Approximate Latent Force Model Inference

Jacob D. Moss*
Computer Lab
University of Cambridge
Cambridge, UK
jm2311@cam.ac.uk

Felix L. Opolka
Computer Lab
University of Cambridge
Cambridge, UK
flo23@cam.ac.uk

Bianca Dumitrascu
Computer Lab
University of Cambridge
Cambridge, UK
bmd39@cam.ac.uk

Pietro Liò
Computer Lab
University of Cambridge
Cambridge, UK
pl219@cam.ac.uk

Abstract

Physically-inspired latent force models offer an interpretable alternative to purely data driven tools for inference in dynamical systems. They carry the structure of differential equations and the flexibility of Gaussian processes, yielding interpretable parameters and dynamics-imposed latent functions. However, the existing inference techniques rely on the exact computation of posterior kernels which are seldom available in analytical form. Applications relevant to practitioners, such as diffusion equations, are hence intractable. We overcome these computational problems by proposing a variational solution to a general class of non-linear and parabolic partial latent force models. We demonstrate the efficacy and flexibility of our framework by achieving competitive performance on several tasks.

1 Introduction & Background

A vast variety of phenomena in the sciences, from systems of interacting particles to the aerodynamics of aircraft, are represented by differential equations. Machine learning approaches such as neural ordinary differential equations (ODEs) [Chen et al., 2018] and non-parametric ODEs [Heinonen et al., 2018] have become popular black-box models. However, the lack of an explicit equation means that these methods lose intepretability and the ability to infer latent terms. On the other hand, Latent Force Models (LFMs) [Lawrence et al., 2007, Alvarez et al., 2009] are a hybrid paradigm combining the structure and extrapolation of explicit models with the flexibility of Gaussian processes (GPs). This fusion is a powerful approach to problems in biophysics [López-Lopera et al., 2019, Moss and Lió, 2020] and stochastic control [Särkkä et al., 2018, Cheng et al., 2020]. However, they often require a intractable integration of the kernel function, severely limiting their usefulness.

Addressing these drawbacks, we introduce *Alfi*, a flexible variational framework for parameter estimation in ordinary and partial differential equation (PDE)-based LFMs. The main contributions are: (i) a gradient-matching step followed by fine-tuning with the full solution, enabling quadratic-time inference in LFMs; and (ii) inference of latent forces from partially known dynamics in temporal and spatio-temporal settings without onerous kernel derivations.

Latent Force Models (LFMs) are differential equation models which leverage explicit dynamics to infer latent forcing terms [Lawrence et al., 2007, Alvarez et al., 2009]. The differential equation, h,

*Correspondence to Jacob Moss jm2311@cam.ac.uk.

Fourth Workshop on Machine Learning and the Physical Sciences (NeurIPS 2021).

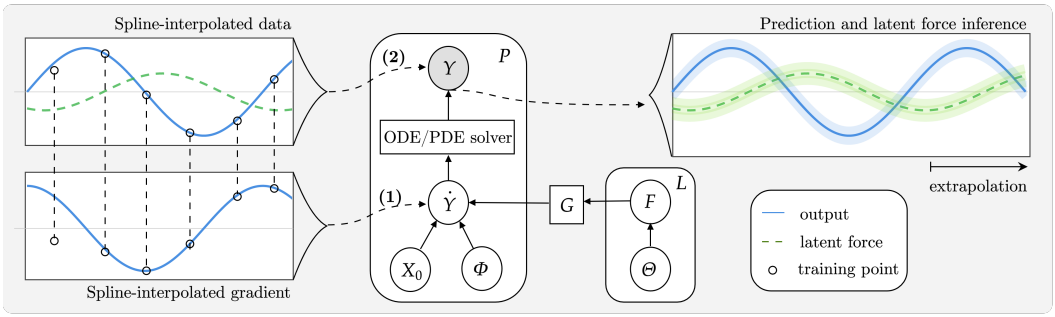


Figure 1: Computational graphical model for *Alfi*. **Step (1)** involves pre-estimating the parameters and latent forces of the differential equation in Eq. 1 by gradient-matching with the spline-interpolated derivative. **Step (2)** involves the more expensive forward solution of the differential equation and backpropagation through the ODE or PDE solver for fine-tuning the parameters.

parameterised by $\mathbf{\Theta}$, enforces a structural relationship between P outputs, $\mathbf{y}(x) \in \mathbb{R}^P$, N inputs, $\mathbf{x} \in \mathbb{R}^N$, and L unobserved latent forces, $\mathbf{f}(x) \in \mathbb{R}^L$. GP priors are assigned to the latent forces, f_i: $f_i \sim \mathcal{GP}(\mathbf{0}, \kappa_i(x, x'))$, and these forces can be mixed by some response function $G(\mathbf{f})$.

$$\overbrace{\mathcal{D}\,\mathbf{y}(\mathbf{x})}^{\text{differential}} = \overbrace{h\big(\mathbf{y}, \mathbf{x}; \mathbf{\Theta}, G(f_1(\mathbf{x}), ..., f_L(\mathbf{x}))\big)}^{\text{differential equation}}, \tag{1}$$

where $\mathcal{D}$ is a differential operator, for example an n$^{\text{th}}$ order (partial) derivative. The covariance between outputs, $\kappa_{\mathbf{y},\mathbf{y}'}(t, t')$, is analytically tractable if G is a linear operator. In these cases, maximum marginal likelihood yields the differential equation parameters and inference can be carried out with standard posterior GP identities (see Rasmussen and Williams [2005]).

