Black Box Probabilistic Numerics

Onur Teymur
University of Kent
Alan Turing Institute

Christopher N. Foley
University of Cambridge
Optima Partners

Philip G. Breen
Roar AI

Toni Karvonen
University of Helsinki
Alan Turing Institute

Chris. J. Oates
Newcastle University
Alan Turing Institute

Abstract

Probabilistic numerics casts numerical tasks, such the numerical solution of differential equations, as inference problems to be solved. One approach is to model the unknown quantity of interest as a random variable, and to constrain this variable using data generated during the course of a traditional numerical method. However, data may be nonlinearly related to the quantity of interest, rendering the proper conditioning of random variables difficult and limiting the range of numerical tasks that can be addressed. Instead, this paper proposes to construct probabilistic numerical methods based only on the final output from a traditional method. A convergent sequence of approximations to the quantity of interest constitute a dataset, from which the limiting quantity of interest can be extrapolated, in a probabilistic analogue of Richardson's deferred approach to the limit. This *black box* approach (1) massively expands the range of tasks to which probabilistic numerics can be applied, (2) inherits the features and performance of state-of-the-art numerical methods, and (3) enables provably higher orders of convergence to be achieved. Applications are presented for nonlinear ordinary and partial differential equations, as well as for eigenvalue problems—a setting for which no probabilistic numerical methods have yet been developed.

1 Introduction

Probabilistic numerics (PN) has attracted significant recent interest from researchers in machine learning, motivated by the possibility of incorporating probabilistic descriptions of *numerical uncertainty* into applications of probabilistic inference and decision support [1, 2]. PN treats the intermediate calculations performed in running a traditional (*i.e.* non-probabilistic) numerical procedure as *data*, which can be used to constrain a random variable model for the quantity of interest [3]. Conjugate Gaussian inference has been widely exploited, with an arsenal of PN methods developed for linear algebra [4–11], cubature [12–30], optimisation [31–36], and differential equations [37–55]. However, nonlinear tasks pose a major technical challenge to this approach, as well as to computational statistics in general, due to the absence of explicit conditioning formulae. Compared to traditional numerical methods, which have benefited from a century or more of sustained research effort, the current scope of PN is limited. The performance gap is broadly characterised by the absence of certain important functionalities—adaptivity, numerical well-conditioning, efficient use of computational resource—all of which contribute to limited applicability in real-world settings.

This article proposes a pragmatic solution that enables state-of-the-art numerical algorithms to be immediately exploited in the context of PN. The idea, which we term *black box probabilistic numerics* (BBPN), is a statistical perspective on *Richardson's deferred approach to the limit* (RDAL) [56]. The starting point for BBPN is a sequence of increasingly accurate approximations produced by a

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

traditional numerical method as its computational budget is increased. Extrapolation of this sequence (to the unattainable limit of 'infinite computational budget') is formulated as a prediction task, to which statistical techniques can be applied. For concreteness, we perform this prediction using *Gaussian process*es (GPs) [57], but other models could be used. Note that we do not aim to remove the numerical analyst from the loop; the performance of BBPN is limited by that of the numerical method on which it is based.

There are three main advantages of BBPN compared to existing methods in PN: (1) BBPN is applicable to any numerical task for which there exists a traditional numerical method; (2) state-of-the-art performance and functionality are automatically inherited from the underlying numerical method; (3) BBPN achieves a provably higher order of convergence relative to a single application of the numerical method on which it is based, in an analogous manner to RDAL. The main limitations of BBPN, compared to existing methods in PN, are: (1) multiple realisations of a traditional numerical method are required (*i.e.* one datum is not sufficient in an extrapolation task), and (2) a joint statistical model has to be built for not just the quantity of interest (as in standard PN), but also for the error associated with the output of a traditional numerical method. The capacity of generic statistical models, such as GPs, to learn salient aspects of this structure from data and to produce meaningful predictions over a range of real-world numerical problems, demands to be investigated.

The article is organised as follows: In Section 2 we recall classical RDAL. In Section 3 we lift RDAL to the space of probability distributions, exploiting GPs to instantiate BBPN and providing a theoretical guarantee that higher order convergence is achieved by using GPs within BBPN. In Section 4 we present a detailed empirical investigation into BBPN, demonstrating its effectiveness on challenging tasks that go beyond the capability of current methods in PN, while also highlighting potential pitfalls. As part of this we perform a comparison of the uncertainty quantification properties of BBPN against earlier approaches. A closing discussion is contained in Section 5.

2 Turning Lead into Gold

Our starting point is the celebrated observation of Richardson [56], that multiple numerical approximations can be combined to produce an approximation more accurate than any of the individual approximations. To see this, consider an intractable scalar quantity of interest $q^* \in \mathbb{R}$, and suppose that q^* can be approximated by a numerical method q that depends on a parameter $h > 0$, such that

$$q(h) = q^* + Ch^\alpha + \mathcal{O}(h^{\alpha+1}) \tag{1}$$

for some $C \in \mathbb{R}$ (which may be unknown) and $\alpha > 0$ (which is assumed known, and called the *order* of the method). Clearly $q(h)$ converges to q^* as $h \to 0$, but we also suppose that the *cost* of computing $q(h)$ increases in the same limit, with exact evaluation of $q(0)$ requiring a hypothetically infinite computational budget. Proposition 1 is the cornerstone of RDAL. It demonstrates that two evaluations of a numerical method of order α can be combined to obtain a numerical method of order $\alpha + 1$. An elementary proof of this foundational result is provided in Appendix A.

Proposition 1. *Let q be a numerical method of order α, as in* (1)*. Fix $\gamma \in (0, 1)$ and let $q_\gamma(h)$ denote the height at which a straight line drawn through the points $(h^\alpha, q(h))$ and $((\gamma h)^\alpha, q(\gamma h))$ intersects the vertical axis in $\mathbb{R}^2$. Then q_γ is a numerical method of order $\alpha + 1$.*

Now consider the natural generalisation of Proposition 1, in which we compute approximations $q(h_i)$ along a decreasing sequence of values $(h_i)_{i=1}^n$. One can then fit a smooth interpolant to the points $\{(h_i^\alpha, q(h_i))\}_{i=1}^n$ (generalising the straight line through two points), then extrapolate this to $h = 0$, to give an estimate for the quantity of interest. This simple idea is widely used in numerical analysis; its potential to radically improve solution accuracy, given only a sequence of simple calculations as input, prompted Press et al. [58, p. 922] to describe it as "turning lead into gold". The practical success of RDAL depends on the choice of interpolant, with polynomial interpolation being most commonly used. Unqualified, RDAL is usually understood to refer to an order $n-1$ polynomial fitted to n points, which produces a numerical method of order $\alpha+n$; see Theorem 9.1 of [59]. Higher-order polynomial extrapolation is known to perform poorly unless the values $(h_i)_{i=1}^n$ are able to be chosen specifically to mitigate Runge's phenomenon [60], motivating the Bulirsch–Stoer algorithm [61], which instead fits a rational function interpolant. This allows both greater expressiveness and robustness than polynomial interpolation (though not necessarily as efficiently [58]). These methods are all situated within the broad category of *extrapolation methods* in numerical analysis; a comprehensive historical survey can be found in [62].

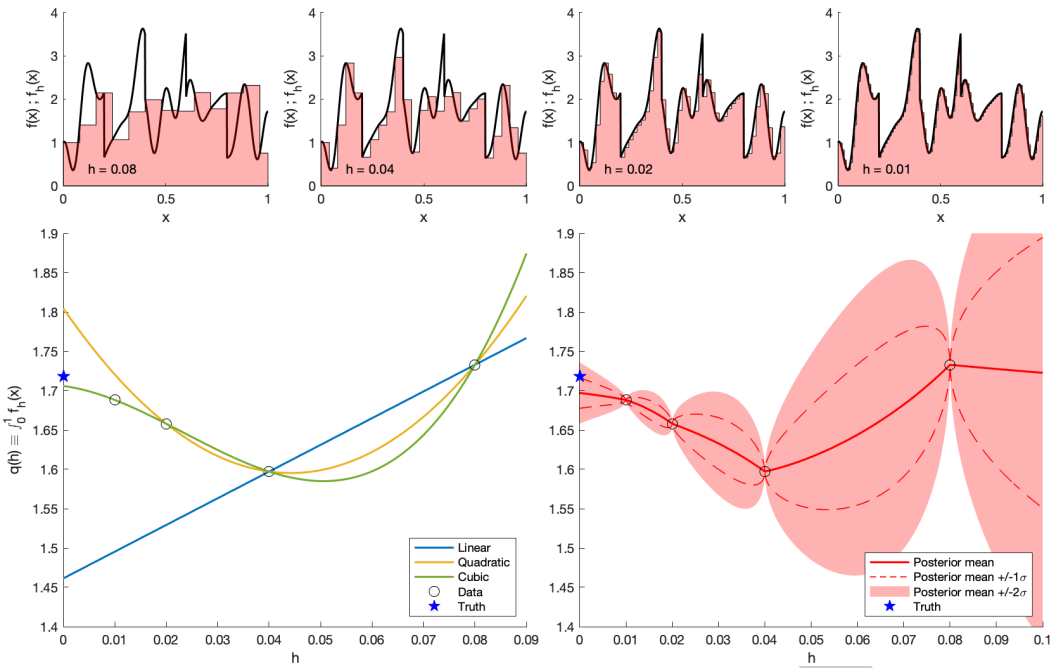

Figure 1: Richardson's deferred approach to the limit (RDAL), applied to the method of Riemann sums with integration bandwidth h and an oscillatory integrand $f(x)$, displayed in the four top panes. The bottom left pane shows linear, quadratic and cubic interpolants converging on the true value of the integral, denoted by a blue star. The strength of RDAL is visible in the fact that the cubic interpolant gives a better estimate than that of the finest-grid Riemann sum. The bottom right pane illustrates *black box probabilistic numerics* (BBPN) in which a Gaussian process (GP) is fitted to the same data. The GP specification is crucial to the performance of BBPN, and is described in Section 3.

Figure 1 presents a simple visual demonstration of RDAL, applied to the method of Riemann sums for an oscillatory 1D integrand. While RDAL gives improved approximations, no quantification of estimation uncertainty is provided. The only attempt of which we are aware to provide uncertainty quantification for RDAL is due to [63], who focused on the Navier–Stokes equation and a specific scalar quantity of interest. Here we go further, proposing the general framework of BBPN and introducing novel methodology that goes beyond scalar quantities of interest. The right-hand pane of Figure 1 displays the outcome of the method we are about to introduce, applied to the same task—observe that the true value of the integral falls within the $\pm 2\sigma$ credible set produced using BBPN. Details of the simulations in this figure are contained in Appendix C.1.

3 Methodology

The core idea of BBPN is to model $q(h)$ as a *stochastic process* $Q(h)$ rather than fit a deterministic interpolant as in RDAL. The distribution of the marginal random variable $Q(0)$ is then interpreted as a representation of the epistemic uncertainty in the quantity of interest $q(0)$. Conjugate Gaussian inference can be performed in BBPN, since one needs only to construct an interpolant. This means the challenge of nonlinear conditioning encountered in PN [4] is avoided, massively extending the applicability of PN. In addition to being able to leverage state-of-the-art numerical methods, the BBPN approach enjoys provably higher orders of convergence relative to a single application of the numerical method on which it is based; see Section 3.3.

3.1 Notation and Setup

Our starting point is to generalise (1) to encompass essentially all numerical tasks, following the abstract perspective of [64]. To do so, we observe that any quantity of interest q^* can be characterised by a sufficiently large collection of real values $q^*(t)$, with t ranging over an appropriate index set T.

