# Bayes Linear Analysis for Ordinary Differential Equations

Matthew Jones[a,c,*], Michael Goldstein[a], David Randell[b], Philip Jonathan[c,d]

[a]*Department of Mathematical Sciences, Durham University, Lower Mountjoy, Stockton Road, Durham, DH1 3LE, UK*
[b]*Shell Global Solutions International B.V., Grasweg 31, 1031 HW Amsterdam, NL*
[c]*Shell Global Solutions UK, Shell Centre, York Road, London, SE1 7NA, UK*
[d]*Department of Mathematics and Statistics, Lancaster University, Lancaster, LA1 4YF, UK*

**Abstract**

Differential equation models are used in a wide variety of scientific fields to describe the behaviour of physical systems. Commonly, solutions to given systems of differential equations are not available in closed-form; in such situations, the solution to the system is generally approximated numerically. The numerical solution obtained will be systematically different from the (unknown) true solution implicitly defined by the differential equations. Even if it were known, this true solution would be an imperfect representation of the behaviour of the real physical system that it was designed to represent. A Bayesian framework is proposed which handles all sources of numerical and structural uncertainty encountered when using ordinary differential equation (ODE) models to represent real-world processes. The model is represented graphically, and the graph proves to be useful tool, both for deriving a full prior belief specification and for inferring model components given observations of the real system. A general strategy for modelling the numerical discrepancy induced through choice of a particular solver is outlined, in which the variability of the numerical discrepancy is fixed to be proportional to the length of the solver time-step and a grid-refinement strategy is used to study its structure in detail. A Bayes linear adjustment procedure is presented, which uses a junction tree derived from the originally specified directed graphical model to propagate information efficiently between model components, lessening the computational demands associated with the inference. The proposed framework is illustrated through application to two examples: a model for the trajectory of an airborne projectile moving subject to gravity and air resistance, and a model for the coupled motion of a set of ringing bells and the tower which houses them.

## 1. Introduction

An ordinary differential equation (ODE) model implicitly defines a set of functions through specification of their derivatives with respect to a single input variable. ODE models are widely used in a range of different scientific fields: for example, in classical mechanics, Newton's second law relates the forces acting on a collection of objects to their acceleration. The resulting ODE model can then be integrated in time to produce a description of the velocities and positions of the objects.

When modelling using ODEs, the system of equations must be solved in order to generate predictions. Exact solution is possible in a limited range of cases, but, in most instances, the solution cannot be obtained directly, and so must be approximated numerically. A variety of different approximation methods exists: broadly, these work by dividing the input domain into a set of intervals, and using a simple approximation to the integral over each of these. Choosing to use a particular numerical scheme to approximate the solution introduces a corresponding systematic discrepancy between the unknown, true solution and the numerical approximation, referred to as the numerical discrepancy. When a numerical solution to an ODE system is used to represent a real physical system, uncertainty about this numerical discrepancy should be accounted for; failure to do so can result in biased and over-confident estimates for any model parameters, and similar

*Corresponding Author: m.j.jones@durham.ac.uk

inaccuracies in predictions for new states (Brynjarsdottir and O'Hagan [5]).
Recently, much effort has been devoted to development of methods which account for uncertainty about the solution of a system of differential equations. Suldin [44], Larkin [28] and later Skilling [41] lay some of the foundations for the field now known as probabilistic numerics. O'Hagan [35], Diaconis [12] and O'Hagan [36] propose Gaussian processes (GPs) as a tool for numerical integration, generating uncertainty specifications for the results of intractable integrals. Oates and Sullivan [33] provide a review of the development of the field of probabilistic numerics, and highlight future research challenges. Hennig et al. [20] and Cockayne et al. [9] are other useful reviews.
Authors following Skilling [41] view solution of a differential equation as an inverse problem. Graepel [18] applies the differential operator corresponding to a given ODE to a GP and samples the forcing function at a number of locations, using the relationship between the process and its derivatives to infer the solution. Chkrebtii et al. [7] develops an approach to uncertainty quantification in ODE models in which a Gaussian process is fitted to the derivative function and integrated over each time-step, and an MCMC routine is used to estimate model parameters. Schober et al. [39] proposes probabilistic ODE solvers using Runge-Kutta means. Schober et al. [40] discuss a probabilistic model for the solution of initial value problems, seeking to leverage the best of standard numerical analysis and probabilistic algorithms for the solution of ODEs. Cockayne et al. [8] presents a probabilistic numerical method for solution of partial differential equations (PDEs) and application to PDE-constrained inverse problems, and considers optimal choice of PDE solver. Kersting and Hennig [26] discuss convergence rates for Gaussian ODE filters. Teymur et al. [45] investigate GP regression for linear multistep solution of ordinary differential equations, and provide proof of the convergence. Teymur et al. [46] consider implicit probabilistic integrators for initial value problems (IVPs) movitated by multistep methods of Conrad et al. [10] and Teymur et al. [45]. Tronarp et al. [47] considers probabilistic numerical approximations to ODEs using GP regression with nonlinear measurement functions and nonlinear Bayesian filtering.
Conrad et al. [10] represent the numerical discrepancies introduced through numerical solution using a sequence of uncorrelated, additive random variables, and demonstrate convergence to the true underlying solution as the time-step length tends to zero. Lie Cheng et al. [31] extend this convergence analysis. Abdulle and Garegnani [1] present a novel probabilistic numerical method for quantifying the uncertainty induced by the time integration of ordinary differential equations (ODEs) by introducing suitable random time steps in a classical time integrator. In the current work, we also propose a statistical model for numerical discrepancy. Mohammadi et al. [32] consider GP emulation of non-linear deterministic simulators such as climate models with multivariate time series outputs. Oates et al. [34] applies probabilistic numerics to governing physical equations of industrial equipment, facilitating improved process monitoring.
Also relevant for the remainder of this article is the large literature on Bayesian uncertainty analysis for linking the output of complex computer models with the real-world systems that they are designed to represent. Craig et al. [11] and Kennedy and O'Hagan [25] both consider aspects of this problem. In the former, a Bayes linear emulator is developed as a surrogate for a large, complex reservoir model, which takes a long time to run: this emulator is used to identify pressure inputs which could explain measurements taken from the real reservoir. In the latter, a similar probabilistic Bayesian model is developed for an atmospheric dispersion model, and this model is used to do inference for a set of model inputs, and then to generate calibrated predictions. In both cases, a structural discrepancy model is used to represent systematic differences between the computer model and the real-world system that it is designed to represent.
In this article, we develop a model which accounts for all important sources of uncertainty encountered when using a set of ODEs to represent the behaviour of a real-world system. The systematic variation of the numerical discrepancy as a function of the input state is accounted for, as is the structural discrepancy between the predictions generated from solution to the ODEs and the real-world system, along with any measurement error. Belief specification and inference for the model components can be carried out in either the fully probabilistic or Bayes linear settings; we consider both cases, favouring the computational simplicity of the Bayes linear analysis. We carry out an initial simulation for a limited number of points in the model input space, refining the solver grid and using the refined solution to quantify uncertainty in the numerical discrepancy as a function of the solver inputs. A graphical representation of the model is used, and proves useful in two respects: it is an efficient way of constructing and representing our prior beliefs about the

model components, and converting the original directed graphical model to a junction tree provides a means of efficiently adjusting beliefs about all model components, allowing for the consideration of problems with a large number of time-steps.
The novel contributions of this article are: a) An inference framework for handling both numerical and structural discrepancies together in a coherent manner; b) Incorporation of systematic variation of the numerical discrepancy as a function of state, time-step and parameter setting; c) A scalable second-order (Bayes linear) framework for analysis in combination with a junction tree representation, enabling efficient local computation and therefore allowing application to larger problems.
The remainder of the article is structured as follows. In Section 2, we outline the structure of a general numerical scheme, and investigate the structure of the corresponding numerical discrepancy. Then, in Section 3, we introduce a general, graphical framework for linking uncertainties about each of the model components. In Section 4, we introduce a modelling strategy for the numerical discrepancy, and derive a full joint prior specification for all components. In Section 5, we consider a simple example, in which we predict the trajectory of a projectile launched with an unknown initial velocity and subject to an uncertain level of air resistance, and in Section 6, we consider a more complex example, in which we account for discrepancies introduced through numerical solution of a coupled bell-tower model. Section 7 provides discussion and considers topics which could be the subject of future research in this area. Supplementary technical details are provided in the appendices.

## 2. Numerical solution of ODEs

We consider the following initial value problem; we specify that the functions $u = \{u_1, \ldots, u_{n_u}\}$, which depend on scalar input $t$ (usually time), satisfy

$$\begin{aligned} \frac{d}{dt}\big(u_i\left(t\right)\big) &= f_i\left(u\left(t\right), t, \xi\left(t\right)\right) \ , \\ u_i\left(t_0\right) &= u_i(t_0) \ , \end{aligned}$$

where $f = \{f_1, \ldots, f_{n_u}\}$ is a set of functions which govern the first-order derivatives of the solution surface, $\xi\left(t\right) = \{\xi_1\left(t\right), \ldots, \xi_{n_\xi}\left(t\right)\}$ is a set of parameters which determine the behaviour of $f$, and $u(t_0) = \{u_1(t_0), \ldots, u_{n_u}(t_0)\}$ is the initial state. The general solution to this system can be written as

$$u_i\left(t\right) = u_i\left(t_0\right) + \int_{t_0}^{t} f_i\left(u\left(s\right), s, \xi\left(s\right)\right) ds \ . \tag{1}$$

In general, this integral cannot be evaluated directly, though, in some cases, an algebraic solution can be found by alternative means (e.g. separation of variables, integrating factors; see e.g. Iserles [22]). In cases where an algebraic solution cannot be found, numerical methods can be used to approximate the solution surface. In Section 2.1, we consider the general structure of numerical approximation techniques for systems of ODEs, and in Section 2.2, we consider the structure of the discrepancy between such a solver and the true solution to the system.

### *2.1. General one-step numerical scheme*

Under a general numerical scheme, the solution is approximated at a set of fixed input settings $t_0 \leq t_1 \leq \ldots \leq t_{n_t}$; the approximation at point $t_k$ is denoted by $\hat{u}\left(t_k\right)$. Fixing $\hat{u}\left(t_0\right) = u(t_0)$, the approximation at time point $t_{k+1}$, $k = 0, 1, \ldots, (n_t - 1)$ is determined by feeding the approximation at time $t_k$ back into a suitably-chosen numerical solution function

$$\hat{u}_i\left(t_{k+1}\right) = \phi_i\left(\hat{u}\left(t_k\right), t_{k+1}, t_k, \xi\right) \ .$$

Many different choices for the function $\phi$ are possible, corresponding to different types of numerical scheme. Two of the most common choices, the Euler and Runge–Kutta schemes, are discussed below.

*Euler scheme.* The simplest approximation strategy is the Euler scheme; the integral (1) is approximated by assuming that the derivative is constant across the time-step $[t_k, t_{k+1}]$

$$\phi_i\left(\hat{u}\left(t_k\right), t_{k+1}, t_k, \xi\right) = \hat{u}\left(t_k\right) + h_k f_i\left(\hat{u}\left(t_k\right), t_k, \xi\right) \ , \tag{2}$$

where $h_k = (t_{k+1} - t_k)$. This scheme is equivalent to the first order Taylor expansion of the solution around the point $t_k$, evaluated at $\{\hat{u}\left(t_k\right), t_{k+1}\}$, and therefore has a local truncation error of order $(h_k)^2$. Euler schemes are easy to implement, and for sufficiently small time-steps can provide a good approximation to the true solution. Euler schemes will we used for the development of the modelling framework in Section 3.

*Runge–Kutta scheme.* A Runge–Kutta scheme of order $q$ gives the following numerical solution function

$$\phi_i\left(\hat{u}\left(t_k\right), t_{k+1}, t_k, \xi\right) = \hat{u}_i\left(t_k\right) + h_k \sum_{l=1}^{q} b_l w_{il} \ ,$$

where $w_{i1} = f_i\left(\hat{u}\left(t_k\right), t_k, \xi\right)$, and $w_{i2}, \ldots, w_{iq}$ are computed as

$$w_{il} = f_i\left(\left(\hat{u}\left(t\right) + h_k \sum_{p=1}^{l-1} a_{lp} w_p\right), \left(t_k + c_l h_k\right), \xi\right) \ ,$$

for $l = 2, 3, \ldots, q$, (Butcher [6]), where $w_p = (w_{1p}, w_{2p}, \ldots, w_{n_u p})^{\mathrm{T}}$ is a vector of coefficients. Different choices of the coefficients $\{a_{lp}, b_l, c_l\}$ give rise to different schemes. The $4^{\text{th}}$ order Runge–Kutta scheme (known as the RK4 scheme) is a popular choice; this has a local truncation error of order $(h_k)^5$.

### *2.2. Numerical discrepancy*

Suppose that the solution $u\left(t_k\right)$ at time $t_k$ is known, and that the system is evolved numerically to time $t_{k+1}$; the discrepancy between the true solution and the numerical approximation at $t_{k+1}$ (often referred to as the local truncation error) is denoted by

$$\eta_i\left(u\left(t_k\right), t_{k+1}, t_k, \xi\right) = u_i\left(t_{k+1}\right) - \hat{u}_i\left(u\left(t_k\right), t_{k+1}, t_k, \xi\right) \ . \tag{3}$$

While $\eta$ is unknown (if $\eta$ were known, $u$ would be known), a lot can be learnt about its structure from further analysis. Taylor expanding the solution around $t_k$ gives

$$\begin{aligned} u_i\left(t_{k+1}\right) &= u_i\left(t_k\right) + (t_{k+1} - t_k)\frac{du}{dt}\left(t_k\right) + \frac{1}{2}(t_{k+1} - t_k)^2 \frac{d^2u}{dt^2}\left(t_k\right) + O((t_{k+1} - t_k)^3) \\ &= u_i\left(t_k\right) + h_k \frac{du}{dt}\left(t_k\right) + \frac{1}{2} h_k^2 \frac{d^2u}{dt^2}\left(t_k\right) + O(h_k^3) \ . \end{aligned}$$

Choosing to approximate the solution at $t_{k+1}$ using an Euler scheme (2) therefore gives the following numerical discrepancy

$$\eta_i\left(u\left(t_k\right), t_{k+1}, t_k, \xi\right) = \frac{1}{2} h_k^2 \frac{d^2u}{dt^2}\left(t_k\right) + \frac{1}{6} h_k^3 \frac{d^3u}{dt^3}\left(t_k\right) + O(h_k^4) \ .$$

This is the sum of the higher-order terms in the Taylor expansion of the solution around $t_k$. Since most numerical schemes are based around series expansions of the solution evaluated at successive time points, similar analysis is possible in many other cases.

It is clear that for general schemes, $\eta$ is a systematic function of the input solution state $u\left(t_k\right)$, parameters $\xi$ and time-step $h_k = (t_{k+1} - t_k)$. Additionally, for small $h_k$, the behaviour of $\eta$ will be mainly determined by the leading-order components of the remaining Taylor expansion components of the solution. This information can be built into any prior specification of our uncertainty about the numerical discrepancy for a particular scheme. In Section 4.1, we develop a suitable form for the numerical discrepancy model using this structural knowledge.

## 3. Building a model

When using an ODE model for the behaviour of a real physical system, discrepancies between model predictions and measurements of the system arise in two main ways:

- the use of a numerical solver to approximate the solution of the ODE model induces a discrepancy between the solution that was implicitly defined and the approximation that can actually be evaluated (see e.g. Conrad et al. [10]);
- even if the true solution to the ODE model were known, this would still generally give an inadequate representation of the real world due to e.g. inadequate description of the physics driving the process, or lack of knowledge about the settings of any model parameters which give an acceptable match to the real world (see, e.g., Goldstein and Rougier [15]).

In this section, we develop a model which represents our uncertainties about these discrepancies for a given ODE model. First, we link the true (unknown) solution to the ODE model to the system that it represents through means of a 'best input' assumption and a structural discrepancy model (Section 3.1.1), and we represent this unknown true solution as the sum of a numerical approximation and an unknown numerical discrepancy component (Section 3.1.2). We then develop a directed graphical representation of the full model, and describe how this representation can be used to efficiently store the prior uncertainty specification for the model components (Section 3.2.1). Finally, we describe how this directed graphical model can be converted into a type of undirected representation known as a junction tree, and how this representation can be exploited to design efficient inference algorithms (Section 3.2.2).
The model presented here is quite a complex one, and the directed graphical model gives a convenient way of simplifying the uncertainty specification which must be made. Direct inference on the directed graphical model is hard, and so conversion to a junction tree representation provides a mechanism for carrying out efficient inference given observation of some of the model components. Section 5 shows a simple example which describes the complete approach taken; additionally, the code used to run the analysis described in Section 5 is available on Github (Jones [23]).