2 Approximate Latent Force Models

We propose *Alfi*, an **a**pproximate **l**atent **f**orce **i**nference framework significantly expanding the class of latent force models. It supports any differentiable kernel, and is agnostic to the functional form of the response to latent forces, G. An example of a non-linear intractable response is a saturating function such as the Hill equation for modelling reaction kinetics in biophysical problems:

$$G(\mathbf{f}(t); \mathbf{w}_j) = \frac{\prod_i f_i(t)^{w_{ji}}}{\prod_i f_i(t)^{w_{ji}} + e^{-w_{j0}}} \tag{2}$$

where w_{ji} are interaction weights, incorporating cooperation/competition between the forces f_i.

2.1 Variational Objective

Let $\hat{\mathbf{Y}} \in \mathbb{R}^{N \times P}$ represent the forward solution of a differential equation at N points given parameters $\mathbf{\Phi}$, a differentiable transformation (G) of latent forces, and input $\mathbf{X} \in \mathbb{R}^{N \times D}$. The input dimensions are $D = 1$ (e.g. time) for ODEs, or $D > 1$ (e.g. time and space) for PDEs. Our formulation extends to arbitrary output dimensions, P. Let $\mathbf{F} = \left[\mathbf{f}_i^\top\right]_i^L$ be our set of independent latent forces where $\mathbf{f}_i \sim \mathcal{GP}(0, \kappa_{\mathbf{\Theta}}(\mathbf{x}, \mathbf{x}'))$ with kernel hyperparameters Θ. We assume a Gaussian likelihood, $p(\mathbf{Y}|\mathbf{F}, \mathbf{\Phi}) = \mathcal{N}(\mathbf{Y}|\hat{\mathbf{Y}}, \sigma^2)$, and seek to optimise parameters by marginalising over the latent forces:

$$p(\mathbf{Y}|\mathbf{\Phi}, \mathbf{\Theta}) = \int \overbrace{p(\mathbf{Y}|\mathbf{F}, \mathbf{\Phi})}^{\text{likelihood}} \prod_i^L \overbrace{p(\mathbf{f_i}|\mathbf{X}, \mathbf{\Theta})}^{\text{GP}} \, \mathrm{d}\mathbf{f}. \tag{3}$$

This marginal is intractable when G is non-linear, so we derive an expectation lower-bound (ELBO) for the marginal likelihood using a variational approximation. We add M inducing points, $\mathbf{u}_i \in \mathbb{R}^{M \times D}$, for each latent force: $\mathbf{U} = \left[\mathbf{u}_i^\top\right]_i^L$. Due to the conditional independence across the latent forces given any mixing parameters (e.g. Eq. 2), a factorised variational distribution is used:

$q(\mathbf{u}) = \prod_i^L \mathcal{N}(\mathbf{m}_i, \mathbf{C}_i)$ with variational parameters $\mathbf{m}_i \in \mathbb{R}^M$ and $\mathbf{C}_i \in \mathbb{R}^{M\times M}$. This yields the ELBO we seek to to optimise (the full derivation is in Appendix C):

$$\log p(\mathbf{Y}|\mathbf{\Phi}, \mathbf{\Theta}) = \mathbb{E}_{q(\mathbf{F})}\left[\log p(\mathbf{y}|\mathbf{F}, \mathbf{\Phi})\right] - \sum_{i=1}^{L} \mathcal{KL}(q(\mathbf{u}_i)||p(\mathbf{u}_i)), \tag{4}$$

where $q(\mathbf{F}) = \int p(\mathbf{F}|\mathbf{U})q(\mathbf{U})\mathrm{d}\mathbf{U}$ and we approximate the expected log-likelihood with Monte Carlo sampling. The variational posterior is $p(\mathbf{f}_i^*|\mathbf{u}_i) = \mathcal{N}(\mathbf{f}_i^*|\mathbf{m}_i^*, \mathbf{S}_i^*)$ where $\mathbf{m}_i^* = \mathbf{K}_{*M}\mathbf{K}_{MM}^{-1}\mathbf{m}_i$ and $\mathbf{S}_i^* = \mathbf{K}_{**} + \mathbf{K}_{*M}\mathbf{K}_{MM}^{-1}(\mathbf{S}_i - \mathbf{K}_{MM})\mathbf{K}_{MM}^{-1}\mathbf{K}_{M*}$ [Hensman et al., 2015].