Definition 2. *A traditional numerical method is defined as a map $q : [0, h_0) \times T \to \mathbb{R}$, for some $h_0 > 0$ such that, for all $t \in T$, the function $h \mapsto q(h, t)$ is continuous at 0 with limit $q(0, t) = q^*(t)$.*

For example, a (univariate) *initial value problem* (IVP), in which t is interpreted as time, can be solved using a traditional numerical method q whose time step size h trades off approximation error against computational cost. The output $q(h, t)$ of such a method represents an approximation to the true solution $q^*(t)$ of the IVP, at each time t for which the solution is defined. In general, depending on the numerical task, the index t could be spatio-temporal, discrete, or even an unstructured set, while the meaning of the index h will depend on the numerical method. Definition 2 thus encompasses, among other things: (1) adaptive integrators for time-evolving *partial differential equations* (PDEs), where $h > 0$ represents a user-specified error tolerance, and the spatio-temporal domain of the solution is indexed by T; (2) iterative methods for approximating the singular values of a $d \times d$ matrix, where for example $h := w^{-1}$ with w the number of iterations performed, and the ordered singular values are indexed by $T = \{1, \ldots, d\}$; and (3) the simultaneous approximation of multiple related quantities of interest, where T indexes not only the domain(s) on which the individual quantities of interest are defined, but also the multiple quantities of interest themselves (this situation arises, for example, in inverse problems that are *PDE-constrained* [65]).

The perspective in Definition 2 is abstract but, as these examples make clear, it will typically only be possible to compute q at certain input values (such as $h = w^{-1}$ for $w \in \mathbb{N}$, or for just a finite collection of inputs t if the index set T is infinite), and furthermore each evaluation is likely to be associated with a computational cost. Thus complete information regarding the map $q : [0, h_0) \times T \to \mathbb{R}$ will not be available in general, and there will therefore remain *epistemic uncertainty* in its complete description. Our aim in Section 3.2, in line with the central philosophical perspective of PN, is to characterise this uncertainty using a statistical model.

3.2 Black Box Probabilistic Numerics

The proposed BBPN approach begins with a *prior* stochastic process Q and constrains this prior using data D. Concretely, we assume that the real values $q(h_i, t_{i,j})$ are provided at a finite set of resolutions $h_1 > \cdots > h_n > 0$ and distinct ordinates $t_{i,1}, \ldots, t_{i,m_i} \in T$. Note that the number of t-ordinates m_i can depend on h_i. Our dataset therefore contains the following information on q:

$$D := \{(h_i, t_{i,j}, q(h_i, t_{i,j})) : \ i = 1, \ldots, n; \ j = 1, \ldots, m_i\}. \tag{2}$$

The stochastic process obtained by conditioning Q on the dataset D, denoted $Q|D$, implies a marginal distribution for $Q(0, \cdot)$, which we interpret as a statistical prediction for the unknown quantity of interest $q^*(\cdot)$. In order for uncertainty quantification in this model to be meaningful, one either requires expert knowledge about the numerical method that generated D, or one must employ a stochastic process that is able to adapt to the data, so that its predictions can be calibrated.

Our goal is to specify a stochastic process model $Q(h, t)$ that behaves in a desirable way under extrapolation to $h = 0$. To this end, we decompose

$$Q(h, t) = Q^*(t) + E(h, t), \tag{3}$$

where $Q^*(t)$ is a prior model for the unknown quantity of interest $q^*(t)$, and $E(h, t)$ is a prior model for the error of the numerical method. It will be assumed that Q^* and E are independent (denoted $Q^* \perp\!\!\!\perp E$), meaning that prior belief about the quantity of interest is independent of prior belief regarding the performance of the numerical method. (This assumption is made only to simplify the model specification, but if detailed insight into the error structure of a numerical method is available then this can be exploited.) Compared to the existing PN methods cited in Section 1, a prior model for the error E is an additional requirement in BBPN.

The error $E(h, t)$ is assumed to vanish[1] as $h \to 0$, meaning that a stationary stochastic process model for $E(h, t)$, and hence for $Q(h, t)$, is inappropriate, and can result in predictions that are both

[1] This statement covers several potentially subtle notions from numerical analysis such as well-posedness of the problem and numerical stability of the algorithm; these are studied in detail in their own right in the literature, and for our purposes it suffices to assume that the error behaves well in the limit.

severely biased as well as under-confident; see Appendix C.2. In the next section, we propose a parsimonious non-stationary GP model for $Q(h,t)$ of the form (3), which combines knowledge of the *order* of the numerical method (only) with data-driven estimation of GP hyperparameters. This setting is practically relevant—the order of a numerical method is typically one of the first theoretical properties that researchers aim to establish while, conversely, for more complex numerical methods the order may actually be the only salient high-level error-characterising property that is known, and thus represent the limit of mathematical insight into the method.

3.3 Gaussian Process BBPN

Gaussian processes provide a convenient model for Q^* and E, since they produce an explicit form for the conditional $Q|D$. The details of conjugate Gaussian inference are standard (see e.g. [57]) and so relegated to Appendix B.1; our focus here is on the specification of GP priors for Q^* and E.

The notation $Q \sim \mathcal{GP}(\mu_Q, k_Q)$ will be used to denote that Q is a GP with mean function μ_Q and covariance function k_Q. With no loss of generality, in what follows we consider centred processes (*i.e.* $\mu_Q = 0$). It will be assumed that $T = T_1 \times \cdots \times T_p$, with each T_i either a discrete or a continuous subset of a Euclidean space, with the Euclidean distance between elements $t_i, t_i' \in T_i$ being denoted $\|t_i - t_i'\|$. (Typical applications involve small p; for example, the domain of a spatio-temporal PDE is typically decomposed as $T = T_1 \times T_2$ where T_1 indexes time and T_2 indexes all spatial dimensions, so that $p = 2$.)

Prior for Q^*: In the absence of detailed prior belief about q^*, we consider the following default prior model. Let $G \sim \mathcal{GP}(0, \sigma^2 \rho_G k_G)$, $Z = (Z_1, \ldots, Z_v) \sim \mathcal{N}(0, \sigma^2 I)$, and let $Z \perp\!\!\!\perp G$. Let $b_1, \ldots, b_v$ be a finite a collection of basis functions and set $b(t) = (b_1(t), \ldots, b_v(t))^\top$. Then set

$$Q^*(t) = Z \cdot b(t) + G(t)$$

where $\sigma^2, \rho_G > 0$ are parameters to be estimated. The basis b will be problem-specific and could be a polynomial basis, Fourier basis, or any number of other bases depending on context. The case $v = 1$ with a constant intercept is closely related to *ordinary kriging* and the case $v > 1$ is closely related to *universal kriging* [66, p. 8]. The apparent redundancy in the parameterisation due to the product $\sigma^2 \rho_G$ will be explained later. Using the notation $t = (t_1, \ldots, t_p)$, we consider a tensor product covariance model $k_G(t, t') = \prod_{i=1}^p k_{G,i}(t_i, t_i')$, $k_{G,i}(t_i, t_i') = \phi_i(\|t_i - t_i'\|/\ell_{t,i})$, for some radial basis functions ϕ_i, scaled to satisfy $\phi_i(0) = 1$, and length-scale parameters $\ell_{t,i} > 0$ to be estimated.

Prior for E: The process $E(h,t)$ is a model for the numerical error $q(h,t) - q^*(t)$, $t > 0$, which may be highly structured. A flexible prior model is therefore required. Moreover the error will, by definition, depend on the order of the numerical method; for successful extrapolation we must therefore encode this order into the model for E. It was earlier argued that a stationary GP is inappropriate, since the error is assumed to be $\mathcal{O}(h^\alpha)$. However, we observe that $h^{-\alpha}(q(h,t) - q^*(t))$ is $\mathcal{O}(1)$, suggesting that this quantity can be modelled using a stationary GP. We therefore take $E \sim \mathcal{GP}(0, \sigma^2 \rho_E k_E)$, where $\rho_E > 0$ is a parameter to be estimated, and

$$k_E((h,t),(h',t')) = (hh')^\alpha \psi(|h - h'|/\ell_h) \cdot k_G(t,t') \tag{4}$$

for a radial basis function ψ, scaled to satisfy $\psi(0) = 1$, and a length-scale parameter $\ell_h > 0$ to be estimated. Note how (4) separates the h and t dependence of E in the prior, and adopts the same covariance model k_G that was used to model the t dependence of G. This can be motivated by the alternative perspective that follows from observing that Q is a GP with covariance function

$$k_Q((h,t),(h',t')) = \sigma^2 \left\{ b(t) \cdot b(t') + \rho_G k_G(t,t') \left(1 + \rho_E \frac{k_E((h,t),(h',t'))}{k_G(t,t')} \right) \right\} \tag{5}$$

where k_E/k_G is a kernel only depending on h. Written this way, the model is seen to perform universal kriging over T with a covariance adjusted by a multiplicative error arising from non-zero values of h.

Higher-order convergence: The GP specification just described is not arbitrary; it ensures that the higher-order convergence property of RDAL is realised in BBPN. Consider again the setting in Proposition 1. Suppose that there exist $L, \varepsilon_0 > 0$ and $\beta \in (0, 1]$ such that $|1 - \psi(\varepsilon)| \leq L\,\varepsilon^\beta$ for all

$\varepsilon \in [0, \varepsilon_0)$. Then the posterior mean $\mathbb{E}[Q(0)|D_h]$ satisfies $|q^* - \mathbb{E}[Q(0)|D_h]| = \mathcal{O}(h^{\alpha+\beta})$ as $h \to 0$. Thus if ψ is Lipschitz (*i.e.* $\beta = 1$), BBPN achieves the same higher-order convergence, $\alpha + 1$, as RDAL. In this context we recall that any Matérn covariance function of smoothness at least $1/2$ is Lipschitz. The proof is provided in Appendix B.2.

Model parameters: The free parameters of our prior model, σ^2, ρ_G, ρ_E, ℓ_h, and the $\ell_{t,i}$ for $i = 1, \ldots, p$, are collectively denoted θ. For all experiments in this article, θ was set using the maximum likelihood[2] estimator θ_{ML}. Our choice of parameterisation ensures that the maximum likelihood estimate for the overall scale σ^2, denoted σ^2_{ML}, has a closed form expression in terms of the remaining parameters. This is analytically derived in Appendix B.3 and can be plugged straight into the likelihood. Gradients with respect to the remaining $3 + p$ parameters are derived in Appendix B.3, and gradient-based optimisation of the log-likelihood was implemented for the remaining parameters.

Remark 3. *GP interpolation, as with classical RDAL, is not parameterisation invariant. Thus some care is required to employ a parameterisation of h that is amenable to the construction of a GP interpolant. The effect of differing parameterisations is explored in Appendix C.2.*

Remark 4. *The classical definition of RDAL presupposes that, in order to employ the method, the order α must be known* a priori *[67]. However if α is not known, the probabilistic perspective affords us the opportunity to learn α as an additional parameter in the statistical model—a procedure with no classical analogue. The feasibility of learning α is explored in Section 4.2.*

Code: Software for BBPN, including code to reproduce the experiments in Section 4, can be downloaded from `github.com/oteym/bbpn`.

4 Experimental Assessment

This section reports a rigorous experimental assessment of BBPN. Firstly, Section 4.1 demonstrates that BBPN is competitive with existing PN methods in the context of *ordinary differential equation*s (ODEs). This result is somewhat surprising, given the black box nature of BBPN compared to the bespoke nature of existing PN methods for ODEs. Secondly, in Section 4.2 we demonstrate the versatility of BBPN by applying it to the nonlinear problem of eigenvalue computation, for which no PN methods currently exist. Finally, in Section 4.3 we use BBPN to provide uncertainty quantification for state-of-the-art numerical methods that aim to approximate the solution of nonlinear PDEs.

