Scalable Multitask Latent Force Models with Applications to Predicting Lithium-ion Concentration

Daniel J. Tait *
Depts. of Comp. Sci. and Statistics
University of Warwick
The Alan Turing Insitutute
dtait@turing.ac.uk

Ferran Brosa Planella
WMG
University of Warwick
The Faraday Institution
ferran.brosa-planella@warwick.ac.uk

Theodoros Damoulas
Depts. of Comp. Sci. and Statistics
University of Warwick
The Alan Turing Insitutute
tdamoulas@turing.ac.uk

W. Dhammika Widanage
WMG
University of Warwick
The Faraday Institution
dhammika.widanalage@warwick.ac.uk

Abstract

It is often necessary in engineering to reduce a comprehensive mathematical model to a simplified representation which admits faster evaluation for control and optimisation applications. However, this reduction can often lead to "missing physics" problem, whereby important dynamics of the true process have been sacrificed for tractability. In this work we consider the augmentation of these mathematically reduced models with adaptive data-driven components to recover this information. By introducing a stochastic differential equation framework with a set of common latent forcing function we allow information to be shared between different operating conditions, referred to as tasks, and to further generalise to new physical scenarios. Unfortunately, standard approaches to this problem scale cubicly in the number of tasks and so we introduce an approximation which achieves linear scaling allowing a large range of different physical scenarios to be investigated. Finally we demonstrate the potential of this method to improve prediction and generalisability in simplified models of Lithium-ion battery dynamics.

1 Introduction

Constructing physically realistic models of the complex dynamical systems which are of interest to mathematicians and engineers often requires the use high dimensional, nonlinear models typical in the form of partial differential equations (PDEs). For a given parameterisation such models can be delicate and expensive to forward simulate from, and such difficulties only get compounded once one also has to consider the problem of parameter inference and calibration. Typically for applications in control and design optimisation, it is necessary to create simplified approximations to the original system. This process often takes the form of an apriori mathematical reduction of the model, and as a result some of the physical realism of the original specification must be sacrificed leading to the problem of missing physics in the reduced model, a loss of information which cannot be recovered from the reduced model regardless of our access to observations from this same system.

Machine learning offers the possibility of recovering this information by combining simplified mechanistic representations with flexible data-driven components to create efficient hybrid models of

*Code to implement the this paper is available at https://github.com/danieljtait/spmelf

Workshop on machine learning for engineering modeling, simulation and design @ NeurIPS 2020

dynamical systems. In the case of continuous time dynamical systems a promising approach is to consider the modulation of the dynamics by Gaussian processes (GPs) which are learned from data, this is the *latent force model* introduced by [Álvarez et al., 2009]. In this work we consider a multitask [Bonilla et al., 2008] framework adapted to dynamical systems in order to construct these hybrid models, each task corresponding to a different operating condition of the system we are studying. Unfortunately, the standard approach to inference in this model would scale cubicly with the number of tasks. We address this problem by introducing an approximation in Section 3 that decouples each of the per-task problems and then constructs the posterior as a product of experts [Hinton, 2002]. Each expert corresponds to the posterior of a linear Gaussian state space model (SSM) and we show in Section 4 that this product can be renormalised as a single equivalent SSM. This allows us to replace the cubic scaling the number of tasks with a linear scaling.

Finally, we demonstrate the ability of this framework to successfully recover missing physics when simplifying complex models describing the dynamics of Li-ion concentration within batteries under charging and discharging. This results in a consistently well-informed simulation of battery state capable of evolving to capture unknown battery characteristics which have been lost under simplification of the original model. Importantly the use of a mathematical reduction combined with our decoupling approximation enables models which are fast enough for control applications, but which also utilise data to maintain fidelity to the true governing dynamics.

2 Background

In this work we consider a collection of P vector valued processes $\mathbf{x}_p(t) \in \mathbb{R}^{D_p}$ for $p = 1, \ldots, P$, where each of the $\mathbf{x}_p$ processes is to be viewed as representing our system of interest under a particular operating condition, for instance an electrochemical system under different discharge rates. We refer to each of the processes $\mathbf{x}_p(t)$ as a *task process* or more simply a task [Bonilla et al., 2008], and our objective is to allow each task to share information, but in such a way that the resulting inference problem remains tractable, even in the case of a large number of tasks.

We want to include an adaptive data-driven information sharing term for each of our continuous time processes and so Gaussian processes are a natural candidate leading to the adoption of a Gaussian process latent variable framework[Lawrence, 2004]. More explicitly we assume that each of these processes interacts through the presence of a common continuous time process, and we denote this set of R common latent functions by $\{g_r(t)\}_{r=1}^R$. To complete the construction of a dynamical system we assume each task can be modelled by a forced stochastic differential equation (SDE) of the form

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{A}_p \mathbf{x}_p(t) + \sum_{r=1}^{R} \mathbf{s}_{pr} g_r(t) + \mathbf{L}_p \mathbf{w}_p(t) \tag{1}$$

where each $\mathbf{w}_p$ is a mean-zero driving white noise process with $\mathbb{E}[\mathbf{w}_p(t)\mathbf{w}_q(t')] = \mathbf{Q}_p \delta(t - t')\delta_{pq}$, for $p = 1, \ldots, P$, and where $\mathbf{L}_p$ are the $D_p \times D_p$ diffusion coefficient matrices. In particular each of these white noise processes is independent for separate tasks, and so the only connection between the models for each task is the shared latent forces.

Each of the vectors $\mathbf{s}_{pr} \in \mathbb{R}^{D_p}$ acts to distribute the scalar perturbation forcing function $g_r(t)$ over the domain of $\mathbf{x}_p$. We shall further assume that each $g_r(t)$ is a Gaussian process (GP) admitting a representation in state space form [Rasmussen and Williams, 2005]. That is there is some state variable $\mathbf{z}_r$, and vector $\mathbf{h}_r \in \mathbb{R}^{M_r}$ such that $g_r(t) = \langle \mathbf{h}_r, \mathbf{z}_r(t) \rangle$, and a matrix $\mathbf{F}_r$ such that

$$\frac{d\mathbf{z}_r}{dt} = \mathbf{F}_r \mathbf{z}_r + \mathbf{L}_{0,r} \mathbf{w}_{0,r}(t).$$

Then if we We write $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_R)^\top$ for the concatenation of these variables, and similarly define $\mathbf{F}$ as the block diagonal matrices with blocks $\mathbf{F}_r$, $r = 1, \ldots, R$ and $\mathbf{H}$ is the $R \times \sum_{r=1}^{R}$ block diagonal matrix with blocks given by row vectors $\mathbf{h}_r^\top$. Then we may write the model for each task with R latent forces as

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{A}_p \mathbf{x}_p(t) + \mathbf{S}_p \mathbf{z}(t) + \mathbf{L}_p \mathbf{w}_p(t), \qquad p = 1, \ldots, P \tag{2a}$$

$$\frac{d\mathbf{z}}{dt} = \mathbf{F}\mathbf{z}_t + L_0 \mathbf{w}_0(t) \tag{2b}$$

where $\mathbb{E}[\mathbf{w}_p(t)\mathbf{w}_q(t')] = \mathbf{Q}_p\delta(t-t')\delta_{pq}$, for $p = 0, 1, \ldots, P$ and where $\mathbf{S}_p = \mathbf{S}_{pr}\mathbf{H}$, with $\mathbf{S}_{pr}$ the matrix with columns $\mathbf{s}_{pr}$.

