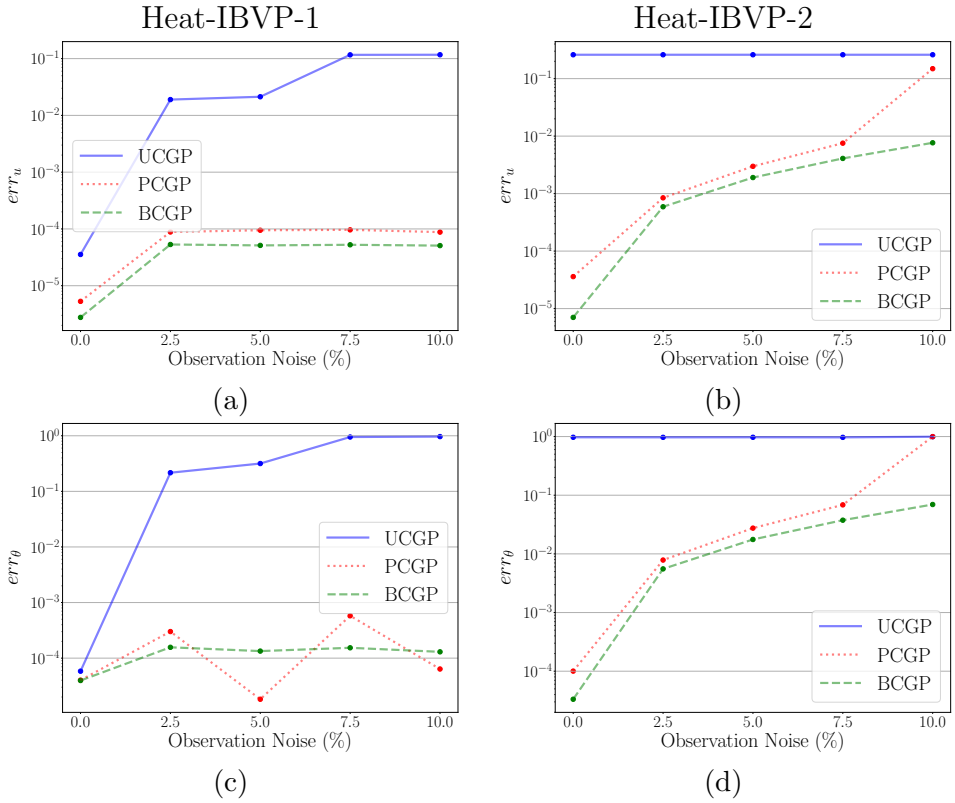


Figure 9: Effect of observation noise on results obtained using UCGPs, PCGPs and BCGPs for Heat-IBVP-1 and Heat-IBVP-2.

performed the experiments with noise levels ranging from 0% to 10%. The results obtained using UCGPs, PCGPs and BCGPs, respectively, are displayed as line plots against noise level in Figure 9. For Heat-IBVP-1, $N_b = 90$ boundary points were used when implementing the PCGP, while $N_b = 100$ points were used for Heat-IBVP-2.

The UCGP obtained strong accuracy for Heat-IBVP-1 in the noise free case; however, performance rapidly deteriorated as noise levels increased. For Heat-IBVP-2, even with low noise levels, the UCGP found the trivial solution and therefore incurred high errors in prediction of u and θ, indicating a lack of robustness of the UCGP to observation noise. The PCGP and BCGP results by contrast exhibited a more gentle deterioration with increasing observation noise. The BCGP results were almost always better, except for $err_{\boldsymbol{\theta}}$ in Heat-IBVP-1, where the accuracy of the PCGP oscillated as observation noise increased. Note also the high errors incurred by the PCGP in Heat-IBVP-2 for 10% observation noise. As discussed in Section 5.3.2, we manually filtered the PCGP optimisation results to remove those which found the trivial zero solution. With 10% observation noise, however, all random restarts found the trivial solution, leading to high errors in this case.

5.4 Wave Equation

The third PDE we considered was the wave equation, a hyperbolic, second-order PDE that is fundamental for describing wave phenomena in areas such as fluid dynamics, electromagnetics and acoustics. Over one spatial dimension, the equation takes the form

$$\mathcal{L}^{\theta}_{x,t}u(x,y) \triangleq \frac{\partial^2}{\partial t^2}u(x,t) - \theta\frac{\partial^2}{\partial x^2}u(x,t) = f(x,t) = 0, \tag{59}$$

with $\theta > 0$ the wave propagation speed. With this PDE, we again considered the inverse problem of learning $\theta = 1$, using two sets of boundary conditions.

5.4.1 Wave-IBVP-1

We first applied the wave equation over spatial domain $\Omega = (0, \pi)$ and time interval $(0, 1]$, with a Cauchy initial condition specified at $t = 0$ and a Neumann boundary condition specified on $\partial\Omega$:

$$\textbf{Wave-IBVP-1:} \begin{cases} u(x,0) = \cos(x), & x \in \bar{\Omega}, \\ \partial_t u(x,0) = \cos^2(x), & x \in \bar{\Omega}, \\ \partial_x u(x,t) = 0, & x \in \partial\Omega,\ t > 0. \end{cases} \tag{60}$$

The separation of variables technique can be used to show that the solution takes the form given in Eq. (61), which is plotted as using a heatmap in Figure 10 (a).

$$u(x,t) = \frac{1}{2}t + \cos(t)\cos(x) + \frac{1}{4}\sin(2t)\cos(2x). \tag{61}$$

We generated a training data set consisting of $N_u = 25$ noise corrupted observations of u, and $N_f = 25$ collocation points for f. We first addressed the problem of learning θ and u from this data set using a UCGP, i.e. ignoring boundary condition information during parameter inference as in Raissi et al. (2017). UCGP performance was poor, with best $err_{\boldsymbol{\theta}}$ value of 4.7×10^{-1}. Furthermore, we found that the parameter estimate was unstable with respect to the initial value of the hyperparameters. The posterior mean of the UCGP incurred $err_{\boldsymbol{u}}$ value of 7.9×10^{-2}. It is plotted in Figure 10 (e), and it is clear that it failed to capture the true function on the upper right of the spatio-temporal domain.

Because this problem involved Neumann boundary conditions, we constructed a BCGP following Eq. (30), adjusted in the manner described in Section 3.2.4 to account for the Cauchy initial condition—see Appendix B.2.5 for details. We found that the BCGP was significantly more accurate than the UCGP under both error metrics, yielding an $err_{\boldsymbol{\theta}}$ value of 1×10^{-2} and $err_{\boldsymbol{u}}$ value of 8.3×10^{-3}. Its posterior mean is displayed in Figure 10 (c), and it exhibits strong agreement with the true solution. Also, in contrast to the UCGP, we found that BCGP results were stable with respect to initialisation during training.

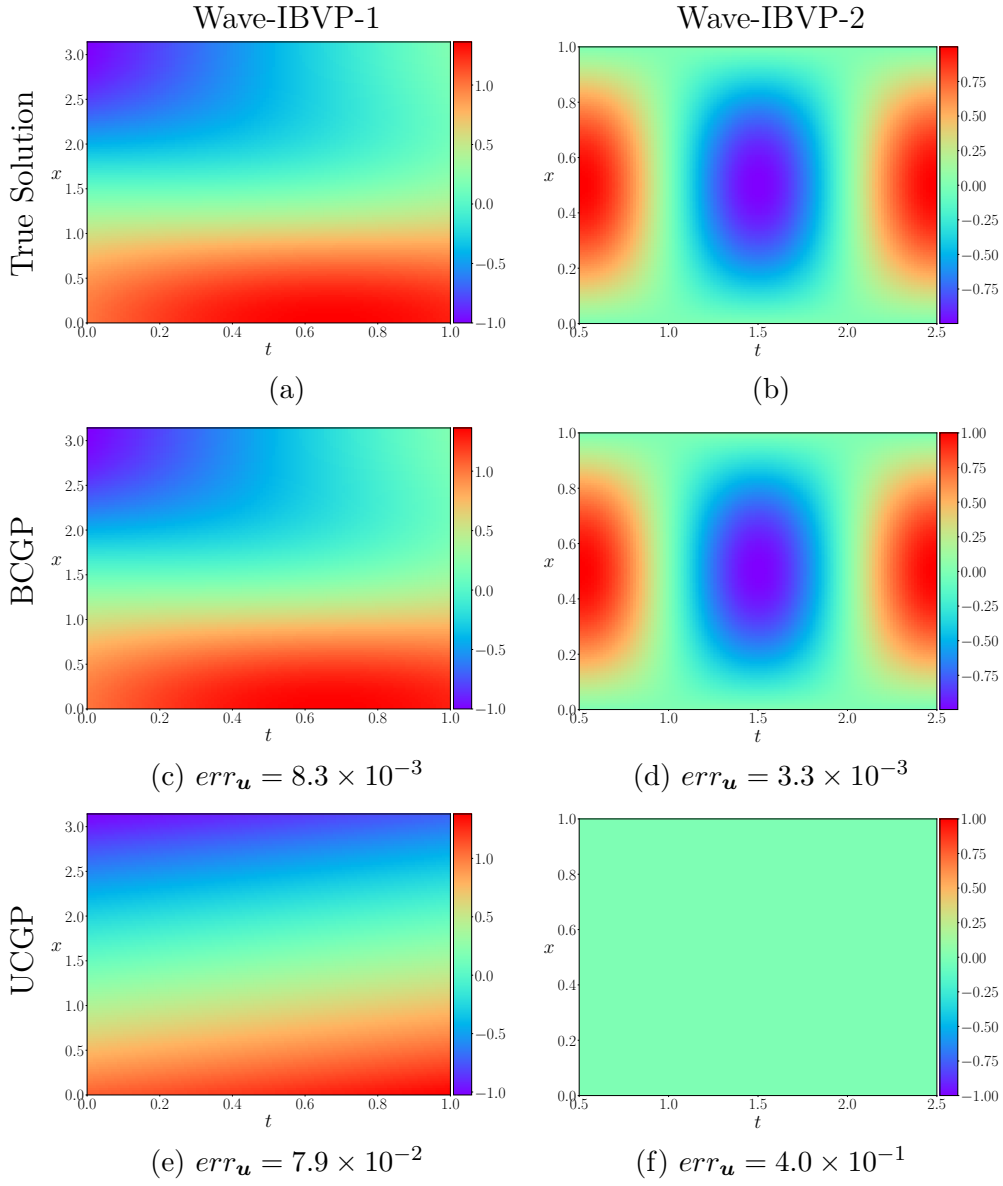


Figure 10: Plots of true and predicted results for Wave-IBVP-1 and Wave-IBVP-2.

5.4.2 Wave-IBVP-2

In our second example involving the wave equation, we considered spatial domain $\Omega = (0, 1)$ over the time interval $(0.5, 2.5]$, and applied a Cauchy initial condition with periodic boundary conditions:

$$\textbf{Wave-IBVP-2:} \begin{cases} u(x,0) = \sin(\pi x), & x \in \bar{\Omega}, \\ \partial_t u(x,0) = 0, & x \in \bar{\Omega}, \\ u(0,t) = u(1,t). \end{cases} \tag{62}$$

This IBVP can be solved using d'Alembert's formula to give the below solution, which is plotted in Figure 10 (b).

$$u(x,t) = \sin(x\pi)\cos((t-0.5)\pi). \tag{63}$$

We again generated $N_u = N_f = 25$ observations, with the objective of learning θ and u. We first applied a UCGP to this problem, which completely failed under each random restart performed. The best result obtained was an estimate of $\hat{\theta} = 0.06$ and $err_{\boldsymbol{u}}$ value of 4.0×10^{-1}. The posterior mean is plotted in Figure 10 (f), from which is apparent that it learned the trivial zero solution.

Because this problem involves periodic boundary conditions, a periodic kernel could be used here to construct a BCGP, adjusted slightly as in Section 5.4.2 above to account for the Cauchy initial condition. However, we found that better performance could be used by designing a periodic BCGP kernel by a modification of the Dirichlet BCGP in Eq. (29)—full details of this construction are given in Appendix B.2.6. In contrast to the UCGP, we found that the BCGP could accurately capture both θ and u. Specifically, the BCGP learned an estimate of $\hat{\theta} = 1.01$ and incurred an $err_{\boldsymbol{u}}$ value of only 3.3×10^{-3}. The posterior mean is plotted in Figure 10 (d), which illustrates that it almost perfectly recovered the true function.