### *3.1. Model structure*

#### *3.1.1. Relating the solution to the system*

The set of real-world quantities represented by the model is denoted by $y(t) = \{y_1(t), \ldots, y_{n_u}(t)\}$. We assume that noise-corrupted measurements of these quantities are available at times $\{t_1, \ldots, t_{n_z}\}$, denoting the set of measurements at time $t_k$ by $z_k = \{z_{1k}, \ldots, z_{n_u k}\}$, where

$$z_{ik} = y_i(t_k) + \epsilon_{ik} \ , \tag{4}$$

and $\epsilon_{ik}$ is an additive measurement error term, which is assumed to be uncorrelated with $y$. When linking the ODE solution $u$ with the real-world quantities $y$, we adopt the strategy outlined in e.g. Goldstein and Rougier [15] and Vernon et al. [49], assuming that there is a 'best input' setting $\xi^*$ such that if we were to evaluate $u$ at this setting, we would obtain all of the information available from the model about the real world. At this point, we relate the solution to the system as

$$y_i(t_k) = u_i(t_k, \xi^*) + \delta_i(t_k) \ , \tag{5}$$

where $\delta = \{\delta_1, \ldots, \delta_{n_u}\}$ is the discrepancy between the solution evaluated at the parameter setting $\xi^*$ and the real world-quantities $y$. $\delta$ is assumed to be uncorrelated with $\{u, \xi^*\}$.
In a Bayesian uncertainty analysis, we make a joint prior uncertainty specification for all components of the expressions (4) and (5), and then use data collected on the system to update these prior beliefs. In Section 3.2, we represent the structure of our model for $u$ through a graph, before using this graph to make a prior specification for the solution.

*3.1.2. Relating the numerical approximation to the solution*

With the numerical discrepancy defined as in (3), our representation of the solution at time $t_{k+1}$ is

$$u_i\left(t_{k+1}, \xi^*\right) = \hat{u}_i\left(u\left(t_k\right), t_k, t_{k+1}, \xi^*\right) + \eta_i\left(u\left(t_k\right), t_k, t_{k+1}, \xi^*\right) \ . \tag{6}$$

The numerical solution $\hat{u}$ is a known function of the solution at time $t_k$, the parameters $\xi^*$ and the initial and final times. As discussed in Section 2.2, the numerical discrepancy $\eta$ is an unknown function of all these components which exhibits strongly systematic behaviour across its input space.

*3.2. Graphical representation*

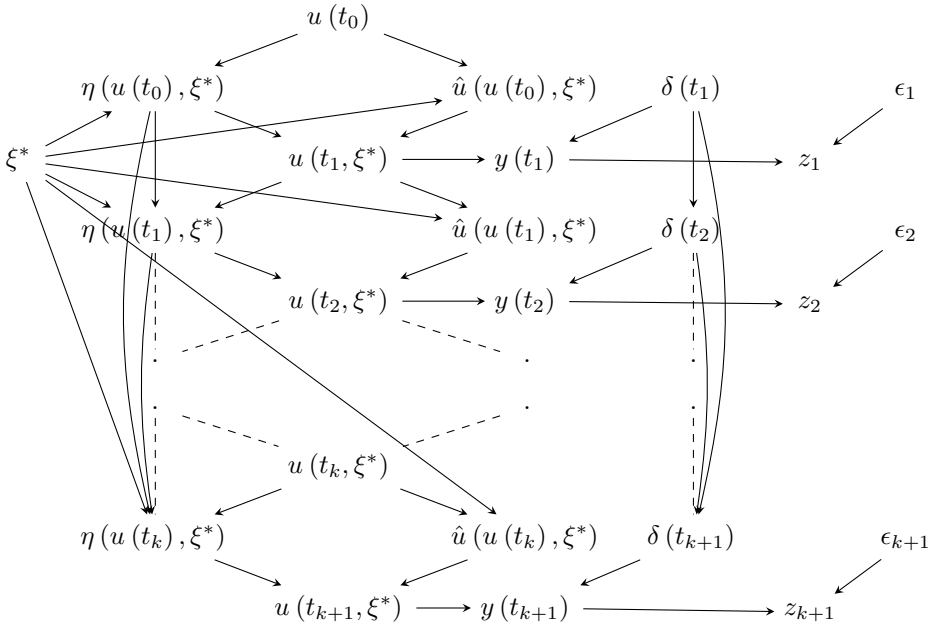



Figure 1: Directed graphical representation of the components of the model, and their relationship to the system values $y$ and the observed data $z$. The dashed lines correspond to edges to model components not explicitly displayed on the diagram.

Graphs provide a useful visual representation of the structure of a statistical model, and can be used to facilitate computationally efficient inference through local computation. Graphical models are useful for both fully-probabilistic (Lauritzen [29]) and Bayes linear (Goldstein and Wilkinson [16]) inference. In a fully probabilistic inference, a full joint prior distribution is specified for all model components; application of Bayes' theorem upon observation of a subset of some of the components leads to a corresponding joint posterior distribution. In a Bayes linear (or second-order) inference, only a second-order specification, consisting of means, variances and covariances for all components, is required; beliefs are then updated in light of observations through Bayes linear adjustment. An introduction to Bayes linear methods is provided in Appendix A, and an introduction to Bayes linear graphical models is provided in Appendix B.1: we recommend the unfamiliar reader to consult these appendices before proceeding. A detailed introduction to the Bayes linear framework is provided by Goldstein and Wooff [17].

The model specification is first made through a directed graphical model (Section 3.2.1), before being converted to a junction tree representation (Section 3.2.2). The advantage of converting the model to the junction tree representation comes from the fact that an efficient procedure is available for Bayes linear belief updating on a junction tree: this procedure is described in Section 4.3, and more detail is given in Appendix B.3. The whole process of starting from a belief specification on a directed graph, converting the

directed representation to a junction tree, and iteratively computing adjusted moments for the individual components is illustrated for a simple model in Appendix B.4.

*3.2.1. Directed graphical model*

Combining the specifications made in equations (4), (5) and (6) suggests the graphical representation shown in Figure 1. This is a directed acyclic graph (DAG), and when supplemented with a consistent node ordering and an appropriate prior belief specification, gives rise to a directed graphical model. This directed graphical model can be used to make a joint prior belief specification for all model components; this prior specification can either be fully probabilistic or second-order (Bayes linear). Using $\{G_1, \ldots, G_{n_G}\}$ to denote the (consistently ordered) full set of nodes, in the fully probabilistic case, our full joint prior specification would consist of probability distributions $p\left(G_k | \mathrm{Pa}\left(G_k\right)\right)$ for each node $G_k$ conditional on its parent nodes $\mathrm{Pa}\left(G_k\right)$. The conditional independence structure of the DAG then determines our full joint prior specification over all components. In the second-order case, our prior specification consists of expectations $\mathrm{E}\left[G_k\right]$ and variances $\mathrm{Var}\left[G_k\right]$ for each node $G_k$, and covariances $\mathrm{Cov}\left[G_k, G_l\right]$ for all pairs of nodes $\{G_k, G_l\}$ connected by an edge; in this case, the belief separation structure implied by the DAG determines the full prior covariance structure for the whole node collection.
In the remainder of this article, we focus on the second-order case, because of its attractive computational properties. In Section 4.1 and Appendix C, we describe a procedure for generating a second-order prior specification on the DAG in Figure 1 for general choices of numerical scheme and numerical discrepancy model. In Section 3.2.2, we describe the conversion of the DAG into a junction tree, which enables efficient inference in the second-order case.

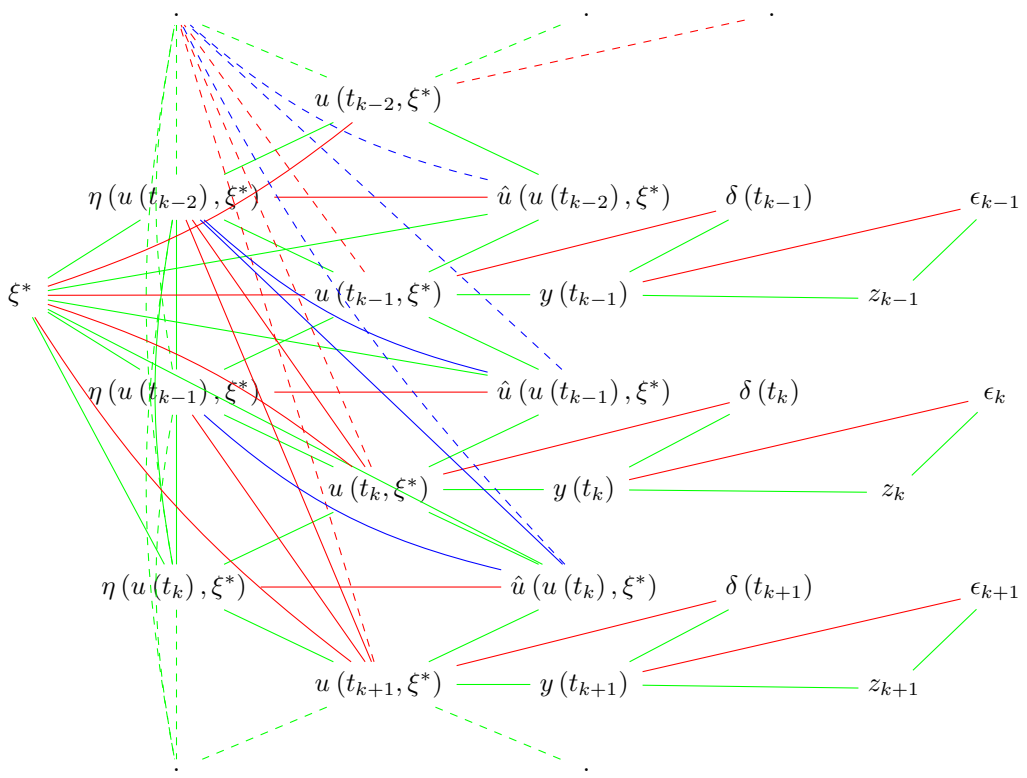



Figure 2: Triangulated moral graph corresponding to a modified version of the DAG shown in Figure 1, in which links between all $\{\delta\left(t_k\right)\}$ components have been dropped. The colour of an edge indicates its origin: green edges were present in the original DAG 1, red edges were introduced through moralization, and blue edges were introduced through triangulating the graph (see Appendix B.2).

*3.2.2. Junction tree*

Appendix B.3 describes a procedure for generating a junction tree from a general directed graphical model. A junction tree is an undirected graphical model which is formed from the cliques of the triangulated moral graph of the original DAG; by construction, the conditional independence structure (or belief separation structure in the second-order case) implied by the junction tree is valid for the original directed graph. Its tree structure means that information obtained by observing the values of certain nodes can be passed easily between the cliques. This property of junction trees allows for the implementation of efficient inference procedures, particularly in the second-order case.
Any junction tree corresponding to the DAG in Figure 1 would be complex, and would be difficult to identify manually. We therefore illustrate the procedure from Appendix B.3 for converting a DAG to a junction tree using a slightly modified version of the DAG in Figure 1, in which all edges between the structural discrepancy components $\{\delta(t_k)\}$ have been eliminated.

*1. Construct the moral graph.* The moral graph is obtained from the modified form of the DAG in Figure 1 by performing the following steps:

- Retaining all edges from the original graph, dropping the associated directions.
  These edges are shown in green in Figure 2.
- Introducing an undirected edge between all pairs of unconnected nodes with a common child in the original DAG:
  - Each numerical discrepancy $\eta(u(t_{k-1}), \xi^*)$ is joined to the corresponding numerical solution $\hat{u}(u(t_{k-1}), \xi^*)$;
  - Each solution component $u(t_k, \xi^*)$ is joined to the numerical discrepancy components $\eta(t_{k-2}, \xi^*), \eta(t_{k-3}, \xi^*), \ldots, \eta(t_1, \xi^*)$ at all earlier times;
  - Each solution component $u(t_k, \xi^*)$ is joined to the parameters $\xi^*$;
  - Each solution component $u(t_k, \xi^*)$ is joined to the discrepancy $\delta(t_k)$ at the corresponding time step;
  - Each measurement error $k$ is joined to the system value $y(t_k)$ at the corresponding time-step.

  These edges are shown in red in Figure 2.

*2. Triangulate the graph.* Inspecting the moral graph from step 1, we identify the need for additional edges to triangulate the graph. Attempting the maximum cardinality search procedure on the moral graph from step 1 (starting from $\xi^*$; Appendix B.3), we see that when labelling the nodes $u(t_k)$ for $k = 2, 3, \ldots, n_t$, the condition that the all the previously labelled neighbours form a complete graph is not satisfied since $\hat{u}(u(t_{k-1}), \xi^*)$ is not connected to any of the $\eta$ variables at times before $t_k$. Therefore, we triangulate the graph by adding edges between $\hat{u}(u(t_{k-1}), \xi^*)$ and $\eta(u(t_l), \xi^*)$ for all $l = 0, \ldots, (k-2)$.
These additional edges are shown in blue in Figure 2.

*3. Perform a maximum cardinality search.* We assign labels to the nodes using the maximum cardinality search procedure outlined in Appendix B.3, confirming as we go that the graph is triangulated. The labels assigned to the nodes in this step will determine the edge structure of the junction tree in step 5.

*4. Identify and label the cliques.* The undirected graph in Figure 2 is a triangulated moral graph corresponding to the modified DAG from Figure 1. We identify the full set of cliques corresponding to this triangulated moral graph: in this case, this set of cliques can be written as $\{Q_1(t_k), Q_2(t_k), Q_3(t_k), Q_4(t_k)\}_{k=1}^{n_t}$, where:

- $Q_1(t_k) = \{\xi^*, \eta(u(t_0)\,\xi^*), \eta(u(t_1), \xi^*), \ldots, \eta(u(t_k), \xi^*), u(t_{k-1}, \xi^*), \hat{u}(u(t_{k-1}), \xi^*)\}$;
- $Q_2(t_k) = \{\xi^*, \eta(u(t_0)\,\xi^*), \eta(u(t_1), \xi^*), \ldots, \eta(u(t_k), \xi^*), \hat{u}(u(t_{k-1}), \xi^*), u(t_k, \xi^*)\}$;
- $Q_3(t_k) = \{u(t_k, \xi^*), y(t_k), \delta(t_k)\}$;

- $Q_4(t_k) = \{y(t_k), z_k, \epsilon_k\}$.

The ordering of the cliques implied by the node ordering obtained from the maximum cardinality search is

$$\{Q_1(t_1), Q_2(t_1), \ldots, Q_1(t_{n_t}), Q_2(t_{n_t}), Q_3(t_1), Q_4(t_1), \ldots, Q_3(t_{n_t}), Q_4(t_{n_t})\}.$$

*5. Create the junction tree.* Having identified the cliques, we use them to construct the junction tree. Using the clique ordering identified by the maximum cardinality search, we join each clique to one of the preceding cliques which contains the intersection between this clique and *all* preceding cliques. For example, clique $Q_3(t_k)$ consists of $\{u(t_k, \xi^*), y(t_k), \delta(t_k)\}$. $y(t_k)$ and $\delta(t_k)$ do not feature in any of the preceding cliques, and therefore do not form part of the intersection between $Q_3(t_k)$ and its predecessors. Therefore, the intersection between $Q_3(t_k)$ and all preceding cliques must be $u(t_k, \xi^*)$, which occurs in $Q_2(t_k)$ and $Q_1(t_{k+1})$ (for $k < n_t$). We can therefore choose either to link $Q_3(t_k)$ to $Q_2(t_k)$ or to $Q_1(t_{k+1})$; we are free to choose either option, and we choose the former. Following this procedure for all cliques produces the junction tree shown in Figure 3.

The cliques $Q_1(t_k)$ and $Q_2(t_k)$ for each time step $t_k$ are linked together to form a chain in the final junction tree (Figure 3), with the cliques $Q_3(t_k)$ and $Q_4(t_k)$ for each time step linked together to form a series of branches from this main chain (attached to it at $Q_2(t_k)$). The system measurement $z_k$ is only found in the clique $Q_4(t_k)$, which is at the end of the branch of the tree corresponding to time $t_k$. This means that, when using the junction tree to update beliefs given observation of $z_k$, information is propagated along the branch corresponding to time $t_k$ (through $Q_3(t_k)$ and $Q_2(t_k)$), then in both directions along the main clique chain (backwards through $Q_1(t_k)$, $Q_2(t_{k-1})$, ..., and forwards through $Q_1(t_{k+1})$, $Q_2(t_{k+1})$), and along the branches corresponding to the other time steps $t_l$, $l \neq k$. Further detail relating to this procedure in the second-order case is provided in Appendix B.3 and in Section 4.3.
In a complex graph, it will be challenging in general to identify a useful junction tree manually. In such problems (e.g. similar to the DAG in Figure 1, with links between both numerical discrepancy components $\{\eta(t_k)\}$ and structural discrepancy components $\{\delta(t_k)\}$ across time-steps), a junction tree can be identified algorithmically. The original DAG is represented as an adjacency matrix, and moralization is relatively simple to implement. Various different algorithms for triangulating graphs exist; a summary is provided by Heggernes [19]. In the examples in Sections 5 and 6, the LB-triang algorithm is adopted. From the resulting triangulated moral graph, the cliques are identified using the Bron-Kerbosch algorithm (Bron and Kerbosch [4]). Once the cliques have been listed, the junction tree structure is relatively easy to identify.