This process is a particular instance of the *latent force model framework* introduced by [Álvarez et al., 2009], and as such one could immediately consider conducting inference in the standard way using filtering methods, for example using the approach in [Hartikainen et al., 2012] or see [Rogers et al., 2020] for a recent discussion in an engineering context. This would involve collecting the components of the system (2) into a single augmented model and then discretising in the usual manner. If we define $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_P)^\top$ then the result is a single SDE of the form

$$\frac{d}{dt}\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\\ \vdots\\ \mathbf{x}_P\\ \mathbf{z}\end{bmatrix} = \begin{bmatrix}\mathbf{A}_1 & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{S}_1\\ \mathbf{0} & \mathbf{A}_2 & \cdots & \mathbf{0} & \mathbf{S}_2\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{A}_P & \mathbf{S}_P\\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{F}\end{bmatrix}\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\\ \vdots\\ \mathbf{x}_P\\ \mathbf{z}\end{bmatrix} + \begin{bmatrix}\mathbf{L}_1\mathbf{w}_1\\ \mathbf{L}_2\mathbf{w}_2\\ \vdots\\ \mathbf{L}_P\mathbf{w}_p\\ \mathbf{L}_0\mathbf{w}_0\end{bmatrix}. \tag{3}$$

Unfortunately, this approach will suffer from fatal scaling issues as either the dimension of tasks, or the number of tasks increases. To see this we will consider applying Kalman-Filtering over N time-steps then writing $D = P^{-1}\sum_{p=1}^{P} D_p$ for the average task dimension filtering methods will scale like $\mathcal{O}(N \cdot (PD + M)^3)$ [Säarkä and Solin, 2019], and so will quickly become infeasible for any combination of large state dimension, or large number of tasks. In the next sections we introduce an approximation which is linear in the number of tasks.

Sensitivity matrix in PDE latent force models For many applications in engineering, and in particular the example we consider in Section 5 we shall be concerned with the case when the matrix $\mathbf{A}_p$ for each task arises from the discretisation of a PDE. With this in mind it is natural to replace the vector valued sensitivity vectors $\mathbf{s}_r$ with functions $s_r(\omega)$ defined over a domain Ω with coordinates $\omega \in \Omega$. For example we could consider a specification of the mechanistic component with Dirichlet boundary conditions $f(\omega, t)$ given by

$$\frac{\partial x(\omega,t)}{\partial t} = \mathcal{A}x + \sum_{r=1}^{R} s_r(\omega)g_r(t) \tag{4a}$$

$$x(\omega, t) = f(\omega, t) \text{ for } x \text{ on } \partial\Omega \tag{4b}$$

for some linear operator $\mathcal{A}$, with obvious extensions to more general boundary conditions.

From (4a) we also have the clear interpretation of the functions $s_r(\omega)$ and their interactions with the latent forces $g_r(t)$. For a given timescale t, the latent forces $g_r(t)$ determine a scalar contribution to the dynamics in (4a), the functions $s_r(\omega)$ then determine how this stochastic perturbation is distributed over the spatial domain. After discretisation, for example by finite differences, finite elements or finite volumes, we produce a discretised version which takes the

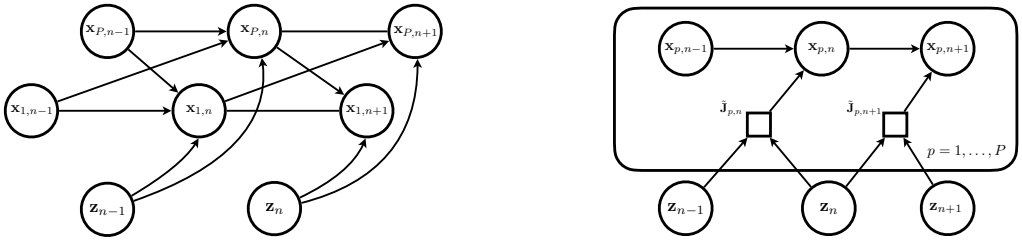
$$\mathbf{x}(t) = \mathbf{A}\mathbf{x}(t) + \sum_{r=1}^{R} \mathbf{s}_r g_r(t). \tag{5}$$

Typically now the components $(\mathbf{s}_r)_i$, $i = 1, \ldots, D$ will represent the function, or a projection of the function, at a collection of nodal coordinates $\omega_i \in \Omega$. In this work we shall consider the case where each of the sensitivity function is given by summing over a set of basis functions $\{\phi_k\}_{k=1}^{Q}$ defined over the domain Ω, and we write

$$s_r(\omega) = \sum_{k=1}^{K} \beta_k \phi_k(\omega).$$

Observation model To complete the modelling setup we require an observation model which connects a variable $\mathbf{y}_p(t)$ to the state variable $\mathbf{x}_p$ for each task. We assume that these observations models are independent for each task and write

$$p(\mathbf{Y} \mid \mathbf{X}) = \prod_{p=1}^{P}\prod_{n=1}^{N} p(\mathbf{y}_{p,n} \mid \mathbf{x}_{p,n}). \tag{6}$$



(a) Full transition model

(b) Decoupled approximate transition model

Figure 1: (a) The graphical model of the system (8) showing the dense connections of the original model (b) The diagram for the approximate model extended to show the deterministic nodes $\tilde{\mathbf{J}}_n(\mathbf{z}_{n-1}, \mathbf{z}_{n-1})$ which are used to condition and so decouple the task models.

where $\mathbf{x}_{p,n}$, respectively $\mathbf{x}_{p,n}$ is an observation of the process at time t_n, for $n = 1, \ldots, N$. An important special case is when the the observation model is a linear Gaussian model and we write

$$p(\mathbf{y}_{p,n} \mid \mathbf{x}_{p,n}) = \mathcal{N}(\mathbf{y}_{p,n} \mid \mathbf{C}_p \mathbf{x}_{p,n}, \mathbf{\Gamma}). \tag{7}$$

The choice of a linear Gaussian observation model allows inference for all the methods considered in this paper to be done using exact Kalman filtering and smoothing.

3 The approximate decoupled task model

In order to achieve an approximation which demonstrates better scaling with respect to the number of tasks we first solve the system (2) explicitly, see [Säarkä and Solin, 2019], on an interval $[t_n, t_n + \Delta t]$, and note that the solution is given by

$$\mathbf{x}_{p,n} = e^{A_p \Delta t} \mathbf{x}_{p,n-1} + \mathbf{J}_{p,n} + \mathbf{W}_{p,n} \tag{8a}$$

$$\mathbf{z}_n = e^{F \Delta t} \mathbf{z}_{n-1} + \mathbf{W}_{0,n} \tag{8b}$$

where we have defined

$$\mathbf{J}_{p,n} = \int_{t_n}^{t_n + \Delta t} e^{A_p (t_n + \Delta t - \tau)} S_p \mathbf{z}(\tau) \, \mathrm{d}\tau,$$

$$\mathbf{W}_{p,n} = \int_{t_n}^{t_n + \Delta t} e^{A_p (t_n + \Delta t - \tau)} L_p \mathbf{w}_p(\tau) \, \mathrm{d}\tau$$

$$\mathbf{W}_{0,n} = \int_{t_n}^{t_n + \Delta t} e^{F (t_n + \Delta t - \tau)} L_0 \mathbf{w}_0(\tau) \, \mathrm{d}\tau$$

Because of the independence of $\mathbf{W}_{p,n}$ and $\mathbf{W}_{q,n}$ for $p \neq q$, we observe that the transitions for each task process will share information only through the common latent force, and so we realise dense connections between each task, this is shown graphically in Figure 1a. Nevertheless, given the limited avenue by which information can be shared between tasks it is worthwhile asking if there exists some conditioning which could introduce an effective decoupling of the tasks.