2.2 Optimisation of Variational Objective

At each epoch, a batch of the latent forces is sampled from the variational posterior using the reparameterisation trick, enabling pathwise gradient estimation. Parameters are pre-estimated in the first few epochs, followed by fine-tuning with the forward solution. With e differential equation evaluations, the computational complexity is $O(NM^2 + e)$. Pseudocode along with sample code snippets are in Appendix B and E. We now detail these steps.

Step (1): Parameter pre-estimation by gradient matching. The forward solution of an ODE or PDE is the most computationally expensive component of our algorithm. As such, we develop a pre-estimation step to initialise the variational **and** differential equation parameters. We implement a gradient-matching algorithm whereby the differential equation is evaluated, with no calls to the solver, at all training points and compared with the interpolated gradient of the data at those points. We chose natural cubic splines for this purpose since they are linear in the number of points; more expensive alternative interpolations are discussed in Appendix B.

Step (2): Forward solution fine-tuning. This step involves executing an appropriate differential equation solver to generate samples from the LFM. For ODEs, we employ an ODE solver, through which the gradients are backpropagated. PDE-based LFMs are more involved since they are much harder to solve than in the ODE case. The GP component uses an anisotropic kernel where each dimension gets a separate lengthscale, since the latent force is likely to vary across time and space at different scales. We solve the PDE using the finite element method (FEM), which first requires converting it into its weak form, which is then solved on a discretised spatial mesh. The weak form is written in the domain-specific Unified Form Language (UFL) [Alnæs et al., 2014]. *Alfi* uses the *FEniCS* package [Logg et al., 2012, Alnæs et al., 2015] for solving the variational problems. The gradients can be determined with reverse-mode algorithmic differentiation using the adjoint equations, computed using the *dolfin-adjoint* package [Mitusch et al., 2019].

3 Experiments

Starting with a simulation of predator-prey dynamics, we demonstrate the ability to infer the prey dynamics without knowledge of prey abundance. We also investigate PDE-based LFMs with the spatio-temporal production of protein in fruit fly embryos with a reaction-diffusion equation.

ODE System: The Lotka-Volterra equations are a system of non-linear first-order differential equations used to model predator-prey dynamics. The population of predators, v, is related to the population of prey, u, with the following pair of equations:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \alpha u - \beta uv, \qquad \frac{\mathrm{d}v}{\mathrm{d}t} = \delta uv - \gamma v, \tag{5}$$

where α, δ are growth rates; β, γ are decay rates; and for clarity $u = u(t)$ and $v = v(t)$.

Assuming the population of prey is unknown and positive, we assign a Gaussian process prior to the function $u(t) \sim \mathcal{GP}(0, \kappa(t, t'))$ and a softplus response, $G(\bullet) = \log(1 + e^{\bullet})$, meaning the solution is intractable. While the the ubiquitous radial basis function (RBF), a universal smooth function approximator, would provide a good interpolation, a more sensible kernel would be periodic:

$$\kappa_{\mathrm{p}}(t, t') = \exp\left(\frac{2\sin^2(\pi|t - t'|)/p}{\ell^2}\right), \tag{6}$$

where p is the period parameter, determining the interval of time after which the function repeats. As seen in Figure 2, the periodic kernel enables realistic extrapolation beyond the training time range. As shown in Appendix G.2, an RBF kernel yields inferior results.

PDE System: Reaction-diffusion equations are second-order parabolic PDEs ubiquitous in the natural sciences. We take a model of gap gene protein production in Drosophila [López-Lopera et al., 2019]:

$$\frac{\partial u(x,t)}{\partial t} = \overbrace{Su(x,t) - \lambda y(x,t)}^{\text{reaction}} + \overbrace{D\frac{\partial^2 u(x,t)}{\partial x^2}}^{\text{diffusion}}, \tag{7}$$

where D is the diffusion constant(s), S is the production rate of the driving mRNA (the latent force) and λ is the decay rate. The output, y, is the protein concentration. We assign an anisotropic RBF kernel prior over the latent force u. Dirichlet boundary conditions are assigned such that $y(x=0,t)=0$ and $y(x=l,t)=0$ where 0 and l are the spatial boundaries.

We start with the weak form for this PDE by multiplying by a test function $v \in V$ and integrating the second derivatives by parts over the spatial domain, Ω.

$$\int_\Omega yv + \Delta t D \nabla_x y \cdot \nabla_x v \, \mathrm{d}\vec{x} = \int_\Omega (y^n + \Delta t S u - \Delta t \lambda y) v \, \mathrm{d}\vec{x}, \tag{8}$$

$$a(y,v) = L_{n+1}(v) \tag{9}$$

where $a(y,v)$ is a bilinear form, v is known as a test function belonging to some Banach space V, and function parameters have been omitted (i.e. $u = u(x,t)$). For completeness, the full derivation and transcription into UFL can be found in Appendix D and E.2.

The results are shown in Table 1, and the full qualitative outputs are in Appendix G.3. We note that when using the parameter pre-estimation procedure, the convergence is significantly sped up and the ELBO starts at a lower initial value, as shown in Figure 3.