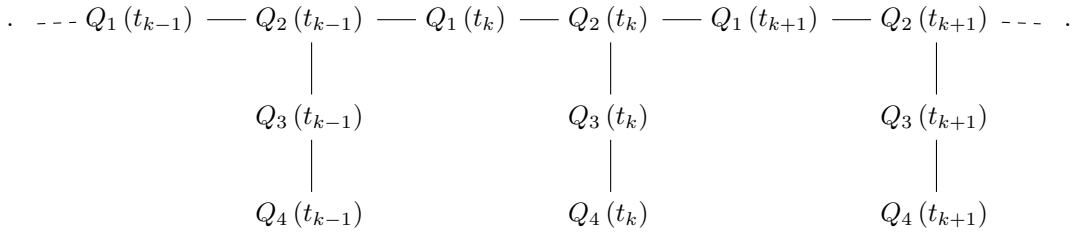



Figure 3: Segment of a junction tree corresponding to the triangulated moral graph in Figure 2. The set of model components contained in each of the cliques $Q_1(t_k)$ to $Q_4(t_k)$ at each time $t_k$ is listed in Section 3.2.2.

## 4. Bayes linear analysis

We outline a general procedure for making a second-order prior specification corresponding to the DAG in Figure 1. In Section 4.1, we discuss the specification of a prior model for the numerical discrepancy components $\eta$, using a grid refinement strategy to investigate its behaviour. In Section 4.2, we combine this prior model for $\eta$ with our prior beliefs about all other components to construct a full joint prior specification over the DAG in Figure 1.

*4.1. Numerical discrepancy model*

Using $\psi(t_k) = \{u(t_{k-1}), t_{k-1}, t_k, \xi\}$ as an abbreviation for the solver inputs at time $t_k$, our model for the numerical discrepancy at time $t_k$ has the common regression plus residual form

$$\eta_i(\psi(t_k)) = \sum_j \beta_{ij} b_j(\psi(t_k)) + r_i(\psi(t_k)) \ , \tag{7}$$

where $b(\psi(t_k)) = \{b_1(\psi(t_k)), \ldots, b_{n_b}(\psi(t_k))\}$ is a set of known basis functions, $\beta = \{\beta_{ij}\}$ $(i = 1, \ldots, n_u,$ $j = 1, \ldots, n_b)$ is a corresponding set of unknown coefficients, and $r(\psi(t_k)) = \{r_1(\psi(t_k)), \ldots, r_{n_u}(\psi(t_k))\}$ is a set of unknown residual components. We assume that $\beta$ and $r$ are a priori uncorrelated. Where possible, the basis functions $\{b_j\}$ should be chosen to reflect important aspects of the leading-order structure of the numerical discrepancy; for example, where an Euler scheme is chosen for the numerical ODE solution, higher-order terms in the Taylor expansion of the solution would generally work well.

*4.1.1. Generating approximate numerical discrepancy data*

As discussed in Section 2.2, the discrepancy between the unknown true solution and the numerical approximation obtained is completely determined by the choice of solver. While we cannot evaluate the numerical discrepancy directly (if we could, it would imply that we knew the true solution), we can perform a prior analysis to investigate its structure further. By choosing a set of initial points which is representative of solution behaviour over the time domain and parameter input space (e.g. using a Latin hypercube design), refining the solver grid, and then representing the numerical discrepancy on the refined grid as a simple random process which is uncorrelated between time steps, we can obtain a set of more accurate estimates of $\eta$, and use these to update the components of the model (7).

To emphasise: since we cannot evaluate $\eta(\psi(t))$ directly, we use differences between solver runs on the original time grid (i.e. the grid used for the analysis outlined in Section 3), and solver runs on a refined grid (e.g. with 100 time-steps on the refined grid for every individual step on the original grid, such that we expect the magnitude of the numerical discrepancy to be considerably smaller) to approximate the true numerical discrepancy corresponding to one time step on the original time grid. On the refined grid, we assume that the numerical discrepancy is zero-mean Gaussian and uncorrelated between solver time-steps. Repeating this simulation a large number of times allows us to generate numerical estimates for the mean and covariance of the numerical discrepancy, which we then use for parameter estimation in the model (7). The procedure used is as follows: we denote the estimates for the discrepancy mean and covariance by $\tilde{\mathrm{E}}[\eta_i(\psi(t))]$ and $\widetilde{\mathrm{Cov}}[\eta_i(\psi(t)), \eta_j(\psi(t))]$, with the tilde indicating that these estimates were calculated using the numerical procedure. Specifically, we perform a single time step on the original grid, and $n^*$ time steps on a refined grid corresponding to the same time interval $h$. On the refined grid, we assume that the numerical discrepancy components $\{\eta_i\}$ are uncorrelated with each other and across time-steps, with mean zero and variance $\{(\sigma_{\eta_i} s(h))^2\}$, for some function $s(.)$ of the time-step $h$. Then for a range of different values of $\psi(t) = \{u(t), t, t+h, \xi\}$:

1. We first evolve the numerical solver to some time point $t$ at the original temporal resolution (including uncorrelated, stochastic $\eta$, as described in step 1), obtaining an initial state $u(t)$.
2. We evolve from time $t$ to time $(t+h)$ on the original time-scale, obtaining $\hat{u}(\psi(t))$.
3. We also evolve from $t$ to $(t+h)$ using the refined temporal resolution: we choose $n^*$ knots $\{\tau_0, \tau_1, \ldots, \tau_{n^*}\}$ with $\tau_0 = t$ and $\tau_{n^*} = t+h$, obtaining $u^*(\psi(t))$.
4. We repeat step 3 a large number of times for different randomly-sampled discrepancies on the refined grid in order to estimate $\{\tilde{\mathrm{E}}[u_i^*(\psi(t))]\}$ and $\{\widetilde{\mathrm{Cov}}[u_i^*(\psi(t)), u_j^*(\psi(t))]\}$.
5. Using these empirically-generated expectations and covariances, we characterise the moments of the numerical discrepancy for a given input setting $\psi(t)$ as

$$\begin{aligned} \tilde{\mathrm{E}}[\eta_i(\psi(t))] &= \tilde{\mathrm{E}}[u_i^*(\psi(t))] - \hat{u}_i(\psi(t)) \ , \\ \widetilde{\mathrm{Cov}}[\eta_i(\psi(t)), \eta_j(\psi(t))] &= \widetilde{\mathrm{Cov}}[u_i^*(\psi(t)), u_j^*(\psi(t))] \ . \end{aligned}$$

*4.1.2. Estimating the parameters of the numerical discrepancy model*

The moments $\{\tilde{\mathrm{E}}[\eta_i(\psi(t))]\}$ and $\{\widetilde{\mathrm{Cov}}[\eta_i(\psi(t)), \eta_j(\psi(t))]\}$ generated using the procedure in Section 4.1.1 are used for an initial Bayes linear adjustment of the model (7); since $\eta$ cannot be evaluated directly, the expectations $\{\tilde{\mathrm{E}}[\eta_i(\psi(t))]\}$ at particular input settings are used as the data for the adjustment, with the covariances $\{\widetilde{\mathrm{Cov}}[\eta_i(\psi(t)), \eta_j(\psi(t))]\}$ being treated as the covariances of the measurement error. The resulting adjusted moments are then used as the prior specification for the numerical discrepancy for the purposes of characterising the full joint prior specification (see Section 4.2). Combining the numerical ODE solver with this prior analysis for the numerical discrepancy generates a more accurate mean prediction and a tighter uncertainty range for the true solution at any given time step, which in turn reduces uncertainty about the starting point for the next evaluation of the numerical solver.
This approach is adopted in the examples in Sections 5 and 6, where the basis functions are simply the leading-order terms of the approximation error induced by truncating the Taylor series. The fitting of this initial model for the numerical discrepancy is described in detail for the projectile example (Section 5) in Appendix D.

*4.2. Second-order prior specification*

We describe a procedure for computing a full, second-order prior specification for all model components from the prior specification for the initial state, the parameters, and the numerical discrepancy function. The individual steps of the procedure are summarised below, with further details in Appendix C.

*Prior uncertainty specification for initial state and numerical discrepancy model.* We begin by making a prior specification for only those components that are not determined by the solution trajectory. First, we specify expectations $\{\mathrm{E}[u_i(t_0)]\}$ and covariances $\mathrm{Cov}[u_i(t_0), u_j(t_0)]$ for the components of the initial state, and expectations $\{\mathrm{E}[\xi_i^*]\}$ and covariances $\mathrm{Cov}[\xi_i^*, \xi_j^*]$ for the 'best input' setting of the model parameters. We choose a form for the numerical discrepancy model and, if appropriate, use the procedure outlined in Section 4.1 to inform out prior belief specification for $\{\beta, r\}$.

*Full prior uncertainty specification.* We convert this initial prior specification into a full prior specification for all components by working along the solution trajectory and computing component moments at each time-step. We can do this in either of two ways: either algebraically, or by sampling trajectories. If choosing to proceed algebraically, we work through the time-steps of the solution, computing the expectation and variance of each component from the prior moments of its parents and the covariance between it and its parent nodes. In almost all cases, the solver function and the numerical discrepancy moments will be non-linear functions of their inputs; therefore, assumptions must be made about the higher-order moments of the model components in order to compute this full specification (e.g. that the higher-order moments correspond to those of a Gaussian distribution). Details of the moments that must be computed in order to characterise the full prior specification are given in Appendix C.
For the sampling approach, we simply make an appropriate distributional assumption for the uncertain components at each time-step, and characterise node expectations, variances and covariances by sampling corresponding numerical solution and numerical discrepancy components at the next time-point. The sampling approach is adopted for the examples presented in Sections 5 and 6, since in both cases, attempting to evaluate the moments of all model components algebraically would be a significant effort, and additional assumptions would have to be made for the higher-order moments in any case.

*4.3. Adjusting moments*

Once the joint prior specification for all components has been characterised (as outlined in Section 4.2), beliefs about the model components can be adjusted given observations of a subset of the measurements $\{z_k\}$ on the real system. The most basic approach that can be taken for adjustment is to simply compute covariances between all pairs of model components (either by making distributional assumptions and sampling on the graph, or by using the DAG 1 to compute the full covariance structure), and to perform a Bayes linear adjustment as outlined in Appendix A.1. For large problems, this approach will quickly prove to be

impractical, since it requires the computation and storage of a large covariance matrix; for very large problems, it may not even be possible to hold all of the relevant covariances in memory simultaneously, leading to the need to re-compute these from the graph as they are required, further adding to the computational burden.
The graph can also be a useful tool when carrying out the adjustment, particularly for problems with a large number of time-steps. The DAG corresponding to a particular model can be converted to a junction tree using the procedure outlined in Appendix B.3; Figure 2 illustrates the conversion procedure for the DAG in Figure 1. On a junction tree, the covariance between any individual node and the observed node can be computed by sequentially propagating covariances between adjacent cliques; the intersection between the sets of nodes in any pair of adjacent cliques is also a separating subset, imposing Bayes linear belief separation properties which can be exploited when performing the inference (see Appendix B.3 for details, and Appendix B.4 for an illustrative example of this procedure).
Propagating the adjustment through the junction tree in this manner means that only the covariances between the nodes in the cliques need to be computed and stored. We begin by adjusting beliefs for all of the nodes belonging to the same clique as the observed data. We then compute the covariance of the data with the nodes in the adjacent cliques through the separating subset, as described in Appendix B.3. Knowledge of these covariances allows us to adjust beliefs for all nodes in these adjacent cliques. Iterating the same procedure allows us to work through all cliques in the junction tree in turn, adjusting beliefs about the nodes in each clique as we reach it. This procedure is used for the adjustment in the bell-tower example presented in Section 6.

## 5. Example: projectile trajectory

We illustrate the model developed in Sections 3 and 4 through application to an example from classical kinematics, taken from Kibble and Berkshire [27]. A projectile of mass $m$ is launched from a point $u(t_0) = (0, 0, 0)$ in three-dimensional space at time $t_0$, with initial velocity $\dot{u}(t_0) = (\dot{u}_1(t_0), 0, \dot{u}_3(t_0)) = (\dot{x}_0, 0, \dot{z}_0)$, and is subject to gravity and air resistance, quantified in terms of the acceleration due to gravity $g$ and the coefficient of linear drag $\lambda$ respectively. We denote the position of the projectile at time $t$ by $u\,(t) = (u_1\,(t)\,, u_2\,(t)\,, u_3\,(t)\,) = (x(t), y(t), z(t))$; application of Newton's second law gives the following second-order ODE system

$$\begin{aligned}
\frac{d^2u_1}{dt^2}\,(t) &= -\gamma \frac{du_1}{dt}\,(t)\ , \\
\frac{d^2u_2}{dt^2}\,(t) &= 0\ , \\
\frac{d^2u_3}{dt^2}\,(t) &= -\gamma \frac{du_3}{dt}\,(t) - g\ ,
\end{aligned}$$

where $\gamma = \lambda/m$. There are no forces acting in the $y$-direction, and so we have $u_2\,(t) = y(t) = 0$ for all times $t$. Integrating the $x$ and $z$ equations once, we obtain

$$\begin{aligned}
\frac{du_1}{dt}\,(t) &= -\gamma u_1\,(t) + \dot{u}_1(t_0)\ , \\
\frac{du_3}{dt}\,(t) &= -\gamma u_3\,(t) - gt + \dot{u}_3(t_0)\ .
\end{aligned}$$

As this is a simple model, a solution is available in closed-form

$$u_1\,(t) = x(t) = \frac{\dot{u}_1(t_0)}{\gamma}\big(1 - e^{-\gamma t}\big)\ , \qquad u_3\,(t) = z(t) = \Big(\frac{\dot{u}_3(t_0)}{\gamma} + \frac{g}{\gamma^2}\Big)\big(1 - e^{-\gamma t}\big) - \frac{gt}{\gamma}\ . \tag{8}$$

In the coming sections, we develop a model for the solution within the framework outlined in Sections 3 and 4. In what follows, we consider only the solution component $u_3\,(t)$ in the $z$-direction, so that $u\,(t) = u_3\,(t)$ only; the same approach could be taken to modelling $u_1\,(t)$.
The code used to implement the analysis described in this section is available on GitHub (Jones [23]).

*5.1. Model*

We outline the structure of the model that will be used for the solution to this equation, and describe the procedure used to generate a corresponding prior uncertainty specification.

*Numerical scheme.* For this problem, we choose a first-order Euler solver

$$\begin{aligned}\phi\left(u\left(t_k\right), t_k, t_{k+1}, \xi\right) &= u\left(t_k\right) + (t_{k+1} - t_k) f\left(u\left(t_k\right), t_k, \xi\right) \\ &= u\left(t_k\right) + (t_{k+1} - t_k)\big( - \gamma u\left(t_k\right) - g t_k + \dot{u}(t_0)\big) \ .\end{aligned}$$

*Numerical discrepancy model.* As discussed in Section 2.2, choosing a first-order Euler scheme means that the numerical discrepancy function is simply the sum of the higher-order components of the Taylor expansion of $u$ around $t_k$. We exploit this information by choosing the second-order term as our sole basis function

$$b\left(\psi\left(t_k\right)\right) = \frac{1}{2}(t_{k+1} - t_k)^2 \frac{d^2 u}{dt^2}\left(t_k\right) \ ,$$

where $\psi\left(t_k\right) = \{u\left(t_k\right), t_{k+1}, t_k, \xi\}$ is the input set (and $\xi = \{\gamma, \dot{u}(t_0)\}$), and where the second derivative is

$$\begin{aligned}\frac{d^2 u}{dt^2}\left(t_k\right) &= -\gamma \frac{du}{dt}\left(t_k\right) - g \\ &= -\gamma f\left(u\left(t_k\right), t_k, \xi\right) - g \ .\end{aligned}$$

We assume that the residual component $r$ has mean zero, and that the covariance between function values at different input settings is

$$\mathrm{Cov}\left[r\left(\psi\right), r\left(\psi'\right)\right] = V\rho\left(\psi, \psi'\right) \ ,$$

where $V$ is the marginal variance parameter, and $\rho\left(\psi, \psi'\right)$ is a correlation function

$$\begin{aligned}\rho\left(\psi, \psi'\right) = \min\left(h_k, h_k'\right) \times \exp\Big[ - \frac{1}{2}\big(&\nu_t (t_k - t_k')^2 + \nu_h (h_k - h_k')^2 + \nu_u (u\left(t_k\right) - u\left(t_k\right)')^2 \\ &+ \nu_\gamma (\gamma - \gamma')^2 + \nu_{\dot{u}} (\dot{u}(t_0) - \dot{u}(t_0)')^2\big)\Big] \ ,\end{aligned}$$

where $\nu = \{\nu_t, \nu_h, \nu_u, \nu_\gamma, \nu_{\dot{u}}\}$ is a set of correlation parameters.

*System relationship.* For the purposes of this simple example, we assume that there is no systematic discrepancy between $u$ and $y$, so that $\delta\left(t_k\right) = 0$ for all $k = 1, 2, \ldots, n_t$. We assume that measurements of $y$ are, however, made with error, and we specify that $\mathrm{Var}\left[\epsilon_k\right] = (0.01)^2$ for all $k = 1, \ldots, n_t$.