The information sharing between a given task and the latent process occurs through the non-zero covariance of $\mathbf{J}_n$ and $\mathbf{W}_{0,n}$ and so by conditioning on $\mathbf{J}_n$ we break the dependence and decouple the different tasks, this is clear from the conditional independence graph in Figure 1b. Unfortunately obtaining the variables $\mathbf{J}_n$ is at least as difficult as solving the original system, and so not immediately useful. However, we can begin to arrive at our decoupling by replacing the integral term $\mathbf{J}_n$ by a suitable quadrature $\mathbf{J}_n \approx \frac{\Delta t}{2} \left(e^{\mathbf{A}_p \Delta t} \mathbf{S}_p \mathbf{z}_{n-1} + \mathbf{S}_p \mathbf{z}_n \right)$. Using this quadrature approximation and (8b) we can write the update (8a) as

$$\begin{aligned} \mathbf{x}_{p,n} &\approx e^{A_p \Delta t} \mathbf{x}_{p,n-1} + \frac{\Delta t}{2} \left(e^{\mathbf{A}_p \Delta t} \mathbf{S}_p \mathbf{z}_{n-1} + \mathbf{S}_p \mathbf{z}_n \right) + \mathbf{W}_{p,n} \\ &= e^{A_p \Delta t} \mathbf{x}_{p,n-1} + \frac{\Delta t}{2} \left(e^{A_p \Delta t} \mathbf{S}_p + \mathbf{S}_p e^{F \Delta t} \right) \mathbf{z}_{n-1} + \frac{\Delta t}{2} \mathbf{S}_p \mathbf{W}_{0,n} + \mathbf{W}_{p,n}, \end{aligned} \tag{9}$$

and so up an error introduced by the quadrature we can write the transition density of the fully augmented model as

$$\begin{bmatrix}\mathbf{x}_n \\ \mathbf{z}_n\end{bmatrix} \Bigg| \begin{bmatrix}\mathbf{x}_{n-1} \\ \mathbf{z}_{n-1}\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\mathbf{x}_n \\ \mathbf{z}_n\end{bmatrix} \Bigg| \begin{bmatrix} e^{\mathbf{A}\Delta t} & \frac{\Delta t}{2}\left(e^{\mathbf{A}\Delta t}\mathbf{S} + \mathbf{S}e^{\mathbf{F}\Delta t}\right) \\ 0 & e^{\mathbf{F}\Delta t}\end{bmatrix} \begin{bmatrix}\mathbf{x}_{n-1} \\ \mathbf{z}_{n-1}\end{bmatrix} \begin{bmatrix}\Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22}\end{bmatrix}\right) \tag{10}$$

where for $p, q = 1, \ldots, P$ we have defined the blocks

$$(\Sigma_{11})_{pq} = \delta_{pq}\mathbb{E}\left[\mathbf{W}_{p,n}, \mathbf{W}_{q,n}^\top\right] + \left(\frac{\Delta t}{2}\right)^2 S_p \mathbb{E}\left[\mathbf{W}_0, \mathbf{W}_0^\top\right] S_q^\top \tag{11a}$$

$$(\Sigma_{12})_p = \frac{\Delta t}{2} S_p \mathbb{E}\left[\mathbf{W}_0 \mathbf{W}_0^\top\right] \tag{11b}$$

and where $\Sigma_{22} = \mathbb{E}\left[\mathbf{W}_0\mathbf{W}_0^\top\right]$, we shall also define $\Sigma_{p,\Delta} \triangleq \mathbb{E}\left[\mathbf{W}_{p,n}\mathbf{W}_{p,n}^\top\right]$.

For $\Delta t \ll 1$ this implies that most of the information in the transition distribution is captured by the auto-covariance of each task noise term and the covariance of the task with the latent force. With the cross-covariance of the tasks being negligible conditional on $\mathbf{x}_{p,n-1}$ and $\mathbf{z}_{n-1}$. We further note that, after conditioning on $\mathbf{z}_n$ and $\mathbf{z}_{n-1}$, then the innovation in (8a) depends only on $\mathbf{W}_{p,n}$ which by construction are independent between tasks. As a result after conditioning the model decouples, and we write our approximation, $\tilde{p}$, to the transition distribution as

$$\begin{aligned}&\tilde{p}(\mathbf{x}_n \mid \mathbf{x}_{n-1}, \mathbf{z}_{n-1}, \mathbf{z}_n) \\ &\quad = \mathcal{N}\left(\mathbf{x}_{p,n} \,\middle|\, e^{A_p\Delta t}\mathbf{x}_{p,n-1} + \frac{\Delta t}{2}\left(e^{A_p\Delta t}S_p\mathbf{z}_{n-1} + \mathbf{S}_p\mathbf{z}_n\right), \mathbb{E}\left[\mathbf{W}_{p,n}\mathbf{W}_{p,n}^\top\right]\right).\end{aligned}$$

The resulting decoupling of the transition model after conditioning on the quadrature terms $\tilde{\mathbf{J}}_{p,n}$ is displayed in Figure 1b.

3.1 Product of state space experts

We now show that the decoupling in the previous section allows us to write the complete posterior up to proportionality as a product of experts [Hinton, 2002], an expert arising from each task trained independently. First we note that using our approximation to the transition model, $\tilde{p}$, we have

$$p(\mathbf{x}_{n+1} \mid \mathbf{x}_n, \mathbf{z}_n, \mathbf{z}_{n+1}) \approx \prod_{i=1}^{P} \tilde{p}(\mathbf{x}_{i,n+1} \mid \mathbf{x}_{i,n}, \mathbf{z}_n, \mathbf{z}_{n+1}), \tag{12}$$

and assuming that the prior for each task is independent so that $p(\mathbf{x}_1) = \prod_{i=1}^{P} p(\mathbf{x}_{i,1})$ then

$$\begin{aligned}p(\mathbf{X} \mid \mathbf{Z}) &= p(\mathbf{x}_1) \prod_{n=2}^{N} p(\mathbf{x}_n \mid \mathbf{x}_{n-1}, \mathbf{z}_{n-1}, \mathbf{z}_n) \\ &\approx \prod_{i=1}^{P} p(\mathbf{x}_{i,1}) \prod_{n=2}^{N} \tilde{p}(\mathbf{x}_{i,n} \mid \mathbf{x}_{i,n-1}, \mathbf{z}_{n-1}, \mathbf{z}_n) \\ &\triangleq \prod_{i=1}^{P} \tilde{p}(\mathbf{X}_i \mid \mathbf{Z})\end{aligned} \tag{13}$$

then using our assumption that the observations are independent for each task we can approximate the joint density as

$$p(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) \approx \prod_{i=1}^{P} p(\mathbf{Y}_i \mid \mathbf{X}_p)\tilde{p}(\mathbf{X}_i \mid \mathbf{Z})p(\mathbf{Z}) \triangleq \prod_{i=1}^{P} \tilde{p}(\mathbf{Y}_i, \mathbf{X}_i, \mathbf{Z}). \tag{14}$$