5.5 Advection-Diffusion Equation

We next considered an advection-diffusion equation

$$\mathcal{L}^{\theta}_{x,t} u(x,y) \triangleq \frac{\partial}{\partial t}u(x,t) - \theta_1 \frac{\partial}{\partial x}u(x,t) - \theta_2 \frac{\partial^2}{\partial x^2}u(x,t) = f(x,t) = 0, \tag{64}$$

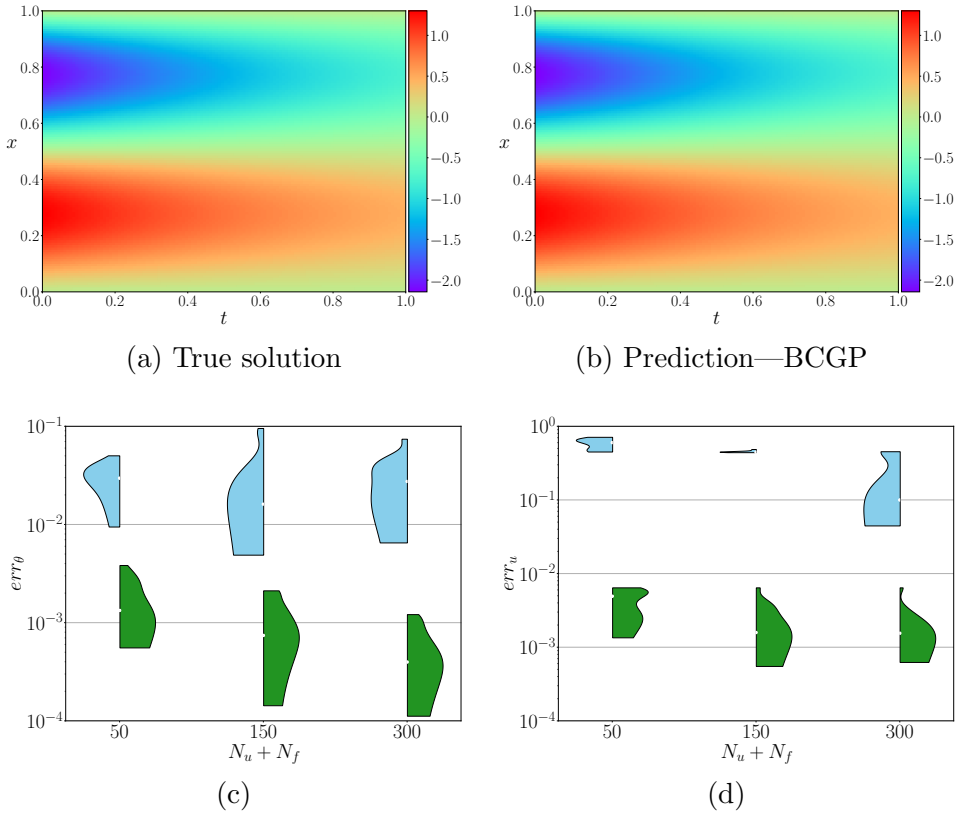
over the spatial domain $\Omega = (0,1)$ and time interval $(0,1]$, under the following Dirichlet initial/boundary conditions:

$$\textbf{Adv-Diff-IBVP:} \begin{cases} u(x,0) = \sin(2\pi x)\exp(x), & x \in \bar{\Omega}, \\ u(x,t) = 0, & x \in \partial\Omega,\ t > 0. \end{cases} \tag{65}$$

The parameter θ_1 in Eq. (64) controls the strength of the advection term, while θ_2 controls the level of diffusion. Setting $\boldsymbol{\theta} = (\theta_1, \theta_2) = (0.05, 0.025)$, it can be directly verified that the true solution takes the form given in Eq. (66), which is visualised in Figure 11 (a).

$$u(x,t) = \sin(2\pi x)\exp\left(-(0.1\pi^2 + 0.025)t + x\right). \tag{66}$$

We used this system to explore how the number of observations N_u and collocation points N_f affects the accuracy of the inference results. Specifically, we generated data sets of sizes $N_u + N_f = \{50, 150, 300\}$ with $N_u = N_f$, where the function space observations $\boldsymbol{y}_u$ were corrupted by 2.5% Gaussian noise. From these data sets, we used BCGPs and UCGPs to learn both the solution function u and parameter vector $\boldsymbol{\theta}$. The BCGPs were specified using mean and kernel functions of the form stated in Eq. (29) for Dirichlet boundary conditions—see Appendix B.2.7 for further details. To give some uncertainty quantification, we repeated the experiments under 15 different random regenerations of each data set. Distribution plots of $err_{\boldsymbol{u}}$ and $err_{\boldsymbol{\theta}}$ against $N_u + N_f$ for both types of model are shown in the bottom row of



(a) True solution (b) Prediction—BCGP

(c) (d)

Figure 11: The top row displays the true solution and BCGP prediction for the Advection-Diffusion-IBVP. In the bottom row, the distributions of UCGP $err_{\boldsymbol{u}}$ and $err_{\boldsymbol{\theta}}$ results are displayed in blue, with BCGP results displayed in green.

Figure 11, where UCGP results are coloured blue, and BCGP results green. Under both error measures, the BCGP outperformed the UCGP by several orders of magnitude. Figure 11 (b) shows the posterior mean of a BCGP trained on a data set with $N_u = N_f = 10$. Even given these sparse data, BCGP was able to accurately capture the true function, with $err_{\boldsymbol{u}}$ value of 9.0×10^{-3} and parameter estimate $\hat{\boldsymbol{\theta}} = (0.054, 0.0252)$.

We remark that we also tried fitting a PCGP for this problem but obtained similar results to the UCGP. For all data set sizes, training yielded high values of both the observation and nugget noise parameters. It is possible that more accurate results could be obtained via a bespoke training routine whereby a bound is placed on the values that the nugget parameter can take (which we found sometimes helps on Heat-IBVP-2). However, we felt that such (possibly application-specific) modifications were beyond the scope of this paper. Note also the contrast to the BCGP method, which does not require any boundary pseudo-observations and therefore no tuning of this type was required.

5.6 Helmholtz Equation

Our final experiment involved the two-dimensional Helmholtz equation

$$\mathcal{L}^{\theta}_{x,y}u(x,y) \triangleq \theta^2 u(x,y) - \frac{\partial^2}{\partial x^2}u(x,y) - \frac{\partial^2}{\partial y^2}u(x,y) = f(x,y) \tag{67}$$

over the spatial domain $\Omega = (0,1)^2$, subject to Dirichlet conditions on the upper right portion of the boundary $\partial\Omega_1$ (see Eq. 73) and Neumann conditions on the lower right boundary $\partial\Omega_2$ (see Eq. 74) as below.

$$\textbf{Helmholtz-BVP:} \begin{cases} u(x,y) = 0, & (x,y) \in \partial\Omega_1, \\ \boldsymbol{n}(x,y) \cdot \nabla u(x,y) = 0, & (x,y) \in \partial\Omega_2. \end{cases} \tag{68}$$

We set $\theta = 3$ and specified the below solution function, from which the form of f can be found by directly evaluating the Helmholtz PDE.

$$u(x,y) = \left(1 - x^2\right)\left(1 - y^2\right) + \cos\left(\frac{\pi x}{2}\right)\left(\exp(-y) + y - (1 + \exp(-1))\right). \tag{69}$$

Once again, the inverse problem of recovering the solution u and PDE parameter θ was considered, in this case given 10 observations each of u and f. Observations in both spaces were subject to white noise, with common standard deviation $\sigma_u = \sigma_f = 0.01$. We modelled this example on Section 4.2 of Gulian et al. (2022), in which BCGPs were constructed using the spectral expansion method (see Section 3.3). This method is not suitable for inverse problems without additional adjustments, however, and therefore θ was fixed to its true value in their work.

To specify a BCGP for this problem, we used mean and kernel functions of the form given Eq. (32), as the BVP involved mixed Dirichlet and Neumann conditions—see Appendix B.2.8 for further details. The BCGP yielded an $err_{\boldsymbol{u}}$ value of 5.0×10^{-3} and $err_{\boldsymbol{\theta}}$ value of 1.2×10^{-2}, in both cases lower than the values $err_{\boldsymbol{u}} = 1.8 \times 10^{-2}$ and $err_{\boldsymbol{\theta}} = 1.0 \times 10^{-1}$ obtained using a UCGP. The posterior means for both models are displayed in the top two rows of Figure 12. The density plots of the prediction error $|u(x) - \hat{u}(x)|$ and the posterior predictive standard deviation $\sigma(x,t)$ in the final two rows of the figure further emphasise the advantage of the BCGP framework. From panels (c) and (e), we see that the highest level of prediction error and uncertainty for the UCGP is at the point $(1,1)$. However, the function value is prescribed at this point and therefore uncertainty should in principle be lowest, which is precisely what is captured by the BCGP in panels (d) and (f).

5.7 Discussion

The BCGP and BCNN comparison in Section 5.2 showed that the BCGP was able to obtain more accurate results from sparse data set sizes, while for larger data set sizes, the results of the two approaches were more closely aligned. The UCGP/PCGP comparisons in Sections 5.3-5.6 demonstrated that the BCGP approach was more robust to data set size and observation noise. In particular, we consistently encountered the trivial solution problem of learning the zero function across the domain when using UCGPs and PCGPs for larger noise levels, whereas the BCGP results exhibited more robustness.

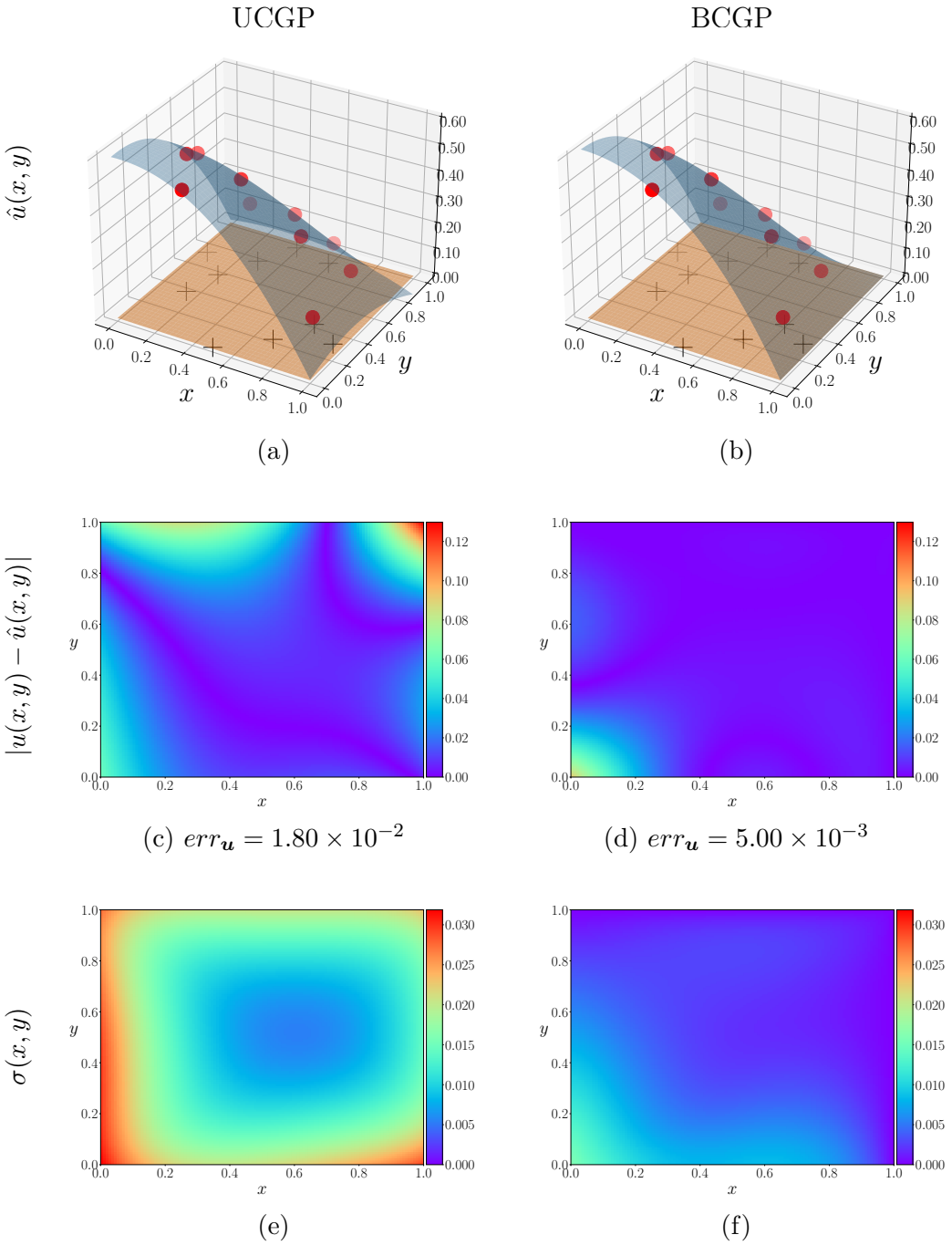


Figure 12: Posterior predictive results for Helmholtz equation using UCGP (left column) and BCGP (right column). The red points in the top row show the observations in u-space, while the black crosses show the input locations for the observations in f-space.

For our inverse problem experiments, more PDE collocation points could have been used, however, we found that this can actually lead to worse inference results for PDEs with $f = 0$ (see Eq. 1). The reason for this counterintuitive result is that introducing more pseudo-observations of $f = 0$ increases the attractor domain of the trivial solution, whereby the posterior of the GP collapses to zero across the domain with very low uncertainty. For an example of this phenomenon, we direct the reader to Section 4.4 of Dalton et al. (2024). We therefore set the number of PDE collocation points N_f equal to the number of solution space observations N_u, which we found to be a good heuristic to mitigate the problem.

6. Conclusions

In this work, we developed a new approach for constructing boundary constrained Gaussian processes (BCGPs) to allow for the exact imposition of Dirichlet, Neumann and mixed boundary conditions. We analysed the representational capacity of our BCGP framework under Dirichlet conditions, and found that our construction can satisfy universal approximation within the space of continuous functions which satisfy the boundary constraint. We also demonstrated an equivalence in the infinite width limit between a boundary-constrained neural network (BCNN) and a BCGP. Finally, extensive numerical experiments were conducted for a range of linear (I)BVPs, where the primary focus was the inference of any unknown parameters of the underlying differential operator. A consistent pattern observed was that the BCGP framework offered results that were robust to initialisation, observation noise and data sparsity.

There are several ways in which future work could expand on that which has been presented in this paper. Firstly, the GPR approach we made use of is limited to the modelling of linear PDEs, as Proposition 6 does not hold for non-linear differential operators. Several approaches have been proposed to extend GP inference to non-linear PDEs (Raissi et al., 2018; Chen et al., 2022; Long et al., 2022). Furthermore, our approach is not appropriate for large data sets, as the computational cost of performing full GP inference would be prohibitive. This issue could be alleviated through the use of so-called sparse GP approaches, which allow for approximate inference with linear complexity in the number of training points (Titsias, 2009). We also only considered boundary information in the design of the hard-constrained mean and covariance functions. However, knowledge of the PDE itself could also be used when designing these functions (Harkonen et al., 2023; Sarkka et al., 2013). Future work could also consider techniques on how to avoid the trivial solution of learning the zero function across the domain in the case of sparse and noisy data. For instance, in practical applications information about the accuracy of the measurements is typically available. This could potentially be used as an informative prior on the observation noise parameter to potentially alleviate the problem. Finally, here we performed inference in an empirical Bayesian manner, which did not yield uncertainty bounds for the parameter estimates. This could be rectified in future work by taking a fully Bayesian approach to inference using Markov chain Monte Carlo, which would allow uncertainty to be more fully quantified.