*Prior specification.* We consider the trajectory of a projectile in the time-interval $[0, 10]$; $n_t = 50$ knots for the solution are selected by specifying that $t_0 = 0$ and $t_{50} = 10$, and then randomly selecting the knots $\{t_1, \ldots, t_{49}\}$ according to a Latin hypercube design. Our initial prior specification is that $\mathrm{E}\left[\gamma\right] = 0.55$ and $\mathrm{Var}\left[\gamma\right] = (0.2)^2$, that $\mathrm{E}\left[\dot{z}_0\right] = 40$ and $\mathrm{Var}\left[\dot{z}_0\right] = (10)^2$, and that $\mathrm{E}\left[u\left(t_0\right)\right] = 0$ and $\mathrm{Var}\left[u\left(t_0\right)\right] = (0.0001)^2$. The first modelling step is to generate an initial fit of the numerical discrepancy model. This is achieved as outlined in Section 4.1, by refining the solver grid, and using samples of the fine-scale solution to generate approximate samples of $\eta$. First, a random time-step is generated on the interval $[\min_k h_k, \max_k h_k]$. Then, the solution value at the start of this interval is fixed by running the deterministic coarse solver to a random initial point. Expectations and covariances for the numerical discrepancy are then generated as outlined in Section 4.1, by refining the interval into 500 equal intervals, running the solver on the fine grid 100 times and computing moments from this sample of discrepancies.

An initial set of 100 numerical discrepancy samples generated as described above is used to perform an initial regression: the prior moments of the regression parameter $\mathrm{E}\left[\beta\right]$ and $\mathrm{Var}\left[\beta\right]$ are fixed to the posterior mean and variance parameters resulting from this fit, and the marginal variance of the residual process $\mathrm{Var}\left[r\right]$ is fixed to the variance of the regression residuals. An additional set of 500 numerical discrepancy

samples is then used to jointly update the regression and residual components. A final set of 100 numerical discrepancy samples is used to check the fitted model, confirming that less than 5% of the samples lie outside 3 standard deviation predictive error bars (Pukelsheim [37]). In this example, the parameters $\nu$ of the correlation function are simply manually set to values which give reasonable predictive performance whilst ensuring that this condition is satisfied: in more complex cases, it may be more practical to use automatic hyperparameter tuning methods (see e.g. Rasmussen and Williams [38], Chapter 5).

*Computing the covariance structure.* Using the limited prior specification outlined above, and the initial numerical discrepancy model fit, we characterise the covariances between all model components as outlined in Section 4.2. In this example, we do this by sampling the solution trajectory 2000 times, assuming a Gaussian distribution for all stochastic components. These samples are used to empirically estimate the covariances corresponding to all edges on the graph.

*5.2. Adjustment*

For 5 different settings of the model parameters, we generate data from the real solution (8), add on synthetic Gaussian noise, and examine the effect of adjustment by a subset of these values. For each trajectory, we make 3 observations, at times selected to give good coverage of the the region of the trajectory where $u(t) > 0$. The prior moments are used to construct a DAG representation of the model, which is then converted to a junction tree representation, using the method outlined in Appendix B.3.
For simplicity in this example, we assume that each numerical discrepancy term is connected only to the numerical discrepancy at the previous time-point in the DAG (dropping the edges to all earlier time points seen in Figure 1). This assumption gives rise to a slightly different junction tree structure form that presented in Section 3.2.2: the four cliques corresponding to each time step are now:

- $Q_1(t_k) = \{\xi^*, \eta(t_k), \hat{u}(t_k), u(t_k)\}$ (for $k = 1, \ldots, n_t$);
- $Q_2(t_k) = \{\xi^*, \eta(t_k), u(t_k), \eta(t_{k+1})\}$ (for $k = 1, \ldots, (n_t - 1)$);
- $Q_3(t_k) = \{\xi^*, u(t_k), \eta(t_{k+1}), \hat{u}(t_{k+1})\}$ (for $k = 0, \ldots, (n_t - 1)$);
- $Q_4(t_k) = \{u(t_k), z_k, \epsilon_k\}$ (for $k = 1, \ldots, n_t$);

where here (and in the rest of this section), we use the notation $\eta(t_k) = \eta(u(t_{k-1}), \xi^*)$ and $\hat{u}(t_k) = \hat{u}(u(t_{k-1}), \xi^*)$ for simplicity. The junction tree obtained by connecting these cliques together is similar to the one shown in Figure 3: the main chain of cliques is formed by linking the cliques $Q_1(t_k)$, $Q_2(t_k)$, $Q_3(t_k)$ together in order, and linking $Q_3(t_k)$ to $Q_1(t_{k+1})$ to generate the links between time steps, with the branch to each $Q_4(t_k)$ attached to the main chain at $Q_3(t_k)$ (though $Q_1(t_k)$ or $Q_2(t_k)$ could also have been chosen). The main clique chain begins with $Q_3(t_0)$ and ends with $Q_1(t_{n_t})$.
The prior moments are updated by propagating the adjustment through the junction tree using the properties outlined in Appendix B.3 and illustrated in Appendix B.4. For clarity, the first steps of the procedure for adjustment given observation of a single $z_k$ ($k < n_t$) are now described in detail.

- **Identify the clique which contains** $z_k$:
  - In this instance, it is the clique $Q_4(t_k) = \{u(t_k), z_k, \epsilon_k\}$.
- **Adjust moments of the other members of the clique**:
  - The covariances $\mathrm{Cov}[u(t_k), z_k]$ and $\mathrm{Cov}[\epsilon_k, z_k]$ are directly available from the junction tree specification.
  - The adjusted expectation for e.g. $u(t_k)$ is therefore

$$\mathrm{E}_{z_k}[u(t_k)] = \mathrm{E}[u(t_k)] + \mathrm{Cov}[u(t_k), z_k]\,\mathrm{Var}[z_k]^{-1}[z_k - \mathrm{E}[z_k]],$$

    with the adjusted variance $\mathrm{Var}_{z_k}[u(t_k)]$ calculated similarly.

- **Identify the neighbouring cliques; compute the intersections between the current clique and its neighbours; move to the neighbours**:
  - In this instance, the sole neighbouring clique is $Q_3(t_k) = \{\xi^*, u(t_k), \eta(t_{k+1}), \hat{u}(t_{k+1})\}$.
  - The intersection with the current (initial) clique is simply $\{u(t_k)\}$.
  - We now move to consider adjustment of $Q_3(t_k)$.
- **Compute the covariance between elements of the current cliques and the data**:
  - Using the junction tree property, the covariance between $z_k$ and any element in $Q_3(t_k)$ is e.g.

$$\mathrm{Cov}[\eta(t_{k+1}), z_k] = \mathrm{Cov}[\eta(t_{k+1}), u(t_k)]\,\mathrm{Var}[u(t_k)]^{-1}\mathrm{Cov}[u(t_k), z_k] ,$$

    where the covariances $\mathrm{Cov}[\eta(t_{k+1}), u(t_k)]$ etc. between elements of the same clique are available from the junction tree specification.
- **Compute the adjusted moments for the current cliques**:
  - Once the covariances of all nodes in $Q_3(t_k)$ with $z_k$ have been identified, adjustments can be carried out in the usual way, e.g.

$$\mathrm{E}_{z_k}[\eta(t_{k+1})] = \mathrm{E}[\eta(t_{k+1})] + \mathrm{Cov}[\eta(t_{k+1}), z_k]\,\mathrm{Var}[z_k]^{-1}[z_k - \mathrm{E}[z_k]] ,$$

    and e.g.

$$\mathrm{Cov}_{z_k}[\eta(t_{k+1}), \hat{u}(t_{k+1})] = \mathrm{Cov}[\eta(t_{k+1}), \hat{u}(t_{k+1})] - \mathrm{Cov}[\eta(t_{k+1}), z_k]\,\mathrm{Var}[z_k]^{-1}\mathrm{Cov}[z_k, \hat{u}(t_{k+1})] .$$

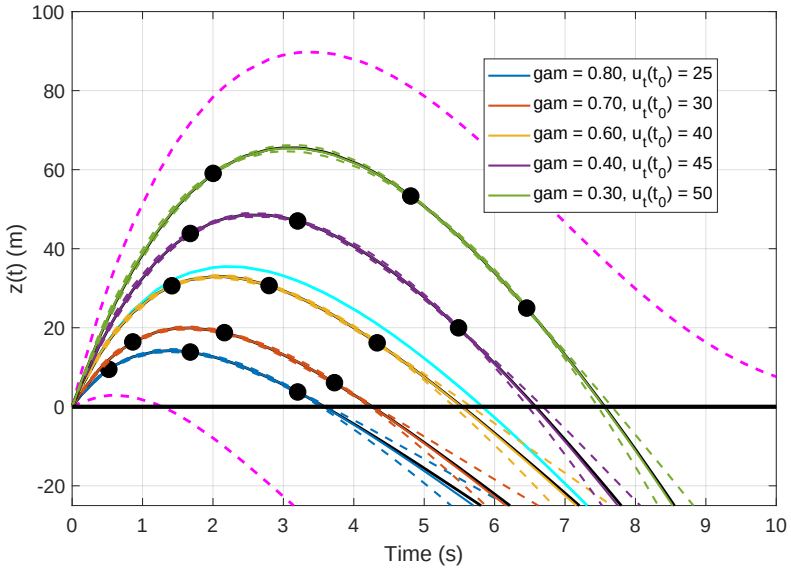
- **Identify the neighbouring cliques (not already visited); compute the intersections between the current cliques and their neighbours; move to the neighbours**:
  - We are at $Q_3(t_k)$; neighbours in the junction tree that we have not yet visited are $Q_2(t_k) = \{\xi^*, \eta(t_k), u(t_k), \eta(t_{k+1})\}$ and $Q_1(t_{k+1}) = \{\xi^*, \eta(t_{k+1}), \hat{u}(t_{k+1}), u(t_{k+1})\}$.
  - Considering $Q_2(t_k)$, the intersection between this node and $Q_3(t_k)$ is the set $S = \{\xi^*, u(t_k), \eta(t_{k+1})\}$.
  - The covariance of any node in $Q_2(t_k)$ with $z_k$ can be computed as e.g.

$$\mathrm{Cov}[\eta(t_k), z_k] = \mathrm{Cov}[\eta(t_k), S]\,\mathrm{Var}[S]^{-1}\mathrm{Cov}[S, z_k] ,$$

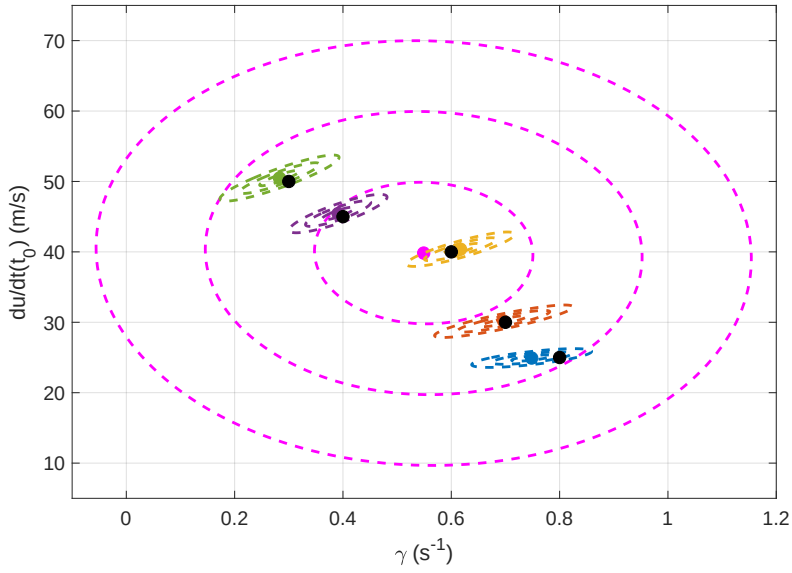
    where $\mathrm{Cov}[S, z_k]$ was already computed for all elements of $S$ when considering adjustment of the previous clique ($Q_3(t_k)$), and $\mathrm{Var}[S]$ is the covariance matrix for the variables in $S$.
  - Having obtained the covariance of all nodes in $Q_2(t_k)$ with $z_k$, adjusted moments can now be computed for all of these variables.
  - Covariances with $z_k$ and corresponding adjusted moments are computed for all elements of the other neighbour, $Q_1(t_{k+1})$, in the same way.

This process of adjusting moments for all nodes in a given clique, then identifying unvisited neighbouring cliques and propagating the covariance with the data $z_k$ out to these neighbours is repeated until all cliques on the junction tree have been visited and adjusted. If other points $z_l$ ($l \neq k$) on the trajectory have also been observed, then we adjust moments further by simply repeating the above procedure for subsequent data points, treating the moments obtained after adjustment by $z_k$ as the prior moments for adjustment by the next data point, and so on.
Figure 4 shows the effect of the adjustment on beliefs about the trajectory and parameter settings in each case. Figure 4(a) shows the prior and adjusted moments of the trajectory for each of the 5 parameter settings used to generate the original data. Our uncertainties about the parameter settings and the build-up

of uncertainty about the state induced by the numerical discrepancy cause the prior error bars to widen over time, meaning that our prior specification allows for a wide range of possible trajectories. Because of the rich covariance structure encoded in the graph, making only 3 observations allows us to make confident and accurate predictions for the remainder of the trajectory. Figure 4(b) shows the prior and adjusted moments of the parameters $\xi^*$ for the same adjustment cases as in Figure 4(a): in the same way, we see that even a small number of observations are enough to do a good job of estimating the parameter settings which generated the trajectory.




(a)




(b)

Figure 4: Prior and adjusted moments for the components of the model presented in Section 5.2. Figure 4(a) shows the moments of the solution trajectory $u\,(t)$: the prior expectations $\mathrm{E}\,[u\,(t_k)\,]$ $(k = 1, 2, \ldots, n_t)$ are shown as a solid cyan line, and prior error bars $\mathrm{E}\,[u\,(t_k)\,] \pm 3\mathrm{Var}\,[u\,(t_k)\,]^{1/2}$ are shown in dashed magenta; adjusted expectations $\mathrm{E}_D\,[u\,(t_k)\,]$ and error bars $\mathrm{E}_D\,[u\,(t_k)\,] \pm 3\mathrm{Var}_D\,[u\,(t_k)\,]^{1/2}$ given different sets of trajectory observations (generated using different parameter settings) are shown as solid and dashed coloured lines respectively; the true trajectory in each case is shown as a black line (largely hidden behind the corresponding adjusted expectation), and the points on the trajectory which are observed are shown as black markers. Figure 4(b) shows the prior and adjusted moments of the parameters $\xi^* = \{\dot{u}_0^*, \gamma^*\}$: the magenta marker shows the prior expectation, and the magenta dashed ellipses show the 1, 2 and 3 standard deviation contours of the prior; the coloured markers and ellipses show the moments after each of the adjustments (colours correspond to those in Figure 4(a)); the true parameter setting in each case is shown as a black marker.

## 6. Example: coupled bell-tower model

Churches around the world have towers which are equipped with sets of bells; these are sounded (rung) to advertise services and to mark important occasions. Sound is generated either by striking a fixed bell with a hammer, or by attaching a clapper to the interior of the bell and allowing the bell to swing in such a way that this clapper strikes its internal surface. In the UK, and in some other countries, it is common to find bells which are hung for 'English style' change ringing (also known as 'full circle' ringing); bells are mounted on a rigid headstock, which is attached using bearings to a rigid frame, allowing it to swing through 360 degrees. A wheel is then attached to the headstock, and the bell is swung using a rope attached to this wheel.

In some towers, the largest bell hung for full circle ringing can weigh more than 1000 kg, and other bells in the ring are generally cast in proportion to the largest bell. The total force exerted on the tower when bells of this size are being rung can be large. It is not uncommon for these forces to induce tower movement which alters the trajectories of the bells, making them difficult to ring. In extreme cases, tower movement can induce structural damage over time. Various authors have used mathematical models to study the effect of change-ringing bells on the tower which houses them: Smith and Hunt [42] derive equations of motion for the tower and the bells, and then use a solution for the tower forced by the solution for the bells to study the response of the tower to different ringing regimes.

In this section, we use the framework outlined in Sections 3 and 4 to handle uncertainties encountered when numerically solving the equations described by Smith and Hunt [42]. The remainder of the section is structured as follows: in Section 6.1, a Lagrangian description of the coupled motion of the tower and the bells is introduced, and the equations of motion are derived. Then, in Section 6.2, a corresponding statistical model is defined through choice of a numerical scheme, discrepancy model and graphical representation. Finally, in Section 6.3, 'observations' of the system are generated through deterministic solution of the system on a finer mesh, and the effect of adjustment by observations of a subset of the (noise-corrupted) solution values is discussed.

### *6.1. Lagrangian bell-tower model*

We use a Lagrangian mechanical model for the motion of the bell-tower system, taken from Smith and Hunt [42]. We denote the displacement of the tower at time $t$ by $x\left(t\right) = \left(x_1\left(t\right), x_2\left(t\right)\right)^{\mathrm{T}}$, where $x_1\left(t\right)$ is the horizontal (East-West) displacement of the tower, and $x_2\left(t\right)$ is its vertical (North-South) displacement. Neglecting the effect of the bells, the Lagrangian for the tower motion is

$$\mathcal{L}_T = \sum_{i=1}^{2} \left[\frac{1}{2} M \dot{x}_i^2 - \frac{1}{2}\kappa_i x_i^2\right] ,$$

where $M$ is the mass of the tower and $\kappa_i$ is the spring constant in the $i^{\text{th}}$ direction. Applying the Euler-Lagrange equation and allowing for damping of the motion, we obtain the equation of motion $\ddot{x}_i + 2\lambda_i \dot{x}_i + \omega_i^2 x_i = 0$ independently for each direction $i = 1, 2$, where $\lambda_i$ is the damping constant and $\omega_i^2 = \kappa_i / M$ the natural frequency for the $i^{\text{th}}$ direction.