Note that each term of the product is a state space model using the decoupled transition density $\tilde{p}$, and we can write the posterior as

$$\begin{aligned} p(\mathbf{X}, \mathbf{Z} \mid \mathbf{Y}) &\propto p(\mathbf{X}, \mathbf{Z}, \mathbf{Y}) \\ &\approx \prod_{i=1}^{P} \tilde{p}(\mathbf{Y}_i, \mathbf{X}_i, \mathbf{Z}) \\ &\propto \prod_{i=1}^{P} \tilde{p}(\mathbf{X}_i, \mathbf{Z} \mid \mathbf{Y}_i). \end{aligned} \tag{15}$$

Each of the terms $\tilde{p}(\mathbf{X}_p, \mathbf{Z} \mid \mathbf{Y}_p)$ may be obtained by independently filtering and smoothing a state space model with transition density $\tilde{p}$ and observations $\mathbf{Y}_p$. While the cost of filtering the original model (2) was $\mathcal{O}(N(DP)^3)$, each of these decoupled tasks can be filtered at a computational cost of only $\mathcal{O}(ND^3)$ and so filtering the complete collection independently has a total complexity of $\mathcal{O}(ND^3P)$. In summary we have reduced the cubic scaling in the number of task with a linear scaling, however we still need to address the problem of normalising (15) to access useful summaries of the posterior such as moments, or to sample from it effectively.

4 Posterior of state space experts

In the previous section we introduced an approximation that effectively decouples our tasks, leading to an approximate posterior (15) which was a product of experts, an expert corresponding to the posterior of a state space model only using the model for the pth task. In this section we demonstrate that this product may be represented as a single state state space model. To do so it will be helpful to introduce a generic state vector $\mathbf{w}$. We shall also denote the posterior conditional on data $\mathbf{Y}$ by $q(\mathbf{w}) = p(\mathbf{w} \mid \mathbf{Y})$, and understand that any densities $q(\cdot)$ are understood to be conditioned on the data, although the algebraic manipulations in this section hold in full generality.

We also recall that the posterior of a linear Gaussian state space model can be obtained by backward smoothing [Bishop, 2006, Säarkä and Solin, 2019], and so the posterior of a state space model can also be represented as the density of a Markov process but now running backwards in the index

$$q_i(\mathbf{w}_1, \ldots, \mathbf{w}_N) = q_i(\mathbf{w}_N) \prod_{n=1}^{N-1} q_i(\mathbf{w}_n \mid \mathbf{w}_{n+1}),$$

where each of the transition models takes the form of a conditional linear Gaussian model which we denote by

$$q_i(\mathbf{w}_n \mid \mathbf{w}_{n+1}) = \mathcal{N}(\mathbf{w}_n \mid \mathbf{A}_{i,n}\mathbf{w}_{n+1} + \mathbf{b}_{i,n}, \Lambda_{i,n}^{-1}). \tag{16}$$

The full density in product of experts form is then given, up to a constant, by

$$q(\mathbf{w}_1, \ldots, \mathbf{w}_N) \propto \prod_{i=1}^{P} q_i(\mathbf{w}_N) \prod_{n=1}^{N-1} q_i(\mathbf{w}_n \mid \mathbf{w}_{n+1}), \tag{17}$$

and we now consider the problem of normalising this final expression to derive expressions for the marginal distributions of the full posterior. Infact we shall show that we can write

$$q(\mathbf{w}_1, \ldots, \mathbf{w}_N) \propto \prod_{n=1}^{N-1} p(\hat{\boldsymbol{\xi}}_n \mid \mathbf{w}_n) q(\mathbf{w}_n \mid \mathbf{w}_{n+1}) q(\mathbf{w}_N) \tag{18}$$

for some set of variables $\hat{\boldsymbol{\xi}}_n$, and therefore posterior may be obtained by applying filtering and smoothing to the backwards model conditioned on the variables $\hat{\boldsymbol{\Xi}} = \{\hat{\boldsymbol{\xi}}_n\}_{n=1}^N$. Note also that the filtering and smoothing for this final posterior is run in the opposite direction for that used for each of the individual experts.

Product of linear Gaussian transition models We show in the appendix that we may write the product of the transition densities, up to a scaling factor independent of $\mathbf{w}_n$ and $\mathbf{w}_{n+1}$, in two

different forms depending on the order which we complete the square

$$\prod_{i=1}^{P} \propto \mathcal{N}(\mathbf{w}_n \mid \mathbf{A}_n \mathbf{w}_{n+1} + \mathbf{b}_n, \mathbf{\Sigma}_n) \times \mathcal{N}(\mathbf{w}_n \mid \boldsymbol{\alpha}_n, \mathbf{\Gamma}_n) \tag{19a}$$

$$\propto \mathcal{N}(\mathbf{w}_n \mid \mathbf{A}'_n \mathbf{w}_{n+1} + \mathbf{b}'_n, \mathbf{\Sigma}'_n) \times \mathcal{N}(\mathbf{w}_{n+1} \mid \boldsymbol{\alpha}'_{n+1}, \mathbf{\Gamma}'_{n+1}) \tag{19b}$$

where in both cases proportionality holds up to a constant independent of $\mathbf{w}_n$ and $\mathbf{w}_{n+1}$. There parameters are given explicitly by

$$\mathbf{A}_n = \mathbf{\Psi}_1^{-\top} \mathbf{\Psi}_{2,n} \qquad \mathbf{A}'_n = \mathbf{\Psi}_{0,n}^{-1} \mathbf{\Psi}_1 \tag{20a}$$

$$\mathbf{b}_n = \mathbf{\Psi}_{1,n}^{-\top} \mathbf{v}_n \qquad \mathbf{b}'_n = \mathbf{\Psi}_{0,n}^{-1} \mathbf{u}_n \tag{20b}$$

$$\mathbf{\Sigma}_n = \left(\mathbf{\Psi}_{1,n} \mathbf{\Psi}_{2,n}^{-1} \mathbf{\Psi}_{1,n}^{\top} \right)^{-1} \qquad \mathbf{\Sigma}'_n = \mathbf{\Psi}_{0,n}^{-1} \tag{20c}$$

$$\boldsymbol{\alpha}_n = \mathbf{\Gamma}_n (\mathbf{u}_n - \mathbf{\Psi}_1^{\top} \mathbf{\Psi}_2^{-1} \mathbf{v}) \qquad \boldsymbol{\alpha}'_{n+1} = \mathbf{\Gamma}'_{n+1} (\mathbf{\Psi}_{1,n}^{\top} \mathbf{\Psi}_{0,n}^{-1} \mathbf{u}_n - \mathbf{v}_n) \tag{20d}$$

$$\mathbf{\Gamma}_n = \left(\mathbf{\Psi}_{0,n} - \mathbf{\Psi}_1 \mathbf{\Psi}_2^{-1} \mathbf{\Psi}_1^{\top} \right)^{-1} \qquad \mathbf{\Gamma}'_{n+1} = \left(\mathbf{\Psi}_{2,n} - \mathbf{\Psi}_{1,n}^{\top} \mathbf{\Psi}_{0,n}^{-1} \mathbf{\Psi}_{1,n} \right)^{-1} \tag{20e}$$

depending on the matrix statistics

$$\mathbf{\Psi}_{0,n} = \sum_{i=1}^{P} \mathbf{\Lambda}_{i,n} \qquad \mathbf{\Psi}_{1,n} = \sum_{i=1}^{P} \mathbf{\Lambda}_{i,n} \mathbf{A}_{i,n} \qquad \mathbf{\Psi}_{2,n} = \sum_{i=1}^{P} \mathbf{A}_{i,n}^{\top} \mathbf{\Lambda}_{i,n} \mathbf{A}_{i,n} \tag{21a}$$

and vector valued statistics

$$\mathbf{u}_n = \sum_{i=1}^{P} \mathbf{\Lambda}_{i,n} \mathbf{b}_{i,n} \quad \mathbf{v}_n = \sum_{i=1}^{P} \mathbf{A}_{i,n}^{\top} \mathbf{\Lambda}_i \mathbf{b}_{i,n}. \tag{21b}$$