Acknowledgments

This work has been funded by the Engineering and Physical Sciences Research Council (EPSRC) of the United Kingdom, grant reference numbers EP/T017899/1, EP/S030875/1 and EP/S020950/1.

Appendix A. Background Material

A.1 Linear differential operators

Following Definition 1 of Chen et al. (2022), we define a linear differential operator as follows.

Definition 17 (Linear Differential Operator) *A linear differential operator on a function $u(\boldsymbol{x})$ takes the form:*

$$\mathcal{L}^{\boldsymbol{\theta}}_{\boldsymbol{x}}[u](\boldsymbol{x}) \triangleq \sum_{i=1}^{L} c_i(\boldsymbol{x}, \boldsymbol{\theta}) \nabla_{\boldsymbol{\alpha}_i} u(\boldsymbol{x}), \qquad \textit{with} \qquad \nabla_{\boldsymbol{\alpha}_i} u \triangleq \frac{\partial^{\alpha_{i,1}}}{\partial x_1^{\alpha_{i,1}}} \cdots \frac{\partial^{\alpha_{i,D}}}{\partial x_D^{\alpha_{i,D}}} u,$$

where $\boldsymbol{\theta}$ parameterises the operator, L is the number of derivatives, $\boldsymbol{\alpha}_i = (\alpha_{i,1}, \cdots, \alpha_{i,D})$ indicates the order of the derivative for each input dimension, $\boldsymbol{x} = (x_1, \cdots, x_D)$ and $c_i(\boldsymbol{x}, \boldsymbol{\theta})$ is the coefficient at $\boldsymbol{x}$.

A.2 Reproducing Kernel Hilbert Spaces

In Section 4.1, we described an RKHS $\mathcal{H}_k$ as a space of "well-behaved" functions for which there exists a unique "reproducing kernel" k. More formally, an RKHS is defined as follows (Kanagawa et al., 2018).

Definition 18 (Reproducing Kernel Hilbert Space) *Let $\mathcal{X}$ be a non-empty set, k a kernel on $\mathcal{X} \times \mathcal{X}$ and $\mathcal{H}$ an $\mathbb{R}$-Hilbert space over $\mathcal{X}$. That is $\mathcal{H}$ is a vector space consisting of functions u that map $\mathcal{X}$ to $\mathbb{R}$, which is equipped with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ that induces a metric for which the space is complete. $\mathcal{H}$ is additionally called a reproducing kernel Hilbert space (RKHS) with reproducing kernel k, if the following are satisfied:*

1. *For all $\boldsymbol{x} \in \mathcal{X}$, we have $k(\cdot, \boldsymbol{x}) \in \mathcal{H}$;*
2. *For all $\boldsymbol{x} \in \mathcal{X}$ and for all $u \in \mathcal{H}$, we have*

$$u(\boldsymbol{x}) = \langle u(\cdot), k(\cdot, \boldsymbol{x}) \rangle_{\mathcal{H}}.$$

The second condition above is called the *reproducing property* of the kernel. As discussed in Section 4.1, the correspondence between kernels and RKHSs is one-to-one (Aronszajn, 1950).

Theorem 19 (Moore–Aronszajn Theorem) *Let $\mathcal{X}$ be a non-empty set. Then for every kernel k on $\mathcal{X} \times \mathcal{X}$, there exists a unique RKHS $\mathcal{H}_k$ for which it is the reproducing kernel, and vice versa.*

The following result allows us to consider the elements of an RKHS in terms of finite weighted sums of its reproducing kernel (Steinward and Christmann, 2008, Theorem 4.21).

Theorem 20 (Reproducing kernel map representation) *Let $\mathcal{X}$ be a non-empty set and k a kernel on $\mathcal{X} \times \mathcal{X}$, with $\mathcal{H}_k$ its associated RKHS. Consider the set of functions*

$$\mathcal{H}_k^{pre} \triangleq \left\{ u(\boldsymbol{x}) = \sum_{i=1}^{N} c_i k\left(\boldsymbol{x}_i, \boldsymbol{x}\right) : N \in \mathbb{N}, c_1, \ldots, c_N \in \mathbb{R}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathcal{X} \right\}. \tag{70}$$

Then $\mathcal{H}_k^{pre} \subset \mathcal{H}_k$ and $\mathcal{H}_k^{pre}$ is dense in $\mathcal{H}_k$ with respect to the metric induced by $\langle \cdot, \cdot \rangle_{\mathcal{H}_k}$.

We remark that the density of $\mathcal{H}_k^{pre}$ in $\mathcal{H}_k$ is equivalent to stating that the topological completion of $\mathcal{H}_k^{pre}$ is equal to $\mathcal{H}_k$ (Searcóid, 2007, Theorem 4.2.1), with appropriate choice of inner product on $\mathcal{H}_k^{pre}$ (Kanagawa et al., 2018, page 11).

The implication of Theorem 20 for GPR is clear by noticing that, in the case of a zero prior mean function, the posterior GP mean $\boldsymbol{\mu}_*$ from Eq. (12) has exactly the form of a weighted sum of evaluations from the chosen kernel as in Eq. (70). Therefore, the RKHS of a given kernel k can be seen intuitively as the space of all posterior means of the GP, as mentioned in Section 4.1.

For the proof of Theorem 13, the following alternative (but equivalent) definition of a kernel is useful (Steinward and Christmann, 2008, Definition 4.1).

Definition 21 *Let $\mathcal{X}$ be a non-empty set. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel if there exists an $\mathbb{R}$-Hilbert space $\mathcal{H}_0$ and a map $\Phi_0 : \mathcal{X} \to \mathcal{H}_0$ such that, for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$, we have*

$$k(\boldsymbol{x}, \boldsymbol{x}') = \left\langle \Phi_0(\boldsymbol{x}), \Phi_0(\boldsymbol{x}') \right\rangle_{\mathcal{H}_0}. \tag{71}$$

We call Φ_0 a feature map of k, and $\mathcal{H}_0$ a feature space.

Similarly, the following alternative feature map representation of an RKHS will be of use (Steinward and Christmann, 2008, Theorem 4.21).

Theorem 22 (Feature map representation of RKHS) *Let $\mathcal{X}$ be a non-empty set and k a kernel on $\mathcal{X} \times \mathcal{X}$ with feature map $\Phi_0 : \mathcal{X} \to \mathcal{H}_0$ where $\mathcal{H}_0$ is an $\mathbb{R}$-Hilbert space. Consider the normed space*

$$\mathcal{H}_{\Phi_0} \triangleq \left\{ u : \mathcal{X} \to \mathbb{R} : \exists\, w \in \mathcal{H}_0 \text{ with } u(\boldsymbol{x}) = \langle w, \Phi_0(x) \rangle_{\mathcal{H}_0} \text{ for all } \boldsymbol{x} \in \mathcal{X} \right\},$$

where

$$\|u\|_{\mathcal{H}_{\Phi_0}} \triangleq \inf \left\{ \|w\|_{\mathcal{H}_0} : w \in \mathcal{H}_0 \text{ with } u = \langle w, \Phi_0(\cdot) \rangle_{\mathcal{H}_0} \right\}.$$

Then, $\mathcal{H}_k = \mathcal{H}_{\Phi_0}$.

We remark that kernels do not have unique feature maps; however, the above construction is independent of the specific choice of Φ_0.

A.3 Topological results

Two topological results are required for the proof of Lemma 27 (see Appendix C.3.2). The first is the *Tietze extension theorem* (van Douwen et al., 1977, Theorem A), and the second the *pasting lemma* (Dugundji, 1966, Theorem III.9.4).

Theorem 23 (Tietze extension theorem) *If A is a normal space and $f : X \to \mathbb{R}$ a bounded continuous function with X a compact subset of A, then there exists a bounded continuous function $F : A \to \mathbb{R}$ such that*

$$\begin{cases} F(x) = f(x), \ x \in X \\ \sup_{x \in A} |F(x)| = \sup_{x \in X} |f(x)| . \end{cases}$$

Theorem 24 (Pasting Lemma) *Let X and Y be closed subsets of a topological space A such that $A = X \cup Y$. If $F : A \to \mathbb{R}$ is continuous when restricted to both X and Y, then F is continuous.*

A.4 Construction of approximate distance functions for line segments

Here we describe the trimming procedure outlined in Section 2.1 of (Biswas and Shapiro, 2004) for constructing a normalised ADF to a line segment in $\mathbb{R}^2$. Note that the following arguments extend to more general curves and surfaces (Shapiro and Tsukanov, 1999b).

A line segment between the points (x_1, y_1) and (x_2, y_2) can be represented as the intersection of an infinite line passing along the segment and a trim region such as a circle—see Figure 3A of (Biswas and Shapiro, 2004) for an illustration. A trimming operation constructs an ADF for the line segment by combining normalised functions for the infinite line and the trim region. A normalised function for the infinite line passing between (x_1, y_1) and (x_2, y_2) can be found as

$$h(x, y) = \frac{(x - x_1)(y_2 - y_1) - (y - y_1)(x_2 - x_1)}{L}$$

where L is the length of the segment. We set the trim region to be a circle with diameter L and centre $(x_c, y_c) = ((x_1 + x_2)/2, (y_1 + y_2)/2)$. This shape can be represented by the inequality $t(x, y) \geq 0$ where

$$t(x, y) = \frac{1}{L}\left[\left(\frac{L}{2}\right)^2 - (x - x_c)^2 - (y - y_c)^2\right].$$

There are several ways in which a normalised ADF ϕ can be constructed from the functions h and t. Here we follow Eq. (5) of (Biswas and Shapiro, 2004) which gives

$$\phi(x, y) = \sqrt{h(x, y)^2 + \frac{\left(\sqrt{t(x, y)^2 + h(x, y)^2} - t(x, y)\right)^2}{4}}, \tag{72}$$

which is twice differentiable at all points away from the line segment.

Appendix B. Additional BCGP Material

B.1 Mean function interpolation on hypercube domains

As discussed in Remarks 2 and 10, the functions b, a and h in Eqs. (2)-(4) may only be known on the boundary of the domain, $\partial\Omega$. In this case, interpolation can be used to define functions defined on $\bar{\Omega}$ which match the boundary values.

Here, we present an interpolation algorithm to construct the mean function $\tilde{m}$ for Dirichlet boundary conditions from Eq. (29) given boundary function $b : \partial\Omega \to \mathbb{R}$ where the domain of interest is the unit cube in $\mathbb{R}^D$, i.e. $\bar{\Omega} = [0,1]^D$. In this case, the boundary $\partial\Omega$ can be decomposed into $2D$ *quasi-disjoint* segments as

$$\partial\Omega = \bigcup_{i=1}^{D} \bigcup_{j=0}^{1} \partial\Omega_{ij}, \text{ where } \partial\Omega_{ij} = \{\boldsymbol{x} = (x_1, \ldots, x_D) \in \Omega : x_i = j\}.$$

The boundary constrained mean function $\tilde{m}$ can then be using the iterative procedure shown in Algorithm 1. For the first input dimension (x_1), the mean is initialised on line 1 by linearly interpolating between the specified boundary functions on $\partial\Omega_{10}$ and $\partial\Omega_{11}$. For each additional dimension $i = 2, \ldots, D$, the mean is augmented by adding a similar linear interpolation for the i^{th} input dimension on line 4. The difference is that the interpolation is done with respect to the augmented boundary functions $\tilde{b}_{i,0}$ and $\tilde{b}_{i,1}$ found on line 3, which ensures that the boundary values for dimensions $l < i$ are not affected by the augmentation procedure.

Algorithm 1 BCGP mean function generator for hypercube

Input: Domain $\bar{\Omega} = [0,1]^D$, boundary function $b : \partial\Omega \to \mathbb{R}$
Output: BCGP mean function $\tilde{m} : \bar{\Omega} \to \mathbb{R}$

1: $\tilde{m}_1(\boldsymbol{x}) = b(\boldsymbol{x} : x_1 = 0)(1 - x_1) + b(\boldsymbol{x} : x_1 = 1)x_1$
2: **for** $i = 2 : D$
3: $\quad \tilde{b}_{i,j}(\boldsymbol{x}) = b(\boldsymbol{x} : x_i = j) - \tilde{m}_{i-1}(\boldsymbol{x} : x_i = j)$ for all $j \in \{0, 1\}$
4: $\quad \tilde{m}_i(\boldsymbol{x}) = \tilde{m}_{i-1}(\boldsymbol{x}) + \tilde{b}_{i,0}(\boldsymbol{x})(1 - x_i) + \tilde{b}_{i,1}(\boldsymbol{x})x_i$
5: **end for**
6: $\tilde{m}(\boldsymbol{x}) \triangleq \tilde{m}_d(\boldsymbol{x})$

The function $\tilde{m}$ exactly satisfies the given Dirichlet boundary conditions:

Proposition 25 *Dirichlet boundary conditions of the form given in Eq. (2), let $\tilde{m}$ be the mean function generated by Algorithm 1. Then we have*

$$\tilde{m}(\boldsymbol{x}) = b(\boldsymbol{x}) \text{ for all } \boldsymbol{x} \in \partial\Omega.$$

This result is proved in Appendix C.5. We use this algorithm to specify the BCGP mean function for Heat-IBVP-1 in Section 5.3.1, adjusted slightly to account for an initial condition—see Appendix B.2.3 below for details.