Default Settings: We use Matérn(1/2) kernels for ϕ_i and ψ, *i.e.* $\phi_i(t_i, t_i') = \exp(-\|t_i - t_i'\|/\ell_{t,i})$, and similarly *mutatis mutandis* for ψ. These kernels impose a minimal continuity assumption on q without additional levels of smoothness being assumed. Sensitivity of results to the choice of kernel is investigated in Appendix C.2.

Performance Metrics: PN is distinguished from traditional numerical analysis by its aim to provide probabilistic uncertainty quantification, but nevertheless approximation accuracy remains important. To perform an assessment on these terms, we considered two orthogonal metrics. Firstly, we compute the *error* of the point estimate (mean), denoted $W := \|\mathbb{E}[Q(0,\cdot)|D] - q^*(\cdot)\|$, where the norm is taken over $t \in T'$ where T' is either T itself or a set of representative elements from T. Secondly, and most importantly from the point of view of PN, we consider the *surprise* $S := \|\mathbb{C}[Q(0,\cdot)|D]^{-1/2}(\mathbb{E}[Q(0,\cdot)|D)] - q^*(\cdot))\|$, where $\mathbb{C}[Q(0,\cdot)|D]$ denotes the posterior covariance matrix. If the true quantity of interest q^* was genuinely a sample from $Q(0,\cdot)|D$, then S^2 would follow a χ^2 distribution with $|T'|$ degrees of freedom. This observation enables the *calibration* of a PN method to be assessed [68]. Both metrics naturally require an accurate approximation to q^* to act as the ground truth, which is available using brute force computation in Sections 4.1 and 4.2 but not in Section 4.3. The role of Section 4.3 is limited to demonstrating BBPN on a problem class that is challenging even for state-of-the-art methods.

[2] Alternative approaches, such as cross-validation, could also be used; see Chapter 5 of [57]. Our choice of maximum likelihood was motivated by the absence of any degrees of freedom (such as the number of folds of cross-validation), which permits a more objective empirical assessment.

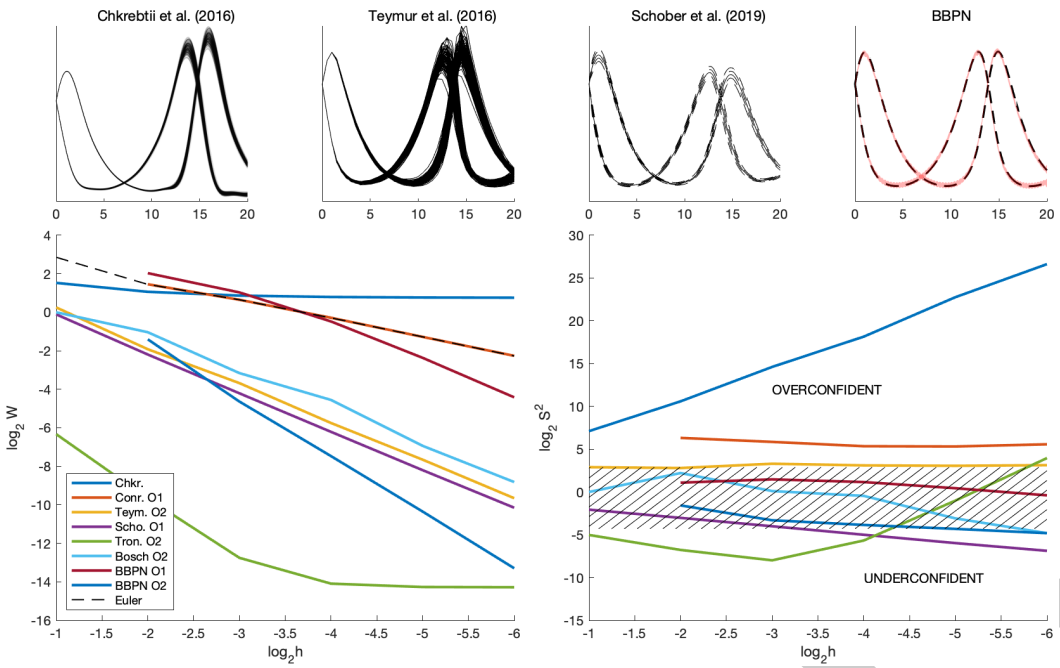

Figure 2: Ordinary differential equations. Top: Output from three existing PN algorithms [39–41] and BBPN, applied to the Lotka–Volterra IVP. Bottom left: The error $\log_2 W$ at the final time point $t_{\text{end}} = 20$, as a function of the time step size h. Bottom right: The surprise $\log_2 S$ at $t_{\text{end}} = 20$, with the central 95% probability band of a χ^2_2 random variable shaded. Methods shown with (where applicable) their order: Chkr. [39]; Conr. O1 [38]; Teym. O2 [41]; Scho. O1 [40]; Tron. O2 [47]; Bosch O2 [55]; BBPN O1 & O2; and (traditional) Euler.

4.1 Ordinary Differential Equations

The numerical solution of ODEs has received considerable attention in PN, with several sophisticated methods available to serve as benchmarks. Here we consider numerical solution of the following Lotka–Volterra IVP, a popular test case in the PN literature:

$$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t} = f(t, \mathbf{y}) = \begin{bmatrix} 0.5y_1 - 0.05y_1y_2 \\ -0.5y_2 + 0.05y_1y_2 \end{bmatrix}, \quad \mathbf{y}(0) = \begin{bmatrix} 20 \\ 20 \end{bmatrix}.$$

The aim in what follows is to approximate the quantity of interest $q^* = \mathbf{y}(t_{\text{end}})$ for $t_{\text{end}} = 20$. The top row of Figure 2 displays output from three distinct PN methods due to [39–41]. (For these plots the coarse step-size $h = 0.5$ was used, so the probabilistic output can be easily visualised.) In each case, these methods treat a sequence of evaluations of the gradient field f as data which are used to constrain a random variable model for the unknown solution of the ODE. Their fundamentally differing character makes direct comparisons challenging, particularly if we are to account for computational cost. However, each algorithm has a recognisable discretisation parameter h, so it remains instructive to study their $h \to 0$ limit. (In most cases h represents a time step size, but the method of [55] is step-size adaptive; in this case h is an error tolerance that is user-specified.) The methods of [38], [39], and [41] require parallel simulations to produce empirical credible sets, and thus have a significant computational cost. The methods of [40] and [47] are based on Gaussian filtering and are less computationally demanding, though in disregarding nonlinearities the description of uncertainty they provide is not as rich. Interestingly, the output from [39] becomes overconfident as $h \to 0$, with S^2 being incompatible with a χ^2_2 random variable, while the output from [40] becomes pessimistic in the same limit. Aside from these two outputs, the other PN methods considered appear to be reasonably calibrated.

To illustrate BBPN, our data consist of the final states produced by either an Euler (order 1) or an Adams–Bashforth (order 2) algorithm, which were performed at different resolutions $\{h_i = 2^{-i}, i =$

$1\ldots,6\}$. The dataset[3] is augmented cumulatively, so that for $i = i'$, all data generated by runs $1,\ldots,i'$ are used. The finest resolution in each case, h_i, is simply denoted h. For this experiment we use a prior with constant intercept, *i.e.* $v = 1$ and $b_1(t) = 1$. The BBPN output, shown in Figure 2, is observed to be calibrated, and the (order 2) output provides the most accurate approximation among all calibrated PN methods considered. Note in particular how BBPN accelerates the convergence of the Euler method from first order to second order, akin to RDAL. In terms of computational cost, BBPN requires running a traditional numerical method at different resolutions, as well as the fitting of a GP. In this experiment, the computational cost of BBPN was intermediate between the filtering approach of [40] and the sampling approaches of [39] and [41]. Further details, including the sources and implementation of all these codes, are given in Appendix C.3.

4.2 Eigenvalue Problems

The calculation of eigenvalues is an important numerical task that has not yet received attention in PN. In this section we apply BBPN to (1) the QR algorithm for matrix eigenvalue discovery, and (2) a cutting-edge adaptive algorithm for the *tensor* eigenproblem, called the shifted power method [69]. In these examples, the order α is *unknown* and we append it to θ as an additional parameter to be estimated using maximum likelihood.

QR Algorithm: Take $T = \{1,\ldots,n\}$. Let $w \in \mathbb{N}$ and define $h \coloneqq w^{-1}$. Given a matrix $A \in \mathbb{R}^{n\times n}$, its Schur form A_∞ is approximated by matrices $A_w \coloneqq R_{w-1}Q_{w-1}$, where R_{w-1} and Q_{w-1} arise as a result of performing a QR decomposition on $A_{w-1} = Q_{w-1}R_{w-1}$, and where $A_0 \coloneqq A$. Then $q(h,\cdot)$ is the vector $\mathrm{diag}(A_{h^{-1}})$, and $q^* = \mathrm{diag}(A_\infty)$ is the vector of eigenvalues of A. As a test problem, whose eigenvalues are available in closed form (see Appendix C.4), we consider the following family of sparse matrices that arise as the discrete Laplace operator in the solution of the Poisson equation by a finite difference method with a five-point stencil. Let the $l \times l$ matrix B and the $ml \times ml$ matrix A be defined by

$$B = \begin{pmatrix} 4 & -1 & & \\ -1 & 4 & -1 & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 \end{pmatrix}, \qquad A = \begin{pmatrix} B & -I & & \\ -I & B & -I & \\ & \ddots & \ddots & -I \\ & & -I & B \end{pmatrix}.$$

BBPN output for this problem is displayed in Figure 3. In the left-hand pane, we take $l = 5, m = 2$ and perform 5 QR iterations, displaying all 10 eigenvalues. In the centre pane, we take $l = 10, m = 10$ and perform 15 iterations. For clarity, this pane only displays the largest few eigenvalues of this 100×100 matrix, and we also show a zoomed-in crop to better demonstrate the extrapolation quality. Both examples show the convergence of $Q(h,\cdot)$ to $q^*(\cdot)$ as $w \to \infty$. Recall that α is *inferred* in these simulations —the maximum likelihood values were, respectively, 1.0186 and 1.0167.

The extrapolation performed by our GP model is seen visually to be effective and almost all true eigenvalues are contained within the $\pm 2\sigma$ credible intervals plotted. For comparison, in Appendix C.2 we contrast the result of using a *stationary* GP model (*i.e.* $\alpha = 0$). The extrapolating properties of that GP are immediately seen to be unsatisfactory, and we support this observation by examining the calibration of the two approaches, in a similar manner to in Section 4.1. This analysis strongly supports our proposed GP specification in Section 3.3.

Shifted Power Method: This iterative algorithm, due to [69] and implemented in [70], finds (random) eigenpairs of higher-order tensor systems, and we include it to demonstrate BBPN on a challenging problem in linear algebra. For $\boldsymbol{\mathcal{A}}$ a symmetric mth-order n-dimensional real tensor, and $\mathbf{x}$ an n-dimensional vector, define

$$\left(\boldsymbol{\mathcal{A}}\mathbf{x}^{m-1}\right)_{i_1} \coloneqq \sum_{i_2=1}^{n} \cdots \sum_{i_m=1}^{n} a_{i_1 i_2 \ldots i_m} x_{i_2} \ldots x_{i_m}\,, \qquad i_1 = 1,\ldots,n,$$

and say that $\lambda \in \mathbb{R}$ is an eigenvalue of $\boldsymbol{\mathcal{A}}$ if there exists $\mathbf{x} \in \mathbb{R}^n$ such that $\boldsymbol{\mathcal{A}}\mathbf{x}^{m-1} = \lambda\mathbf{x}$ and $\mathbf{x}^\top\mathbf{x} = 1$. Here we take $n = m = 6$ and produce a random symmetric tensor using the `create_problem`

[3] In this experiment the two components of q^* were treated as *a priori* independent, but this is not a specific requirement of BBPN and dependence between outputs can in principle also be encoded into the GP model.