Recursive normalization In order to normalise our product of state space experts, and so construct the single state space model which is equivalent to this product, we will frequently need to evaluate integrals of the form

$$I_n(\mathbf{w}_n) \triangleq \int \prod_{i=1}^{P} \prod_{m=1}^{n-1} q_i(\mathbf{w}_m \mid \mathbf{w}_{m+1}) \, \mathrm{d}\mathbf{w}_1 \cdots \mathrm{d}\mathbf{w}_{n-1}.$$

These can be evaluated recursively by first noting that

$$\begin{aligned} I_{n+1}(\mathbf{w}_{n+1}) &= \int \prod_{i=1}^{P} \prod_{m=1}^{n} q_i(\mathbf{w}_m \mid \mathbf{w}_{m+1}) \, \mathrm{d}\mathbf{w}_1 \cdots \mathrm{d}\mathbf{w}_n \\ &= \int \prod_{i=1}^{P} q_i(\mathbf{w}_n \mid \mathbf{w}_{n+1}) \left(\int \prod_{i=1}^{P} \prod_{m=1}^{n-1} q_i(\mathbf{w}_m \mid \mathbf{w}_{m+1}) \, \mathrm{d}\mathbf{w}_1 \cdots \mathrm{d}\mathbf{w}_{n-1} \right) \mathrm{d}\mathbf{w}_n \\ &= \int \prod_{i=1}^{P} q_i(\mathbf{w}_n \mid \mathbf{w}_{n+1}) I_n(\mathbf{w}_n) \, \mathrm{d}\mathbf{w}_n \end{aligned} \tag{22}$$

and then using the initial condition $I_1(\mathbf{w}_1) = 1$. Since each of the densities $q_i(\mathbf{w}_n \mid \mathbf{w}_{n+1})$ is the Gaussian function (16), then the integral terms will be Gaussian also, and moreover we can evaluate each I_n up to any multiplicative constants which do not depend on $\mathbf{w}_n$. In particular each $I_n(\mathbf{w}_n)$ is going to be a Gaussian function, say

$$I_n(\mathbf{w}_n) \propto \mathcal{N}(\mathbf{w}_n \mid \mathbf{m}_n, \mathbf{S}_n) \tag{23}$$

the parameters of which are determined recursively through the use of (19b) giving

$$\begin{aligned} I_{n+1}(\mathbf{w}_{n+1}) &\propto \int \prod_{i=1}^{P} \mathcal{N}(\mathbf{w}_n \mid \mathbf{A}_{i,n} \mathbf{w}_{n+1} + \mathbf{b}_{i,n}, \mathbf{\Lambda}_{i,n}^{-1}) \mathcal{N}(\mathbf{w}_n \mid \mathbf{S}_n) \, \mathrm{d}\mathbf{w}_n \\ &\propto \mathcal{N}(\mathbf{w}_{n+1} \mid \boldsymbol{\alpha}'_{n+1}, \mathbf{\Gamma}'_{n+1}) \\ &\quad \times \int \mathcal{N}(\mathbf{w}_n \mid \mathbf{A}'_n \mathbf{w}_{n+1} + \mathbf{b}'_n, \mathbf{\Sigma}'_n) \mathcal{N}(\mathbf{w}_n \mid \mathbf{m}_n, \mathbf{S}_n) \, \mathrm{d}\mathbf{w}_n \\ &\propto \mathcal{N}(\mathbf{w}_{n+1} \mid \boldsymbol{\alpha}'_{n+1}, \mathbf{\Gamma}'_{n+1}) \mathcal{N}(\mathbf{m}_n \mid \mathbf{A}'_n \mathbf{w}_{n+1} + \mathbf{b}'_n, \mathbf{\Sigma}'_n + \mathbf{S}_n) \\ &\propto \mathcal{N}(\mathbf{w}_{n+1} \mid \mathbf{m}_{n+1}, \mathbf{S}_{n+1}) \end{aligned} \tag{24}$$

where we have used standard results for the posterior of linear Gaussian models

$$\mathbf{m}_{n+1} = \mathbf{S}_{n+1}\left\{(\mathbf{A}'_n)^\top\left(\mathbf{S}_n + \boldsymbol{\Sigma}'^{-1}_n\right)^{-1}\left(\mathbf{m}_n - \mathbf{b}'_n\right) + (\boldsymbol{\Gamma}'_{n+1})^{-1}\boldsymbol{\alpha}'_{n+1}\right\} \tag{25a}$$

$$\mathbf{S}_{n+1} = \left((\boldsymbol{\Gamma}'_{n+1})^{-1} + (\mathbf{A}'_n)^\top\left(\mathbf{S}_n + \boldsymbol{\Sigma}'_n\right)^{-1}\mathbf{A}_n\right)^{-1}. \tag{25b}$$

The equivalent transition model Using the results of the previous section we are now ready to write the equivalent model for the new transition density. We first recall that up to a normalising constant our full density is given by

$$q(\mathbf{w}_1, \ldots, \mathbf{w}_N) \propto \prod_i q_i(\mathbf{w}_N) \prod_{m=1}^{N-1} q_i(\mathbf{w}_m \mid \mathbf{w}_{m+1}), \tag{26}$$

then for $j = 0, \ldots, N-1$, and where we evaluate to unity any product for which the lower and upper limits are equal. Then

$$\begin{aligned} q(\mathbf{w}_{N-j}, \ldots, \mathbf{w}_N) &\propto \int \prod_i q_i(\mathbf{w}_N) \prod_{m=1}^{N-1} q_i(\mathbf{w}_m \mid \mathbf{w}_{m+1})\, \mathrm{d}\mathbf{w}_1 \cdots \mathrm{d}\mathbf{w}_{N-j-1} \\ &= \prod_i q_i(\mathbf{w}_N) \prod_{m=N-j}^{N} q_i(\mathbf{w}_m \mid \mathbf{w}_{m+1}) \\ &\quad \times \int \prod_{i=1}^{P} \prod_{n=1}^{N-j-1} q_i(\mathbf{w}_n \mid \mathbf{w}_{n+1})\, \mathrm{d}x_1 \cdots \mathrm{d}\mathbf{w}_{N-j-1} \\ &= \prod_{i=1}^{P} q_i(\mathbf{w}_N) \prod_{m=N-j}^{N} q_i(\mathbf{w}_m \mid \mathbf{w}_{m+1}) \times I_{N-j}(\mathbf{w}_{N-j}). \end{aligned} \tag{27}$$

In particular we have that the transition model becomes, up to normalising constants

$$q(\mathbf{w}_N) \propto \prod_{i=1}^{N} q_i(\mathbf{w}_N) \times I_N(\mathbf{w}_N) \tag{28a}$$

$$q(\mathbf{w}_n \mid \mathbf{w}_{n+1}) \propto \prod_i q_i(\mathbf{w}_n \mid \mathbf{w}_{n+1}) \times I_n(\mathbf{w}_n), \qquad n = 1, \ldots, N-1. \tag{28b}$$