B.2 Explicit form of BCGPs used in numerical experiments

B.2.1 Explicit form of BCGP used for Poisson-BVP-1

We used normalised ADFs $\phi_1(x) = x$ and $\phi_2(x) = 2 - x$ for the boundary segments $\partial\Omega_1 = \{0\}$ and $\partial\Omega_2 = \{2\}$ respectively, and found a joined ADF ϕ by inputting ϕ_1 and ϕ_2 into Eq. (25).

Eq. (50) specifies mixed boundary conditions of the form given in Eq. (4), with specific values $b = 0$, $a = -1$ and $h = 0$. A mixed BCGP can then be constructed for this problem my plugging these values of ϕ_1, ϕ_2, ϕ b, a and h into Eqs. (32)-(33), to give mean function $\tilde{m}(x) = 0$ and kernel function

$$\tilde{k}(x, x') = \mathcal{B}_x^{m_1} \mathcal{B}_{x'}^{m_1} k_1(x, x') + \mathcal{B}_x^{m_2} \mathcal{B}_{x'}^{m_2} k_2(x, x')$$

in which $\mathcal{B}_x^{m_1}$ and $\mathcal{B}_x^{m_2}$ are given by

$$\begin{aligned}
\mathcal{B}_x^{m_1}[\cdot](x) &= (\phi_1(x) - \phi(x)\left[\phi_1(x) + \nabla\phi_2(x) \cdot \nabla\phi_1(x)\right])\,[\cdot](x) \\
&\quad - \phi(x)\phi_1(x)\nabla\phi_2(x) \cdot \nabla[\cdot](x), \\
\mathcal{B}_x^{m_2}[\cdot](x) &= \phi(x)\phi_2(x)[\cdot](x).
\end{aligned}$$

In the same way, a BCNN can be specified as

$$\tilde{u}_{nn}(x) = \mathcal{B}_x^{m_1}[\hat{u}_{nn}^{(1)}](x) + \mathcal{B}_x^{m_2}[\hat{u}_{nn}^{(2)}](x),$$

where $\hat{u}_{nn}^{(1)}$ and $\hat{u}_{nn}^{(2)}$ are neural networks and $\mathcal{B}_x^{m_1}$ and $\mathcal{B}_x^{m_2}$ are as given above.

B.2.2 Explicit form of BCGP used for Poisson-BVP-3

To construct an ADF $\phi_1(x, y)$ for the upper curved portion of the boundary $\partial\Omega_1$, we first constructed individual ADFs to each of the 100 line segments along the boundary (as in Figure 1 b), and then joined them into a single ADF ϕ_1 using Eq. (25) (as in Figure 3 b). For the lower flat boundary $\partial\Omega_2$, we simply set $\phi_2(x, y) = y$, and then found joined ADF by inputting ϕ_1 and ϕ_2 into Eq. (25).

Eq. (52) specifies mixed boundary conditions of the form given in Eq. (4), with specific values $b = 0$, $a = 0$ and $h = 0$. A mixed BCGP can then be constructed for this problem by plugging these values of ϕ_1, ϕ_2, ϕ b, a and h into Eqs. (32)-(33), to give mean function $\tilde{m}(x, y) = 0$ and kernel function

$$\tilde{k}([x, y], [x', y']) = \mathcal{B}_{x,y}^{m_1} \mathcal{B}_{x'}^{m_1} k_1([x, y], [x', y']) + \mathcal{B}_{x,y}^{m_2} \mathcal{B}_{x'}^{m_2} k_2([x, y], [x', y'])$$

in which $\mathcal{B}_{x,y}^{m_1}$ and $\mathcal{B}_{x,y}^{m_2}$ are given by

$$\begin{aligned}
\mathcal{B}_{x,y}^{m_1}[\cdot](x, y) &= (\phi_1(x, y) - \phi(x, y)\nabla\phi_2(x, y) \cdot \nabla\phi_1(x, y))\,[\cdot](x, y) \\
&\quad - \phi(x, y)\phi_1(x, y)\nabla\phi_2(x, y) \cdot \nabla[\cdot](x, y), \\
\mathcal{B}_{x,y}^{m_2}[\cdot](x, y) &= \phi(x, y)\phi_2(x, y)[\cdot](x, y).
\end{aligned}$$

In the same way, a BCNN can be specified as

$$\tilde{u}_{nn}(x, y) = \mathcal{B}_{x,y}^{m_1}[\hat{u}_{nn}^{(1)}](x, y) + \mathcal{B}_{x,y}^{m_2}[\hat{u}_{nn}^{(2)}](x, y),$$

where $\hat{u}_{nn}^{(1)}$ and $\hat{u}_{nn}^{(2)}$ are neural networks and $\mathcal{B}_{x,y}^{m_1}$ and $\mathcal{B}_{x,y}^{m_2}$ are as given above.

B.2.3 Explicit form of BCGP used for Heat-IBVP-1

Because this IBVP only involves Dirichlet boundary conditions, a BCGP of the form given in Eq. (29) can be used, given the following ADF for the spatial boundary $\partial\Omega = \{\pi/2, -\pi/2\}$ and initial time $t = 0$

$$\phi(x,t) = \left(x + \frac{\pi}{2}\right)\left(\frac{\pi}{2} - x\right)t.$$

Since the Dirichlet values in Eq. (55) are only known at the initial / boundary points, Algorithm 1 can be followed to find a mean function $\tilde{m}$ which interpolates these values into the interior of the spatio-temporal domain. The final BCGP then has the below form.

$$\begin{aligned}\tilde{m}(x,t) &= 2\pi x\exp(-t) + (1-t)(\sin(x) - 2\pi x),\\ \tilde{k}([x,t],[x',t']) &= \phi(x,t)\phi(x',t')k([x,t],[x',t']).\end{aligned}$$

B.2.4 Explicit form of BCGP used for Heat-IBVP-2

A BCGP for this problem can be constructed following Eq. (29) by treating the time input as another spatial boundary. In this case, we can specify an ADF for the boundary $\partial\Omega$ and initial time $t = 0$ using the product join from Eq. (29) as

$$\phi(x,y,t) = x(\pi - x)y(\pi - y)t.$$

Then, notice that the initial condition in Eq. (57) also satisfies the homogeneous boundary conditions; therefore we can treat this function in the same manner as $b(\boldsymbol{x})$ in Eq. (29). This means that the following form of mean and kernel functions yield an appropriate BCGP:

$$\begin{cases}\tilde{m}(x,y,t) = \sin(x)\sin(y),\\ \tilde{k}([x,y,t],[x',y',t']) = \phi(x,y,t)\phi(x',y',t')k([x,y,t],[x',y',t']).\end{cases}$$

B.2.5 Explicit form of BCGP used for Wave-IBVP-1

We constructed a mean function $\tilde{m}$ for the Cauchy initial condition in the same manner as the example in Section 3.2.4, to give

$$\tilde{m}(x,t) = \cos(x) + t\cos^2(x).$$

Furthermore, note that this mean function also satisfies the vanishing spatial derivative boundary condition in Eq. (60), and therefore it is appropriate to use as the mean function in a BCGP for this IBVP.

Constructing an associated boundary-constrained covariance function involved two stages. Firstly, we designed a covariance function to satisfy the Neumann boundary by inputting spatio-temporal kernels $k_1([x,t];[x',t'])$ and $k_2([x,t];[x',t'])$ into Eq. (30) with $a = 0$ and ADF $\phi(x) = x(\pi - x)$. Secondly, we simply multiplied this kernel by factors of time squared (as in Section 3.2.4) so that all samples from a zero-mean GP with this kernel were constrained to have zero time derivative at $t = 0$. This yielded the following boundary constrained kernel:

$$\tilde{k}([x,t],[x',t']) = t(t')^2\left[\mathcal{B}_x^{r_1}\mathcal{B}_{x'}^{r_1}k_1([x,t],[x',t']) + \mathcal{B}_x^{r_2}\mathcal{B}_{x'}^{r_2}k_2([x,t],[x',t'])\right],$$

where the linear operators $\mathcal{B}_x^{r_1}$ and $\mathcal{B}_x^{r_2}$ are given by

$$\mathcal{B}_x^{r_1}[\cdot](x,t) = [\cdot](x,t) - \phi(x)\nabla\phi(x)\cdot\nabla[\cdot](x,t),$$
$$\mathcal{B}_x^{r_2}[\cdot](x,t) = \phi(x)^2[\cdot](x,t).$$

B.2.6 Explicit form of BCGP used for Wave-IBVP-2

Designing a BCGP mean and kernel under periodic boundary conditions involves two stages.

Firstly, we design a mean and kernel assuming homogeneous Dirichlet boundary conditions (i.e. $b = 0$ in Eq. 2). Note that the initial condition in Eq. (62) satisfies this condition, so we use it as the mean function $\tilde{m}$ in Eq. (29). Then, treating time as another spatial boundary, we define an ADF for Eq. (29) as

$$\phi(x,t) = x(1-x)t^2$$

where t is squared to ensure the derivative process satisfies the Cauchy initial condition in the IBVP.

In the second stage, we place a GP prior on the value of the function at the boundary, which is solely a function of time and not space. Adding this to the above GP gives final mean and kernel of the form

$$\begin{aligned}\tilde{m}(x,t) =& \sin(x\pi),\\ \tilde{k}([x,t],[x',t']) =& \phi(x,t)\phi(x',t')k_1([x,t],[x',t']) + t^2(t')^2k_2(t,t'),\end{aligned}$$

where again the t squared factor in front of k_2 ensure the initial condition is satisfied.

B.2.7 Explicit form of BCGP used for Adv-Diff-IBVP

As in Appendix B.2.4 above, a BCGP for this problem can be constructed following Eq. (29) by treating the time input as another spatial boundary. In this case, we can specify an ADF for the boundary $\partial\Omega$ and initial time $t = 0$ using the product join from Eq. (24) as

$$\phi(x,t) = x(1-x)t.$$

Then, notice that the initial condition in Eq. (65) also satisfies the homogeneous boundary conditions. Therefore, we can treat this function the same as $b(\boldsymbol{x})$ in Eq. (29). This means that the following form of mean and kernel functions yield an appropriate BCGP:

$$\begin{cases}\tilde{m}(x,t) = \sin(2\pi x)\exp(x),\\ \tilde{k}([x,t],[x',t']) = \phi(x,t)\phi(x',t')k([x,t],[x',t']).\end{cases}$$

B.2.8 Explicit form of BCGP used for Helmholtz-BVP

The Dirichlet portion of the boundary $\partial\Omega_1$ is formed by the upper and right portions of the square spatial domain. Specifically, we have

$$\partial\Omega_1 = \left\{(x,y)\in[0,1]^2 : x = 1 \text{ or } y = 1\right\}. \tag{73}$$

The Neumann portion of the boundary $\partial\Omega_2$ is formed by the left and lower portions of the domain:

$$\partial\Omega_2 = \left\{(x, y) \in [0, 1]^2 : x = 0 \text{ or } y = 0\right\}. \tag{74}$$

We specified an ADF ϕ_1 for $\partial\Omega_1$ by joining the functions $\phi_1^{(1)}(x, y) = 1 - x$ and $\phi_1^{(2)}(x, y) = 1 - y$ using Eq. (25). Similarly, an ADF ϕ_2 for $\partial\Omega_1$ was specified by joining the functions $\phi_2^{(1)}(x, y) = x$ and $\phi_2^{(2)}(x, y) = y$ in the same manner. Finally, a joint ADF ϕ was found by joining ϕ_1 and ϕ_2, again using Eq. (25).

The mixed boundary conditions in Eq. (68) are the same as those given in Eq. (52). Therefore, a mixed BCGP (see Eq. 4) can be specified for this problem using ϕ_1, ϕ_2 and ϕ described above, where $\tilde{m}$ and $\tilde{k}$ have the same form as presented in Appendix B.2.2.

Appendix C. Proofs of Theoretical Results

C.1 Proof of Proposition 8

Proof Proving Proposition 8 simply requires the solution structure $\tilde{u}$ from Eq. (27) to be plugged in to the Robin condition from Eq. (3), before then showing that the constraint is satisfied under the assumption that ϕ is a normalised ADF for the boundary $\partial\Omega$. We present these steps in the following.

Assume $\boldsymbol{x} \in \partial\Omega$. Then $\nabla\tilde{u}(\boldsymbol{x})$ can be evaluated by applying the chain rule to each term in Eq. (27) and ignoring those terms with involving $\phi(\boldsymbol{x}) = 0$ to yield:

$$\nabla\tilde{u}(\boldsymbol{x}) = \nabla\hat{u}_1(\boldsymbol{x}) + \nabla\phi(\boldsymbol{x})a(\boldsymbol{x})\hat{u}_1(\boldsymbol{x}) - \nabla\phi(\boldsymbol{x})\nabla\phi(\boldsymbol{x}) \cdot \nabla\hat{u}_1(\boldsymbol{x}) - \nabla\phi(\boldsymbol{x})h(\boldsymbol{x}).$$

The inner product $\boldsymbol{n}(\boldsymbol{x}) \cdot \nabla\tilde{u}(\boldsymbol{x})$ with $\boldsymbol{n}(\boldsymbol{x})$ the outward normal vector is then given by

$$\boldsymbol{n}(\boldsymbol{x})\cdot\nabla\hat{u}_1(\boldsymbol{x})+\boldsymbol{n}(\boldsymbol{x})\cdot\nabla\phi(\boldsymbol{x})a(\boldsymbol{x})\hat{u}_1(\boldsymbol{x})-(\boldsymbol{n}(\boldsymbol{x})\cdot\nabla\phi(\boldsymbol{x}))(\nabla\phi(\boldsymbol{x})\cdot\nabla\hat{u}_1(\boldsymbol{x}))-\boldsymbol{n}(\boldsymbol{x})\cdot\nabla\phi(\boldsymbol{x})h(\boldsymbol{x}).$$

Since ϕ is a normalised ADF, $\nabla\phi(\boldsymbol{x}) = -\boldsymbol{n}(\boldsymbol{x})$ when $\boldsymbol{x} \in \partial\Omega$ (see Eq. 23), meaning the above expression simplifies to give

$$\begin{aligned}\boldsymbol{n}(\boldsymbol{x}) \cdot \nabla u(\boldsymbol{x}) &= \boldsymbol{n}(\boldsymbol{x}) \cdot \nabla\hat{u}_1(\boldsymbol{x}) - a(\boldsymbol{x})\hat{u}_1(\boldsymbol{x}) - \boldsymbol{n}(\boldsymbol{x}) \cdot \nabla\hat{u}_1(\boldsymbol{x}) + h(\boldsymbol{x}) \\ &= -a(\boldsymbol{x})\hat{u}_1(\boldsymbol{x}) + h(\boldsymbol{x}).\end{aligned}$$

Since $\phi(\boldsymbol{x}) = 0$ on the boundary $\partial\Omega$, $a(\boldsymbol{x})\tilde{u}(\boldsymbol{x})$ takes the form

$$a(\boldsymbol{x})\tilde{u}(\boldsymbol{x}) = a(\boldsymbol{x})\hat{u}_1(\boldsymbol{x}).$$

Therefore,

$$\begin{aligned}\boldsymbol{n}(\boldsymbol{x}) \cdot \nabla\tilde{u}(\boldsymbol{x}) + a(\boldsymbol{x})\tilde{u}(\boldsymbol{x}) &= -a(\boldsymbol{x})\hat{u}_1(\boldsymbol{x}) + h(\boldsymbol{x}) + a(\boldsymbol{x})\hat{u}_1(\boldsymbol{x}) \\ &= h(\boldsymbol{x}),\end{aligned}$$

as required.