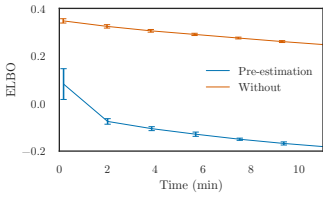


Figure 3: Pre-estimation: gradient-matching improves convergence.

4 Discussion

Alfi enables approximate inference in linear, non-linear, ordinary, and partial latent force models. The output of *Alfi* is a conditional Gaussian process, meaning the resolution, uniformity, and range of evaluation points is arbitrary. In addition, the error of the result is tunable using the tolerances in the solver. When compared with analytical LFMs, our method overcomes the complex derivations of integrated kernels, which are either intractable or involve complicated convolutions [Lawrence et al., 2007]. Since FEM is limited by the curse of dimensionality, so will *Alfi*'s performance on PDEs. Regardless, the severity is minimal since many physical problems are in 3D or 4D.

Ethical considerations: There could be sensitive applications (e.g. medical time-series) where blind faith in inferred forces can lead to detrimental effects on the patient. This is a consideration inherent in all models, however in this case the latent nature could be more pernicious.

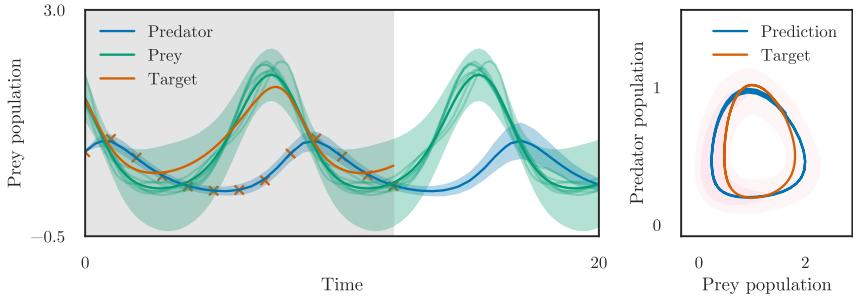


Figure 2: *Alfi* with a periodic kernel on the Lotka-Volterra task. Orange crosses indicate the training datapoints. Blue and green lines are used for predator and prey respectively, orange line for **unseen** prey ground truth. *Left:* the output dynamics with the training data as crosses along with the latent dynamics. *Right:* the phase space (u, v) alongside the unseen ground truth.

Table 1: Results of *Alfi* versus the closest model in the literature, GP-mRNA [López-Lopera et al., 2019]. *Alfi* obtains a comparable performance in most cases, although the higher latent force inference error is an avenue for further research. For metrics explanation see Appendix G.1.

Model	Trunk Gap Gene	$Q^2[\%] \uparrow$ mRNA $\mu \pm \sigma$	$Q^2[\%] \uparrow$ Gap Protein $\mu \pm \sigma$	$68 - \mathrm{CA}_{\pm\sigma}[\%]$ mRNA $\mu \pm \sigma$	$68 - \mathrm{CA}_{\pm\sigma}[\%]$ Gap Protein $\mu \pm \sigma$
GP-mRNA	kr	$\mathbf{90.5 \pm 2.0}$	90.8 ± 0.6	-3.7 ± 6.7	21.8 ± 4.0
	kni	81.1 ± 2.5	88.7 ± 0.8	18.3 ± 6.3	35.3 ± 5.3
	gt	$\mathbf{91.2 \pm 1.9}$	$\mathbf{92.3 \pm 0.6}$	-0.7 ± 3.3	25.8 ± 1.8
*Alfi**	kr	82.4 ± 1.2	$\mathbf{95.7 \pm 0.2}$	3.3 ± 3.5	19.4 ± 2.0
	kni	77.2 ± 2.6	86.9 ± 2.9	$\mathbf{-1.5 \pm 1.6}$	$\mathbf{12.4 \pm 9.1}$
	gt	72.4 ± 3.3	86.5 ± 1.2	-3.9 ± 4.4	$\mathbf{5.8 \pm 14.3}$

References

Martin S Alnæs, Anders Logg, Kristian B Ølgaard, Marie E Rognes, and Garth N Wells. Unified form language: A domain-specific language for weak formulations of partial differential equations. *ACM Transactions on Mathematical Software (TOMS)*, 40(2):1–37, 2014.

Martin S. Alnæs, Jan Blechta, Johan Hake, August Johansson, Benjamin Kehlet, Anders Logg, Chris Richardson, Johannes Ring, Marie E. Rognes, and Garth N. Wells. The fenics project version 1.5. *Archive of Numerical Software*, 3(100), 2015. doi: 10.11588/ans.2015.100.20553.

Mauricio Alvarez, David Luengo, and Neil D Lawrence. Latent force models. In *Artificial Intelligence and Statistics*, pages 9–16. PMLR, 2009.

Kolja Becker, Eva Balsa-Canto, Damjan Cicin-Sain, Astrid Hoermann, Hilde Janssens, Julio R Banga, and Johannes Jaeger. Reverse-engineering post-transcriptional regulation of gap genes in drosophila melanogaster. *PLoS Comput Biol*, 9(10):e1003281, 2013.