Note that the term $I_n(\mathbf{w}_n)$ gives the additional correction needed to the density obtained by normalising the product of transition densities alone. Using (19a) we can rewrite this as

$$\begin{aligned} q(\mathbf{w}_n \mid \mathbf{w}_{n+1}) &\propto \mathcal{N}(\mathbf{w}_n \mid \mathbf{A}_n\mathbf{w}_{n+1} + \mathbf{b}_n, \boldsymbol{\Sigma}_n) \\ &\quad \times \mathcal{N}(\mathbf{w}_n \mid \boldsymbol{\alpha}_n, \boldsymbol{\Gamma}_n) \times \mathcal{N}(\mathbf{w}_n \mid \mathbf{m}_n, \mathbf{S}_n) \\ &\propto \mathcal{N}(\mathbf{w}_n \mid \mathbf{A}_n\mathbf{w}_{n+1} + \mathbf{b}_n, \boldsymbol{\Sigma}_n)\mathcal{N}(\mathbf{w}_n \mid \hat{\boldsymbol{\xi}}_n, \hat{\mathbf{S}}_n) \end{aligned} \tag{29}$$

where

$$\hat{\mathbf{S}}_n = \left(\boldsymbol{\Gamma}_n^{-1} + \mathbf{S}_{n+1}^{-1}\right)^{-1} \tag{30a}$$

$$\hat{\boldsymbol{\xi}}_n = \hat{\mathbf{S}}_{n+1}\left(\boldsymbol{\Gamma}_n^{-1}\boldsymbol{\alpha}_n + \mathbf{S}_n^{-1}\mathbf{m}_n\right). \tag{30b}$$

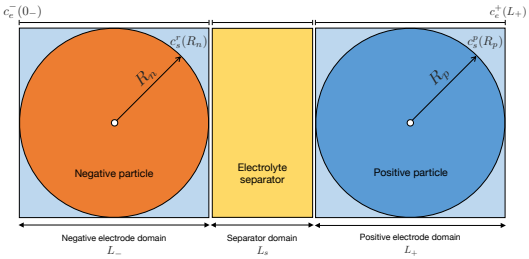
These are the parameters which will be used to evaluate the product of experts posterior (15) by treating it as a single equivalent state space model run backwards in time with the transition model (29), in particular we can write this as

$$q(\mathbf{w}_1, \ldots, \mathbf{w}_N) \propto q(\mathbf{w}_N) \prod_{n=1}^{N-1} \mathcal{N}(\hat{\boldsymbol{\xi}}_n \mid \mathbf{w}_n, \hat{\mathbf{S}}_n)\mathcal{N}(\mathbf{w}_n \mid \mathbf{A}_n\mathbf{w}_{n+1} + \mathbf{b}_n, \boldsymbol{\Sigma}_n) \tag{31}$$

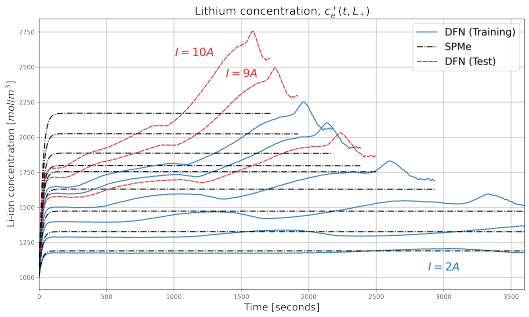
which is the density of a linear Gaussian state space model running backwards with states $\mathbf{w}_n$ and "pseudo-observations" given by the statistics $\hat{\boldsymbol{\Xi}} = \{\hat{\boldsymbol{\xi}}_n\}_{n=1}^{N-1}$. Therefore we can normalise

the marginal components of (31) by running filtering and smoothing conditioned on these set of pseudo-observations. Finally in our particular applications our state vectors will be $\mathbf{w}_n = \mathbf{z}_n$ or $\mathbf{w}_n = (\mathbf{x}, \mathbf{z}_n)^\top$ depending on whether we require the joint distribution or just the distribution of the shared latent forces. Note in particular that for prediction on a new task we only require the latent forces, and therefore we are carrying out filtering and smoothing on a model of state dimension M. We also note that typically this posterior will depend on additional parameters, however now that we have access to a linear Gaussian state space model for the state variables it is straightforward to optimise these using standard methods such as expectation maximisation, see [Bishop, 2006].

5 Prediction of Lithium-ion dynamics under varying C-rates



(a) SPMe geometry



(b) Training data

Figure 2: (a) Depiction of the single particle model with electrolyte (SPMe). Also depicted are the four variables observed for each experiment, the Lithium-ion concentration for each solid particle $c_s^\pm$ and the electrolyte concentration at each endpoint, $c_e^-(0_-)$ and $c_e^+(L_+)$. (b) Trajectories of c_e^+ for our training and test data from the DFN model. Also shown are the trajectories of the SPMe model at the same discharge rates, these trajectories are close for slow discharge at $I = 2$, and increasingly diverges as the current increases to $I = 10A$.

The Doyle-Fuller-Newman model (DFN) ([Fuller et al., 1994]) has become a benchmark for describing the behaviour of Li-ion batteries. It accounts for the porous microstructure of the electrodes using effective parameters to describe transport at the macro-scale coupled with spherical particles representative of the micro-scale. Unfortunately, to provide a comprehensive description of the underlying physical mechanism across multiple-scales this model requires a large system of PDEs over a complex geometry connected by algebraic relationships. This makes the model computationally demanding to forward simulate from, and therefore impedes calibration and control applications which would require frequent use calls to an expensive numerical solver.

This has motivated work in creating simplified approximations of the underlying dynamics, an important class are the single particle models which aim to approximate the electrodes of the battery by a single representative particle, and further simplify the dynamics. The single particle model with electrolyte (SPMe) [Moura et al., 2017, Marquis et al., 2019] was introduced to provide a computationally more efficient approximation better suited to control applications. Of course creating this simplified approximation necessarily leads to a loss in the physical realism of the model, and therefore the problem of missing physics in the reduced approximation. This phenomena can be observed in Figure 2b where we plot realisations of the DFN and an equivalently parameterised SPMe model for various speeds of discharge. At slower discharge rates the agreement between the models is good, however at higher rates the governing dynamics of the DFN become increasingly nonlinear, and as such the accuracy of the SPMe approximation begins to break down. In this experiment we examine the potential of a shared set of latent forces to recover this information which we refer to the SPMe + LF model.

Model description The SPMe decomposes the battery into three domains: a single representative particle in the negative electrode, a single particle in the positive electrode and the electrolyte. We define the concentrations in these domains by $c_{\mathrm{s}}^-(r)$ for the concentration in the negative particle at a point given by radius r, and similarly $c_{\mathrm{s}}^+(r)$ for the concentration in the positive. The concentration of ions in the electrolyte is denoted by $c_{\mathrm{e}}(\omega)$ and defined across the entire domain $0 < \omega < L$. This domain can be decomposed in the negative electrode domain ($0 < \omega < L^-$), the separator domain ($L^- < \omega < L - L^+$) and the positive electrode domain ($L - L^+ < \omega < L$).

In all domains we assume that the diffusion coefficient is independent of the concentration value, so that the model so described gives a system of three linear partial differential equations. The electrolyte equation is defined to be linear within each subdomain, even though the parameters might change across subdomains.