■

C.2 Derivation of Eq. (28)

Our derivation follows that described by Rvachev and Sheiko (1995) and Sukumar and Srivastava (2022). The objective is to design trial functions $\tilde{u}_1$ and $\tilde{u}_2$ such that

$$\tilde{u}_1(\boldsymbol{x}) = b(\boldsymbol{x}), \ \boldsymbol{x} \in \partial\Omega_1, \tag{75}$$
$$\boldsymbol{n}(\boldsymbol{x}) \cdot \nabla\tilde{u}_1(\boldsymbol{x}) + a(\boldsymbol{x})\tilde{u}_1(\boldsymbol{x}) = 0, \quad \boldsymbol{x} \in \partial\Omega_2, \tag{76}$$

and

$$\tilde{u}_2(\boldsymbol{x}) = 0, \quad \boldsymbol{x} \in \partial\Omega_1, \tag{77}$$
$$\boldsymbol{n}(\boldsymbol{x}) \cdot \nabla\tilde{u}_2(\boldsymbol{x}) + a(\boldsymbol{x})\tilde{u}_2(\boldsymbol{x}) = h(\boldsymbol{x}), \ \boldsymbol{x} \in \partial\Omega_2. \tag{78}$$

The solution structure $\tilde{u}(\boldsymbol{x}) = \tilde{u}_1(\boldsymbol{x}) + \tilde{u}_2(\boldsymbol{x})$ will then exactly satisfy the mixed BVP from Eq. (4).

In the below we will assume that $\hat{u}_1, \hat{u}_2, b, a$ and h are smooth functions, ϕ_1 is an ADF for $\partial\Omega_1$, ϕ_2 is a normalised ADF for $\partial\Omega_2$ and finally that the joined ADF ϕ is normalised for $\partial\Omega_2$ also.

The function $b(\boldsymbol{x}) + \phi_1(\boldsymbol{x})\hat{u}_1(\boldsymbol{x})$ will satisfy the Dirichlet condition in Eq. (75) but not the Robin condition in Eq. (76) given arbitrary choice of continuous trial function $\hat{u}_1$. To satisfy the Robin condition, we plug this function into Eq. (27) with $h = 0$ and ignoring the $\hat{u}_2$ term whose role will be fulfilled by $\tilde{u}_2$ below. Doing so gives

$$\tilde{u}_1(\boldsymbol{x}) = (1 + \phi(\boldsymbol{x})a(\boldsymbol{x}))\left(b(\boldsymbol{x}) + \phi_1(\boldsymbol{x})\hat{u}_1(\boldsymbol{x})\right) - \phi(\boldsymbol{x})\nabla\phi_2(\boldsymbol{x}) \cdot \nabla\left(b(\boldsymbol{x}) + \phi_1(\boldsymbol{x})\hat{u}_1(\boldsymbol{x})\right).$$

A simple solution structure $\tilde{u}_2$ which satisfies Eqs.(77) and (78) is given by

$$\tilde{u}_2(\boldsymbol{x}) = \phi(\boldsymbol{x})\left[\phi_2(\boldsymbol{x})\hat{u}_2(\boldsymbol{x}) - h(\boldsymbol{x})\right].$$

Expanding $\tilde{u}_1$ using the chain rule and adding the result to $\tilde{u}_2$ yields the solution structure in Eq. (28).

C.3 Proof of Theorem 13

Proof Let both $u \in \mathcal{H}_{bc}$ and $\varepsilon > 0$ be arbitrary. We need to find some $\tilde{u} \in \mathcal{H}_{\tilde{k}}$ such that $\|u - \tilde{u}\|_\infty \leq \varepsilon$. We will assume that the input scale has been chosen such that $\sup_{\boldsymbol{x}\in\bar{\Omega}} \phi(\boldsymbol{x}) = 1$. This is without loss of generality, since otherwise the arguments we present below can be rescaled with respect to the supremum value.

We first introduce the following lemma, which is proved in Appendix C.3.1. Recall that ϕ is an ADF (see Eq. 22) for the boundary $\partial\Omega$.

Lemma 26 *For any $\varepsilon' > 0$, there exists $\delta \in (0, 1)$ such that $\phi(\boldsymbol{x}) \leq \delta$ implies $|u(\boldsymbol{x})| \leq \varepsilon'$.*

Choose $\delta \in (0, 1)$ such that Lemma 26 holds for $\varepsilon' = \frac{\varepsilon}{3} > 0$. We next introduce some notation:

$$\begin{cases} \mathcal{I}_\delta = \{\boldsymbol{x} \in \bar{\Omega} : \phi(\boldsymbol{x}) > \delta\} \\ \partial\mathcal{I}_\delta = \{\boldsymbol{x} \in \bar{\Omega} : \phi(\boldsymbol{x}) = \delta\} \\ \bar{\mathcal{I}}_\delta = \mathcal{I}_\delta \cup \partial\mathcal{I}_\delta \\ \bar{\Omega} \setminus \mathcal{I}_\delta = \{\boldsymbol{x} \in \bar{\Omega} : \phi(\boldsymbol{x}) \leq \delta\} \end{cases} \tag{79}$$

Note that $\delta \in (0,1)$ implies $\mathcal{I}_\delta, \partial\mathcal{I}_\delta$ and $\bar{\Omega} \setminus \mathcal{I}_\delta$ are all non-empty, because $\phi : \bar{\Omega} \to [0,1]$ is continuous.

We define the continuous function $z_u : \Omega \to \mathbb{R}$ as

$$z_u(\boldsymbol{x}) \triangleq \frac{u(\boldsymbol{x})}{\phi(\boldsymbol{x})}. \tag{80}$$

Now, consider the following lemma, which is proved in Appendix C.3.2.

Lemma 27 *There exists $\bar{z}_u \in \mathcal{C}(\bar{\Omega})$ such that*

$$\begin{cases} \bar{z}_u(\boldsymbol{x}) = z_u(\boldsymbol{x}),\ \boldsymbol{x} \in \bar{\mathcal{I}}_\delta \\ \sup\limits_{\boldsymbol{x} \in \bar{\Omega} \setminus \mathcal{I}_\delta} |\bar{z}_u(\boldsymbol{x})| = |z_u(\boldsymbol{p})|\,,\ \boldsymbol{p} \in \partial\mathcal{I}_\delta. \end{cases} \tag{81}$$

Since we have assumed that k is a universal kernel (see Definition 12) on $\bar{\Omega}$, there exists $\hat{z} \in \mathcal{H}_k$ such that

$$\|\bar{z}_u - \hat{z}\|_\infty \leq \frac{\varepsilon}{3}. \tag{82}$$

Consider now the function $\tilde{u} : \bar{\Omega} \to \mathbb{R}$ defined as

$$\tilde{u}(\boldsymbol{x}) \triangleq \phi(\boldsymbol{x})\hat{z}(\boldsymbol{x}). \tag{83}$$

By the following lemma (which is proved in Appendix C.3.3), $\tilde{u} \in \mathcal{H}_{\tilde{k}}$.

Lemma 28 *Let $k : \bar{\Omega} \times \bar{\Omega} \to \mathbb{R}$ be a kernel, and $\tilde{k}$ a boundary-constrained kernel of the form given in Eq. (29) with ϕ an ADF for $\bar{\Omega}$. Then $g \in \mathcal{H}_k$ implies $h \in \mathcal{H}_{\tilde{k}}$, where $h(\boldsymbol{x}) = \phi(\boldsymbol{x})g(\boldsymbol{x})$.*

We claim that $\|u - \tilde{u}\|_\infty \leq \varepsilon$—to prove this, we use the partition $\bar{\Omega} = \partial\Omega \cup \mathcal{I}_\delta \cup \{\Omega \setminus \mathcal{I}_\delta\}$ (see Eq. 79), and consider each subdomain in turn:

Case I: $\boldsymbol{x} \in \partial\Omega$. Clearly $\tilde{u}(\boldsymbol{x}) = 0$ in this case, since $\phi(\boldsymbol{x}) = 0$ for $\boldsymbol{x} \in \partial\Omega$ and $\hat{z}$ is a continuous function. Since $u(\boldsymbol{x}) = 0$ if $\boldsymbol{x} \in \partial\Omega$ by assumption, $\|u - \tilde{u}\|_\infty \leq \varepsilon$ holds.

Case II: $\boldsymbol{x} \in \mathcal{I}_\delta$. By Eq. (82) we have that

$$|\bar{z}_u(\boldsymbol{x}) - \hat{z}(\boldsymbol{x})| \leq \frac{\varepsilon}{3}.$$

Recall that $\bar{z}_u(\boldsymbol{x}) = z_u(\boldsymbol{x})$ if $\boldsymbol{x} \in \mathcal{I}_\delta$ (see Eq. 81). Therefore

$$|z_u(\boldsymbol{x}) - \hat{z}(\boldsymbol{x})| \leq \frac{\varepsilon}{3}.$$

Since $\phi(\boldsymbol{x}) \in (0,1]$ if $\boldsymbol{x} \in \mathcal{I}_\delta$, we additionally have

$$\begin{aligned} \phi(\boldsymbol{x})\,|z_u(\boldsymbol{x}) - \hat{z}(\boldsymbol{x})| &\leq \frac{\varepsilon}{3} \\ \Rightarrow\ |\phi(\boldsymbol{x})z_u(\boldsymbol{x}) - \phi(\boldsymbol{x})\hat{z}(\boldsymbol{x})| &\leq \frac{\varepsilon}{3}. \end{aligned}$$

Finally, note that $u(\boldsymbol{x}) = \phi(\boldsymbol{x})z_u(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathcal{I}_\delta$ by construction of z_u (see Eq. 80) and $\tilde{u}(\boldsymbol{x}) = \phi(\boldsymbol{x})\hat{z}(\boldsymbol{x})$ by definition (see Eq. 83). Therefore

$$|u(\boldsymbol{x}) - \tilde{u}(\boldsymbol{x})| \leq \frac{\varepsilon}{3}.$$

Case III: $\boldsymbol{x} \in \Omega \setminus \mathcal{I}_\delta$. By Eq. (82), we again know that

$$|\bar{z}_u(\boldsymbol{x}) - \hat{z}(\boldsymbol{x})| \leq \frac{\varepsilon}{3}.$$

This inequality can be expressed equivalently as

$$\begin{aligned} -\frac{\varepsilon}{3} \leq \bar{z}_u(\boldsymbol{x}) - \hat{z}(\boldsymbol{x}) \leq \frac{\varepsilon}{3} \\ \Rightarrow\ -\bar{z}_u(\boldsymbol{x}) - \frac{\varepsilon}{3} \leq -\hat{z}(\boldsymbol{x}) \leq -\bar{z}_u(\boldsymbol{x}) + \frac{\varepsilon}{3}. \end{aligned}$$

Since ϕ is strictly positive in Ω, the above inequalities hold when multiplied by $\phi(\boldsymbol{x})$:

$$\begin{aligned} -\phi(\boldsymbol{x})\bar{z}_u(\boldsymbol{x}) - \phi(\boldsymbol{x})\frac{\varepsilon}{3} \leq -\phi(\boldsymbol{x})\hat{z}(\boldsymbol{x}) \leq -\phi(\boldsymbol{x})\bar{z}_u(\boldsymbol{x}) + \phi(\boldsymbol{x})\frac{\varepsilon}{3} \\ \Rightarrow\ -\phi(\boldsymbol{x})\bar{z}_u(\boldsymbol{x}) - \frac{\varepsilon}{3} \leq -\tilde{u}(\boldsymbol{x}) \leq -\phi(\boldsymbol{x})\bar{z}_u(\boldsymbol{x}) + \frac{\varepsilon}{3}, \end{aligned}$$

where in the last line we make use of the fact that $\phi(\boldsymbol{x}) \leq 1$ and $\tilde{u}$ has the form given in Eq. (83). Since inequalities are not affected by the addition of constants, the above implies

$$u(\boldsymbol{x}) - \phi(\boldsymbol{x})\bar{z}_u(\boldsymbol{x}) - \frac{\varepsilon}{3} \leq u(\boldsymbol{x}) - \hat{u}(\boldsymbol{x}) \leq u(\boldsymbol{x}) - \phi(\boldsymbol{x})\bar{z}_u(\boldsymbol{x}) + \frac{\varepsilon}{3}.$$