Ben Calderhead, Mark Girolami, and Neil D Lawrence. Accelerating bayesian inference over nonlinear differential equations with gaussian processes. In *Advances in neural information processing systems*, pages 217–224. Citeseer, 2009.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*, 2018.

Li-Fang Cheng, Bianca Dumitrascu, Michael Zhang, Corey Chivers, Michael Draugelis, Kai Li, and Barbara Engelhardt. Patient-specific effects of medication using latent force models with gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 4045–4055. PMLR, 2020.

Frank Dondelinger, Dirk Husmeier, Simon Rogers, and Maurizio Filippone. Ode parameter inference using adaptive gradient matching with gaussian processes. In *Artificial intelligence and statistics*, pages 216–228. PMLR, 2013.

Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *arXiv preprint arXiv:1809.11165*, 2018.

Jouni Hartikainen, Mari Seppanen, and Simo Sarkka. State-space inference for non-linear latent force models with application to satellite orbit prediction. *arXiv preprint arXiv:1206.4670*, 2012.

Pashupati Hegde, Markus Heinonen, Harri Lähdesmäki, and Samuel Kaski. Deep learning with differential gaussian process flows. *arXiv preprint arXiv:1810.04066*, 2018.

Markus Heinonen, Cagatay Yildiz, Henrik Mannerström, Jukka Intosalmi, and Harri Lähdesmäki. Learning unknown ode models with gaussian processes. In *International Conference on Machine Learning*, pages 1959–1968. PMLR, 2018.

James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable variational gaussian process classification. 2015.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. *arXiv preprint arXiv:2005.08926*, 2020.

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improving variational inference with inverse autoregressive flow. *arXiv preprint arXiv:1606.04934*, 2016.

Neil D Lawrence, Guido Sanguinetti, and Magnus Rattray. Modelling transcriptional regulation using gaussian processes. *Advances in Neural Information Processing Systems*, 19:785, 2007.

Anders Logg, Kent-Andre Mardal, Garth N. Wells, et al. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012. ISBN 978-3-642-23098-1. doi: 10.1007/978-3-642-23099-8.

Andrés Felipe López-Lopera, Nicolas Durrande, and Mauricio Alexander Alvarez. Physically-inspired gaussian process models for post-transcriptional regulation in drosophila. *IEEE/ACM transactions on computational biology and bioinformatics*, 2019.

Sebastian K Mitusch, Simon W Funke, and Jørgen S Dokken. dolfin-adjoint 2018.1: automated adjoints for fenics and firedrake. *Journal of Open Source Software*, 4(38):1292, 2019.

Jacob Moss and Pietro Lió. Gene regulatory network inference with latent force models, 2020.

Alexander Norcliffe, Cristian Bodnar, Ben Day, Jacob Moss, and Pietro Lio. Neural ode processes. *arXiv preprint arXiv:2103.12413*, 2021.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

Simo Särkkä, Mauricio A Álvarez, and Neil D Lawrence. Gaussian process latent force models for learning and stochastic control of physical systems. *IEEE Transactions on Automatic Control*, 64 (7):2953–2960, 2018.

Michalis K Titsias, Neil D Lawrence, and Magnus Rattray. Efficient sampling for gaussian process inference using control variables. In *NIPS*, pages 1681–1688. Citeseer, 2008.

Wil Ward, Tom Ryder, Dennis Prangle, and Mauricio Alvarez. Black-box inference for non-linear latent force models. In *International Conference on Artificial Intelligence and Statistics*, pages 3088–3098. PMLR, 2020.

Philippe Wenk, Alkis Gotovos, Stefan Bauer, Nico S Gorbach, Andreas Krause, and Joachim M Buhmann. Fast gaussian process based gradient matching for parameter identification in systems of nonlinear odes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1351–1360. PMLR, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 4.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 4
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 4

(b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] In the supplementary material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] In Appendix H
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] In Appendix F
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] In the Supplementary Materials, where the datasets are included.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] The data used does not contain personally identifiable or offensive information.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Comparison

Machine learning methods for differential equations typically assume the equation is unknown, and follow either a GP or neural network approach. For example, differential GPs [Hegde et al., 2018], generalise the discrete layers of a deep GP to a sequence of infinitesimal steps in which the input is warped through a vector field defined by an stochastic differential equation (SDE). However, the SDE acts only as a mechanism for constructing a model rather than as an actual physical model. A similar approach also treats the ODE as unknown using a Gaussian process and uses the full solution along with sensitivity equations to extract gradients [Heinonen et al., 2018]. This model, however, only uses the conditional mean of the Gaussian process, whereas *Alfi* also captures the covariance. The neural network methods, popularised by Chen et al. [2018], represent the ODE function as a neural network, where the weights are optimised by backpropagation through an ODE solver or via adjoint equations. Some recent extensions add stochasticity in the form of initial condition distributions [Norcliffe et al., 2021]. Unknown dynamics models lack the ability to infer latent forces, and therefore cannot be compared directly with LFMs.