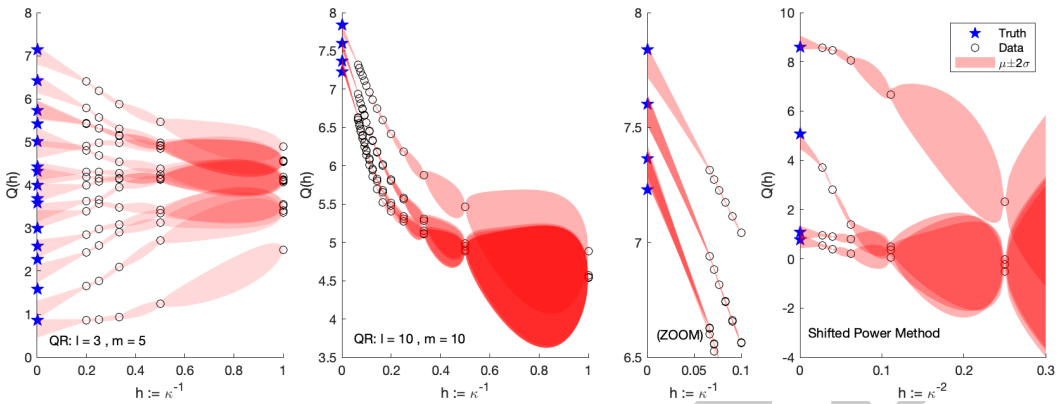

Figure 3: Eigenvalue problems. Left and centre: QR algorithm. Right: Shifted power method. Details of each simulation are given in the main text. All plots show red shaded $\pm 2\sigma$ credible intervals, numerical data as black circles, and true eigenvalues as blue stars.

function of [70]. Two parameterisations of q were considered; $h := w^{-1}$ and $h := w^{-2}$, where w denotes the number of iterations performed, with results based on the latter parameterisation presented in the right-hand pane of Figure 3. (The maximum likelihood value for α in this example was 1.3318.) It can be seen that, after 5 iterations, BBPN is more accurate than each of the individual approximations on which it was trained. The choice of parameterisation affects the performance of BBPN, an issue we explore further in Appendix C.2.

In this example there is no additional computational cost to BBPN in the data collection stage, since the dataset is generated during a single run of an iterative numerical method. Therefore the only overhead is due to fitting the GP; though for this example this cost is itself negligible. All details, including a systematic assessment of error W and surprise S as h is varied for both the QR algorithm and the shifted power method, are given in Appendix C.4.

Remark 5. *In this section we have implicitly modelled eigenvalues as* a priori *independent, for simplicity of exposition. Heuristics from random matrix theory suggest that, when treated probabilistically, eigenvalues may be better modelled with some non-trivial dependence structure. We note that this additional structure can be easily incorporated into the prior GP model for BBPN.*

4.3 Partial Differential Equations

To demonstrate the potential of BBPN on a challenging problem for which state-of-the-art numerical methods are required, we consider numerical solution of the *Kuramoto–Sivashinsky equation* [71, 72]

$$\partial_t u + \partial_x^4 u + \partial_x^2 u + u \partial_x u = 0. \tag{6}$$

This equation is used to study a variety of reaction-diffusion systems, producing complex spatio-temporal dynamics which exhibit temporal chaos—the characteristics of which depend strongly on the amplitude of the initial data and the domain length. We consider a flat initial condition $u(x, 0) = \exp(-0.01x^2)$ with periodic boundary condition on the domain $0 \leq x \leq 1$, and numerically solve to $t = 200$. Our quantity of interest is therefore $u(x, 200)$ for the domain $x \in [0, 1]$.

To obtain numerical solutions to (6) we transfer the problem into Fourier space and apply the popular fourth-order time-differencing ETD RK4 numerical scheme; see Appendix C.5 and [73]. ETD RK4 was designed to furnish fourth-order numerical solutions to time-dependent PDEs which combine low-order nonlinear terms with higher-order linear terms, as in (6). Computing accurate approximations to the solution of chaotic, stiff PDEs is a challenging problem for existing PN methods because computationally demanding high-order approximations across both spatial and temporal domains are required. Here, we assess BBPN applied to sequences of six runs of ETDRK4, with minimum temporal step size $h = \delta t$ and, for simplicity, a fixed spatial step size $\delta x = 0.001$ throughout.[4] A

[4]For the $h = 0.002$ simulation in 4, we have $h_i \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5\}$, and for the $h = 0.01$ simulation we have $h_i \in \{0.002, 0.005, 0.01, 0.02, 0.05, 0.1\}$

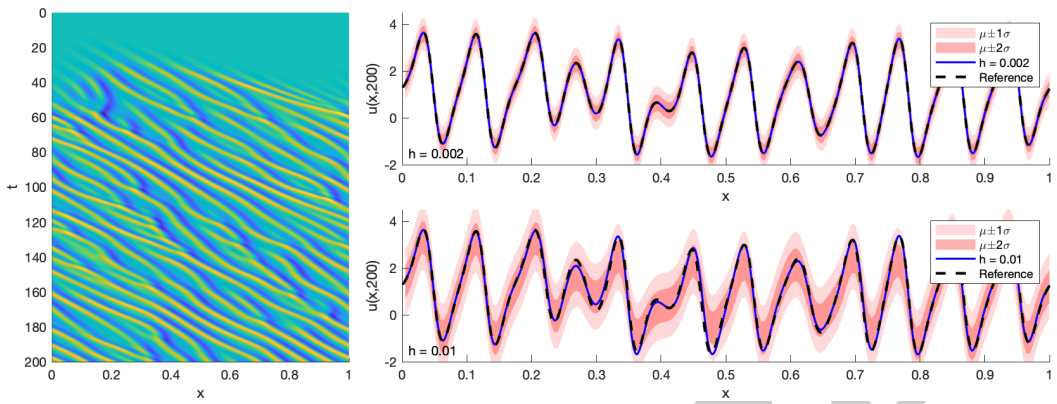

Figure 4: Partial differential equations. Left: Solution to the Kuramoto–Sivashinsky equation. Right: Approximation of the solution at the final time point ($t = 200$) using BBPN, based on minimum time step sizes $h \in \{0.01, 0.002\}$. Posterior mean (blue) and credible regions (shaded) are displayed. Also shown is a reference solution (dashed black) obtained by taking $h = 0.0005$, but note that this may differ from the true solution due to the challenging nature of the numerical task.

reference solution was generated by taking $h = 0.0005$, but this cannot of course be guaranteed to be an accurate approximation to the true solution of (6).

Results shown in Figure 4 are encouraging; not only can accurate approximations be produced, but the associated uncertainty regions appear to be reasonably well calibrated, insofar as the magnitude of the uncertainty is consistent with the magnitude of the discrepancy between the posterior mean and the reference solution. Full details of these simulations are contained in Appendix C.5.

5 Discussion

This paper presented *black box probabilistic numerics*, a simple yet powerful framework that bridges the gap between existing PN methods and the numerical state-of-the-art. Positive results were presented on the important problems of numerically approximating ODEs, eigenvalues, and PDEs. Our main technical contribution is a probabilistic generalisation of Richardson's deferred approach to the limit, which may be of independent interest.

The main drawbacks, compared to existing PN, are a possibly increased computational cost and the additional requirement to model the error of a traditional numerical method. Compared to existing PN, in which detailed modelling of the inner workings of numerical algorithms are exploited, only the order of the numerical method is used in BBPN (and we can even dispense with that, as in Section 4.2), which may reduce its expressiveness in some settings. However, despite the black box approach, BBPN was no less accurate than existing PN in our experiments, and in fact the higher-order convergence property may enable BBPN to out-perform existing PN.

Some avenues for further research (that we did not consider in the present article) include the use of more flexible and/or computationally cheaper alternatives to GPs, the adoption of principles from experimental design to sequentially select resolutions h_i given an overall computational budget, and the simultaneous use of different traditional (or even probabilistic) numerical methods within BBPN.

Acknowledgments and Disclosure of Funding

This work was supported by the Lloyd's Register Foundation programme on data-centric engineering at the Alan Turing Institute, UK. The authors wish to thank Jon Cockayne and Ilse Ipsen for feedback on earlier versions of the manuscript, and Nicholas Krämer for assistance with the `probnum` package.

References

[1] P. Hennig, M. A. Osborne, and M. Girolami, "Probabilistic numerics and uncertainty in computations," *Proc. R. Soc. A*, vol. 471, no. 2179, p. 20150142, 2015.

[2] C. J. Oates and T. J. Sullivan, "A modern retrospective on probabilistic numerics," *Stat. Comput.*, vol. 29, no. 6, pp. 1335–1351, 2019.

[3] J. Cockayne, C. J. Oates, T. J. Sullivan, and M. Girolami, "Bayesian probabilistic numerical methods," *SIAM Rev.*, vol. 61, no. 4, pp. 756–789, 2019.

[4] J. Cockayne, C. J. Oates, I. C. Ipsen, M. Girolami, *et al.*, "A Bayesian conjugate gradient method (with discussion)," *Bayesian Anal.*, vol. 14, no. 3, pp. 937–1012, 2019.

[5] S. Bartels, J. Cockayne, I. C. Ipsen, and P. Hennig, "Probabilistic linear solvers: a unifying view," *Stat. Comput.*, vol. 29, no. 6, pp. 1249–1263, 2019.

[6] J. Wenger and P. Hennig, "Probabilistic linear solvers for machine learning," *NeurIPS 34*, 2020.

[7] P. Hennig, "Probabilistic interpretation of linear solvers," *SIAM J. Optim.*, vol. 25, no. 1, pp. 234–260, 2015.

[8] T. W. Reid, I. C. Ipsen, J. Cockayne, and C. J. Oates, "A probabilistic numerical extension of the conjugate gradient method," *arXiv:2008.03225*, 2020.

[9] F. Schäfer, T. J. Sullivan, and H. Owhadi, "Compression, inversion, and approximate PCA of dense kernel matrices at near-linear computational complexity," *Multiscale Model. Simul.*, vol. 19, no. 2, pp. 688–730, 2021.

[10] S. Bartels and P. Hennig, "Probabilistic approximate least-squares," *Proc. 19th Int. Conf. Artificial Intelligence and Statistics*, vol. 51, pp. 676–684, 2016.

[11] J. Cockayne, I. C. Ipsen, C. J. Oates, and T. W. Reid, "Probabilistic iterative methods for linear systems," *arXiv:2012.12615*, 2020.

[12] P. Diaconis, "Bayesian numerical analysis," *Statistical Decision Theory and Related Topics IV*, vol. 1, pp. 163–175, 1988.

[13] A. O'Hagan, "Some Bayesian numerical analysis," *Bayesian Statistics 4: Proc. 4th Valencia International Meeting*, pp. 345–363, 1992.

[14] M. Fisher, C. Oates, C. Powell, and A. Teckentrup, "A locally adaptive Bayesian cubature method," *PMLR*, vol. 108, pp. 1265–1275, 2020.

[15] J. Prüher and S. Särkkä, "On the use of gradient information in Gaussian process quadratures," in *IEEE 26th Int. Workshop on Machine Learning for Signal Processing*, 2016.