The intercalated lithium concentration is modelled by a diffusion equation in a spherical particle that is representative of the electrode. Therefore, two diffusion equations need to be solved, one for each electrode, which can be written as

$$\frac{\partial c_{\mathrm{s}}^{\pm}}{\partial t} = D_{\mathrm{s}}^{\pm} \frac{1}{r^2} \frac{\partial}{\partial r} \left(r^2 \frac{\partial c_{\mathrm{s}}^{\pm}}{\partial r} \right), \qquad \text{in } 0 < r < R^{\pm}, \tag{32}$$

$$\frac{\partial c_{\mathrm{s}}^{\pm}}{\partial r} = 0, \qquad \text{at } r = 0, \tag{33}$$

$$D_{\mathrm{s}}^{\pm} \frac{\partial c_{\mathrm{s}}^{\pm}}{\partial r} = \pm \frac{I(t)}{a^{\pm} L^{\pm} A F}, \qquad \text{at } r = R^{\pm}, \tag{34}$$

$$c_{\mathrm{s}}^{\pm} = c_0^{\pm}, \qquad \text{at } t = 0, \tag{35}$$

where $D_{\mathrm{s}}^{\pm}$ is the diffusion coefficient, $R^{\pm}$ is the particle radius, $I(t)$ is the applied current, $a^{\pm}$ is the particle surface area per unit of volume, $L^{\pm}$ is the electrode thickness, A is the electrode plate area, F is the Faraday constant, and $c_0^{\pm}$ is the initial concentration. In the parameters, the superscripts $+$ and $-$ denote the positive and negative electrode, respectively.

The last PDE of the model is the electrolyte equation, which can be written as

$$\varepsilon(\omega) \frac{\partial c_{\mathrm{e}}}{\partial t} = \frac{\partial}{\partial \omega} \left(\varepsilon(\omega)^{1.5} D_{\mathrm{e}} \frac{\partial c_{\mathrm{e}}}{\partial \omega} \right) + (1 - t^{+}) J(\omega), \qquad \text{in } 0 < \omega < L, \tag{36}$$

$$\frac{\partial c_{\mathrm{e}}}{\partial \omega} = 0, \qquad \text{at } \omega = 0, L, \tag{37}$$

$$c_{\mathrm{e}} = c_{\mathrm{e}0}, \qquad \text{at } t = 0, \tag{38}$$

where

$$\varepsilon(\omega) = \begin{cases} \varepsilon^{-}, & \text{in } 0 \le \omega < L^{-}, \\ \varepsilon^{\mathrm{s}}, & \text{in } L^{-} \le \omega < L - L^{+}, \\ \varepsilon^{+}, & \text{in } L - L^{+} \le x \le L, \end{cases} \quad J(\omega) = \begin{cases} \frac{I(t)}{L^{-} A F}, & \text{in } 0 \le \omega < L^{-}, \\ 0, & \text{in } L^{-} \le \omega < L - L^{+}, \\ -\frac{I(t)}{L^{+} A F}, & \text{in } L - L^{+} \le x \le L. \end{cases} \tag{39}$$

Here, $\varepsilon(\omega)$ is the porosity, D_{e} is the ion diffusion coefficient, t^{+} the cation transference number, $J(\omega)$ is the volumetric reaction current density and c_{e0} is the initial ion concentration. Each part of the domain (negative electrode, separator and positive electrode) can have different porosity, so ε is defined to be piecewise constant where ε^{-}, ε^{s} and ε^{+} are the porosities in the negative electrode, semparator and positive electrode, respectively.

To produce the discrete operator $\mathbf{A}$ for each task the finite volumes method is used to discretise each of these three PDEs to give a matrix operator over each of the particles and the electrolyte, and these are then combined to give a single block diagonal matrix over the whole domain. In this particular application it turns out that the matrices $\mathbf{A}_p$ for each task will be the same, because the per-task variations only occur from changing the input current which enters as a deterministic forcing term. The results of the previous sections hold in the case of additional deterministic forcing with minor adaptations, indeed the desired modifications can be achieved by taking the deterministic forcing as the mean functions for the per-task driving white noise process $\mathbf{w}_p(t)$, $p = 1, \ldots, P$.

As discussed in Section 2 we will specify our model by way of a sensitivity function $s_r(\omega)$ where ω is a spatial co-ordinate of our battery domain. In this model we have effectively three domains, the two solid particles and the electrolyte which we denote by $\Omega_s^{\pm}$ and Ω_e respectively. The electrolyte domain is further subdivided as in Figure 2a for the positive and negative electrodes and the separator and we denote these further subdomains by Ω_e^i with $i \in \{+, s, -\}$. This gives a total of five domains and we are going to allow different perturbations to each domain by specifying a separate model for each domain. To also extrapolate onto previously unseen models we want this function to depend on the constant discharge current I, leading to the specification

$$s_r(\omega, I_p) = \sum_{k=1}^{Q^{\Omega_d}} \beta_k^{\Omega_d} \phi_{k,\Omega_d}(\omega, I_p), \qquad \omega \in \Omega_d, \quad p = 1, \ldots, P.$$

for $\Omega_d \in \{\Omega_s^+, \Omega_s^-, \Omega_e^-, \Omega_e^s, \Omega_e^+\}$, and where Q_d^Ω is the number of basis functions ϕ_{k,Ω_d} used. In the experiments reported for this paper we will use a cubic polynomials for each domain.

Data generation The DFN is assumed to capture the true dynamics of the underlying process, and therefore we use the PyBaMM package [Sulzer et al., 2020a] to simulate trajectories of this model at various rates of discharge by running simulations with a constant rate of discharge on $P = 10$ currents evenly spaced in $[I_{min}, I_{max}]$ with $I_{min} = 2A$ and $I_{min} = 8A$. We are then interested in the ability of our model to forward simulate the dynamics using the learned values of the latent forces. That is we are going to predict trajectories from the model at previously unseen discharge currents using the learned latent forces. We shall consider two forms of this challenge, first an interpolation type problem where we aim to predict at a discharge current within the interval $[I_{min}, I_{max}]$, and then the harder task of extrapolating onto a discharge current greater than I_{max}. For the interpolation challenge we take $I = 7A$ and for the extrapolation problem we take $I \in \{9A, 10A\}$ to investigate increasing degrees of extrapolation, these target trajectories are also visualised in Figure 2b.

Results The main results for the prediction experiment are displayed in Figure 3 where we plot forward simulations of our model conditional on the trained latent force posterior. For the interpolation problem in Figure 3a we see that the SPMe + LF model does a good job of capturing the true dynamics of the DFN model with the true trajectory contained entirely within the ± 2 standard deviation intervals of the predictive distribution marginals, by comparison the SPMe model does a poor job of capturing the behaviour of the DFN model, quickly reaching a steady state and missing the qualitative features of the dynamics.

Inspecting also the more challenging extrapolation we see that our model continues to do a good job of prediction at 9A, which we recall is a full one amp higher than our maximum training current. For the most extreme prediction problem at 10A we see that while the performance deteriorates, the result is still substantially better than the SPMe model alone, indicating that by sharing latent forces we have successfully learned a model that is able to capture the loss in physics that occurs when simplifying the complex DFN model to the SPMe version, and so learn a model that can extrapolate into domains with increasingly nonlinear physics, these results are also summarised in Table 1.