Latent force models (LFMs) are between the explicit and unknown differential equation paradigms; the equation depends on a set of latent forces which are inferred using the explicit dynamics. There are several other approaches for overcoming the intractable posterior in non-linear LFMs, namely the Laplace approximation [Lawrence et al., 2007] and MCMC [Titsias et al., 2008]. Hartikainen et al. [2012] show the connection between non-linear latent force models and non-linear SDEs, thereby enabling an approximation using Gaussian filtering and smoothing. Neural network approximations have also been attempted, where an inverse autoregressive flow [Kingma et al., 2016] is used to model the variational posterior [Ward et al., 2020].

B Training Scheme

Natural cubic splines: Our pre-estimation procedure is similar to neural controlled differential equations Kidger et al. [2020] only in the choice of spline. Our interpolant is not fed through a solver–it is used only for gradient estimation. There are many possible interpolation approaches to use instead of cubic splines. One popular method is using a Gaussian process [Calderhead et al., 2009, Dondelinger et al., 2013, Wenk et al., 2019] with a differentiable kernel (and mean) function. Gaussian processes with such kernels are closed under differentiation, and the covariance terms can be found as $\text{cov}(\dot{y}_j(t), \dot{y}_j(t')) = \frac{\partial^2 K(t,t')}{\partial t \partial t'}$. Cubic splines are linear in the number of points, in contrast to GPs which are cubic (or quadratic for non-naive implementation [Gardner et al., 2018]). Since cubic splines are used in pre-estimation, only the 1st and 2nd derivatives are continuous. If the model has higher order derivatives, a higher order spline can be used or the higher-order derivatives ignored—only an approximation is desired in this step.

Natural gradient optimisation: optimising the variational parameters using gradient descent in parameter space assumes Euclidean geometry. Variational parameters exist in distribution space, so we should use a more appropriate geometry within which we calculate distance, such as KL-divergence. Natural gradient optimisation takes steps in this space.

Algorithm 1: *Alfi* Scheme: Steps (1) and (2)

Input: $\mathbf{x}, \mathbf{y}$
initialization;
for $i \leftarrow 1$ **to** T **do**
    latent_forces = variational_posterior($\mathbf{x}$);
    if *preestimation* **then**
        target $\leftarrow$ spline_interpolate_gradient($\mathbf{x}, \mathbf{y}$) ; `// target is gradient`
        pred $\leftarrow$ de_func(initial_conditions, latent_forces);
    else
        target $\leftarrow \mathbf{y}$; `// target is solutions`
        pred $\leftarrow$ solve(de_func, initial_conditions, latent_forces);
    end
    loss $\leftarrow$ elbo(pred, target)
end

C ELBO Derivation

For completeness, we include here the derivation of the ELBO in Equation 4.

$$\log p(\mathbf{Y}|\mathbf{\Theta}, \mathbf{\Phi}) = \log \iint p(\mathbf{Y}|\mathbf{F})p(\mathbf{F}|\mathbf{U})p(\mathbf{U})\mathrm{d}\mathbf{U}\mathrm{d}\mathbf{F}, \tag{10}$$

$$= \log \iint \frac{p(\mathbf{Y}|\mathbf{F})p(\mathbf{F}|\mathbf{U})p(\mathbf{U})p(\mathbf{F}|\mathbf{U})q(\mathbf{U})}{p(\mathbf{F}|\mathbf{U})q(\mathbf{U})}\mathrm{d}\mathbf{U}\mathrm{d}\mathbf{F} \tag{11}$$

We now apply Jensen's inequality:

$$\geq \iint p(\mathbf{F}|\mathbf{U})q(\mathbf{U}) \log \frac{p(\mathbf{Y}|\mathbf{F})p(\mathbf{U})}{q(\mathbf{U})}\mathrm{d}\mathbf{U}\mathrm{d}\mathbf{F}, \tag{12}$$

$$= \int q(\mathbf{U}) \left[\int p(\mathbf{F}|\mathbf{U}) \log p(\mathbf{Y}|\mathbf{F})\mathrm{d}\mathbf{F} + \log \frac{p(\mathbf{U})}{q(\mathbf{U})} \right] \mathrm{d}\mathbf{U}, \tag{13}$$

$$= \mathbb{E}_{q(\mathbf{U})} \left[\mathbb{E}_{p(\mathbf{F}|\mathbf{U})} \left[\log p(\mathbf{Y}|\mathbf{F}) \right] \right] - \mathcal{KL}(q(\mathbf{U})||p(\mathbf{U})), \tag{14}$$

$$= \mathbb{E}_{q(\mathbf{F})} \left[\log p(\mathbf{Y}|\mathbf{F}) \right] - \sum_{i}^{L} \mathcal{KL}(q(\mathbf{u}_i)||p(\mathbf{u}_i)), \tag{15}$$

D Heat Equation Weak Form Derivation

We go through the complete instructive derivation for the heat equation's variational form below.