Independently of the tower, the Lagrangian for the motion of the bells is

$$\mathcal{L}_B = \sum_{j=1}^{n_\theta} \left[\frac{1}{2} m_j r_{g_j}^2 \dot{\theta}_j^2 + \frac{1}{2} m_j r_{c_j}^2 \dot{\theta}_j^2 + m_j g \cos\theta_j\right] ,$$

where $m_j$ is the mass of the $j^{\text{th}}$ bell, $r_{c_j}$ is the distance from the pivot to the centre of mass for bell $j$, and $r_{g_j}$ is the $j^{\text{th}}$ radius of gyration. In this instance, applying the Euler-Lagrange equations yields $l_j \ddot{\theta}_j + g \sin\theta_j = 0$, where $l_j = (r_{g_j}^2 + r_{c_j}^2)/r_{c_j}$.

We combine these Lagrangians, modifying the kinetic energy of each bell to include a horizontal forcing due to the motion of the tower, to obtain the following Lagrangian for the coupled bell-tower system

$$\mathcal{L} = \sum_{i=1}^{2} \left[\frac{1}{2} M \dot{x}_i^2 - \frac{1}{2}\kappa_i x_i^2\right] + \sum_{i=1}^{2}\sum_{j=1}^{n_\theta} \left[d_{ij} m_j r_{c_j} \dot{\theta}_j \dot{x}_i \cos\theta_j\right] + \sum_{j=1}^{n_\theta} \left[\frac{1}{2} m_j l_j r_{c_j} \dot{\theta}_j^2 + m_j g \cos\theta_j\right] , \tag{9}$$

where $d_j = (d_{1j}, d_{2j})^{\mathrm{T}}$ is a unit vector along the swing direction of bell $j$. Applying the Euler-Lagrange equations to (9), we obtain the following equations of motion

$$\ddot{x}_i + 2\lambda_i \dot{x}_i + \omega_i^2 x_i = -\sum_{j=1}^{n_\theta} d_{ij} \frac{m_j r_{c_j}}{M} \left[ \ddot{\theta}_j \cos\theta_j - \dot{\theta}_j^2 \sin\theta_j \right], \tag{10}$$

$$l_j \ddot{\theta}_j + g \sin\theta_j = -\sum_{i=1}^{2} d_{ij} \ddot{x}_i \cos\theta_j \ . \tag{11}$$

| Bell | cwt | $m$ qr | lb | $r_c$ (mm) | $l$ (mm) | Swing (handstroke) |
|---|---|---|---|---|---|---|
| 1 | 6 | 1 | 27 | 407 | 628 | N-S |
| 2 | 7 | 0 | 8 | 405 | 665 | S-N |
| 3 | 7 | 3 | 10 | 417 | 665 | S-N |
| 4 | 7 | 3 | 14 | 429 | 712 | S-N |
| 5 | 9 | 3 | 25 | 417 | 752 | S-N |
| 6 | 11 | 0 | 8 | 428 | 790 | W-E |
| 7 | 12 | 2 | 8 | 403 | 856 | W-E |
| 8 | 15 | 3 | 1 | 438 | 887 | W-E |
| 9 | 21 | 2 | 9 | 451 | 984 | W-E |
| 10 | 28 | 0 | 6 | 559 | 1027 | S-N |

Table 1: Fixed bell parameters (mass $m$, distance $r_c$ from pivot to centre of mass and parameter $l$, computed as in Section 6.1) used as input to equations (10) and (11). The weights and layout correspond to those of the bells at Durham Cathedral (Dove [13]), and the other bell parameters are those of the old bells at Great St Mary, Cambridge (taken from Smith and Hunt [42])

*6.2. Model specification*

We develop a statistical model for the solution of the system (10) and (11) that fits within the framework outlined in Sections 3 and 4.

*Numerical scheme.* We again choose a first-order Euler scheme for this problem. We stack the components outlined in Section 6.1 into a $2 \times (2 + n_\theta)$-dimensional solution state vector

$$u(t) = \begin{pmatrix} \dot{x}(t) \\ \dot{\theta}(t) \\ x(t) \\ \theta(t) \end{pmatrix} \ .$$

Combining the equations of motion (10) and (11) with the trivial equations of motion $\frac{dx}{dt} = \dot{x}$ and $\frac{d\theta}{dt} = \dot{\theta}$ gives the representation

$$A \frac{du}{dt}(t) = b,$$

where

$$A(u(t), \xi) = \begin{pmatrix} A_{\dot{x}\dot{x}} & A_{\dot{x}\dot{\theta}} & 0 & 0 \\ A_{\dot{\theta}\dot{x}} & A_{\dot{\theta}\dot{\theta}} & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{pmatrix} , \qquad b(u(t), \xi) = \begin{pmatrix} b_{\dot{x}} \\ b_{\dot{\theta}} \\ b_x \\ b_\theta \end{pmatrix} ,$$

where $b_x = \dot{x}$, $b_\theta = \dot{\theta}$, and the elements of the remaining components are

$$A_{\dot{x}_i \dot{x}_j} = I_{ij} \ , \qquad A_{\dot{x}_i \dot{\theta}_j} = d_{ij} \frac{m_j r_{c_j}}{M} \cos\theta_j \ ,$$

$$A_{\dot{\theta}_i \dot{x}_j} = d_{ji} \cos\theta_i \ , \qquad A_{\dot{\theta}_i \dot{\theta}_j} = I_{ij} \ ,$$

$$b_{\dot{x}_i} = \sum_{j=1}^{n_\theta} d_{ij} \frac{m_j r_{c_j}}{M} \dot{\theta}_j^2 \sin\theta_j - 2\lambda_i \dot{x}_i - \omega_i^2 x_i \ , \qquad b_{\dot{\theta}_j} = -g \sin\theta_j \ .$$

Using this notation, we can write the equations of motion as

$$\frac{du}{dt}(t) = f(u(t), \xi) = A^{-1} b \ ,$$

where $\xi = \{\lambda, \omega\}$, and the first-order Euler scheme for this problem is

$$\hat{u}_i(u(t_k), t_k, t_{k+1}, \xi) = u(t_k) + (t_{k+1} - t_k) f_i(u(t_k), \xi) \ .$$

*Numerical discrepancy model.* As in the example presented in Section 5, we use the higher-order derivatives of $f$ as the basis for our numerical discrepancy model

$$b(\psi) = \begin{pmatrix} \frac{1}{2} h_k^2 \frac{d^2 u}{dt^2}(\psi) \\ \frac{1}{6} h_k^3 \frac{d^3 u}{dt^3}(\psi) \end{pmatrix} \ ,$$

where again, $\psi = \{u(t_k), t_k, t_{k+1}, \xi\}$ is an abbreviation for the set of solver inputs. The second-order derivatives are

$$\frac{d}{dt}\Big(A \frac{du}{dt}\Big) = \frac{db}{dt} \ ,$$

$$\frac{dA}{dt}\frac{du}{dt} + A\frac{d^2u}{dt^2} = \frac{db}{dt} \ ,$$

$$\frac{d^2u}{dt^2} = A^{-1}\Big[\frac{db}{dt} - \frac{dA}{dt}\frac{du}{dt}\Big] \ ,$$

and repeating the same procedure, we find that the third-order derivatives are

$$\frac{d^3u}{dt^3} = A^{-1}\Big[\frac{d^2b}{dt^2} - \Big(\frac{d^2A}{dt^2}\frac{du}{dt} + 2\frac{dA}{dt}\frac{d^2u}{dt^2}\Big)\Big] \ .$$

For the residual process, we simply specify that

$$\mathrm{Cov}\left[r_i(\psi), r_j(\psi')\right] = V_{ij} I(\psi = \psi') \ ,$$

so that the residual is uncorrelated across the input space and between outputs.

*System relationship.* We assume that there is a systematic discrepancy between the true solution to the equation and the real bell-tower system (as in equation 5). We believe that $\mathrm{E}\left[\delta_i(t_k)\right] = 0$ for all $i, k$, and that

$$\mathrm{Cov}\left[\delta_i(t_k), \delta_j(t_l)\right] = W_{ij} \exp\Big[-\frac{\nu_t}{2}(t_k - t_l)^2\Big] \ ,$$

for all $|k - l| \leq 2$, with $\mathrm{Cov}\left[\delta_i(t_k), \delta_j(t_l)\right] = 0$ otherwise, where $\nu_{t_k} = 1$ is a correlation parameter, and $W_{ii} = (10^{-5})^2$ for all tower components and $(1)^2$ for all bell-related components (with $W_{ij} = 0$ for all $j \neq i$). We assume that measurements on the system are made subject to error (as in equation 4), and we specify that $\mathrm{Var}\left[\epsilon_i\right] = (10^{-6})^2$ for all tower-related components and $(10^{-2})^2$ for all bell-related components (we assume that measurement errors are uncorrelated across model components).

In this example, we illustrate the process of belief adjustment using 'observations' of a numerically-generated trajectory (obtained by running the solver at a higher temporal resolution): systematic discrepancy and measurement error contributions are therefore also generated using the above specifications and added to the synthetic trajectories. The above parameters are therefore treated as known and fixed, and are not estimated from the data as part of the analysis. For analyses using real data, it will be necessary to estimate such parameters from the data, for example using methods similar to those outlined in Rasmussen and Williams [38] (Chapter 5).

*Prior specification.* Using the components specified above, we proceed to generate a joint prior specification corresponding to the graphical model. We solve for a grid of 500 evenly-spaced intervals between $t_0 = 0$ and $t = 10$, so that $h_k = 0.02$ for all $k$. Our second-order belief specification for the components of the initial state is as follows

$$\begin{aligned} \mathrm{E}\left[\dot{x}_i\left(t_0\right)\right] &= 0\ , & \mathrm{E}\left[\dot{\theta}_i\left(t_0\right)\right] &= 0\ , & \mathrm{E}\left[x_i\left(t_0\right)\right] &= 0\ , & \mathrm{E}\left[\theta_i\left(t_0\right)\right] &= 160\ , \\ \mathrm{Var}\left[\dot{x}_i\left(t_0\right)\right] &= (10^{-4})^2\ , & \mathrm{Var}\left[\dot{\theta}_i\left(t_0\right)\right] &= (10^{-4})^2\ , & \mathrm{Var}\left[x_i\left(t_0\right)\right] &= (10^{-4})^2\ , & \mathrm{Var}\left[\theta_i\left(t_0\right)\right] &= (0.1)^2\ , \end{aligned}$$

and for the parameters $\xi^* = \{\lambda^*, \omega^*\}$, we specify that $\mathrm{E}\left[\lambda_i^*\right] = 1$, $\mathrm{Var}\left[\lambda_i^*\right] = (0.1)^2$, $\mathrm{E}\left[\omega_i^*\right] = 1.5$ and $\mathrm{Var}\left[\omega_i^*\right] = (0.1)^2$ (for $i = 1, 2$ in both cases). An initial fit of the discrepancy model is carried out using the grid refinement procedure outlined in Section 4.1 and described in detail in Section 5.2.

In this example, we use the junction tree for analysis; we obtain this junction tree from the initial DAG defined by the conditional independence structure of the model. First, we sample 500 solution trajectories, assuming a Gaussian distribution for the initial state, the numerical discrepancy, the structural discrepancy and the measurement error. Using these samples, we empirically estimate the covariances corresponding to the edges on the DAG. We then proceed to convert the DAG into a junction tree which encodes the same belief separation properties, using the procedure outlined in Appendix B.3. First, we moralise the graph, adding additional edges into the adjacency matrix between pairs of nodes with common children. Then, we triangulate the graph, using the 'LB-Triang' algorithm for computing minimal triangulations presented by Berry et al. [2]. The nodes are labelled according to a maximum cardinality search, and the cliques are identified using the Bron-Kerbosch algorithm (see, for example Himmel et al. [21]). The identified cliques are then joined together into a tree as described in Appendix B.3. Once the cliques of the junction tree have been identified, the covariances corresponding to the edges within cliques are computed from the covariances on the edges of the original graph.

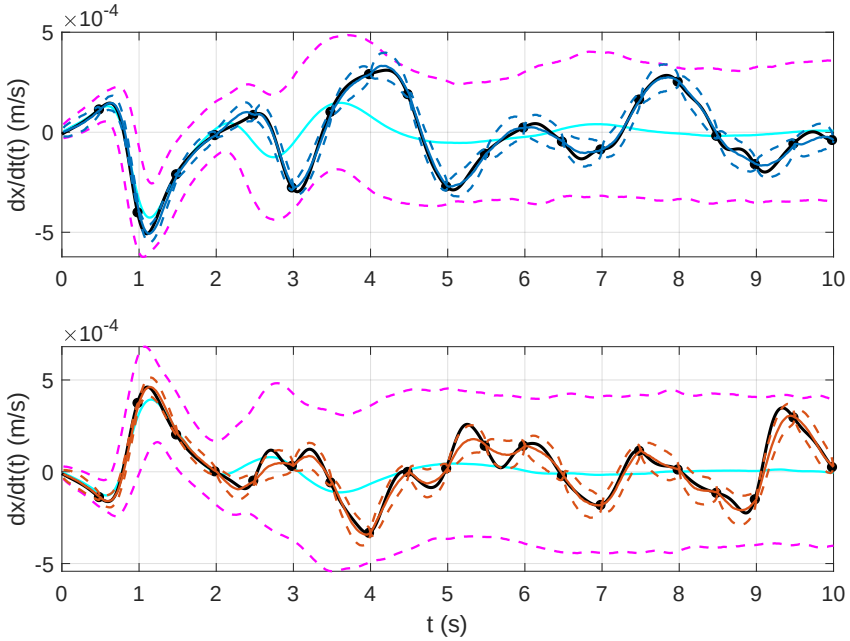
### *6.3. Adjustment*

Having represented the prior joint covariance structure using a junction tree, we proceed to use the junction tree as a framework for updating beliefs given data, as outlined in Appendix B.3. Data observed on a real-world bell-tower system is not available, and so we illustrate the adjustment process by adjusting using data simulated using an Euler solver run on a refined grid. We divide each of the initial time-steps into 200 sub-intervals, generate the numerical solution on this grid, and add on a randomly-sampled (Gaussian) structural discrepancy and measurement error term at each time-point.

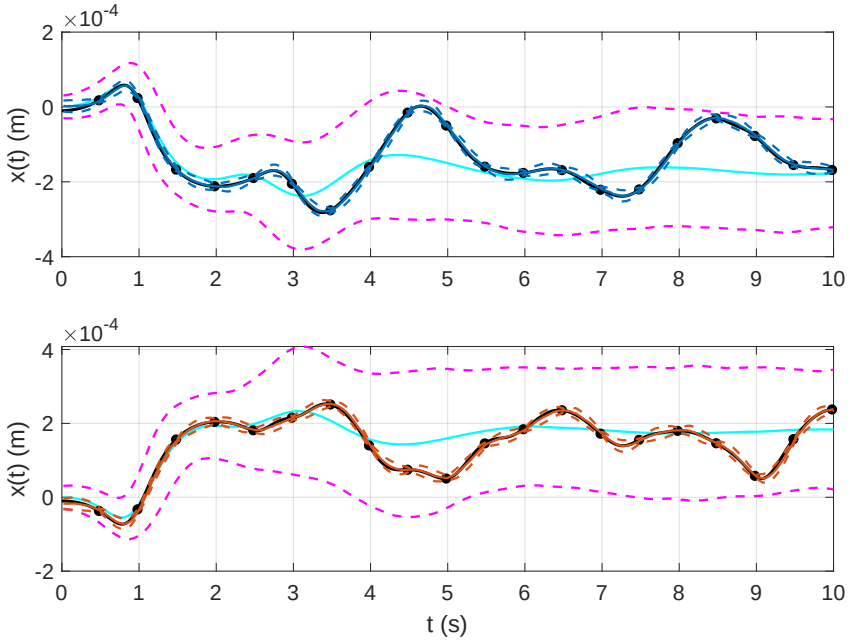
We 'observe' all components of this synthetic solution at every $25^{\text{th}}$ time-point (i.e. the collection $D = \{z_{25}, z_{50}, \ldots, t_{475}, z_{500}\}$). The junction tree is used to adjust by each of these measurements: the covariance $\mathrm{Cov}\left[., z_k\right]$ of each node on the graph with the observed data $z_k$ is computed by propagation of the covariance along the tree, and these covariances are used to adjust the expectations and covariances within each clique when it is reached. We adjust by each of the observed $z_k$ in turn, using the set of adjusted moments after each pass through the graph as the prior for the adjustment by the next data point. After adjustment by all observed data, we compare the data with the adjusted moments for all time points $t_k$, and confirm that 95% of the $z_k$ lie within 3 standard deviations of the adjusted mean.

Figures 5 and 6 show the prior and adjusted moments for some of the components of the system value $y\left(t\right)$ for all knots $t_k$ on the solution trajectory. In the prior, there is a great deal of uncertainty about both $u\left(t\right)$ and the discrepancy $\delta\left(t\right)$. While our uncertainty about the ODE solution at the start of the trajectory is

relatively low, it increases rapidly in time, owing to the build-up of numerical and parameter uncertainty between time-steps. After adjustment by 20 data points observed at evenly-spaced knots, we are able to make confident and accurate predictions for the system value $y = u + \delta$ at the remaining times, though for later times, some of the variation in the data due to the discrepancy is incorrectly attributed to the ODE solution.