Furthermore, since δ has been chosen such that $\boldsymbol{x} \in \Omega \setminus \mathcal{I}_\delta$ implies $|u(\boldsymbol{x})| \leq \frac{\varepsilon}{3}$, we have:

$$\begin{aligned} -\frac{\varepsilon}{3} - \phi(\boldsymbol{x})\bar{z}_u(\boldsymbol{x}) - \frac{\varepsilon}{3} \leq u(\boldsymbol{x}) - \hat{u}(\boldsymbol{x}) \leq \frac{\varepsilon}{3} - \phi(\boldsymbol{x})\bar{z}_u(\boldsymbol{x}) + \frac{\varepsilon}{3} \\ \Rightarrow\ -\frac{2\varepsilon}{3} - \phi(\boldsymbol{x})\bar{z}_u(\boldsymbol{x}) \leq u(\boldsymbol{x}) - \hat{u}(\boldsymbol{x}) \leq \frac{2\varepsilon}{3} - \phi(\boldsymbol{x})\bar{z}_u(\boldsymbol{x}). \end{aligned} \tag{84}$$

We now proceed to show that $|\phi(\boldsymbol{x})\bar{z}_u(\boldsymbol{x})| \leq \frac{\varepsilon}{3}$:

$$\begin{aligned} |\phi(\boldsymbol{x})\bar{z}_u(\boldsymbol{x})| &= \phi(\boldsymbol{x})\,|\bar{z}_u(\boldsymbol{x})| \ \text{(since } \phi(\boldsymbol{x}) > 0 \text{ in } \Omega) \\ &\leq \phi(\boldsymbol{x}) \sup_{\boldsymbol{x}' \in \bar{\Omega} \setminus \mathcal{I}_\delta} \left|\bar{z}_u(\boldsymbol{x}')\right| \\ &= \phi(\boldsymbol{x})\,|z_u(\boldsymbol{p})|\,,\ \boldsymbol{p} \in \partial\mathcal{I}_\delta \text{ (by Eq. 81).} \end{aligned}$$

Finally, since $\phi(\boldsymbol{p}) \geq \phi(\boldsymbol{x}) > 0$ by Eq. (79), we have

$$\begin{aligned} |\phi(\boldsymbol{x})\bar{z}_u(\boldsymbol{x})| &\leq \phi(\boldsymbol{p})\,|z_u(\boldsymbol{p})| \\ &= |\phi(\boldsymbol{p})z_u(\boldsymbol{p})| \\ &= |u(\boldsymbol{p})| \ \text{(by Eq. 80)} \\ &\leq \frac{\varepsilon}{3} \ \text{(by choice of } \delta). \end{aligned}$$

The above implies that Eq. (84) can be simplified to yield

$$\begin{aligned} -\frac{2\varepsilon}{3} - \frac{\varepsilon}{3} \leq\ & u(\boldsymbol{x}) - \tilde{u}(\boldsymbol{x}) \leq \frac{2\varepsilon}{3} - \frac{\varepsilon}{3} \\ \Rightarrow\ -\varepsilon \leq\ & u(\boldsymbol{x}) - \tilde{u}(\boldsymbol{x}) \leq \varepsilon \\ \Rightarrow\ & |u(\boldsymbol{x}) - \tilde{u}(\boldsymbol{x})| \leq \varepsilon. \end{aligned}$$

Putting all three cases together, we have $\|u - \tilde{u}\|_\infty \leq \varepsilon$ as required.

■

C.3.1 Proof of Lemma 26

Proof We first introduce two sub-lemmas, where in each case $d(\boldsymbol{x})$ is the exact distance function from Eq. (21) and $S \triangleq \sup_{\boldsymbol{x}\in\bar{\Omega}} d(\boldsymbol{x})$. Lemma 29 is proved in Appendix C.3.4, and Lemma 30 is proved in Appendix C.3.5.

Lemma 29 *For any $\varepsilon' > 0$, there exists $\delta' \in (0, S)$ such that $d(\boldsymbol{x}) \leq \delta'$ implies $|u(\boldsymbol{x})| \leq \varepsilon'$.*

Lemma 30 *For any $\delta' \in (0, S)$, there exists $\delta \in (0, 1)$ such that $\phi(\boldsymbol{x}) \leq \delta$ implies $d(\boldsymbol{x}) \leq \delta'$.*

The desired result then holds by setting δ' in Lemma 29 equal to δ' in Lemma 30.

■

C.3.2 Proof of Lemma 27

Proof We begin by remarking that the sets $\partial\mathcal{I}_\delta$, $\bar{\mathcal{I}}_\delta$ and $\bar{\Omega} \setminus \mathcal{I}_\delta$ defined in Eq. (79) are all closed sets under the Euclidean topology because $\phi \in \mathcal{C}(\bar{\Omega})$. Furthermore, they are all compact, since they are subsets of the bounded set $\bar{\Omega}$. Also, note that $\bar{\Omega} \setminus \mathcal{I}_\delta \subset \mathbb{R}^D$ is normal (and therefore can be used in place of the normal space A in Theorem 23) because all metrizable spaces are normal.

Consider the function $z_u : \partial\mathcal{I}_\delta \to \mathbb{R}$. Since $\partial\mathcal{I}_\delta$ is compact and $\bar{\Omega} \setminus \mathcal{I}_\delta$ is normal, by Theorem 23 there exists a function $z'_u \in \mathcal{C}(\bar{\Omega} \setminus \mathcal{I}_\delta)$ such that

$$\begin{cases} z'_u(\boldsymbol{x}) = z_u(\boldsymbol{x}),\ \boldsymbol{x} \in \partial\mathcal{I}_\delta \\ \sup\limits_{\boldsymbol{x}\in\bar{\Omega}\setminus\mathcal{I}_\delta} \left|z'_u(\boldsymbol{x})\right| = \sup\limits_{\boldsymbol{x}\in\partial\mathcal{I}_\delta} \left|z_u(\boldsymbol{x})\right|. \end{cases} \tag{85}$$

Now, define the function $\bar{z}_u : \bar{\Omega} \to \mathbb{R}$ by

$$\bar{z}_u(\boldsymbol{x}) = \begin{cases} z'_u(\boldsymbol{x}),\ \boldsymbol{x} \in \bar{\Omega} \setminus \mathcal{I}_\delta \\ z_u(\boldsymbol{x}),\ \boldsymbol{x} \in \mathcal{I}_\delta. \end{cases} \tag{86}$$

To show that $\bar{z}_u \in \mathcal{C}(\bar{\Omega})$, we use Theorem 24. Consider the closed sets $\bar{\Omega} \setminus \mathcal{I}_\delta$ and $\bar{\mathcal{I}}_\delta$. Clearly, $\bar{\Omega} = \{\bar{\Omega} \setminus \mathcal{I}_\delta\} \cup \bar{\mathcal{I}}_\delta$ (see Eq. 79). Furthermore, the restriction of $\bar{z}_u$ to $\bar{\Omega} \setminus \mathcal{I}_\delta$ (i.e. the function $\bar{z}_u : \bar{\Omega} \setminus \mathcal{I}_\delta \to \mathbb{R}$) is continuous because in this case, it equals $z'_u \in \mathcal{C}(\bar{\Omega} \setminus \mathcal{I}_\delta)$ by

construction in Eq. (86). Similarly, the function $\bar{z}_u : \bar{\mathcal{I}}_\delta \to \mathbb{R}$, is continuous because in this case, it equals $z_u \in \mathcal{C}(\bar{\mathcal{I}}_\delta)$ by Eqs.(85) and (86). Therefore, by Theorem 24, $\bar{z}_u \in \mathcal{C}(\bar{\Omega})$.

Additionally, we have that

$$\sup_{\boldsymbol{x}\in\bar{\Omega}\setminus\mathcal{I}_\delta} |\bar{z}_u(\boldsymbol{x})| = \sup_{\boldsymbol{x}\in\partial\mathcal{I}_\delta} |z_u(\boldsymbol{x})| .$$

Finally, since z_u is continuous and $\partial\mathcal{I}_\delta$ compact, by the extreme value theorem there exists $\boldsymbol{p} \in \partial\mathcal{I}_\delta$ such that

$$\begin{aligned} z_u(\boldsymbol{p}) &= \sup_{\boldsymbol{x}\in\partial\mathcal{I}_\delta} |z_u(\boldsymbol{x})| \\ &= \sup_{\boldsymbol{x}\in\bar{\Omega}\setminus\mathcal{I}_\delta} |\bar{z}_u(\boldsymbol{x})| . \end{aligned}$$

■

C.3.3 Proof of Lemma 28

Proof Let $g \in \mathcal{H}_k$ be arbitrary. By Theorem 22, there exists a feature map $\Phi_0 : \bar{\Omega} \to \mathcal{H}_0$ with $\mathcal{H}_0$ an $\mathbb{R}$-Hilbert space so that, for all $\boldsymbol{x} \in \bar{\Omega}$, we have

$$g(\boldsymbol{x}) = \langle w, \Phi_0(\boldsymbol{x})\rangle_{\mathcal{H}_0}$$

for some $w \in \mathcal{H}_0$. Therefore, for all $\boldsymbol{x} \in \bar{\Omega}$, we have

$$\begin{aligned} h(\boldsymbol{x}) &= \phi(\boldsymbol{x})g(\boldsymbol{x}) \\ &= \phi(\boldsymbol{x}) \langle w, \Phi_0(\boldsymbol{x})\rangle_{\mathcal{H}_0} \\ &= \langle w, \phi(\boldsymbol{x})\Phi_0(\boldsymbol{x})\rangle_{\mathcal{H}_0} \end{aligned}$$

Now consider the forward map $\tilde{\Phi}_0 : \bar{\Omega} \to \mathcal{H}_0$ defined as $\tilde{\Phi}_0(\boldsymbol{x}) \triangleq \phi(\boldsymbol{x})\Phi_0(\boldsymbol{x})$. Note that $\tilde{\Phi}_0$ maps into $\mathcal{H}_0$ because $\phi(\boldsymbol{x}) \in \mathbb{R}$ for all $\boldsymbol{x} \in \bar{\Omega}$ and $\mathcal{H}_0$ is an $\mathbb{R}$-Hilbert space.

Let $\boldsymbol{x}, \boldsymbol{x}' \in \bar{\Omega}$ be arbitrary. Then

$$\begin{aligned} \left\langle \tilde{\Phi}_0(\boldsymbol{x}), \tilde{\Phi}_0(\boldsymbol{x}')\right\rangle_{\mathcal{H}_0} &= \left\langle \phi(\boldsymbol{x})\Phi_0(\boldsymbol{x}), \phi(\boldsymbol{x}')\Phi_0(\boldsymbol{x}')\right\rangle_{\mathcal{H}_0} \\ &= \phi(\boldsymbol{x})\phi(\boldsymbol{x}') \left\langle \Phi_0(\boldsymbol{x}), \Phi_0(\boldsymbol{x}')\right\rangle_{\mathcal{H}_0} \\ &= \phi(\boldsymbol{x})\phi(\boldsymbol{x}')k(\boldsymbol{x}, \boldsymbol{x}') \text{ (by Eq. 71)} \\ &= \tilde{k}(\boldsymbol{x}, \boldsymbol{x}'). \end{aligned}$$

Therefore, $\tilde{\Phi}_0$ is a feature map for $\tilde{k}$ (see Definition 21). This means we have

$$h(\boldsymbol{x}) = \left\langle w, \tilde{\Phi}_0(\boldsymbol{x})\right\rangle_{\mathcal{H}_0} ,$$

where $\tilde{\Phi}_0 : \bar{\Omega} \to \mathcal{H}_0$ with $\mathcal{H}_0$ an $\mathbb{R}$-Hilbert space and $w \in \mathcal{H}_0$. This implies by Theorem 22 that $h \in \mathcal{H}_{\tilde{\Phi}_0} = \mathcal{H}_{\tilde{k}}$. Since g was chosen arbitrarily, this holds for all $g \in \mathcal{H}_k$. ■

C.3.4 Proof of Lemma 29

Proof Let $\epsilon' > 0$ be arbitrary. Since $\bar{\Omega}$ is compact and $u : \bar{\Omega} \to \mathbb{R}$ is continuous, u is also uniformly continuous. Coupled with the fact that u is zero on $\partial\Omega$, this implies the existence of $\delta' \in (0, S)$ such that, for all $\boldsymbol{x}_b \in \partial\Omega$, $\|\boldsymbol{x} - \boldsymbol{x}_b\|_2 \leq \delta'$ implies $|u(\boldsymbol{x})| \leq \epsilon'$.

Suppose now that $d(\boldsymbol{x}) \leq \delta'$. Because d is continuous and $\partial\Omega$ is compact, by the extreme value theorem there exists $\boldsymbol{x}_b \in \partial\Omega$ such that

$$\begin{aligned} d(\boldsymbol{x}) &= \inf_{\boldsymbol{x}' \in \partial\Omega} \|\boldsymbol{x} - \boldsymbol{x}'\|_2 \\ &= \|\boldsymbol{x} - \boldsymbol{x}_b\|_2 \\ &\leq \delta'. \end{aligned}$$

Therefore, by the construction of δ' above, $|u(\boldsymbol{x})| \leq \epsilon'$.