$$\partial_t y(x,t) = Su(x,t) - \lambda y(x,t) + D\nabla_x^2 y(x,t) \tag{16}$$

The first step in FEM is to find the variational form. We henceforth omit function parameters (e.g. $u = u(x,t)$). First, we take the finite backwards difference as follows $\forall n = 0, 1, 2, ...$:

$$\partial_t y^{n+1} \approx \frac{y^{n+1} - y^n}{\Delta t} = Su - \lambda y + D\nabla_x^2 y \tag{17}$$

We rearrange to find a bilinear form on the left-hand-side and a linear form on the right:

$$y^{n+1} + \Delta t\lambda y^{n+1} - \Delta t D\nabla^2 y^{n+1} = y^n + \Delta t S u \tag{18}$$

For brevity we now drop the $n+1$ indices ($y = y^{n+1}$). The next step in FEM is to multiply both sides by a test function $v \in V$ and integrate over the spatial domain Ω. The Dirichlet boundary conditions are defined such that $y(x=0,t) = 0$ and $y(x=l,t) = 0$ where 0 and l are the spatial boundaries.

$$\int_\Omega (yv + \Delta t\lambda yv - \Delta t D\nabla_x^2 yv)\mathrm{d}\vec{x} = \int_\Omega (y^n + \Delta t S u)\, v\, \mathrm{d}\vec{x} \tag{19}$$

We can integrate the second derivatives by parts:

$$\int_\Omega \Delta t D\nabla_x^2 yv \, \mathrm{d}\vec{x} = D \underbrace{[\nabla_x y \cdot v]_0^l}_{0} - \int_\Omega \Delta t D\nabla_x y \cdot \nabla_x v \, \mathrm{d}\vec{x} \tag{20}$$

where $[\nabla_x y \cdot v]_0^l$ vanished due to boundary conditions. This yields the variational (weak) form:

$$\int_\Omega yv + \Delta t\lambda yv + \Delta t D\nabla_x y \cdot \nabla_x v \, \mathrm{d}x = \int_\Omega (y^n + \Delta t S u)v \, \mathrm{d}\vec{x} \tag{21}$$

$$a(y,v) = L_{n+1}(v) \tag{22}$$

This weak form can then be written in UFL almost verbatim.

E Code

E.1 Differences between an ODE- or PDE-based LFM

Both `OrdinaryLFM()` (for ODE-based LFMs) and `PartialLFM()` (for PDE-based LFMs) inherit from the `VariationalLFM()` class which provides methods for sampling from the variational GP.

Depending on which class is used, the practitioner must implement the `ode_func` or `pde_func` function, which are used by the solvers to calculate the forward solution, and the pre-estimator (`PreEstimator`) to calculate data gradients.

An example LFM for a transcriptional regulation model from is written below, demonstrating how simple *Alfi* makes constructing LFMs:

```
class TranscriptionLFM(OrdinaryLFM):
    def __init__(self, num_outputs, gp_model, config: VariationalConfiguration, **kwargs):
        super().__init__(num_outputs, gp_model, config, **kwargs)
        self.decay_rate = Parameter(torch.rand((self.num_outputs, 1)))
        self.basal_rate = Parameter(torch.rand((self.num_outputs, 1)))
        self.sensitivity = Parameter(torch.rand((self.num_outputs, 1)))
        # optionally add any constraints...

    def odefunc(self, t, h):
        f_latents = self.f
        f_latents = self.G(f_latents)  # optionally add non-linearity
        dh = self.basal_rate + self.sensitivity * f_latents - self.decay_rate * h
        return dh
```

E.2 Unified Form Language for PDEs

The `PartialLFM` requires an additional module called `fenics_module`. This is defined in Unified Form Language (UFL) [Alnæs et al., 2014]. An example UFL snippet for the reaction-diffusion equation from Section 3 is shown below:

```
mesh = UnitIntervalMesh(20)
V = FunctionSpace(mesh, 'P', 1)

u = TrialFunction(V)
v = TestFunction(V)

a = (1 + dt * decay) * u * v * dx
a = a + dt * diffusion * inner(grad(u), grad(v)) * dx
L = (u_prev + dt * sensitivity * latent_force) * v * dx

bc = DirichletBC(V, 0, 'on_boundary')

y = Function(V)
solve(a == L, y, bc)
```

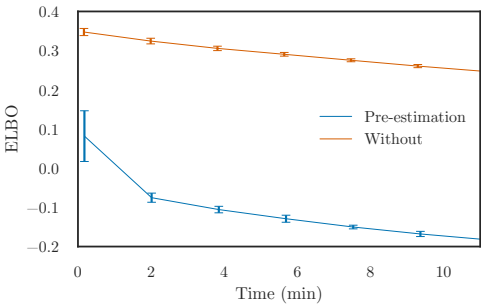
F Time profiling

As shown in Figure 4, the convergence of the variational LFM has a faster or comparable training time to the analytical solution, as shown in Figure 4b for the 1D case.