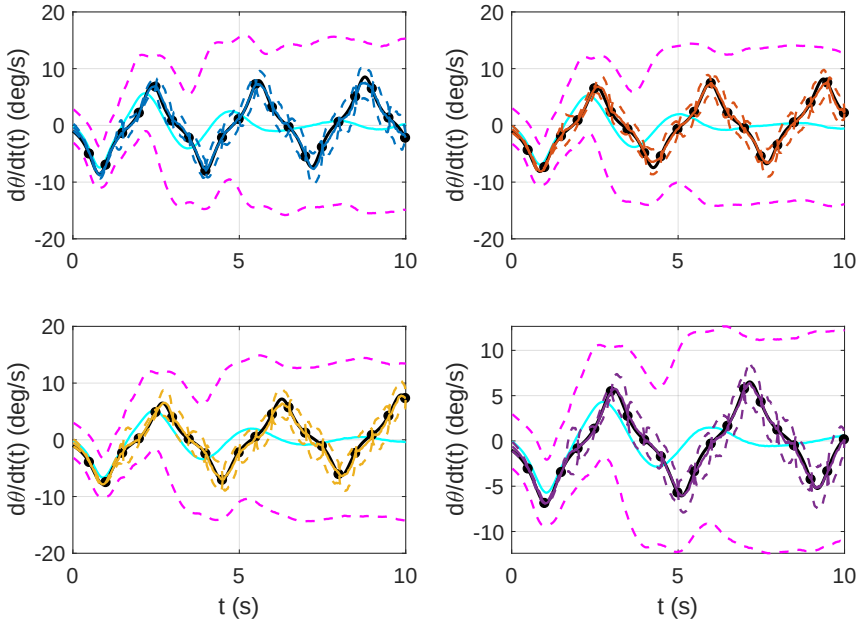



(a) Tower velocity components

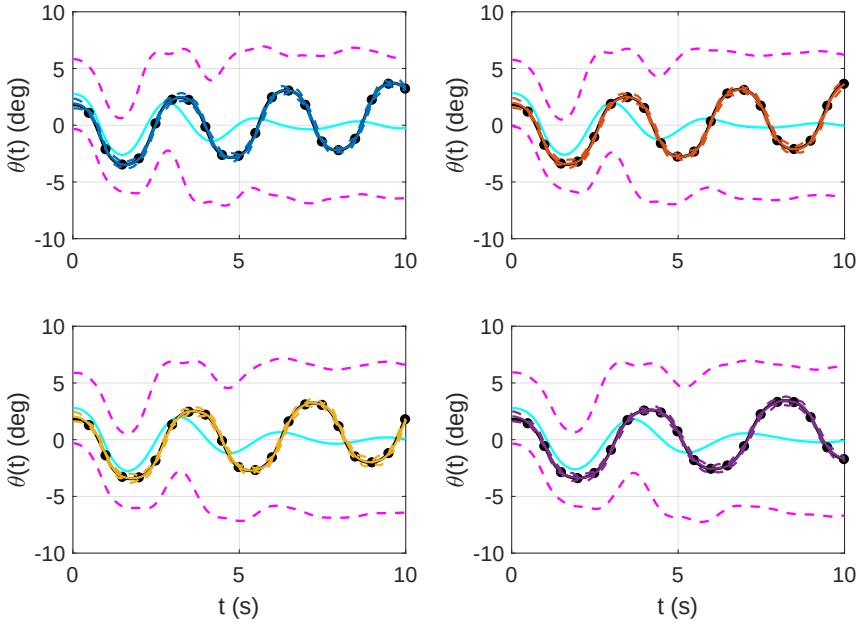



(b) Tower positions

Figure 5: Prior and adjusted moments of the $\dot{x}$ and $x$ components of the solution $y(t)$ after adjustment as outlined in Section 6.3. In all figures, the prior expectations $\mathrm{E}[y(t_k)]$ are shown in cyan, and prior error bars $\mathrm{E}[y(t_k)] \pm 3\mathrm{Var}[y(t_k)]^{1/2}$ are shown in dashed magenta. The adjusted expectations $\mathrm{E}_D[y(t_k)]$ are shown as solid coloured lines, and the adjusted error bars $\mathrm{E}_D[y(t_k)] \pm 3\mathrm{Var}_D[y(t_k)]^{1/2}$ are shown as dashed coloured lines. The synthetic solution from which the data used for the adjustment were sampled is shown in black. Figure 5(a) shows both components of the tower velocity $\dot{x}_i(t_k)$ for $i = 1, 2$ (in m/s), and Figure 5(b) shows the tower positions $x_i(t_k)$ (in m) for $i = 1, 2$.




(a) Bell velocities




(b) Bell positions

Figure 6: Prior and adjusted moments of the $\dot{\theta}$ and $\theta$ components of the solution $y(t)$ after adjustment as outlined in Section 6.3. Colours schemes used for the plot components correspond to those used in Figure 5 (cyan and magenta for priors, black for data used in adjustment, coloured lines for adjusted moments). Figure 6(a) shows the bell angular velocities $\dot{\theta}_i(t_k)$ (in deg/s) for bells $i = \{1, 4, 6, 10\}$, and Figure 6(b) shows the bell positions $\theta_i(t_k)$ (in deg/s) for bells $i = \{1, 4, 6, 10\}$.

## 7. Discussion

Commonly, it is not possible to directly evaluate the solution to a system of ODEs, and so a numerical solution is used as an approximation. The discrepancy between this approximate solution and the real solution is unknown by definition. In this article, we present a framework which represents the structure of our uncertainty about the true solution to the system of ODEs, and which links this to the real-world system that the ODEs are designed to represent. The framework is most naturally represented as a graph; this way of representing the model is also useful for the design of inference algorithms. We discussed the structure of the numerical discrepancy function, and proposed a procedure for reducing uncertainty about this component along the solution trajectory through a grid refinement procedure.

We perform Bayes linear analysis of the model, using the graph to structure our belief specification and to carry out updating given observations of the system at certain times. The use of a Bayes linear graphical model provides access to tools which reduce the computational burden associated with both the prior uncertainty specification and the inference procedure. We can store a limited prior specification on the edges of the graph, and use this specification to efficiently re-construct the covariance between any pair of nodes as required. Additionally, converting the originally specified DAG to a junction tree means that inference can be performed by propagating the adjustment along the cliques of the diagram, leading to an update procedure whose complexity scales linearly with the number of time steps (for a fixed number of links between discrepancy components across time-steps). Structuring the model in this way allows for a much richer description of the covariance structure of the solution trajectory than would have been obtainable through simply fitting an emulator to the solver output, and consequently allows us to produce more confident and accurate predictions for the system value at time points where it has not been observed, given measurement at a particular set of time points.

Related future work will focus on two main areas. First, the framework can be extended to deal with partial differential equation (PDE) models, in which the solution surface is implicitly defined through a set of relationships between partial derivatives of the solution with respect to a number of input variables, and specification of appropriate boundary conditions. PDE models commonly describe the spatio-temporal evolution of a particular physical system; for example, the diffusion and convection-diffusion equations can be used to describe the evolution of gas concentrations (Stockie [43]), and the Euler and Navier-Stokes equations can be used as a model for the behaviour of fluids (see, e.g. Vallis [48], Blazek [3]). For PDE models, it is even more common that an analytical solution to the system of equations is not available. Numerical methods for PDEs generally approximate the solution at a fixed set of points in space, and evolve the approximation in time using a transfer function derived from the original equation. A number of different kinds of numerical schemes exist in this context: finite difference schemes (see, e.g. LeVeque [30]) construct the transfer function using finite difference approximations to partial derivative terms, in much the same way as the Euler scheme in the ODE case; finite element schemes (see, e.g., Iserles [22]) introduce a basis representation of the solution, and find a weak solution by imposing that the integral of this approximate solution against a class of test functions vanishes; finite volume schemes (see, e.g. Eymard et al. [14]) use conservation laws implied by particular PDEs to to derive relationships between quantities on spatially- and temporally-adjacent mesh elements. In the PDE case, the numerical discrepancy introduced through use of a particular solver will be a function of both space and time, requiring the use of a more complex model. There are also additional complexities in the case of the finite element and finite volume methods, since these generally give rise to implicit numerical schemes, in which we must invert a model parameter-dependent matrix in order to time-evolve the solution.

Second, further work will focus on the treatment of higher-order behaviour of the solution trajectory (or the solution surface, in the case of a PDE). In the examples presented in Sections 5 and 6, the prior second-order specification is derived by making a Gaussian distributional assumption for each of the numerical discrepancy terms, sampling the solution trajectory a number of times and deriving expectations, variances and covariances empirically from these samples. This approach was adopted because the non-linearity in the expression for the numerical solution means that assumptions about the higher-order moments of the model components are required; of course, this also makes the solution dependent on a particular choice of distribution. The Bayes linear approach does not, however, limit us to consideration of only the first-

and second-order moments of a collection of quantities; we may make a second-order prior specification and carry out a Bayes linear adjustment for as many integer powers of a component as we choose. The existing framework can be adapted to accommodate beliefs about higher order components, eliminating the need to make distributional assumptions and sample, and capturing higher order effects in the adjustment process, all whilst retaining the attractive computational properties of the Bayes linear analysis.

**Acknowledgements**

Matthew Jones was supported by an EPSRC CASE Studentship in partnership with Shell Projects and Technology.

## A. Bayes linear analysis

This appendix summarises aspects of Bayes linear methodology required for the analysis performed in the main article. Appendix A.1 details the extent of the prior specification that we must make, and details how our beliefs should be updated on learning the values of a set of quantities; Appendix A.2 introduces the concepts of Bayes linear sufficiency and belief separation.

### *A.1. Bayes linear adjustment*

We are uncertain about the values of a collection of quantities $C$; suppose that we observe the values of a subset $D = \{D_1, \ldots, D_{n_D}\}$ of $C$. A simple way of modifying our beliefs about the subset $B = \{B_1, \ldots, B_{n_B}\}$ is through linear fitting; for a single element $B_i$, we seek the linear combination

$$\mathrm{E}_D\left[B_i\right] = \sum_{k=0}^{n_D} h_k D_k \ ,$$

which minimises

$$\mathrm{E}\left[\left(B_i - \sum_{k=0}^{n_D} h_k D_k\right)^2\right] \ ,$$

where $D_0 = 1$. Optimising over $h = (h_1, \ldots, h_{n_D})^{\mathrm{T}}$, we find that (Goldstein and Wooff [17])

$$\mathrm{E}_D\left[B_i\right] = \mathrm{E}\left[B_i\right] + \mathrm{Cov}\left[B_i, D\right] \mathrm{Var}\left[D\right]^{-1}\left[D - \mathrm{E}\left[D\right]\right] \ .$$

The quantity $\mathrm{E}_D\left[B_i\right]$ is referred to as the adjusted expectation. For the whole collection, we have the following definitions.

**Definition A.1.** *The **adjusted expectation** of a collection $B$ given observation of a collection $D$ is*

$$E_D\left[B\right] = E\left[B\right] + Cov\left[B, D\right] Var\left[D\right]^{-1}\left[D - E\left[D\right]\right] \ .$$

**Definition A.2.** *The **adjusted variance** of a collection $B$ given observation of a collection $D$ is*

$$Var_D\left[B\right] = Var\left[B\right] - Cov\left[B, D\right] Var\left[D\right]^{-1} Cov\left[D, B\right] \ .$$

**Definition A.3.** *The **adjusted covariance** between collections $A$ and $B$ given observation of a collection $D$ is*

$$Cov_D\left[A, B\right] = Cov\left[A, B\right] - Cov\left[A, D\right] Var\left[D\right]^{-1} Cov\left[D, B\right] \ .$$

*A.2. Belief separation*

In a probabilistic analysis, conditional independence statements are of considerable importance; we say that the collection $B$ is conditionally independent of the collection $A$ given the collection $D$ if $p(B|A,D) = p(B|D)$. In a Bayes linear analysis, the corresponding specification is one of Bayes linear sufficiency.

**Definition A.4.** *The collection $D$ is **Bayes linear sufficient** for $A$ for adjusting $B$ if*

$$E_{D\cup A}[B] = E_D[B] \ .$$

Specifying that $D$ is Bayes linear sufficient for $A$ for adjusting $B$ implies that adjustment by $A$ provides no further information about $B$ after adjustment by $D$. This is equivalent to the following specification.

**Definition A.5.** *The collection $D$ **separates** $A$ and $B$, written*

$$\lfloor A \perp B \rfloor / D \ ,$$

*if $D$ is Bayes linear sufficient for $A$ for adjusting $B$.*

## B. Bayes linear graphical models

Graphs are used as a convenient qualitative and quantitative representation of the structure of our beliefs about the quantities in a problem. In the probabilistic case, a graph encodes a set of conditional independence judgements that we make; in the Bayes linear case, the graph encodes our judgements about belief separations between quantities. A detailed introduction to Bayes linear graphical models can be found in Goldstein and Wooff [17] (Chapter 10). This section presents aspects of the treatment of Goldstein and Wooff which are relevant to the analysis performed in the rest of the article.

*B.1. Directed graphical models*

A directed acyclic graph (DAG) is a collection of nodes $\{G_1, \ldots, G_{n_G}\}$, with each node $G_i$ representing a collection of quantities $\{X_{i1}, \ldots, X_{im_i}\}$. Some of the nodes are joined by directed edges, subject to the condition that there can be no directed cycles. If a directed edge runs from node $G_i$ to node $G_j$, we say that node $G_i$ is a parent of node $G_j$, and that node $G_j$ is a child of node $G_i$; the set of parents of a node $G_i$ is denoted by $\mathrm{Pa}(G_i)$, and the set of its child nodes is denoted by $\mathrm{Ch}(G_i)$.
We assign numerical labels to the nodes as follows: we find any node with no parents and label this as node 1; then, we assign the labels $2, 3, \ldots, n_G$ in order to any node whose parents are all already numbered. Any ordering of the nodes in which it is not possible to reach a lower-numbered node from a higher-numbered node is referred to as an ordering which is consistent with the graph; the procedure described here allows us to construct at least one ordering which is consistent with any given DAG.
We can now define a directed Bayes linear (second-order) graphical model as follows.

**Definition B.1.** *A **directed Bayes linear graphical model** is a model in which, when $G_1, \ldots, G_{n_G}$ is a consistent node ordering, each node $G_k$ is separated by its parent nodes from all lower-numbered nodes in the list, i.e. that*

$$\lfloor G_k \perp G(k-1) \rfloor / Pa(G_k) \ ,$$

*where $G(j) = G_1 \cup \ldots \cup G_j$ is the collection of predecessor nodes.*

The following belief separation property follows directly from the structure of the graph.

**Theorem B.1.** *On the DAG with (consistently ordered) nodes $G_1, \ldots, G_{n_G}$, the covariance between the sets of quantities $X_j$ and $X_k$ represented by the pair of nodes $G_j$ and $G_k$, with $j < k$ is*

$$Cov[X_k, X_j] = Cov[X_k, Pa(G_k)]\, Var[Pa(G_k)]^{-1} Cov[Pa(G_k), X_j] \ .$$

*B.2. Undirected graphical models*

Belief separation judgements can also be represented using undirected graphical models. An undirected graph is one in which the edges between nodes have no associated direction. While it is generally intuitively more difficult to specify an undirected graphical model directly, it is straightforward to construct an undirected graph which preserves important aspects of the belief structure imposed by a given DAG. When adjusting beliefs, it is more natural to work with the undirected graph, as its global Markov property is preserved under belief adjustment.
On an undirected graph, we say that a collection of nodes $W$ separates collections of nodes $U$ and $V$ if all paths from nodes in $U$ to nodes in $V$ pass through a node in $W$. The undirected graphical model associated with a particular directed graphical model is known as the moral graph, and is constructed as follows.

**Definition B.2.** *The **moral graph** associated with a particular directed graphical model is constructed by:*

- *'marrying the parents', i.e. drawing an (undirected) arc between all pairs of nodes which share a common child node;*
- *dropping all edge directions.*

For the moral graph, we have the following belief separation structure

**Theorem B.2.** *If, for three collections of nodes $U$, $V$ and $W$, $W$ separates $U$ from $V$ on the moral graph corresponding to our original DAG, then $\lfloor U \perp V \rfloor / W$.*

*B.3. Junction trees*

Directed and undirected trees have attractive computational properties, and it is always possible to convert a given directed graphical model into a type of undirected tree known as a junction tree, using the following procedure.