[16] A. Gessner, J. Gonzalez, and M. Mahsereci, "Active multi-information source Bayesian quadrature," *PMLR*, vol. 115, pp. 712–721, 2020.

[17] T. Karvonen, S. Särkkä, and C. J. Oates, "Symmetry exploits for Bayesian cubature methods," *Stat. Comput.*, vol. 29, no. 6, pp. 1231–1248, 2019.

[18] H. R. Chai and R. Garnett, "Improving quadrature for constrained integrands," *PMLR*, vol. 89, pp. 2751–2759, 2019.

[19] R. Jagadeeswaran and F. J. Hickernell, "Fast automatic Bayesian cubature using lattice sampling," *Stat. Comput.*, vol. 29, no. 6, pp. 1215–1229, 2019.

[20] T. Karvonen and S. Särkkä, "Classical quadrature rules via Gaussian processes," *IEEE 27th Int. Workshop on Machine Learning for Signal Processing*, 2017.

[21] T. Karvonen, C. J. Oates, and S. Särkkä, "A Bayes–Sard cubature method," *NeurIPS 31*, 2018.

[22] M. Osborne, R. Garnett, S. Roberts, C. Hart, S. Aigrain, and N. Gibson, "Bayesian quadrature for ratios," *PMLR*, vol. 22, pp. 832–840, 2012.

[23] X. Xi, F.-X. Briol, and M. Girolami, "Bayesian quadrature for multiple related integrals," *PMLR*, vol. 80, pp. 5373–5382, 2018.

[24] F.-X. Briol, C. J. Oates, M. Girolami, and M. A. Osborne, "Frank-Wolfe Bayesian quadrature: probabilistic integration with theoretical guarantees," *NeurIPS 28*, 2015.

[25] T. Gunter, M. A. Osborne, R. Garnett, P. Hennig, and S. J. Roberts, "Sampling for inference in probabilistic models with fast Bayesian quadrature," *NeurIPS 27*, 2014.

[26] M. Kennedy, "Bayesian quadrature with non-normal approximating functions," *Stat. Comput.*, vol. 8, no. 4, pp. 365–375, 1998.

[27] A. O'Hagan, "Bayes–Hermite quadrature," *J. Stat. Plan. Infer.*, vol. 29, no. 3, pp. 245–260, 1991.

[28] F. Larkin, "Gaussian measure in Hilbert space and applications in numerical analysis," *Rocky Mountain J. Math.*, vol. 2, no. 3, pp. 379–422, 1972.

[29] C. E. Rasmussen and Z. Ghahramani, "Bayesian Monte Carlo," *NeurIPS 15*, pp. 505–512, 2003.

[30] F.-X. Briol, C. J. Oates, M. Girolami, M. A. Osborne, D. Sejdinovic, *et al.*, "Probabilistic integration: a role in statistical computation?," *Stat. Sci.*, vol. 34, no. 1, pp. 1–22, 2019.

[31] J. Močkus, "On Bayesian methods for seeking the extremum and their application.," *IFIP Congress*, pp. 195–200, 1977.

[32] J. Močkus, V. Tiesis, and A. Zilinskas, "The application of Bayesian methods for seeking the extremum," *Towards Global Optimization*, vol. 2, no. 2, pp. 117–129, 1978.

[33] J. Močkus, *Bayesian Approach to Global Optimization*. Springer, 1989.

[34] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," *NeurIPS 25*, 2012.

[35] P. Hennig and M. Kiefel, "Quasi-Newton methods: A new direction," *JMLR*, vol. 14, no. 1, p. 843–865, 2013.

[36] M. Mahsereci and P. Hennig, "Probabilistic line searches for stochastic optimization," *NeurIPS 28*, 2015.

[37] J. Skilling, "Bayesian solution of ordinary differential equations," *Maximum Entropy and Bayesian Methods*, pp. 23–37, 1992.

[38] P. R. Conrad, M. Girolami, S. Särkkä, A. Stuart, and K. Zygalakis, "Statistical analysis of differential equations: Introducing probability measures on numerical solutions," *Stat. Comput.*, vol. 27, no. 4, pp. 1065–1082, 2016.

[39] O. A. Chkrebtii, D. A. Campbell, B. Calderhead, and M. A. Girolami, "Bayesian solution uncertainty quantification for differential equations," *Bayesian Anal.*, vol. 11, no. 4, pp. 1239–1267, 2016.

[40] M. Schober, S. Särkkä, and P. Hennig, "A probabilistic model for the numerical solution of initial value problems," *Stat. Comput.*, vol. 29, no. 1, pp. 99–122, 2019.

[41] O. Teymur, K. Zygalakis, and B. Calderhead, "Probabilistic linear multistep methods," *NeurIPS 29*, 2016.

[42] O. Teymur, B. Calderhead, H. C. Lie, and T. J. Sullivan, "Implicit probabilistic integrators for ODEs," *NeurIPS 31*, 2018.

[43] P. Hennig and S. Hauberg, "Probabilistic solutions to differential equations and their application to Riemannian statistics," *PMLR*, vol. 33, pp. 347–355, 2013.

[44] H. Kersting and P. Hennig, "Active uncertainty calibration in Bayesian ODE solvers," *Proc. 32nd Conf. Uncertainty in Artificial Intelligence*, pp. 309–318, 2016.

[45] M. Schober, D. Duvenaud, and P. Hennig, "Probabilistic ODE solvers with Runge-Kutta means," *NeurIPS 27*, 2014.

[46] H. Owhadi, "Bayesian numerical homogenization," *Multiscale Model. Sim.*, vol. 13, no. 3, pp. 812–828, 2015.

[47] F. Tronarp, H. Kersting, S. Särkkä, and P. Hennig, "Probabilistic solutions to ordinary differential equations as nonlinear Bayesian filtering: a new perspective," *Stat. Comput.*, vol. 29, no. 6, pp. 1297–1315, 2019.

[48] J. Wang, J. Cockayne, and C. Oates, "On the Bayesian solution of differential equations," *arXiv:1805.07109*, 2018.

[49] J. Cockayne, C. Oates, T. Sullivan, and M. Girolami, "Probabilistic numerical methods for PDE-constrained Bayesian inverse problems," *AIP Conference Proceedings*, vol. 1853, p. 060001, 2017.

[50] O. A. Chkrebtii and D. A. Campbell, "Adaptive step-size selection for state-space probabilistic differential equation solvers," *Stat. Comput.*, vol. 29, no. 6, pp. 1285–1295, 2019.

[51] H. Owhadi and C. Scovel, *Operator-Adapted Wavelets, Fast Solvers, and Numerical Homogenization: From a Game Theoretic Approach to Numerical Approximation and Algorithm Design*. Cambridge University Press, 2019.

[52] A. Abdulle and G. Garegnani, "Random time step probabilistic methods for uncertainty quantification in chaotic and geometric numerical integration," *Stat. Comput.*, vol. 30, no. 4, pp. 907–932, 2020.

[53] H. Kersting, T. J. Sullivan, and P. Hennig, "Convergence rates of Gaussian ODE filters," *Stat. Comput.*, vol. 30, no. 6, pp. 1791–1816, 2020.

[54] J. Wang, J. Cockayne, O. Chkrebtii, T. J. Sullivan, and C. J. Oates, "Bayesian numerical methods for nonlinear partial differential equations," *arXiv:2104.12587*, 2021.

[55] N. Bosch, P. Hennig, and F. Tronarp, "Calibrated Adaptive Probabilistic ODE Solvers," *PMLR*, vol. 130, pp. 3466–3474, 2021.

[56] L. F. Richardson and J. A. Gaunt, "VIII. The deferred approach to the limit," *Philos. Trans. R. Soc. A*, vol. 226, no. 636–646, pp. 299–361, 1927.

[57] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[58] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007.

[59] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer, 2008.

[60] C. Runge, "Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten," *Zeitschrift für Mathematik und Physik*, vol. 46, no. 224–243, p. 20, 1901.

[61] R. Bulirsch and J. Stoer, "Fehlerabschätzungen und Extrapolation mit rationalen Funktionen bei Verfahren vom Richardson-Typus," *Numer. Math.*, vol. 6, no. 1, pp. 413–427, 1964.

[62] D. C. Joyce, "Survey of extrapolation processes in numerical analysis," *SIAM Rev.*, vol. 13, no. 4, pp. 435–490, 1971.

[63] T. A. Oliver, N. Malaya, R. Ulerich, and R. D. Moser, "Estimating uncertainties in statistics computed from direct numerical simulation," *Phys. Fluids*, vol. 26, no. 3, p. 035101, 2014.

[64] B. Chartres and R. Stepleman, "A general theory of convergence for numerical methods," *SIAM J. Numer. Anal.*, vol. 9, no. 3, pp. 476–492, 1972.

[65] J. C. de los Reyes, *Numerical PDE-Constrained Optimization*. Springer, 2015.

[66] M. L. Stein, *Interpolation of Spatial Data: Some Theory for Kriging*. Springer, 2012.

[67] C. Burg and T. Erwin, "Application of Richardson extrapolation to the numerical solution of partial differential equations," *Numerical Methods for PDEs*, vol. 25, no. 4, pp. 810–832, 2009.

[68] J. Cockayne, M. M. Graham, C. J. Oates, T. J. Sullivan, and O. Teymur, "Testing whether a learning procedure is calibrated," *arXiv:2012.12670*, 2021.

[69] T. G. Kolda and J. R. Mayo, "Shifted power method for computing tensor eigenpairs," *SIAM J. Matrix Anal. A.*, vol. 32, no. 4, pp. 1095–1124, 2011.

[70] B. W. Bader, T. G. Kolda, *et al.*, "Tensor toolbox for MATLAB, version 3.2.1," 2021.

[71] Y. Kuramoto, "Diffusion-induced chaos in reaction systems," *Prog. Theor. Phys. Supp.*, vol. 64, pp. 346–367, 1978.

[72] G. Sivashinsky, "Nonlinear analysis of hydrodynamic instability in laminar flames," *Acta Astronaut.*, vol. 4, no. 11, pp. 1177–1206, 1977.

[73] A. Kassam and L. N. Trefethen, "Fourth-order time stepping for stiff PDEs," *SIAM J. Sci. Comput.*, vol. 26, pp. 1214–1233, 2005.

[74] T. Karvonen, G. Wynne, F. Tronarp, C. J. Oates, and S. Särkkä, "Maximum likelihood estimation and uncertainty quantification for Gaussian process approximation of deterministic functions," *SIAM/ASA J. Uncertainty Quantification*, vol. 8, no. 3, pp. 926–958, 2020.

[75] H. Kersting and M. Mahsereci, "A Fourier state space model for Bayesian ODE filters," *arXiv:2007.09118*, 2020.

Supplementary Material

These appendices contain supplementary material for the paper *Black Box Probabilistic Numerics*.

A Proof of Proposition 1

The equation of the straight line through two points (x_1, y_1) and (x_2, y_2) is given by

$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1}.$$

Substituting the points $(h^\alpha, q(h))$ and $((\gamma h)^\alpha, q(\gamma h))$, and taking $x = 0$, we have

$$y = q(h) - \frac{q(\gamma h) - q(h)}{\gamma^\alpha - 1}.$$

By the assumption that q is of order α, we have the expansions $q(h) = q^* + Ch^\alpha + \mathcal{O}(h^{\alpha+1})$ and $q(\gamma h) = q^* + C(\gamma h)^\alpha + \mathcal{O}(h^{\alpha+1})$, and then by substitution and straightforward cancellation we find

$$y = q^* + \mathcal{O}(h^{\alpha+1}).$$

Therefore the y-intercept of the line is an approximation of q^* of order $\alpha + 1$.