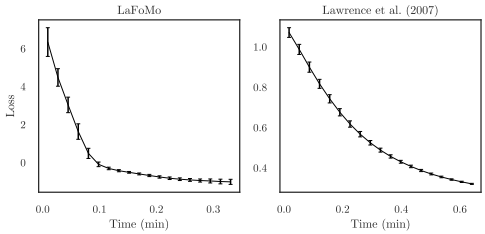
G Additional Results

G.1 Metrics

We use the $Q^2 = 1 - \mathrm{SMSE}$ criterion, where SMSE is the mean-squared-error (MSE) normalised by the target variance, overcoming the sensitivity of MSE to the scale of the target values [Rasmussen and Williams, 2005]. In addition, to give more insights into the more challenging spatio-temporal experiments, we also quote coverage interval accuracy deviation ($68\% - \mathrm{CA}_{\pm\sigma}$), which measures the proportion of targets within 1 standard deviation of the predictive distribution.



(a) Latent Force Model. Training times for the reaction-diffusion PDE LFM with and without pre-estimation. Training was run on a Titan Xp GPU.



(b) Convergence times of *Alfi* compared with the analytical model in Lawrence et al. [2007]. We found that increasing the learning rate for the latter model lead to instabilities in the cholesky decompositions. Both were run on an AMD Ryzen 5600X CPU.

Figure 4: Time profiling.

G.2 Lotka-Volterra

In Figure 5, the results for the Lotka-Volterra task are shown when using an RBF kernel instead of the periodic kernel.

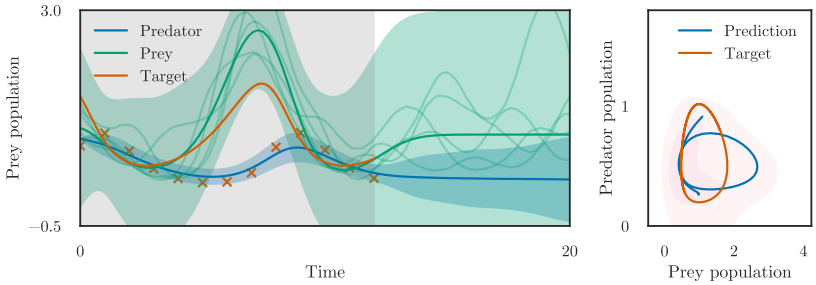


Figure 5: *Alfi* with an RBF kernel on the Lotka-Volterra task. This demonstrates inferior extrapolation performance compared with a periodic kernel, since a periodic kernel has additional inductive biases.

G.3 Drosophila

In Figure 6, the results for the reaction-diffusion experiment with the Drosophila embryogenisis dataset are shown.

H Hyperparameters

Hyperparameter optimisation is typically carried out with maximum marginal likelihood, or a lower-bound thereof. The hyperparameters were initial set to reasonable values: lengthscales of kernels were set to the smallest distance between datapoints, and rates to a uniform random number between 0 and 1. For the comparison with GP-mRNA [López-Lopera et al., 2019], we kept the experimental conditions the same: the rates were fixed to the values reported by Becker et al. [2013], and the lengthscales were learnt by maximising the ELBO.

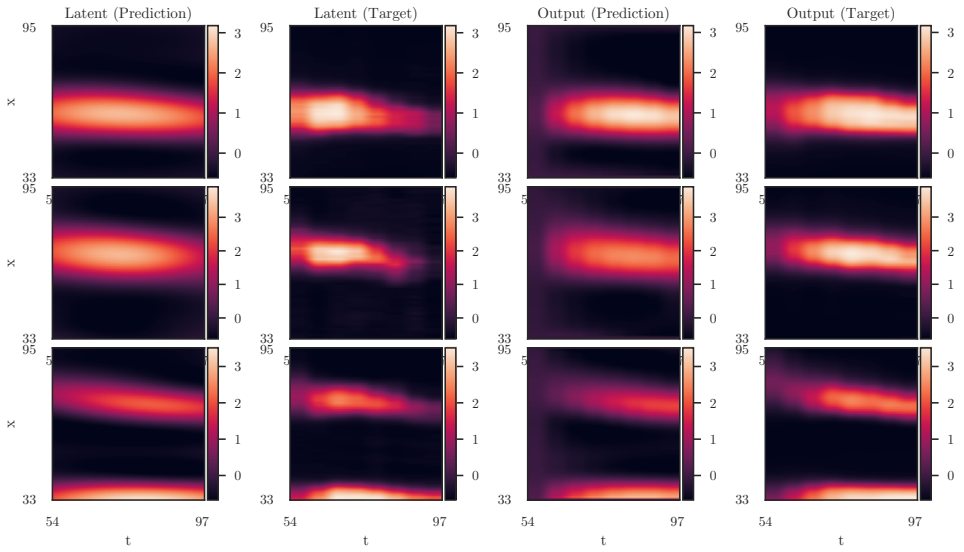


Figure 6: *Alfi* on the Drosophila reaction-diffusion task. *First and second columns*: the latent mRNA concentration alongside ground truth. *Third and fourth columns:* protein concentration alongside ground truth.