**Definition B.3.** *We construct a **junction tree** corresponding to a particular DAG using the following procedure:*

1. ***Construct the moral graph** corresponding to the DAG as outlined in definition B.2;*
2. ***Triangulate the graph**: add in sufficient edges to ensure that there are no cycles of length 4 or more without a chord;*
3. ***Perform a maximum cardinality search**:*
   - *Choose an arbitrary node and label it as node 1;*
   - *Assign the labels $2, 3, \ldots, n_G$ sequentially to the node on the graph with the largest number of labelled neighbours.*

   *If and only if the graph is triangulated, then each time we label a new node, the labelled neighbours of this node will form a complete graph (they will all be neighbours of each other).*
4. ***Identify and label the cliques**: the cliques are the maximal sets of nodes which form complete sub-graphs. We label the cliques in order of the highest-labelled node that they contain.*
5. ***Create the junction tree**: Each clique is a node in the junction tree. The intersection of the nodes in a clique with the nodes in all lower-numbered cliques will be contained within at least one of the lower-numbered cliques; we draw an edge between the clique and one of the lower-numbered cliques which contains the intersection.*

*Covariances between different cliques.* For any two adjacent cliques $A$ and $B$ on the junction tree, we have the following property: if $Z = A \cap B$ is the set of nodes belonging to both cliques, and $U = A \setminus Z$, $V = B \setminus Z$, then $\lfloor U \perp V \rfloor / Z$, since $Z$ separates $U$ and $V$ on the corresponding undirected graph. For nodes $U_i$ and $V_j$, therefore, we have that $\mathrm{Cov}\left[U_i, V_j\right] = \mathrm{Cov}\left[U_i, Z\right] \mathrm{Var}\left[Z\right]^{-1} \mathrm{Cov}\left[Z, V_j\right]$. This property forms the basis of an efficient algorithm for updating all nodes on the graph: when a node $D$ is observed, we compute $\mathrm{Cov}\left[G_i, D\right]$ for each node by propagating the covariance through the intersections of the cliques. Appendix B.4 illustrates this procedure through application to a simple example.

*Implementation.* For moderately complex DAGs (e.g. the one displayed in Figure 1, with most of the links between numerical discrepancy/structural discrepancy components retained), the operations of triangulating the moral graph and subsequently identifying the cliques would be difficult to perform manually. When implementing an analysis of this kind, there are a number of algorithms which can be used to perform these tasks. For the triangulation operation, Heggernes [19] provides a survey of a number of different triangulation algorithms, and Berry et al. [2] present some recent work in this area. For identifying the cliques of the triangulated graph, the Bron-Kerbosch algorithm is a popular choice: Bron and Kerbosch [4] first presented the algorithm, and more modern treatments are due to, e.g. Himmel et al. [21], Kazals and Karande [24].

*B.4. Example: adjustment on a junction tree*

We illustrate the procedure for adjusting moments on a junction tree in application to a simple example. Figure B.7 shows a DAG representing a particular belief structure for a 7-variable model. Also shown is the corresponding moral graph, in which the parents $B$ and $C$ of $D$ and the parents $E$ and $F$ of $G$ have been connected, and all edge directions have been dropped. Note that this graph happens to be already triangulated- it can be seen that there are no cycles of length 4 or more without a chord. A schematic of the corresponding junction tree is also shown: the 5 cliques are $Q_1 = \{A, B, C\}$, $Q_2 = \{B, C, D\}$, $Q_3 = \{C, D, F\}$, $Q_4 = \{D, E, F\}$ and $Q_5 = \{E, F, G\}$. One possible node labelling resulting from the maximum cardinality search is for the nodes to be labelled in alphabetical order (or reverse alphabetical order): in any case, the cliques will be linked together in a chain to form the junction tree, as shown in Figure B.7. Suppose that $G$ has been observed, and we wish to compute the adjusted moments

$$\begin{aligned} \mathrm{E}_G[A] &= \mathrm{E}[A] + \mathrm{Cov}[A, G]\,\mathrm{Var}[G]^{-1}\big[G - \mathrm{E}[G]\big] , \\ \mathrm{Var}_G[A] &= \mathrm{Var}[A] - \mathrm{Cov}[A, G]\,\mathrm{Var}[G]^{-1}\mathrm{Cov}[G, A] . \end{aligned}$$

To evaluate either of these quantities, $\mathrm{Cov}[A, G]$ is required; this is not specified directly, and must be computed using the graph.
$G$ is a member of clique $Q_5$ only, and $A$ is a member of clique $Q_1$ only; therefore, to obtain adjusted beliefs about $A$, the adjustment is propagated along the chain of cliques from $Q_5$ to $Q_1$. Using the belief separation properties of the junction tree, the covariance between $G$ and any member of $Q_4 = \{D, E, F\}$ can be computed: for example, in the case of $D$

$$\mathrm{Cov}[D, G] = \mathrm{Cov}[D, S_{45}]\,\mathrm{Var}[S_{45}]^{-1}\mathrm{Cov}[S_{45}, G] ,$$

where $S_{45} = \{E, F\}$ is the subset of nodes which separates the cliques $Q_4$ and $Q_5$. In the same way, the covariance between any node in clique $Q_3 = \{C, D, F\}$ and $G$ can now be computed: for example, in the case of $C$

$$\mathrm{Cov}[C, G] = \mathrm{Cov}[C, S_{34}]\,\mathrm{Var}[S_{34}]^{-1}\mathrm{Cov}[S_{34}, G] ,$$

and since, by definition, the members of the separating subset $S_{34} = \{D, F\}$ are also members of clique $Q_4 = \{D, E, F\}$, from above we have that

$$\mathrm{Cov}[C, G] = \mathrm{Cov}[C, S_{34}]\,\mathrm{Var}[S_{34}]^{-1}\mathrm{Cov}[S_{34}, S_{45}]\,\mathrm{Var}[S_{45}]^{-1}\mathrm{Cov}[S_{45}, G] .$$

Working along the chain of cliques in the tree in this way gives the result

$$\begin{aligned} \mathrm{Cov}[A, G] = {} & \mathrm{Cov}[A, S_{12}]\,\mathrm{Var}[S_{12}]^{-1}\mathrm{Cov}[S_{12}, S_{23}]\,\mathrm{Var}[S_{23}]^{-1} \\ & \times \mathrm{Cov}[S_{23}, S_{34}]\,\mathrm{Var}[S_{34}]^{-1}\mathrm{Cov}[S_{34}, S_{45}]\,\mathrm{Var}[S_{45}]^{-1}\mathrm{Cov}[S_{45}, G] . \end{aligned}$$

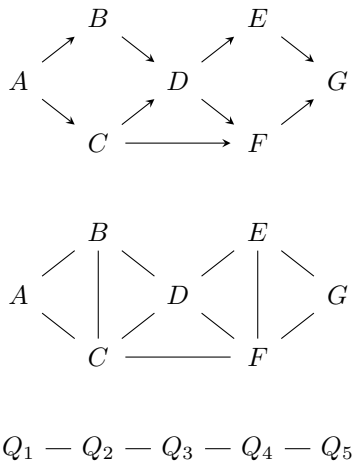



Figure B.7: The DAG used in the example from Appendix B.4, along with the corresponding moral graph and a corresponding junction tree. The cliques of the junction tree are $Q_1 = \{A, B, C\}$, $Q_2 = \{B, C, D\}$, $Q_3 = \{C, D, F\}$, $Q_4 = \{D, E, F\}$ and $Q_5 = \{E, F, G\}$.

## C. Computing the full prior specification

We provide details of the moments that must be computed in order to characterise the prior corresponding to the DAG 1. Having made the limited prior specification for the initial state $u(t_0)$ and the best input parameters $\xi^*$ as described in Section 4.2, we work through the solution trajectory, computing the moments of the solver, numerical discrepancy and solution at each stage. We cycle through the following steps for each time $t_k$, $k = 0, 1, 2, \ldots, (n_t - 1)$

*Solver.* The expectation of the solution component $\hat{u}_i(t_{k+1}) = \hat{u}_i(u(t_k), \xi^*)$ is computed as

$$\mathrm{E}\left[\hat{u}_i(t_{k+1})\right] = \mathrm{E}\left[\mathrm{E}\left[\hat{u}_i(u(t_k), \xi^*) \middle| u(t_k), \xi^*\right]\right],$$

and the covariance between any pair of solution components is

$$\begin{aligned}\mathrm{Cov}\left[\hat{u}_i(t_{k+1}), \hat{u}_j(t_{k+1})\right] = {} & \mathrm{E}\left[\mathrm{Cov}\left[\hat{u}_i(u(t_k), \xi^*), \hat{u}_j(u(t_k), \xi^*) \middle| u(t_k), \xi^*\right]\right] \\ & + \mathrm{Cov}\left[\mathrm{E}\left[\hat{u}_i(u(t_k), \xi^*) \middle| u(t_k), \xi^*\right], \mathrm{E}\left[\hat{u}_j(u(t_k), \xi^*) \middle| u(t_k), \xi^*\right]\right].\end{aligned}$$

The outer expectations and covariances are taken with respect to $\{u(t_k), \xi^*\}$. For solvers which contain polynomial terms of order 2 or greater, or other non-linear functions, we must make assumptions about higher-order moments of $\{u(t_k), \xi^*\}$ if we are to compute theses expectations and covariances. These moments can be specified directly, or be obtained from a suitable probability distribution $p(u(t_k), \xi^*)$ characterised using the second order moments of $\{u(t_k), \xi^*\}$.

*Numerical discrepancy.* For the numerical discrepancy $\eta_i(t_{k+1}) = \eta_i(u(t_k), \xi^*)$, we have

$$\mathrm{E}\left[\eta_i(t_{k+1})\right] = \mathrm{E}\left[\mathrm{E}\left[\eta_i(u(t_k), \xi^*) \middle| u(t_k), \xi^*\right]\right],$$

and the covariance between two components of the numerical discrepancy is

$$\begin{aligned}\mathrm{Cov}\left[\eta_i(t_{k+1}), \eta_j(t_{k+1})\right] = {} & \mathrm{E}\left[\mathrm{Cov}\left[\eta_i(u(t_k), \xi^*), \eta_j(u(t_k), \xi^*) \middle| u(t_k), \xi^*\right]\right] \\ & + \mathrm{Cov}\left[\mathrm{E}\left[\eta_i(u(t_k), \xi^*) \middle| u(t_k), \xi^*\right], \mathrm{E}\left[\eta_j(u(t_k), \xi^*) \middle| u(t_k), \xi^*\right]\right].\end{aligned}$$

Again, the outer expectations and covariances in these expressions are taken with respect to $\{u(t_k), \xi^*\}$. The covariances of the discrepancy components with their parent nodes are

$$\mathrm{Cov}\left[\eta_i(t_{k+1}), u_j(t_k)\right] = \mathrm{E}\left[\mathrm{E}\left[\eta_i(u(t_k), \xi^*)\middle| u(t_k), \xi^*\right] u_j(t_k)\right] - \mathrm{E}\left[\eta_i(t_{k+1})\right]\mathrm{E}\left[u_j(t_k)\right] ,$$

and

$$\begin{aligned}\mathrm{Cov}\left[\eta_i(t_{k+1}), \eta_j(t_l)\right] = \mathrm{E}\Big[&\mathrm{Cov}\left[\eta_i(u(t_k), \xi^*), \eta_j(u(t_{l-1}), \xi^*)\middle| u(t_k), u(t_{l-1}), \xi^*\right]\Big] \\ &+ \mathrm{Cov}\left[\mathrm{E}\left[\eta_i(u(t_k), \xi^*)\middle| u(t_k), \xi^*\right], \mathrm{E}\left[\eta_j(u(t_{l-1}), \xi^*)\middle| u(t_{l-1}), \xi^*\right]\right] .\end{aligned}$$

*Solution.* Once the moments of $\hat{u}(t_{k+1})$ and $\eta(t_{k+1})$ have been computed, it is relatively straightforward to compute the expectation and covariance of the solution $u(t_{k+1}) = \hat{u}(t_{k+1}) + \eta(t_{k+1})$ at time $t_{k+1}$

$$\begin{aligned}\mathrm{E}\left[u_i(t_{k+1})\right] &= \mathrm{E}\left[\hat{u}_i(t_{k+1})\right] + \mathrm{E}\left[\eta_i(t_{k+1})\right] , \\ \mathrm{Cov}\left[u_i(t_{k+1}), u_j(t_{k+1})\right] &= \mathrm{Cov}\left[\hat{u}_i(t_{k+1}), \hat{u}_j(t_{k+1})\right] + \mathrm{Cov}\left[\hat{u}_i(t_{k+1}), \eta_j(t_{k+1})\right] \\ &\quad + \mathrm{Cov}\left[\eta_i(t_{k+1}), \hat{u}_j(t_{k+1})\right] + \mathrm{Cov}\left[\eta_i(t_{k+1}), \eta_j(t_{k+1})\right] .\end{aligned}$$

The covariances of the solution components with each of the components of its parent nodes in the DAG 1 are similarly easy to compute

$$\begin{aligned}\mathrm{Cov}\left[u_i(t_{k+1}), \hat{u}_j(t_{k+1})\right] &= \mathrm{Cov}\left[\hat{u}_i(t_{k+1}), \hat{u}_j(t_{k+1})\right] + \mathrm{Cov}\left[\eta_i(t_{k+1}), \hat{u}_j(t_{k+1})\right] , \\ \mathrm{Cov}\left[u_i(t_{k+1}), \eta_j(t_{k+1})\right] &= \mathrm{Cov}\left[\hat{u}_i(t_{k+1}), \eta_j(t_{k+1})\right] + \mathrm{Cov}\left[\eta_i(t_{k+1}), \eta_j(t_{k+1})\right] .\end{aligned}$$

**Algorithm 1** Compute second-order prior specification corresponding to the DAG 1; details of the individual calculations are provided in Section 4.2.

Make limited prior specification $\{\mathrm{E}[u_i(t_0)]\}$, $\mathrm{Cov}[u_i(t_0), u_j(t_0)]$, $\{\mathrm{E}[\xi_i^*]\}$ and $\mathrm{Cov}\left[\xi_i^*, \xi_j^*\right]$.
**for** $k = 0, 1, \ldots, (n_t - 1)$ **do**
  **Solver**
    Compute $\{\mathrm{E}[\hat{u}_i(t_{k+1})]\}$, $\mathrm{Cov}[\hat{u}_i(t_{k+1}), \hat{u}_j(t_{k+1})]\}$ for $i, j = 1, \ldots, n_u$
    Compute $\{\mathrm{Cov}[\hat{u}_i(t_{k+1}), u_j(t_k)]\}\}$ for $i, j = 1, \ldots, n_u$
  **Numerical discrepancy**
    Compute $\{\mathrm{E}[\eta_i(t_{k+1})]\}$, $\mathrm{Cov}[\eta_i(t_{k+1}), \eta_j(t_{k+1})]\}$ for $i, j = 1, \ldots, n_u$
    Compute $\{\mathrm{Cov}[\eta_i(t_{k+1}), u_j(t_k)]\}\}$ for $i, j = 1, \ldots, n_u$
    Compute $\{\mathrm{Cov}[\eta_i(t_{k+1}), \eta_i(t_l)]\}\}$ for $i, j = 1, \ldots, n_u$, $l = 1, \ldots, k$
  **Solution**
    Compute $\{\mathrm{E}[u_i(t_{k+1})]\}$, $\mathrm{Cov}[u_i(t_{k+1}), u_j(t_{k+1})]$ for $i, j = 1, \ldots, n_u$
    Compute $\{\mathrm{Cov}[u_i(t_{k+1}), \eta_j(t_{k+1})]\}$ for $i, j = 1, \ldots, n_u$
    Compute $\{\mathrm{Cov}[u_i(t_{k+1}), \hat{u}_j(t_{k+1})]\}$ for $i, j = 1, \ldots, n_u$
**end for**

## D. Numerical discrepancy modelling

This appendix gives details relating to the numerical discrepancy fitting procedure (Section 4.1) as applied to the projectile trajectory example presented in Section 5. Each of the steps outlined in Section 4.1 is described below:

1. We specify that $s(h) = h$ and that $\sigma_{\eta_i} = 1$, so that the standard deviation of the numerical discrepancy is $h$, the time-step length. This value of $\sigma_{\eta_i}$ is selected manually to ensure that the refined solution $u^*$ lies within the error bars $\hat{u}(\psi) \pm 2h$ for all input settings and sampled trajectories.

2. We generate a set of 500 inputs $\psi = \{u(t), t, t+h, \xi\}$ for data-generation by using a Latin hypercube in $\{t, t+h, \xi\}$, and by evolving from $u(0) = 0$ to $t$ on the time scale of the coarse grid.
3. For each setting of $\psi$ generated in step 2, we obtain the corresponding numerical solution $\hat{u}(\psi(t))$ using the Euler scheme.
4. For each setting of $\psi$ from step 2, we also sub-divide the interval $[t, t+h]$ into $n^* = 500$ evenly-spaced intervals, and obtain samples of $u^*(\psi(t))$ for each input setting by evolving from $t$ to $t+h$ on the refined grid, at each step including a mean-zero Gaussian numerical discrepancy with standard deviation as specified in 1.
5. For each input setting, we sample $u^*$ 100 times by repeating step 4.
6. The samples generated in 5 are used to empirically determine the moments $\tilde{\mathrm{E}}[\eta(\psi(t))]$ and $\widetilde{\mathrm{Var}}[\eta(\psi(t))]$ at each of the input settings fixed in 2.