B Gaussian Processes for BBPN

This appendix contains full details of how analytic conditioning formulae are obtained and how maximum likelihood estimates are calculated.

B.1 Conditioning Formulae

It will be convenient to introduce *lexicographic ordering*, where the indices

$$\{(i, j) : j = 1, \ldots, m_i,\ i = 1, \ldots, n\} \tag{7}$$

are ordered first by i and then, for indices with the same i, by j. Let $h_{(l)}$ and $t_{(l)}$ denote, respectively, the values of h_i and $t_{i,j}$ corresponding to the l'th ordered pair (i, j) in (7). Let $\mathbf{q}$ represent a column vector of length $m := \sum_{i=1}^{n} m_i$, with entries $\mathbf{q}_l := q(h_{(l)}, t_{(l)})$ in lexicographic order.

From (5), the prior model for Q described in Section 3.3 has covariance function

$$k_Q((h, t), (h', t')) = \sigma^2[b(t) \cdot b(t') + \rho_G k_G(t, t') + \rho_E k_E((h, t), (h', t'))], \tag{8}$$

where the additivity follows from the assumptions $Q^* \perp\!\!\!\perp E$ and $Z \perp\!\!\!\perp G$. Let K_Q be an $m \times m$ matrix and $\mathbf{k}_Q(h, t)$ be an $m \times 1$ column vector with entries of the form

$$(K_Q)_{l,l'} := k_Q((h_{(l)}, t_{(l)}), (h_{(l')}, t_{(l')}))\ , \quad (\mathbf{k}_Q(h, t))_l := k_Q((h_{(l)}, t_{(l)}), (h, t)). \tag{9}$$

Then standard Gaussian conditioning formulae (eg. Equation 2.19 in [57]) demonstrate that the conditional process $Q|D$ has mean and covariance functions

$$\mu_{Q|D}(h, t) = \mathbf{k}_Q(h, t)^\top K_Q^{-1} \mathbf{q} \tag{10}$$

$$k_{Q|D}((h, t), (h', t')) = k_Q((h, t), (h', t')) - \mathbf{k}_Q(h, t)^\top K_Q^{-1} \mathbf{k}_Q(h', t') \tag{11}$$

The mean and covariance functions of the marginal process $Q(0, \cdot)|D$ are extracted by setting h equal to 0 in Equations (10) and (11).

B.2 Proof of Higher-Order Convergence Result in Section 3.3

For a scalar quantity of interest, the full covariance function in (5) is

$$k_Q(h, h') = a_1 + a_2(hh')^\alpha \psi\left(\frac{|h - h'|}{\ell_h}\right)$$

for certain positive constants a_1 and a_2. For $\gamma \in [0,1]$, denote

$$\psi_h = \psi\left(\frac{(1-\gamma)h}{\ell_h}\right).$$

Then the conditional mean at $h=0$, given the data $D_h = \{(h, q(h)), (\gamma h, q(\gamma h))\}$, is

$$\begin{aligned}
\mathbb{E}[Q(0)|D_h] &= \begin{pmatrix} q(h) \\ q(\gamma h) \end{pmatrix}^\top \begin{pmatrix} a_1 + a_2 h^{2\alpha} & a_1 + a_2\gamma^\alpha\psi_h h^{2\alpha} \\ a_1 + a_2\gamma^\alpha\psi_h h^{2\alpha} & a_1 + a_2\gamma^{2\alpha} h^{2\alpha} \end{pmatrix}^{-1} \begin{pmatrix} a_1 \\ a_1 \end{pmatrix} \\
&= \frac{q(h)\gamma^\alpha(\gamma^\alpha - \psi_h) + q(\gamma h)(1-\gamma^\alpha\psi_h)}{a_1 a_2(1 - 2\gamma^\alpha\psi_h + \gamma^{2\alpha})h^{2\alpha} + a_2^2\gamma^{2\alpha}(1-\psi_h^2)h^{4\alpha}} a_1 a_2 h^{2\alpha} \\
&= \frac{q(h)\gamma^\alpha(\gamma^\alpha - \psi_h) + q(\gamma h)(1-\gamma^\alpha\psi_h)}{a_1(1 - 2\gamma^\alpha\psi_h + \gamma^{2\alpha}) + a_2\gamma^{2\alpha}(1-\psi_h^2)h^{2\alpha}} a_1.
\end{aligned}$$

Inserting $q(h) = q^* + Ch^\alpha + \mathcal{O}(h^{\alpha+1})$ and $q(\gamma h) = q^* + C\gamma^\alpha h^\alpha + \mathcal{O}(h^{\alpha+1})$ in the above equation yields

$$\begin{aligned}
\left|q^* - \mathbb{E}[Q(0)|D_h]\right| &= q^*\left|1 - \frac{\gamma^\alpha(\gamma^\alpha - \psi_h) + 1 - \gamma^\alpha\psi_h}{a_1(1 - 2\gamma^\alpha\psi_h + \gamma^{2\alpha}) + a_2\gamma^{2\alpha}(1-\psi_h^2)h^{2\alpha}} a_1\right| \\
&\quad + \left|\frac{\gamma^\alpha(\gamma^\alpha - \psi_h) + \gamma^\alpha(1-\gamma^\alpha\psi_h)}{a_1(1 - 2\gamma^\alpha\psi_h + \gamma^{2\alpha}) + a_2\gamma^{2\alpha}(1-\psi_h^2)h^{2\alpha}}\right| |C|\, a_1 h^\alpha \\
&\quad + \left|\frac{\gamma^\alpha(\gamma^\alpha - \psi_h) + 1 - \gamma^\alpha\psi_h}{a_1(1 - 2\gamma^\alpha\psi_h + \gamma^{2\alpha}) + a_2\gamma^{2\alpha}(1-\psi_h^2)h^{2\alpha}}\right| a_1 \mathcal{O}(h^{\alpha+1}) \\
&\leq q^*\left|\frac{a_2\gamma^{2\alpha}(1-\psi_h^2)}{a_1(1 - 2\gamma^\alpha\psi_H + \gamma^{2\alpha}) + a_2\gamma^{2\alpha}(1-\psi_h^2)h^{2\alpha}}\right| h^{2\alpha} \\
&\quad + \left|\frac{\gamma^\alpha(1+\gamma^\alpha)}{1 - 2\gamma^\alpha\psi_h + \gamma^{2\alpha}}\right| |C|\, |1 - \psi_h| h^\alpha \\
&\quad + \mathcal{O}(h^{\alpha+1}).
\end{aligned}$$

It follows from the Hölder assumption $|1 - \psi(\varepsilon)| \leq L\,\varepsilon^\beta$ that $|1 - \psi_h| = \mathcal{O}(h^\beta)$. Therefore the second term, which dominates the right-hand side, is of order $\mathcal{O}(h^{\alpha+\beta})$. This concludes the proof.

B.3 Maximum Likelihood Estimation

The parameters θ of the covariance function k_Q are estimated from data using maximum likelihood. Recall that (with α known) θ consists of the parameters σ, ρ_G, ρ_E, ℓ_h, and the $\ell_{t,i}$ for $i = 1, \ldots, p$. This parameterisation is deliberately chosen to enable the maximum likelihood estimator σ_{ML} to be computed as an explicit function of the remaining components of θ. It is convenient to express

$$k_Q((h,t),(h',t')) = \sigma^2 \overline{k}_Q((h,t),(h',t'))$$

where $\overline{k}_Q((h,t),(h',t'))$ is (8) with $\sigma = 1$. Analogously define $\overline{K}_Q$ as in (9) but with $\sigma = 1$. The log-likelihood of observing the dataset D in (2) under the model for Q defined in (3) can then be expressed as

$$\mathcal{L}(\theta) = -\frac{m}{2}\log(2\pi) - m\log\sigma - \frac{1}{2}\log|\overline{K}_Q| - \frac{1}{2\sigma^2}\mathbf{q}^\top \overline{K}_Q^{-1}\mathbf{q}, \tag{12}$$

where we note that $\overline{K}_Q$ does not depend on σ but can depend on all the other components of θ. In the case of the overall amplitude parameter σ, it is possible to obtain an analytic expression for the value σ_{ML} by differentiating and setting $\partial\mathcal{L}/\partial\sigma = 0$ [74]. This gives

$$\sigma_{\text{ML}}^2 = \frac{\mathbf{q}^\top \overline{K}_Q^{-1}\mathbf{q}}{m} \tag{13}$$

Plugging $\sigma = \sigma_{\mathrm{ML}}$ into (12) gives

$$\mathcal{L}(\theta|\sigma = \sigma_{\mathrm{ML}}) = -\frac{m}{2}\log(\mathbf{q}^\top \overline{K}_Q^{-1}\mathbf{q}) - \frac{1}{2}\log|\overline{K}_Q| + C \tag{14}$$

where C is a constant in θ. From here, we employ numerical optimisation to maximise (14) over the remaining $3 + p$ degrees of freedom in θ.

It is important to ensure that numerical optimisation is successful, otherwise conclusions provided by BBPN could be an artefact of failure of the numerical optimisation method. To this end, we undertake robust gradient-based optimisation on (14), using `MATLAB`'s packaged `fmincon` routine. This requires calculation of the gradients of (14) and explicit formulae will now be provided.

By differentiating (14) we have

$$\partial_\theta \mathcal{L}(\theta|\sigma = \sigma_{\mathrm{ML}}) = \frac{m}{2}\frac{\mathbf{q}^\top \overline{K}_Q^{-1}(\partial_\theta \overline{K}_Q)\overline{K}_Q^{-1}\mathbf{q}}{\mathbf{q}^\top \overline{K}_Q^{-1}\mathbf{q}} - \frac{1}{2}\mathrm{tr}\big(\overline{K}_Q^{-1}(\partial_\theta \overline{K}_Q)\big) \tag{15}$$

Define the matrices

$$(B)_{l,l'} := b(t_{(l)}) \cdot b(t_{(l')}) \;,\; (K_G)_{l,l'} := k_G(t_{(l)}, t_{(l')}) \;,\; (K_E)_{l,l'} := k_E((h_{(l)}, t_{(l)}), (h_{(l')}, t_{(l')})).$$

Then $\overline{K}_Q = B + \rho_G K_G + \rho_E K_E$, and it follows that

$$\begin{aligned} \partial_{\rho_G} K_Q &= K_G \;, & \partial_{\ell_h} K_Q &= \rho_E \partial_{\ell_h} K_E \;, \\ \partial_{\rho_E} K_Q &= K_E \;, & \partial_{\ell_{t,i}} K_Q &= \rho_G \partial_{\ell_{t,i}} K_G + \rho_E \partial_{\ell_t} K_E \end{aligned}$$

The low-level terms such as $\partial_{\ell_h} K_E$ can readily be computed by hand and will depend on the radial basis functions ϕ_i and ψ adopted in K_G and K_E. Note that if $\alpha > 0$ is treated as unknown and appended to the parameter vector θ, as in Section 4.2, a similar calculation can be performed to obtain the gradient with respect to α of (14).

The convergence of this gradient-based optimisation approach to a minimum of $\mathcal{L}(\theta)$ is verified empirically in Appendix C.3.2.

C Details of Empirical Assessment

This appendix contains full details for all experiments described in the main text.

C.1 Riemann Sum Illustration in Figure 1

Figure 1 considers the function $f(x) = \sin^2(4\pi x) + \exp(x) - \frac{5}{2}x^4 + \frac{1}{2}\cos(16\pi x) + \frac{1}{4}\cos(20\pi x)$. The quantity of interest q^* is the integral $\int_0^1 f(x)\,\mathrm{d}x$, which has the exact value $(\mathrm{e} - 1) \approx 1.71828$.