■

C.3.5 Proof of Lemma 30

Proof Let $\delta' \in (0, S)$ be arbitrary. Consider the superlevel set of d with respect to δ',

$$L^+_{\delta'}(d) = \{\boldsymbol{x} \in \Omega : d(\boldsymbol{x}) \geq \delta'\}.$$

Note that $L^+_{\delta'}(d)$ is non-empty because $\delta' < S = \sup_{\boldsymbol{x} \in \bar{\Omega}} d(\boldsymbol{x})$. Furthermore, the continuity of d implies that $L^+_{\delta'}(d)$ is closed and therefore compact as Ω is bounded. Now consider the infimum of ϕ over this set,

$$\inf_{\boldsymbol{x}' \in L^+_{\delta'}(d)} \phi(\boldsymbol{x}').$$

Since the infimum is being taken over a compact set and ϕ is continuous, by the extreme value theorem there exists $\boldsymbol{x} \in L^+_{\delta'}(d)$ such that

$$\inf_{\boldsymbol{x}' \in L^+_{\delta'}(d)} \phi(\boldsymbol{x}') = \phi(\boldsymbol{x}) = \Delta > 0.$$

Note that $\Delta > 0$ because $\phi(\boldsymbol{x}) = 0 \iff \boldsymbol{x} \in \partial\Omega \iff d(\boldsymbol{x}) = 0$. Set $\delta = \Delta/2 \in (0, 1)$. Suppose that $\boldsymbol{x} \in \bar{\Omega}$ is such that $\phi(\boldsymbol{x}) \leq \delta$. By construction of δ, we know that $\boldsymbol{x} \notin L^+_{\delta'}(d)$. Therefore $d(\boldsymbol{x}) < \delta' \implies d(\boldsymbol{x}) \leq \delta'$, as required.

■

C.4 Proof of Theorem 16

Proof We do not directly make use of Theorem 15 in the proof, and instead, for completeness, we show all required steps fully, which mirror those from the proof of Theorem 15 given in (Murphy, 2023, Section 18.7.1). We denote the trainable parameters of the neural network $\hat{u}_{nn}$ (Eq. 40) used in the definition of the BCNN $\tilde{u}_{nn}(\boldsymbol{x})$ (Eq. 39) as

$$\boldsymbol{\omega} = \{b^{(1)}, w_1^{(1)}, \ldots, w_H^{(1)}, b_1^{(0)}, \ldots, b_H^{(0)}, \boldsymbol{w}_1^{(0)}, \ldots, \boldsymbol{w}_H^{(0)}\}.$$

Let $\boldsymbol{x} \in \bar{\Omega}$ be arbitrary. Then, the expected value of $\tilde{u}_{nn}(\boldsymbol{x})$ under the prior distributions from Eq. (41) is given by:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\omega}}\left[\tilde{u}_{nn}(\boldsymbol{x})\right] &= \mathbb{E}_{\boldsymbol{\omega}}\left[\tilde{m}(\boldsymbol{x}) + \phi(\boldsymbol{x})\hat{u}_{nn}(\boldsymbol{x})\right] \\
&= \mathbb{E}_{\boldsymbol{\omega}}\left[\tilde{m}(\boldsymbol{x}) + \phi(\boldsymbol{x})\left(b^{(1)} + \sum_{j=1}^{H} w_j^{(1)} h_j(\boldsymbol{x})\right)\right] \\
&= \mathbb{E}_{\boldsymbol{\omega}}\left[\tilde{m}(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{\omega}}\left[\phi(\boldsymbol{x})\left(b^{(1)} + \sum_{j=1}^{H} w_j^{(1)} h_j(\boldsymbol{x})\right)\right] \\
&= \tilde{m}(\boldsymbol{x}) + \phi(\boldsymbol{x})\mathbb{E}_{\boldsymbol{\omega}}\left[b^{(1)} + \sum_{j=1}^{H} w_j^{(1)} h_j(\boldsymbol{x})\right] \\
&= \tilde{m}(\boldsymbol{x}) + \phi(\boldsymbol{x})\left(\mathbb{E}_{\boldsymbol{\omega}}\left[b^{(1)}\right] + \sum_{j=1}^{H} \mathbb{E}_{\boldsymbol{\omega}}\left[w_j^{(1)} h_j(\boldsymbol{x})\right]\right) \\
&= \tilde{m}(\boldsymbol{x}) + \phi(\boldsymbol{x})\left(\underbrace{\mathbb{E}_{\boldsymbol{\omega}}\left[b^{(1)}\right]}_{=0} + \sum_{j=1}^{H} \underbrace{\mathbb{E}_{\boldsymbol{\omega}}\left[w_j^{(1)}\right]}_{=0} \mathbb{E}_{\boldsymbol{\omega}}\left[h_j(\boldsymbol{x})\right]\right) \\
&= \tilde{m}(\boldsymbol{x}).
\end{aligned}$$

We remark that $\mathbb{E}_{\boldsymbol{\omega}}\left[w_j^{(1)} h_j(\boldsymbol{x})\right] = \mathbb{E}_{\boldsymbol{\omega}}\left[w_j^{(1)}\right]\mathbb{E}_{\boldsymbol{\omega}}\left[h_j(\boldsymbol{x})\right]$ holds because $w_j^{(1)}$ is independent of $h_j(\boldsymbol{x})$ for all $j = 1, \ldots, H$ under our prior assumption for $\boldsymbol{\omega}$ (see Eq. 41).

Now let $\boldsymbol{x}, \boldsymbol{x}' \in \Omega$ be arbitrary. Then by Eq. (39) and Eq. (41), the covariance between the corresponding outputs $\tilde{u}_{nn}(\boldsymbol{x})$ and $\tilde{u}_{nn}(\boldsymbol{x}')$ is given by

$$\begin{aligned}
\mathrm{Cov}\left(\tilde{u}_{nn}(\boldsymbol{x})\tilde{u}_{nn}(\boldsymbol{x}')\right) &= \mathbb{E}_{\boldsymbol{\omega}}\left[\left(\tilde{u}_{nn}(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{\omega}}\left[\tilde{u}_{nn}(\boldsymbol{x})\right]\right)\left(\tilde{u}_{nn}(\boldsymbol{x}') - \mathbb{E}_{\boldsymbol{\omega}}\left[\tilde{u}_{nn}(\boldsymbol{x}')\right]\right)\right] \\
&= \mathbb{E}_{\boldsymbol{\omega}}\left[\phi(\boldsymbol{x})\hat{u}_{nn}(\boldsymbol{x})\phi(\boldsymbol{x}')\hat{u}_{nn}(\boldsymbol{x}')\right] \\
&= \phi(\boldsymbol{x})\phi(\boldsymbol{x}')\mathbb{E}_{\boldsymbol{\omega}}\left[\hat{u}_{nn}(\boldsymbol{x})\hat{u}_{nn}(\boldsymbol{x}')\right].
\end{aligned}$$

We furthermore have that

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\omega}}\left[\hat{u}_{nn}(\boldsymbol{x})\hat{u}_{nn}(\boldsymbol{x}')\right] &= \mathbb{E}_{\boldsymbol{\omega}}\left[\left(b^{(1)} + \sum_{j=1}^{H} w_j^{(1)} h_j(\boldsymbol{x})\right)\left(b^{(1)} + \sum_{j=1}^{H} w_j^{(1)} h_j(\boldsymbol{x}')\right)\right] \\
&= \sigma_{b^{(1)}}^2 + \sum_{j=1}^{H} \frac{\omega}{H}\mathbb{E}_{\boldsymbol{\omega}}\left[h_j(\boldsymbol{x})h_j\left(\boldsymbol{x}'\right)\right],
\end{aligned}$$

where the final expression holds because $\mathbb{E}_{\boldsymbol{\omega}}\left[b^{(1)}\right] = 0$ and $\mathrm{Cov}\left(w_i^{(1)}, w_j^{(1)}\right) = 0$ for $i \neq j$. Note that all $b_j^{(0)}\left(\boldsymbol{w}_j^{(0)}\right)$ share a common prior. Therefore, $\mathbb{E}_{\boldsymbol{\omega}}\left[h_j(\boldsymbol{x})h_j\left(\boldsymbol{x}'\right)\right]$ is the same for all $j = 1, \ldots, H$. This means we have

$$\mathbb{E}_{\boldsymbol{\omega}}\left[\hat{u}_{nn}(\boldsymbol{x})\hat{u}_{nn}(\boldsymbol{x}')\right] = \sigma_{b^{(1)}}^2 + \omega\mathbb{E}_{\boldsymbol{\omega}}\left[h_1(\boldsymbol{x})h_1\left(\boldsymbol{x}'\right)\right] \triangleq k_{nn}\left(\boldsymbol{x}, \boldsymbol{x}'\right).$$

We remark that k_{nn} is the form of the kernel for the infinite width limit of the unconstrained neural network $\hat{u}_{nn}$ given in Eq. (42) (Murphy, 2023, Section 18.7.1). We therefore have

$$\begin{aligned}\mathrm{Cov}\big(\tilde{u}_{nn}(\boldsymbol{x}), \tilde{u}_{nn}(\boldsymbol{x}')\big) &= \phi(\boldsymbol{x})\phi(\boldsymbol{x}')k_{nn}(\boldsymbol{x}, \boldsymbol{x}') \\ &= \tilde{k}_{nn}(\boldsymbol{x}, \boldsymbol{x}'),\end{aligned}$$

i.e. the covariance between any two function outputs is given by the boundary constrained kernel $\tilde{k}_{nn}$ generated by using k_{nn} in the construction of a Dirichlet BCGP (see Eq. 29).

Now, consider the limit as $H \to \infty$. Note that the contribution of the hidden units (the h_j's) is an infinite sum of random variables with shared mean and variance, respectively. Furthermore, the variance is bounded because we are assuming φ is bounded. Therefore, we conclude by the Central Limit Theorem that the contribution of the hidden units converges in distribution to a Gaussian. Therefore, the output of the BCNN itself at any point $\boldsymbol{x} \in \bar{\Omega}$ converges to a Gaussian, with mean $\tilde{m}(\boldsymbol{x})$ and covariance $\tilde{k}_{nn}(\boldsymbol{x}, \boldsymbol{x})$. Furthermore, given any set of inputs $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)} \in \bar{\Omega}$ with $N \geq 2$ the joint distribution of the corresponding outputs $(\tilde{u}_{nn}(\boldsymbol{x}^{(1)}), \tilde{u}_{nn}(\boldsymbol{x}^{(2)}), ..., \tilde{u}_{nn}(\boldsymbol{x}^{(N)}))^\top$ converges to a multivariate Gaussian, where the cross covariance terms are found as $\mathrm{Cov}\left(\tilde{u}_{nn}(\boldsymbol{x}^{(i)}), \tilde{u}_{nn}(\boldsymbol{x}^{(j)})\right) = \tilde{k}_{nn}\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right)$

Therefore, in the limit, the function $\tilde{u}_{nn}$ exactly satisfies the definition of a Gaussian process (Definition 4) over $\bar{\Omega}$ with mean function $\tilde{m}$ and covariance function $\tilde{k}_{nn}$ so we conclude

$$\tilde{u}_{nn}(\boldsymbol{x}) \to \mathcal{GP}(\tilde{m}(\boldsymbol{x}), \tilde{k}_{nn}(\boldsymbol{x}, \boldsymbol{x}')).$$

■

C.5 Proof of Proposition 25

Proof

We prove that for all $i = 1, \ldots, D$, $\tilde{m}_i(\boldsymbol{x})$ exactly satisfies boundary conditions on $\bigcup_{l=0}^{i} \bigcup_{j=0}^{1} \partial\Omega_{lj}$. Since $\tilde{m}(\boldsymbol{x}) \triangleq \tilde{m}_d(\boldsymbol{x})$, this is sufficient to prove the claim.

We proceed by induction. Firstly, note by construction of $\tilde{m}_1$ that $\tilde{m}_1(\boldsymbol{x} : x_1 = 0) = b(\boldsymbol{x} : x_1 = 0)$ and $\tilde{m}_1(\boldsymbol{x} : x_1 = 1) = b(\boldsymbol{x} : x_1 = 1)$, i.e $\tilde{m}_1(\boldsymbol{x})$ satisfies the specified boundary conditions on $\partial\Omega_{10}$ and $\partial\Omega_{11}$.

Next, assume that $\tilde{m}_{i-1}(\boldsymbol{x})$ exactly satisfies boundary conditions on $\bigcup_{l=0}^{i-1} \bigcup_{j=0}^{1} \partial\Omega_{lj}$ for $i = 2, \ldots, D$. Consider the output of $\tilde{m}_i(\boldsymbol{x})$ with $x_i = j$ for $j \in \{0, 1\}$, that is, $\tilde{m}_i(\boldsymbol{x} : x_i = j)$. We have

$$\begin{aligned}\tilde{m}_i(\boldsymbol{x} : x_i = j) &= \tilde{m}_{i-1}(\boldsymbol{x} : x_i = j) + \tilde{b}_{i,0}(\boldsymbol{x} : x_i = j)(1 - j) + \tilde{b}_{i,1}(\boldsymbol{x} : x_i = j)j \\ &= \tilde{m}_{i-1}(\boldsymbol{x} : x_i = j) + \tilde{b}_{i,j}(\boldsymbol{x} : x_i = j) \\ &= \tilde{m}_{i-1}(\boldsymbol{x} : x_i = j) + b(\boldsymbol{x} : x_i = j) - \tilde{m}_{i-1}(\boldsymbol{x} : x_i = j) \\ &= b(\boldsymbol{x} : x_i = j).\end{aligned}$$

That is, $\tilde{m}_i$ satisfies the boundary conditions on $\partial\Omega_{i0}$ and $\partial\Omega_{i1}$.

Now, we show that $\tilde{m}_i$ satisfies the boundary conditions on $\partial\Omega_{ij}$ for $l < i$. Since $\tilde{m}_{i-1}$ satisfies these conditions by assumption, by construction of $\tilde{m}_i$ we simply need to prove

that for all $l < i$ we have $\tilde{b}_{i,j}(\boldsymbol{x} : x_l = q) = 0$ for $j, q \in \{0, 1\}$. To see this, let $l < i$ be arbitrary. We then have

$$\begin{aligned}\tilde{b}_{i,j}(\boldsymbol{x} : x_l = q) &= b(\boldsymbol{x} : x_i = j, x_l = q) - \tilde{m}_{i-1}(\boldsymbol{x} : x_i = j, x_l = q) \\ &= 0.\end{aligned}$$

The above equality holds because $\tilde{m}_{i-1}$ equals b where $x_l = q \in \{0, 1\}$ by assumption. Therefore $\tilde{m}_i(\boldsymbol{x}) = \tilde{m}_{i-1}(\boldsymbol{x})$ where $\boldsymbol{x} \in \partial\Omega_{l0} \cup \partial\Omega_{l1}$, Since l was arbitrary, this holds for all $l < i$, i.e. boundary conditions are satisfied for all $\bigcup_{l=0}^{i-1}\bigcup_{j=0}^{1} \partial\Omega_{lj}$. Combined with the above result for $\partial\Omega_{i0} \cup \partial\Omega_{i1}$, we conclude by induction that $\tilde{m}$ satisfies boundary conditions on all of $\partial\Omega$.