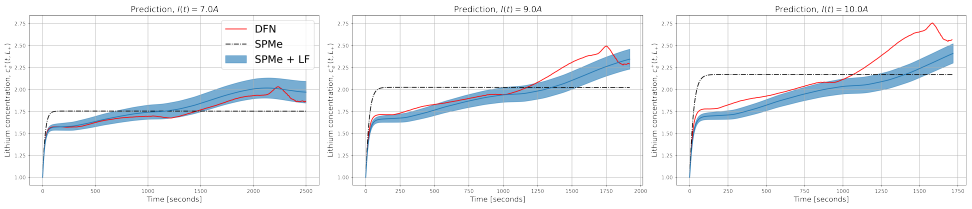
Finally, we recall our remarks in Section 3 that our approximation is as a product of experts, one arising from a fit to each task SMM independently. In Figure 4a we plot the latent force posterior for four of the independently trained tasks. We observe that the learned force for the task with $I = 2A$ is almost constant, we also recall from Figure 2b that at this low current discharge the SPMe already provided a good approximation to the DFN model, and therefore the fact that a very simple perturbation sufficed for this task coincides with our expectations. Conversely for faster discharge we see that more complex latent forces were required. In Figure 4b we plot the posterior learned from the joint model, we observe that this full posterior is most similar to that learned for the independent tasks with the faster discharge, indicating that these tasks have a greater influence on the final learned posterior.

Table 1: Prediction error on unseen discharge experiments. Reported are the mean squared errors in dimensionless quantities for both the SPMe model and N=100 samples from the SPMe + LF model when compared to the ground truth from the DFN model. See [Sulzer et al., 2020b,a] for the relevant scaling factors used when converting to dimensionless variables.

	Prediction Current		
Name	7	9	10
SPMe	0.017	0.054	0.088
SPMe + LF	0.006	0.005	0.023

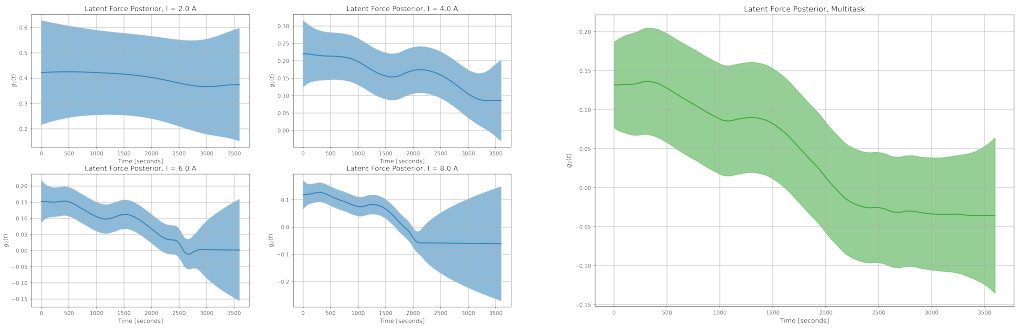
6 Discussion

In this work we have demonstrated that physics informed machine learning allows for an effective combination of mathematical model reduction, and data informed recovery of any resulting information loss. While the general recipe of "mathematical model + data-driven term" is simple, in practice care must be given to the specification of the data-driven term, and how to ensure that this term allows for efficient inference but also allows for the extraction of succificent information from the data to be useful for new tasks. In this work we have adopted a multi-task structure to achieve this information recovery, and then introduced an approximation which allows us to consider combinations of large numbers of tasks, or tasks with high state dimensions, in a scalable manner. This has reduced a cubic scaling in the number of tasks to parallel implementations of each tasks.



(a) Prediction at $I = 7A$ (b) Prediction at $I = 9A$ (c) Prediction at $I = 10A$

Figure 3: Prediction of electrolyte Lithium-ion concentration, $c^+(t, L^+$, from our trained model. (a) Shows the results of the interpolation type task. (b) Shows the results when predicting at a current higher than the range used when training, and (c) shows the same results for an even higher current. Also shown are the forward simulations from the SPMe simplification alone at the same currents.



(a) Per-task posterior for $g_2(t)$ (b) Multitask posterior for $g_2(t)$

Figure 4: A example of the learned posterior for one of the three latent forces in the experiment. (a) Shows the values for this latent force which would have been obtained training on the currents $I \in \{2, 4, 6, 8\}$ Amperes independently, these are the per-task experts for the posterior (b) Shows the resulting posterior after combining using the method discussed in Section 4.

We have then demonstrated the promising applicability of this framework for prediction of battery dynamics under varying discharge rates. While we have considered linear observation models in principle this method could be extended to more general filtering based approximations such as the extended Kalman filter [Säarkä and Solin, 2019] for nonlinear problems. Future work will also consider further quantification of the approximation error introduced by our product of SSM experts approximation.

Acknowledgments and Disclosure of Funding

This work is supported by HVM Catapult project number 8205, The Faraday Institution [EP/S003053/1 grant number FIRG003] and Lloyd's Register Foundation together with the Alan Turing Institute.

References

Mauricio Álvarez, David Luengo, and Neil D. Lawrence. Latent force models. volume 5 of *Proceedings of Machine Learning Research*, pages 9–16, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR.

Edwin V Bonilla, Kian M. Chai, and Christopher Williams. Multi-task gaussian process prediction. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 153–160. Curran Associates, Inc., 2008.

Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, August 2002. ISSN 0899-7667. doi: 10.1162/089976602760128018.

Neil D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 329–336. MIT Press, 2004.

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

Jouni Hartikainen, Mari Seppänen, and Simo Särkkä. State-space inference for non-linear latent force models with application to satellite orbit prediction. In *The 29th International Conference on Machine Learning (ICML 2012), University of Edinburgh, Scotland, from June 26 to July 1 2012*, pages 1–8. ACM, 2012. ISBN 978-1-4503-1285-1.

T.J. Rogers, K. Worden, and E.J. Cross. On the application of gaussian process latent force models for joint input-state-parameter estimation: With a view to bayesian operational identification. *Mechanical Systems and Signal Processing*, 140:106580, 2020. ISSN 0888-3270. doi: https://doi.org/10.1016/j.ymssp.2019.106580.

Simo Säarkä and Arno Solin. *Applied Stochastic Differential Equations*. Cambridge University Press, Cambridge, 2019.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Thomas F. Fuller, Marc Doyle, and John Newman. Simulation and Optimization of the Dual Lithium Ion Insertion Cell. *Journal of The Electrochemical Society*, 141(1):1–10, 1994. ISSN 0013-4651. doi: 10.1149/1.2054684.

Scott J. Moura, Federico Bribiesca Argomedo, Reinhardt Klein, Anahita Mirtabatabaei, and Miroslav Krstic. Battery State Estimation for a Single Particle Model with Electrolyte Dynamics. *IEEE Transactions on Control Systems Technology*, 25(2):453–468, 2017. ISSN 10636536. doi: 10.1109/TCST.2016.2571663.

Scott G. Marquis, Valentin Sulzer, Robert Timms, Colin P. Please, and S. Jon Chapman. An Asymptotic Derivation of a Single Particle Model with Electrolyte. *Journal of The Electrochemical Society*, 166(15):A3693–A3706, nov 2019. ISSN 0013-4651. doi: 10.1149/2.0341915jes. URL `http://jes.ecsdl.org/lookup/doi/10.1149/2.0341915jes`.

Valentin Sulzer, Scott G Marquis, Robert Timms, Martin Robinson, and S Jon Chapman. Python battery mathematical modelling (pybamm). *ECSarXiv. February*, 7, 2020a.

Valentin Sulzer, Scott G. Marquis, Robert Timms, Martin Robinson, and S. Jon Chapman. Python Battery Mathematical Modelling (PyBaMM). *ECSarXiv*, 2020b. doi: 10.1149/OSF.IO/67CKJ.