BBPN was applied to the method of Riemann sums. The convergence of this method is first order, and we set $\alpha = 1$ accordingly. We choose a range of step-sizes h between 0.01 an 0.08, with the Riemann sum approximations plotted in the left pane of Figure 1. Hyperparameters of the GP were set using maximum likelihood approach, as described in Appendix B.3.

C.2 Sensitivity to Prior Specification

In this section we consider the effect of varying several of the choices made during the specification of our prior model. The suitability of our non-stationary GP model is considered in Appendix C.2.2. The effect of the choice of parametrisation for h is considered in Appendix C.2.2. The choice of the kernel functions ϕ_i and ψ is discussed in Appendix C.2.3. Finally, the nature and number of the finite-dimensional basis terms b_i is discussed in Appendix C.2.4. In each case we explore the impact of these aspects of the prior specification by reproducing figures from the main text under different settings within the GP model.

C.2.1 Stationary / Non-Stationary Error Model

Since the error $E(h,t)$ is assumed to vanish in the limit $h \to 0$, and since its scale is assumed to depend on the order α of the underlying numerical method, we specified a non-stationary GP in (4). For the QR algorithm example in Section 4.2, we now contrast this with the same analysis performed with the stationary GP whose covariance function is

$$\tilde{k}_E((h,t),(h',t')) = \psi\left(|h-h'|/\ell_h\right) \cdot k_G(t,t') \tag{16}$$

i.e. setting $\alpha = 0$ in (4).

From Figure 5 (right), we see that the extrapolation is extremely poor when a stationary GP is used. Moreover, the use of a stationary GP leads in this case to over-confident predictions, with the true eigenvalues belonging outside of the $\pm 2\sigma$ credible intervals. This provides strong support for the use of the non-stationary GP that we propose in the main text.

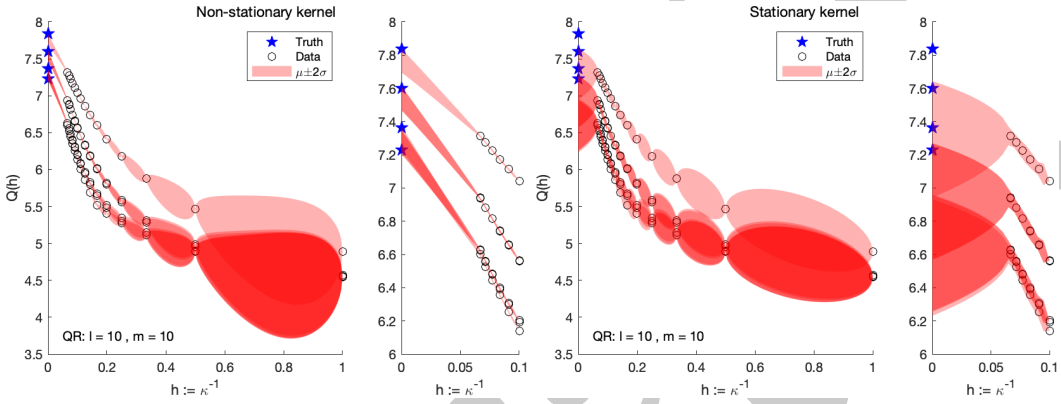

Figure 5: Comparison of stationary (4), on left, and non-stationary (16), on right, covariance functions for the QR algorithm example detailed in Section 4.2.

C.2.2 Parameterisation of h

The choice of parameterisation of h is also crucial to the operation of BBPN. While it is sometimes the case that an 'obvious' parameterisation exists (such as the step-size in a time-stepping method, where the order α specifically refers to this quantity; or the overall tolerance level of a numerical method) this is, unfortunately, not always true. If some heuristic reasoning for determining this parameterisation is not available, we recommend some prior experimentation and comparison with calibration metrics such as surprise, introduced in Section 4.

For the QR algorithm example in Section 4.2, Figure 6 shows the effect of replacing the parameterisation $h := \kappa^{-1}$ (as in Figure 3) with $h := \kappa^{-1/2}$ and $h := \kappa^{-2}$. Although BBPN continues to work, to an extent, with these alternative parametrisations, its predictive performance is somewhat diminished.

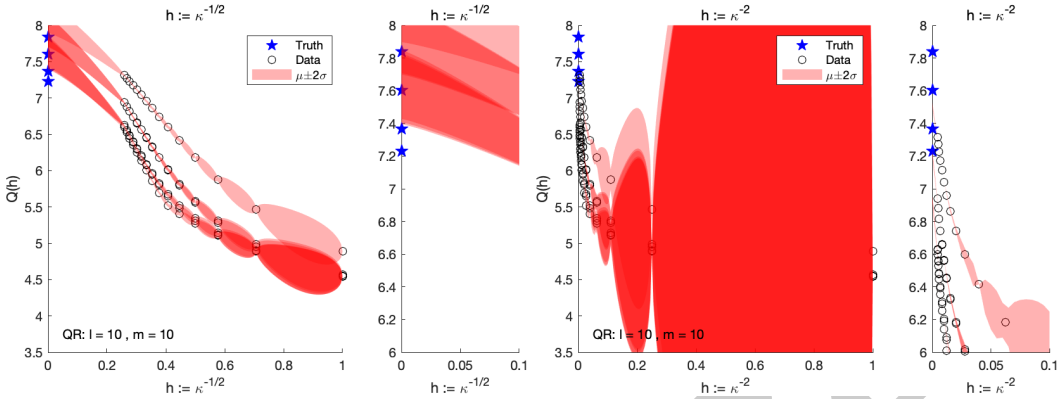

Figure 6: Comparison of different parameterisations for h relative to the number of iterations κ of the QR algorithm; $h := \kappa^{-1/2}$ (left); $h := \kappa^{-2}$ (right)

C.2.3 Choice of Radial Basis Functions ϕ_i and ψ

For all simulations in this article we specified Matérn 1/2 kernels for ϕ_i and ψ. The motivation for this, stated in the preliminary notes in Section 4, is to impose the minimal continuity assumption on q but not to assume additional levels of smoothness where this cannot be justified *a priori*.

Figure 7 shows the effect of specifying instead Matérn 3/2 or Gaussian kernels for ϕ_i and ψ in the Riemann sum test problem in Figure 1, contrasting with the Matérn 1/2 kernel used there. In all cases, the same process of gradient-based optimisation was employed to automate the setting of the kernel hyperparameters. The additional smoothness of the mean interpolant is clearly visible in the higher Matérn and Gaussian cases, but note also the difference in scale of the $\pm 2\sigma$ region. In particular, the use of smoother kernels is associated with higher confidence in the predictive output, with the Gaussian kernel producing the largest value of the surprise S^2 (though this was still within the central 95% region for a χ^2 distribution, so we do not reject the hypothesis that the BBPN output is calibrated). On balance we err on the side of caution and recommend the Matérn 1/2 kernel for applications of BBPN.

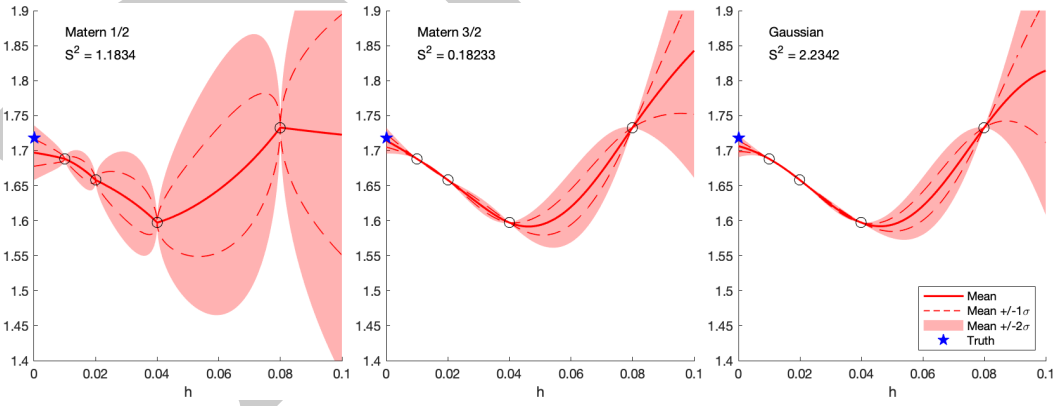

Figure 7: Comparison of different kernel types for the radial basis functions ϕ_i and ψ. Matérn 1/2 (left); Mateérn 3/2 (centre); and Gaussian (right).

C.2.4 Choice of Basis Functions b_i

In this section we demonstrate the purpose of including basis functions b_i in the model for $G(t)$. To do so, we plot the output of the BPPN procedure for the PDE example in Figure 4, since this example has non-trivial 't' domain (though the variable called t in the model definition in Section 3 is in fact called x here). The effect of including a constant basis function (*i.e.* $v = 1$ and $b_1(t) = 1$) is to allow the model a non-zero mean in t. For this example, the dynamics are mostly above the 0 level and even a simple global mean would be more likely between 1 and 2. Omitting the basis function (*i.e.* $v = 0$), as shown in the bottom pane of Figure 8, inflates the covariance to compensate for this misfit, and in this case results in an underconfident model.

In this example, it is unlikely that the additional inclusion of higher-order polynomial basis functions would be of use. Indeed our experiments showed this. However the oscilliating nature of the dynamics across the range of t suggests a Fourier basis may be an appropriate mean model. Ideas along these lines are partially explored in [75], and a fuller investigation in the context of BBPN will be the subject of future work.

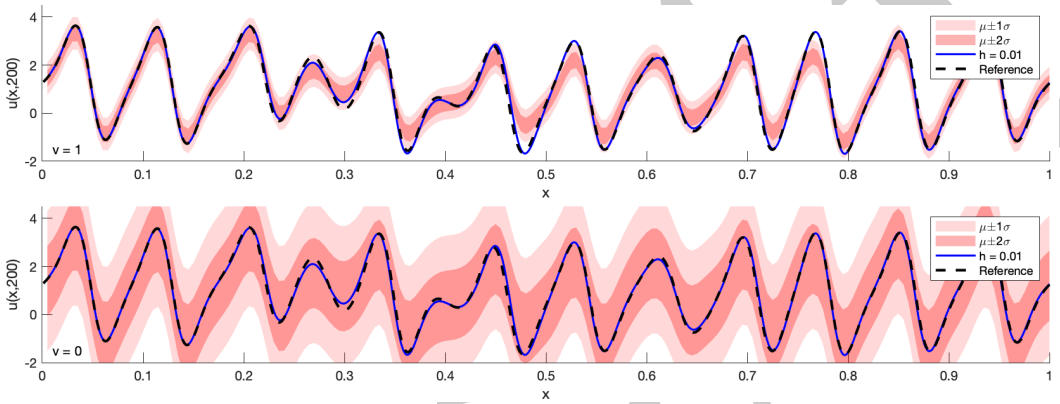

Figure 8: Comparison of the inclusion and exclusion of the first polynomial basis function (top: $v = 1$, bottom: $v = 0$) for the model in Section 4.3.

C.3 Ordinary Differential Equations

Here we provide full details for the ODE experiment in the main text. In Appendix C.3.1 we explain how all the probabilistic ODE solvers that we considered in the main text were implemented. Then, in Appendix C.3.2, we present evidence that the gradient-based optimisation approach we employed to estimate the GP hyperparameters in BBPN has successfully converged.