■

References

Robert J Adler. *The Geometry of Random Felds*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2010.

Martin Alnæs, Jan Blechta, Johan Hake, August Johansson, Benjamin Kehlet, Anders Logg, Chris Richardson, Johannes Ring, Marie E Rognes, and Garth N Wells. The FEniCS project version 1.5. *Archive of numerical software*, 3(100), 2015.

Mauricio A Alvarez, David Luengo, and Neil D Lawrence. Linear latent force models using Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

Sokratis J Anagnostopoulos, Juan Diego Toscano, Nikolaos Stergiopulos, and George Em Karniadakis. Residual-based attention in physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 421:116805, 2024.

Nachman Aronszajn. Theory of Reproducing Kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950. Publisher: American Mathematical Society.

A.V. Artiukh, M.V. Sidorov, and S.M. Lamtyugova. R-functions and nonlinear galerkin method for solving the nonlinear stationary problem of flow around body of revolution. *Nonlinear Dynamics and Systems Theory*, 21:138–149, 2021.

Mikhail Basarab, Alain Giani, and Philippe Combette. Thermal accelerometer simulation by the R-functions method. *Applied Sciences*, 10(23):8373, 2020.

Henri Berestycki, Louis Nirenberg, and SR Srinivasa Varadhan. The principal eigenvalue and maximum principle for second-order elliptic operators in general domains. *Communications on Pure and Applied Mathematics*, 47(1):47–92, 1994.

Jens Berg and Kaj Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28–41, 2018.

Arpan Biswas and Vadim Shapiro. Approximate distance fields with non-vanishing gradients. *Graphical Models*, 66(3):133–159, 2004.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

Solveig Bruvoll and Michael S Floater. Transfinite mean value interpolation in general dimension. *Journal of Computational and Applied Mathematics*, 233(7):1631–1639, 2010.

Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (PINNs) for fluid mechanics: A review, 2021.

Jialei Chen, Zhehui Chen, Chuck Zhang, and C. F. Jeff Wu. Apik: Active physics-informed kriging model with partial differential equations. *SIAM/ASA Journal on Uncertainty Quantification*, 10(1):481–506, 2022.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

Sébastien Da Veiga and Amandine Marrel. Gaussian process modeling with inequality constraints. In *Annales de la Faculté des sciences de Toulouse: Mathématiques*, volume 21, pages 529–555, 2012.

David Dalton, Dirk Husmeier, and Hao Gao. Physics-informed graph neural network emulation of soft-tissue mechanics. *Computer Methods in Applied Mechanics and Engineering*, 417:116351, 2023.

David Dalton, Dirk Husmeier, and Hao Gao. Physics and Lie symmetry informed Gaussian processes. In *International Conference on Machine Learning (ICML)*, 2024.

Liang Ding, Simon Mak, and C. F. Jeff Wu. BdryGP: a new Gaussian process model for incorporating boundary information, 2019.

Frank Dondelinger, Maurizio Filippone, Simon Rogers, and Dirk Husmeier. ODE parameter inference using adaptive gradient matching with Gaussian processes. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2013.

James Dugundji. *Topology*. Prentice Hall, 1966.

Christopher Dyken and Michael S Floater. Transfinite mean value interpolation. *Computer Aided Geometric Design*, 26(1):117–134, 2009.

Joaquim Cezar Felipe, Caetano Traina, and Agma Juci Machado Traina. A new family of distance functions for perceptual similarity retrieval of medical images. *Journal of Digital Imaging*, 22:183–201, 2009.

Michael S Floater and Francesco Patrizi. Transfinite mean value interpolation over polygons. *Numerical Algorithms*, 85(3):995–1003, 2020.

Michael Freytag, Vadim Shapiro, and Igor Tsukanov. Field modeling with sampled distances. *Computer-Aided Design*, 38(2):87–100, 2006.

Sarah F Frisken and Ronald N Perry. Designing with distance fields. *ACM SIGGRAPH 2006 Courses*, pages 60–66, 2006.

Elmer Gilbert and Daniel Johnson. Distance functions and their application to robot path planning in the presence of obstacles. *IEEE Journal on Robotics and Automation*, 1(1): 21–30, 1985.

Thore Graepel. Solving noisy linear operator equations by Gaussian processes: Application to ordinary and partial differential equations. In *International Conference on Machine Learning (ICML)*, 2003.

Mamikon Gulian, Ari L. Frankel, and Laura P. Swiler. Gaussian process regression constrained by boundary value problems. *Computer Methods in Applied Mechanics and Engineering*, 388:114117, 2022.

Marc Harkonen, Markus Lange-Hegermann, and Bogdan Raita. Gaussian process priors for systems of linear partial differential equations with constant coefficients. In *International Conference on Machine Learning (ICML)*, 2023.

Bjørn Sand Jensen, Jens Brehm Nielsen, and Jan Larsen. Bounded Gaussian process regression. In *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2013.

Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K. Sriperumbudur. Gaussian processes and kernel methods: A review on connections and equivalences, 2018. arXiv:1807.02582 [cs, stat].

Leonid Vitalevich Kantorovich, Vladimir Ivanovich Krylov, Curtis D Benster, and George Weiss. Approximate methods of higher analysis. *Physics Today*, 13(1):74–76, 1960.

George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021. Number: 6 Publisher: Nature Publishing Group.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

RA Kolyada, KV Maksimenko-Sheiko, and TI Sheiko. R-functions method in the mathematical modeling of convective heat exchange in an octahedral fuel assembly with 37 fuel elements. *Journal of Mathematical Sciences*, 238:154–164, 2019.

Tomislav Kosta and Igor Tsukanov. Meshfree modeling of dynamic response of mechanical structures. *Meccanica*, 49:2399–2418, 2014.

Lidiya Kurpa, Olga Mazur, and Igor Tsukanov. Application of R-functions theory to study parametric vibrations and dynamical stability of laminated plates. In *International Conference on Nonlinear Dynamics*, 2013.

Markus Lange-Hegermann. Linearly constrained Gaussian processes with boundary conditions. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1090–1098, 2021. ISSN: 2640-3498.

Raphael Leiteritz and Dirk Pflüger. How to avoid trivial solutions in physics-informed neural networks, 2021. arXiv:2112.05620 [cs, stat].

Zhaohui Li and Matthias Hwai Yong Tan. Improving Gaussian process emulators with boundary information. In *Artificial Intelligence, Big Data and Data Science in Statistics: Challenges and Solutions in Environmetrics, the Natural Sciences and Technology*, pages 171–192. Springer, 2022.

Songming Liu, Hao Zhongkai, Chengyang Ying, Hang Su, Jun Zhu, and Ze Cheng. A unified hard-constraint framework for solving geometrically complex PDEs. 2022.

Da Long, Zheng Wang, Aditi Krishnapriyan, Robert Kirby, Shandian Zhe, and Michael Mahoney. AutoIP: A united framework to integrate physics into Gaussian processes. In *International Conference on Machine Learning (ICML)*, 2022.

Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

Björn Lütjens, Catherine H. Crawford, Mark Veillette, and Dava Newman. PCE-PINNs: Physics-informed neural networks for uncertainty propagation in ocean modeling. *CoRR*, abs/2105.02939, 2021.

Denis Maillet. A review of the models using the cattaneo and vernotte hyperbolic heat equation and their experimental validation. *International Journal of Thermal Sciences*, 139:424–432, 2019.

Arman Melkumyan. Operator induced multi-task Gaussian processes for solving differential equations. In *Advances in Neural Information Processing Systems Workshop: New Directions in Multiple Kernel Learning*. MIT Press, 2012.

Charles A. Micchelli, Yuesheng Xu, and Haizhang Zhang. Universal Kernels. *Journal of Machine Learning Research*, 7(95):2651–2667, 2006.

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

Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer, 1996.

Vien Minh Nguyen-Thanh, Xiaoying Zhuang, and Timon Rabczuk. A deep energy method for finite deformation hyperelasticity. *European Journal of Mechanics - A/Solids*, 80: 103874, 2020.

Ola Nilsson. *Level-set methods and geodesic distance functions*. PhD thesis, Linköping Universisty Electronic Press, 2009.

Stanley Osher, Ronald Fedkiw, and K Piechor. Level set methods and dynamic implicit surfaces. *Applied Mechanics Reviews*, 57(3):B15–B15, 2004.

Samunda Perera, Nick Barnes, Xuming He, Shahram Izadi, Pushmeet Kohli, and Ben Glocker. Motion segmentation of truncated signed distance function based volumetric surfaces. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 1046–1053, 2015.

Marvin Pförtner, Ingo Steinwart, Philipp Hennig, and Jonathan Wenger. Physics-informed Gaussian process regression generalizes linear pde solvers, 2024.

GP Purja Pun, Rohit Batra, Rampi Ramprasad, and Yuri Mishin. Physically informed artificial neural networks for atomistic modeling of materials. *Nature Communications*, 10(1):2339, 2019.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Machine learning of linear differential equations using Gaussian processes. *Journal of Computational Physics*, 348: 683–693, 2017.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Numerical Gaussian processes for time-dependent and nonlinear partial differential equations. *SIAM Journal on Scientific Computing*, 40(1):A172–A198, 2018. Publisher: Society for Industrial and Applied Mathematics.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

Carl Edward Rasmussen and Christopher KI Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA., 2006.

Jaakko Riihimäki and Aki Vehtari. Gaussian processes with monotonicity information. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

Vladimir L Rvachev and Tatyana I Sheiko. R-functions in boundary value problems in mechanics. *Applied Mechanics Reviews*, 48(4):151–188, 1995.

Vladimir L Rvachev, Tatyana I Sheiko, Vadim Shapiro, and Igor Tsukanov. On completeness of RFM solution structures. *Computational Mechanics*, 25(2):305–317, 2000.

Vladimir L Rvachev, Tatyana I Sheiko, Vadim Shapiro, and Igor Tsukanov. Transfinite interpolation over implicitly defined sets. *Computer Aided Geometric Design*, 18(3):195–220, 2001.

Simo Sarkka, Arno Solin, and Jouni Hartikainen. Spatiotemporal learning via infinite-dimensional Bayesian filtering and smoothing: A look at Gaussian process regression through Kalman filtering. *IEEE Signal Processing Magazine*, 30(4):51–61, 2013.

Mícheál Ó Searcóid. *Metric Spaces*. Springer London, London, 2007.

Vadim Shapiro. Semi-analytic geometry with R-functions. *Acta Numerica*, 16:239–303, 2007.

Vadim Shapiro and Igor Tsukanov. Meshfree simulation of deforming domains. *Computer-Aided Design*, 31(7):459–471, 1999a.

Vadim Shapiro and Igor Tsukanov. Implicit functions with guaranteed differential properties. In *ACM symposium on solid modeling and applications*, 1999b.

Hailong Sheng and Chao Yang. PFNN: A penalty-free neural network method for solving a class of second-order boundary-value problems on complex geometries. *Journal of Computational Physics*, 428:110085, 2021. arXiv:2004.06490 [cs, math].

M. Sidorov and A. Artyukh. Mathematical modeling and numerical analysis of nonstationary plane-parallel flows of viscous incompressible fluid by R-functions and Galerkin method. *ECONTECHMOD*, 3:3–11, 2014.

Ercan Solak, Roderick Murray-Smith, WE Leithead, Douglas Leith, and Carl Rasmussen. Derivative observations in Gaussian process models of dynamic systems. In *Advances in neural information processing systems*, 2002.

Arno Solin and Manon Kok. Know your boundaries: Constraining Gaussian processes by variational harmonic features. In *Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2019.

Arno Solin and Simo Särkkä. Hilbert space methods for reduced-rank Gaussian process regression. *Statistics and Computing*, 30(2):419–446, 2020.

Fangying Song, Chuanju Xu, and George Em Karniadakis. Computing fractional Laplacians on complex-geometry domains: algorithms and simulations. *SIAM Journal on Scientific Computing*, 39(4):A1320–A1344, 2017.

Ingo Steinward and Andreas Christmann. *Support Vector Machines*. Information Science and Statistics. Springer New York, New York, NY, 2008.

N. Sukumar and Ankit Srivastava. Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 389:114333, 2022.

Laura P Swiler, Mamikon Gulian, Ari L Frankel, Cosmin Safta, and John D Jakeman. A survey of constrained Gaussian process regression: Approaches and implementation challenges. *Journal of Machine Learning for Modeling and Computing*, 1(2), 2020.

Matthias Tan. Gaussian process modeling with boundary information. *Statistica Sinica*, 2016.

Michalis Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

Igor Tsukanov and Vadim Shapiro. Meshfree modeling and analysis of physical fields in heterogeneous media. *Advances in Computational Mathematics*, 23:95–124, 2005.

Eric K van Douwen, David J Lutzer, and Teodor C Przymusiński. Some extensions of the Tietze-Urysohn theorem. *The American Mathematical monthly*, 84(6):435–441, 1977.

Jiahao Zhang, Shiqi Zhang, and Guang Lin. PAGP: A physics-assisted Gaussian process framework with active learning for forward and inverse problems of partial differential equations, 2022. arXiv:2204.02583 [cs, math, stat].

Olek C Zienkiewicz, Robert L Taylor, and Jian Z Zhu. *The Finite Element Method: Its Basis and Fundamentals*. Elsevier, 2005.