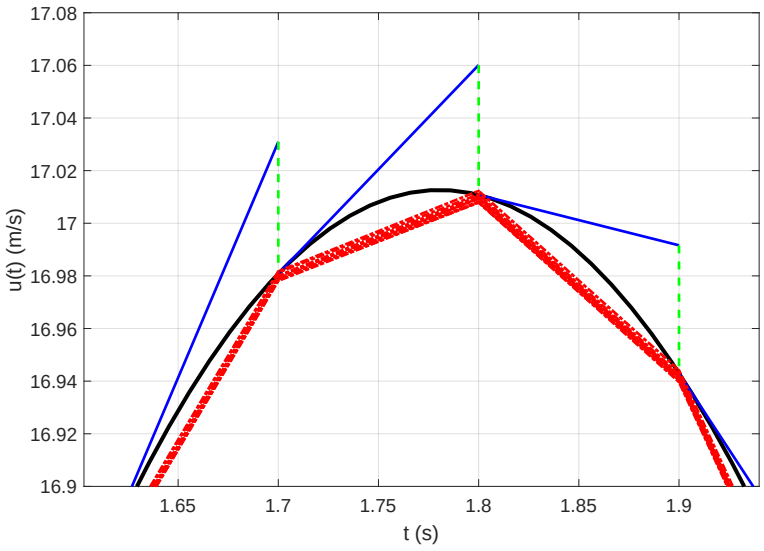
The moments $\tilde{\mathrm{E}}[\eta(\psi(t))]$ and $\widetilde{\mathrm{Var}}[\eta(\psi(t))]$ generated for specific input settings $\psi$ using this procedure are then used as data to adjust beliefs about the numerical discrepancy over the whole space of possible inputs. The expectations obtained in step 6 are treated as observations of the numerical discrepancy at the corresponding inputs, with the variances from step 6 treated as measurement error variances.
In more detail, the data generated using the above procedure are used to adjust the moments of the numerical discrepancy model (7). First, a set of 50 points is used to generate the prior moments for the regression coefficients $\{\beta_j\}$ and an estimate of the marginal variance $V$ of the residual process. Then, a set of 500 points is used to jointly adjust the moments of the regression coefficients and the residual process. Denoting the set of calculated $\tilde{\mathrm{E}}[\eta(\psi(t))]$ values by $D$, the resulting adjusted expectation for an evaluation of the numerical discrepancy $\eta$ at any input point $\psi$ is

$$\mathrm{E}_D[\eta(\psi)] = \sum_j \mathrm{E}_D[\beta_j] b_j(\psi) + \mathrm{E}_D[r(\psi)] ,$$

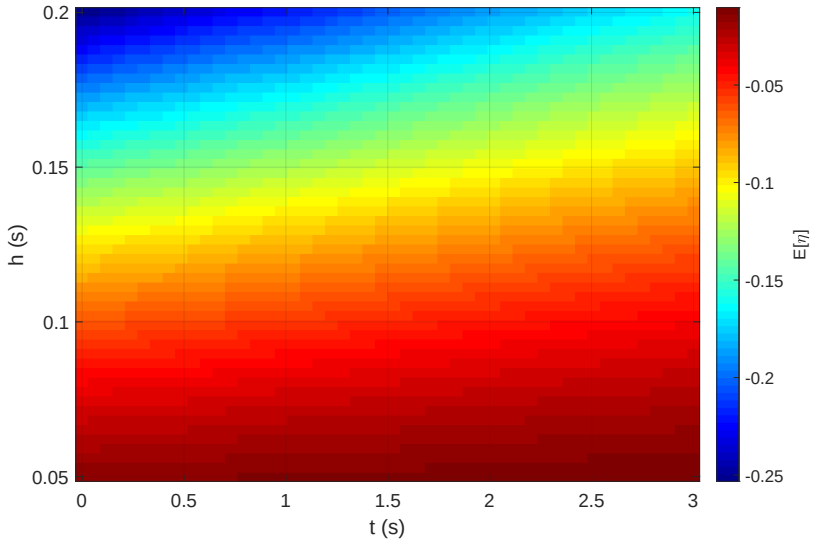
and the covariance between evaluations of $\eta$ at inputs $\psi$ and $\psi'$ is

$$\begin{aligned}\mathrm{Cov}_D[\eta(\psi), \eta(\psi')] = &\sum_{j,l} \mathrm{Cov}_D[\beta_j, \beta_l] b_j(\psi) b_l(\psi') + \sum_j \mathrm{Cov}_D[\beta_j, r(\psi')] b_j(\psi) \\ &+ \sum_l \mathrm{Cov}_D[r(\psi), \beta_l] b_l(\psi') + \mathrm{Cov}_D[r(\psi), r(\psi')] .\end{aligned}$$

Recall from Section 4.1 that the $\{b_j(\psi(t))\}$ are the (deterministic) basis functions (generally chosen to be higher-order terms from the Taylor expansion of the solution). Once the moments of the model have been adjusted, a final set of 100 data points is used to test the quality of the model predictions. The adjusted moments are used as input to the sampling procedure used to characterise the moments for the full model, detailed in Section 4.2.
Aspects of the fitted numerical discrepancy model are illustrated in Figure D.8. Figure 8(a) demonstrates how the model for $\eta$ is used to apply a correction to the numerical solution at successive time-steps, in order to generate more accurate predictions for the true solution. Figure 8(b) shows the mean prediction generated by the fitted model for a grid of values of $t$ and $h$.




(a)




(b)

Figure D.8: Illustration of the numerical discrepancy model for the projectile trajectory example presented in Section 5 and referred to in Appendix D. Figure 8(a) demonstrates the relationship between the true solution, the numerical approximation and the numerical discrepancy model. The true solution (for $\gamma = 0.15$ and $\dot{u}(t_0) = 20$) in this time interval is shown in black; numerical approximations $\hat{u}$ are generated at intervals of $h = 0.1$ seconds are generated (starting from the true solution in each case), and the corresponding trajectories are shown in blue; the mean correction $\mathrm{E}[\eta]$ applied by the numerical discrepancy model is shown in dashed green; and 10 trajectories generated by sampling numerical discrepancy values from a Gaussian distribution are shown as red dashed lines. Figure 8(b) shows the expectation $\mathrm{E}[\eta]$ of the numerical discrepancy for a range of different $\{t, h\}$ values (assuming evolution begins from the true solution in each case).

## References


[1] A. Abdulle and G. Garegnani. Random time step probabilistic methods for uncertainty quantification in chaotic and geometric numerical integration. *Statistics and Computing*, 2020. ISSN 15731375. doi: 10.1007/s11222-020-09926-w. URL https://doi.org/10.1007/s11222-020-09926-w.

[2] A. Berry, E. Dahlhaus, P. Heggernes, and G. Simonet. Sequential and parallel triangulating algorithms for elimination game and new insights on minimum degree. *Theoretical Computer Science*, 409:601–616, 2008.

[3] J. Blazek. *Computational Fluid Dynamics: Principles and Applications*. Elsevier Science, 2005. ISBN 9780080445069. doi: doi.org/10.1016/B978-008044506-9/50004.

[4] C. Bron and J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16: 575–577, 1973. doi: 10.1145/362342.362367.

[5] J. Brynjarsdottir and A. O'Hagan. Learning about physical parameters: the importance of model discrepancy. *Inverse Problems*, xx:1–21, 2014. ISSN 13616420. doi: 10.1088/0266-5611/30/11/114007.

[6] J.C. Butcher. Coefficients for the study of Runge-Kutta integration processes. *Journal of the Australian Mathematical Society*, 3:185–201, 1963. doi: doi.org/10.1017/S1446788700027932.

[7] O. A. Chkrebtii, D. A. Campbell, B. Calderhead, and M. A. Girolami. Bayesian solution uncertainty quantification for differential equations. *Bayesian Analysis*, 11(4):1269–1273, 2016. ISSN 19316690. doi: 10.1214/16-BA1036.

[8] J. Cockayne, C. J. Oates, T. Sullivan, and M. Girolami. Probabilistic meshless methods for partial differential equations and Bayesian inverse problems. *arXiv preprint arXiv:1605.07811*, pages 1–52, 2016.

[9] J. Cockayne, C. Oates, T. Sullivan, and M. Girolami. Bayesian probabilistic numerical methods. *Forthcoming*, 2017. URL https://arxiv.org/pdf/1702.03673.pdf.

[10] P. R. Conrad, M. Girolami, S. Särkkä, A. Stuart, and K. Zygalakis. Statistical analysis of differential equations: introducing probability measures on numerical solutions. *Statistics and Computing*, 27(4):1065–1082, Jul 2017. ISSN 1573-1375. doi: 10.1007/s11222-016-9671-0. URL https://doi.org/10.1007/s11222-016-9671-0.

[11] P. S. Craig, M. Goldstein, J. C. Rougier, and A. H. Seheult. Bayesian forecasting for complex systems using computer simulators. *Journal of the American Statistical Association*, 96(454):717–729, 2001. ISSN 0162-1459. doi: 10.1198/016214501753168370.

[12] P. Diaconis. Bayesian numerical analysis. In S. Gupta J. Berger, editor, *Statistical Decision Theory and Related Topics IV*, pages 163–175. Springer-Verlag, 1988.

[13] Dove. Dove's Guide: Durham Cathedral, 2015. URL http://dove.cccbr.org.uk/detail.php?searchString=Durham+Cath{&}Submit=+Go+{&}DoveID=DURHAM.

[14] R. Eymard, T. Gallouet, and R. Herbin. Finite Volume Methods: Schemes and Analysis. Technical report, 2008.

[15] M. Goldstein and J. C. Rougier. Probabilistic formulations for transferring inferences from mathematical models to physical systems. *SIAM Journal on Scientific Computing*, 26(0):467–487, 2004. ISSN 1064-8275. doi: 10.1137/S106482750342670X. URL http://dx.doi.org/10.1137/S106482750342670X.

[16] M. Goldstein and D.J. Wilkinson. Bayes linear analysis for graphical models: The geometric approach to local computation and interpretive graphics. *Statistics and Computing*, 10:311–324, 2000. doi: https://doi.org/10.1023/A:1008977409172.

[17] M. Goldstein and D. Wooff. *Bayes Linear Statistics: Theory and Methods*. Wiley, Chichester, 2007.

[18] T. Graepel. Solving noisy linear operator equations by Gaussian processes: application to ordinary and partial differential equations. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, page 234–241. AAAI Press, 2003. ISBN 1577351894.

[19] P. Heggernes. Minimal triangulations of graphs: A survey. *Discrete Mathematics*, 306:297–317, 2006. doi: https://doi.org/10.1016/j.disc.2005.12.003.

[20] P. Hennig, M.A. Osborne, and M. Girolami. Probabilistic numerics and uncertainty in computations. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 471, 2015.

[21] A.S. Himmel, H. Molter, R. Niedermeier, and M. Sorge. Enumerating maximal cliques in temporal graphs. pages 337–344, 2016.

[22] Arieh Iserles. *A first course in the numerical analysis of differential equation*. 2008. ISBN 9780521734905.

[23] M.J. Jones. bayes-linear-odes. https://github.com/mjj89/bayes-linear-odes, 2020.

[24] F. Kazals and C. Karande. A note on the problem of reporting maximal cliques. *Theoretical Computer Science*, 407: 564–568, 2008.

[25] M C Kennedy and A O'Hagan. Bayesian calibration of computer models. *J. R. Stat. Soc. Ser. B*, 63:425–464, 2001. URL http://dx.doi.org/10.1111/1467-9868.00294.

[26] H. Kersting and P. Hennig. Active uncertainty calibration in Bayesian ODE solvers. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'16, page 309–318, Arlington, Virginia, USA, 2016. AUAI Press. ISBN 9780996643115.

[27] T. W. B. Kibble and F. H. Berkshire. *Classical Mechanics*. Imperial College Press, London, 2004. ISBN 978-1-86094-435-2.

[28] F . M . Larkin. Gaussian measure in Hilbert space and applications in numerical analysis. *The Rocky Mountain Journal of Mathematics*, 2:379–421, 1972.

[29] S.L. Lauritzen. *Graphical Models*. Oxford Statistical Science. Clarendon Press, 1996. ISBN 9780191591228.

[30] R.J. LeVeque. *Numerical Methods for Conservation Laws*. Birkhauser, 1992. ISBN 0-8176-2723-5.

[31] H. Lie Cheng, A. M. Stuart, and T. J. Sullivan. Strong convergence rates of probabilistic integrators for ordinary differential equations. *Statistics and Computing*, 29(6):1265–1283, 2019. ISSN 15731375. doi: 10.1007/s11222-019-09898-6.

[32] H. Mohammadi, P. Challenor, and M. Goodfellow. Emulating dynamic non-linear simulators using Gaussian processes.

*Computational Statistics and Data Analysis*, 139:178–196, 2019. ISSN 01679473. doi: 10.1016/j.csda.2019.05.006. URL https://doi.org/10.1016/j.csda.2019.05.006.

[33] C. J. Oates and T. J. Sullivan. A modern retrospective on probabilistic numerics. *Statistics and Computing*, 29(6):1335–1351, 2019. ISSN 15731375. doi: 10.1007/s11222-019-09902-z. URL https://doi.org/10.1007/s11222-019-09902-z.

[34] C. J. Oates, J. Cockayne, R. G. Aykroyd, and M. Girolami. Bayesian probabilistic numerical methods in time-dependent state estimation for industrial hydrocyclone equipment. *Journal of the American Statistical Association*, 114(528):1518–1531, 2019. ISSN 1537274X. doi: 10.1080/01621459.2019.1574583. URL https://doi.org/10.1080/01621459.2019.1574583.

[35] A. O'Hagan. Monte Carlo is fundamentally unsound. *Journal of the Royal Statistical Society. Series D*, 36(2):247–249, 1987.

[36] A. O'Hagan. Bayes–Hermite quadrature. *Journal of Statistical Planning and Inference*, 29(3):245–260, 1991. ISSN 03783758. doi: 10.1016/0378-3758(91)90002-V. URL http://www.sciencedirect.com/science/article/pii/037837589190002V.

[37] F. Pukelsheim. The three sigma rule. *The American Statistician*, 48(2):88–91, 1994. doi: 10.1080/00031305.1994.10476030. URL http://www.jstor.org/stable/2684253{%}5Cnpapers2://publication/uuid/9ACCD11F-DDFC-4267-84CB-68C447DC1CCC.

[38] C.E. Rasmussen and C.K.I. Williams. *Gaussian processes for machine learning*. The MIT Press, 2005. ISBN 026218253X. URL http://www.gaussianprocess.org/gpml/.

[39] M. Schober, D.K. Duvenaud, and P. Hennig. Probabilistic ODE solvers with Runge-Kutta means. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, page 739–747, Cambridge, MA, USA, 2014. MIT Press.

[40] M. Schober, S. Särkkä, and P. Hennig. A probabilistic model for the numerical solution of initial value problems. *Statistics and Computing*, Jan 2018. ISSN 1573-1375. doi: 10.1007/s11222-017-9798-7. URL https://doi.org/10.1007/s11222-017-9798-7.

[41] J. Skilling. Bayesian solution of ordinary differential equations. In C. R. Smith, G. J. Erickson, and P. O. Neudorfer, editors, *Maximum Entropy and Bayesian Methods: Seattle, 1991*, pages 23–37. Springer Netherlands, Dordrecht, 1992. ISBN 978-94-017-2219-3. doi: 10.1007/978-94-017-2219-3_2. URL https://doi.org/10.1007/978-94-017-2219-3_2.

[42] R. Smith and H. Hunt. Vibration of bell towers excited by bell ringing — a new approach to analysis. In *International Conference on Noise and Vibration Engineering*, 2008. ISBN 9781615671915. URL http://www2.eng.cam.ac.uk/{~}hemh1/isma2008.pdf.

[43] J. M. Stockie. The mathematics of atmospheric dispersion modeling. *SIAM Review*, 53:349–372, 2011. ISSN 0036-1445. doi: 10.1137/10080991X. URL http://epubs.siam.org/doi/pdf/10.1137/10080991X{%}5Cnpapers2://publication/uuid/75D838F0-6E28-4CFA-965F-568BCC65CA79.

[44] A. V. Suldin. The method of regression in the theory of approximation. *Uchenye Zapiski Kazanskogo Universiteta (Book 6)*, 123:3–35, 1963.

[45] O. Teymur, K. Zygalakis, and B. Calderhead. Probabilistic linear multistep methods. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4321–4328. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6356-probabilistic-linear-multistep-methods.pdf.

[46] O. Teymur, B. Calderhead, H. Cheng Lie, and T. J. Sullivan. Implicit probabilistic integrators for ODEs. *Advances in Neural Information Processing Systems*, 2018-December(NeurIPS):7244–7253, 2018. ISSN 10495258.

[47] F. Tronarp, H. Kersting, S. Särkkä, and P. Hennig. Probabilistic solutions to ordinary differential equations as nonlinear Bayesian filtering: a new perspective. *Statistics and Computing*, 29(6):1297–1315, 2019. ISSN 15731375. doi: 10.1007/s11222-019-09900-1. URL https://doi.org/10.1007/s11222-019-09900-1.

[48] G. K. Vallis. *Atmospheric and Oceanic Fluid Dynamics*. Cambridge University Press, Cambridge, U.K., 2006.

[49] I. Vernon, M. Goldstein, and R. G. Bower. Galaxy formation: a Bayesian uncertainty analysis. *Bayesian Analysis*, 5(4):619–669, 2010. ISSN 1936-0975. doi: 10.1214/10-BA524. URL http://projecteuclid.org/euclid.ba/1340110846.