C.3.1 Details of Implementation

In this section we describe in detail the sources and licences of the codes, as well as the settings used, to perform the comparison experiments in Section 4.1. These codes are from different sources, span several years in release date, and are coded in different languages. They also accept inputs and give outputs in mutually inconsistent forms. This makes a 'cloned-repository' solution from which results could be reproduced automatically impractical. In the interests of maximum possible transparency we manually collect and present code sources and parameter values here in the hope that interested readers will not find it difficult to reproduce our results locally if required. Recall that our simulations consist of varying input h.

The one-step-ahead sampling model of Chkrebtii *et al.* [39] (labelled 'Chkr.' in Figure 2) was run using `MATLAB` code from `https://git.io/J33lL` with `nsolves` $= 100$, `N` $= \lceil 20/h \rceil$, `nevalpoints` $= 1001$ and the `lambda` and `alpha` hyperparameters left at their default values (which depend on $\mathbb{N}$, and therefore h). This software has no explicitly-stated licence.

The perturbed integrator approach of Conrad et al. [38] and Teymur et al. [41] (labelled 'Conr. O1' and 'Teym. O2' was run using `MATLAB` code provided to us by the authors of the latter paper and not, as far as we are aware, publicly released.

The Gaussian filtering approach of Schober et al. [40], Tronarp et al. [47] and Bosch et al. [55] (labelled 'Scho. O1', 'Tron. O2' and 'Bosch O2') was run by installing the `Python` package `probnum` and using the function `probsolve_ivp`. 'Scho. O1' uses non-adaptive step-sizes and takes `algo_order` $= 1$, and `method` $= \texttt{'EK0'}$; 'Tron. O2' uses non-adaptive step-sizes and takes `algo_order` $= 2$, and `method` $= \texttt{'EK1'}$; while 'Bosch O2' uses *adaptive* step-sizes and takes `algo_order` $= 2$, and `method` $= \texttt{'EK1'}$. In the latter case, h is taken as the relative tolerance `rtol` instead of the step-size. This software is Copyright of the ProbNum Development Team and is released under an MIT licence.

The reference solution used in calculating errors was calculated using `MATLAB`'s in-build `ode45` function with tolerances set using `odeset('RelTol',3e-14,'AbsTol',1e-20)`

It is difficult to fairly compare the wall-clock times of these codes, particularly since they are written in different languages and are therefore run in different environments. For the example simulation in Figure 2, none of the examples took more than a few seconds on a 2018 MacBook Pro, and some were virtually instant.

All publicly-available codes were downloaded or cloned on 22 April 2021.

C.3.2 Parameter Identifiability

In order to assess the robustness of our gradient-based optimisation procedure for maximum likelihood estimation, we consider again the Lotka–Volterra model. Here we will vary each parameter ℓ_t, ℓ_h, ρ_G and ρ_E in turn, holding all other parameters fixed at the values produced by the gradient-based optimisation method. The resulting plots are given in Figure 9.

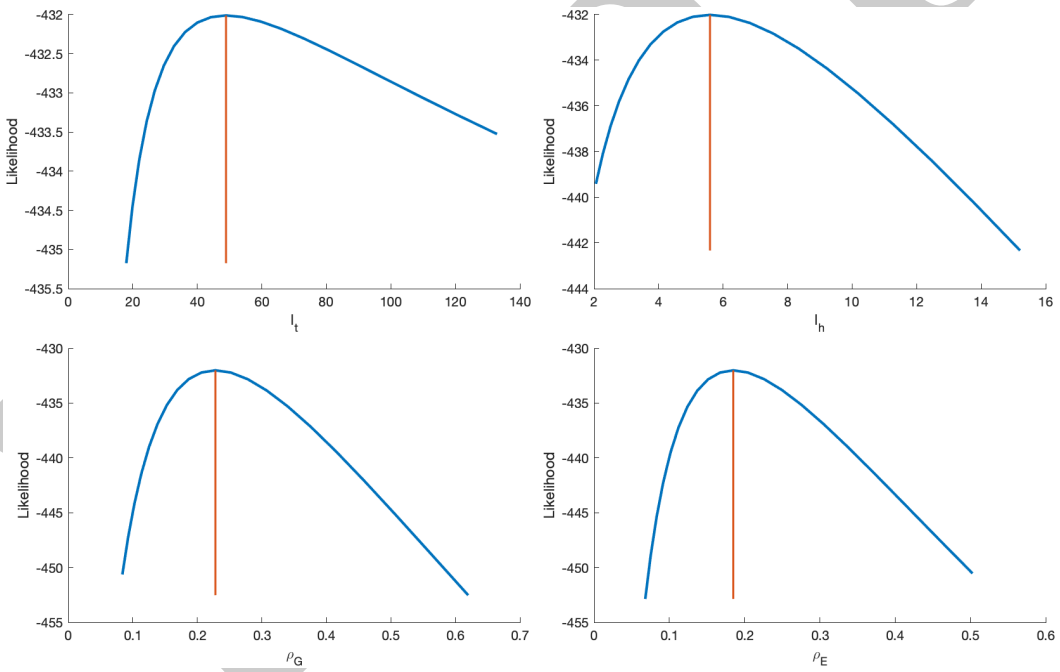

Figure 9: Variation of model likelihood in the neighourhoods of the maximum likelihood values found by `MATLAB`'s `ode45` optimiser. In each, case, the remaining parameters were fixed at the maximum likelihood value. The values determined by the optimiser are shown with a vertical orange line.

In this application, at least, we can be reasonably confident that the optimisation procedure has located a global maximiser in 4D (though, strictly speaking, we cannot confirm this from the univariate plots in Figure 9). In general, and as is common in GP modelling, model fit should always be assessed, in order to be confident in the data-driven nature of the GP output, something particularly salient in numerical applications where calibration is of paramount importance.

C.4 Eigenvalue Problems

In this section we provide certain further details for the eigenvalue problem presented in Section 4.2. We first note that the matrix A defined there can be shown to have exact eigenvalues $4-2\cos(p\pi/(l+1))-2\cos(q\pi/(m+1)); p=1,\ldots,l; q=1,\ldots,m$. The knowledge of the true values is required to facilitate the following analysis. For this section we take $l=3$ and $m=5$, as in the left-hand panes of Figure 3.

In a similar manner to Figure 2, we plot in the left-hand pane of Figure 10 the (log-) error W for several methods —the classic QR algorithm in green, then the traditional extrapolation methods of Richardson and Bulirsch–Stoer (using the data obtained in the run of the QR algorithm) in red and yellow respectively. From the definition of W, this 'combined absolute error' is formed by considering the norm of the error vector of *all* eigenvalues. (The centre pane gives the (log-) maximum relative error w, *i.e.* $\max_i[(\hat{\lambda}_i-\lambda_i)/\lambda_i]$, where $\hat{\lambda}$ is the vector of true eigenvalues, and is provided since this is a more familiar presentation of error in eigenvalue problems in numerical analysis.) It is seen that polynomial and even rational function interpolation are not robust in this setting, and give errors significantly larger than simply the most accurate single QR-produced estimate. BBPN does not suffer the same issue, possibly because the nonparametric interpolant has favourable stability properties, and it is somewhat competitive with the traditional QR algorithm, at the cost of additional computation but with the additional richness of output that a PN method provides.

The right-hand pane shows the (log-squared-) surprise of *individual* eigenvalues of the 15×15 matrix, plotted over the 95% central probability region of a χ^2_1 random variable. This shows that the predictions provided for the majority of the 15 eigenvalues are well-calibrated, but that a small number of predictions are overconfident. This is a promising early result for a problem with no previous PN method in existence, as well as one in which α has to be inferred due to the absence of a canonical parameterisation for h; see Appendix C.2.2.

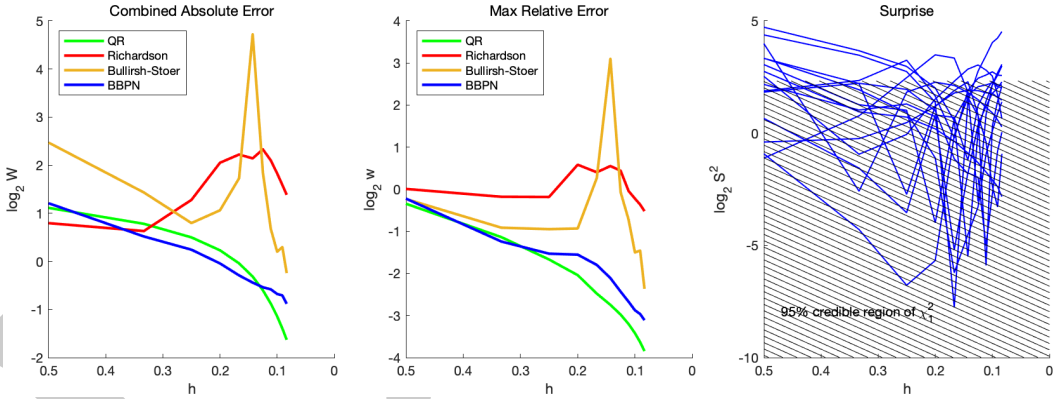

Figure 10: Eigenvalue Problems: Left: the combined absolute error W for the classic QR algorithm (green), the traditional extrapolation methods of Richardson (red) and Bulirsh–Stoer (yellow) using the data obtained in the run of the QR algorithm, and BBPN (blue). Centre: the maximum relative error w for the same methods. Right: the surprise S of *individual* eigenvalues of the 15×15 matrix, plotted over the 95% central probability region of a χ^2_1 random variable.

C.5 Kuramoto–Sivashinsky Equation (KSE)

In this section we provide further detail for the PDE problem presented in Section 4.3.

Numerical solutions to the *Kuramoto–Sivashinsky equation* (KSE) were computed on the spatial grid $x\in\{0,0.001,0.002,\ldots,1\}$ and over time segments $t_{i,j}=jh_i$ for $j\in\{0,1,\ldots,m_j\}$, where $m_j=\lfloor 200/h_i\rceil$ with $\lfloor\bullet\rceil$ denoting the nearest integer function and h_j the time-step parameter. After transformation into Fourier space, solutions were computed using a fourth-order Runge–Kutta numerical integrator ETDRK4 [73].

C.5.1 Fourier Transform to Employ the ETDRK4 Numerical Integration Scheme

We discretise the spatial domain using a Fourier spectral transformation. That is, we set

$$u(x,t) \approx \sum_{k \in \Omega_k} \tilde{u}_k(t) \exp^{ikx/L},$$

in (6), where Ω_k denotes the set of wave-numbers. Doing so returns the Fourier transformed KSE,

$$\frac{d}{dt}\tilde{u}_k(t) + \left(\frac{k^4}{L^4} - \frac{k^2}{L^2}\right)\tilde{u}_k(t) + \frac{ik}{2L}\tilde{v}_k(t) = 0, \qquad t > 0, \tag{17}$$

where

$$\tilde{v}_k(t) = \frac{1}{2\pi L}\int_{-\pi L}^{\pi L} u^2(x,t)\exp^{-ikx/L}\,dx \approx \frac{1}{N}\sum_{l=0}^{N-1} u^2(x_l,t)\exp^{-ikx_l/L}$$

with $N = 1/\delta x$ and δx denoting the spatial step-size, and on assuming that both the solution and spatial derivative are periodic in x, *i.e.*,

$$u(x,t) = u(x+2\pi L,t) \quad \text{and} \quad \frac{\partial}{\partial x}u(x,t) = \frac{\partial}{\partial x}u(x+2\pi L,t), \qquad t \geq 0,$$

for some user defined length scale L (which we take to be $L = 1/2\pi$ in our simulation). See [73] for a complete description of the fourth-order ETDRK4 scheme, as well as example `MATLAB` code used to compute solutions to (17).