Probabilistic ODE Solutions in Millions of Dimensions

Nicholas Krämer [*1] **Nathanael Bosch** [*1] **Jonathan Schmidt** [*1] **Philipp Hennig** [1 2]

Abstract

Probabilistic solvers for ordinary differential equations (ODEs) have emerged as an efficient framework for uncertainty quantification and inference on dynamical systems. In this work, we explain the mathematical assumptions and detailed implementation schemes behind solving high-dimensional ODEs with a probabilistic numerical algorithm. This has not been possible before due to matrix-matrix operations in each solver step, but is crucial for scientifically relevant problems—most importantly, the solution of discretised partial differential equations. In a nutshell, efficient high-dimensional probabilistic ODE solutions build either on independence assumptions or on Kronecker structure in the prior model. We evaluate the resulting efficiency on a range of problems, including the probabilistic numerical simulation of a differential equation with millions of dimensions.

1. Introduction

Problem Statement This paper discusses a class of algorithms that computes the solution of initial value problems based on ordinary differential equations (ODEs), i.e. finding a function y that satisfies

$$\dot{y}(t) = f(y(t), t), \tag{1}$$

for all $t \in [t_0, t_{\max}]$, as well as the initial condition $y(t_0) = y_0 \in \mathbb{R}^d$. Usually, f is non-linear, in which case the solution of Equation (1) cannot generally be derived in closed form and has to be approximated numerically. We continue the work of *probabilistic* numerical algorithms for ODEs (Schober et al., 2019; Tronarp et al., 2019; Kersting et al., 2020b; Tronarp et al., 2021; Bosch et al., 2021; Krämer & Hennig, 2020). Like other filtering-based ODE solvers ("ODE filters"), the algorithm used herein translates the numerical approximation of ODE solutions to a problem of probabilistic inference. The resulting (approximate) posterior distribution quantifies the uncertainty associated with the unavoidable discretisation error (Bosch et al., 2021) and provides a language that integrates well with other data inference schemes (Kersting et al., 2020a; Schmidt et al., 2021). The main difference to prior work is that we focus on the setting where the dimension d of the ODE is high, that is, say, $d \gg 100$. (It is not clearly defined at which point an ODE counts as high-dimensional, but $d \approx 100$ is already a scale of problems in which previous state-of-the-art probabilistic ODE solvers faced computational challenges.)

Motivation and Impact High-dimensional ODEs describe the interaction of large networks of dynamical systems and appear in many disciplines in the natural sciences. The perhaps most prominent example is the simulation of discretised partial differential equations. There, the dimension of the ODE equals the number of grid points used to discretise the problem (with e.g. finite differences; Schiesser, 2012). More recently, ODEs gained popularity in machine learning through the advent of neural ODEs (Chen et al., 2018) or physics-informed neural networks (Raissi et al., 2019). With the growing complexity of the model, each of the above can quickly become high-dimensional. If such use cases shall gain from probabilistic solvers, fast algorithms for large ODE systems are crucial.

Prior Work and State-of-the-Art Many non-probabilistic ODE solvers, for example, explicit Runge–Kutta methods, have a computational complexity linear in the ODE dimension d (Hairer et al., 1993). Explicit Runge–Kutta methods are often the default choices in ODE solver software packages. Compared to the efficiency of the methods provided by DifferentialEquations.jl (Rackauckas & Nie, 2017), SciPy (Virtanen et al., 2020), or Matlab (Shampine & Reichelt, 1997), probabilistic methods have lacked behind so far. Intuitively, ODE filters are a fusion of ODE solvers and Gaussian process models—two classes of algorithms that suffer from high dimensionality. More precisely, the problem is that probabilistic solvers require matrix-matrix operations at

[*]Equal contribution [1]University of Tübingen, Tübingen, Germany [2]Max Planck Institute for Intelligent Systems, Tübingen, Germany. Correspondence to: Nicholas Krämer <nicholas.kraemer@uni-tuebingen.de>, Nathanael Bosch <nathanael.bosch@uni-tuebingen.de>, Jonathan Schmidt <jonathan.schmidt@uni-tuebingen.de>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

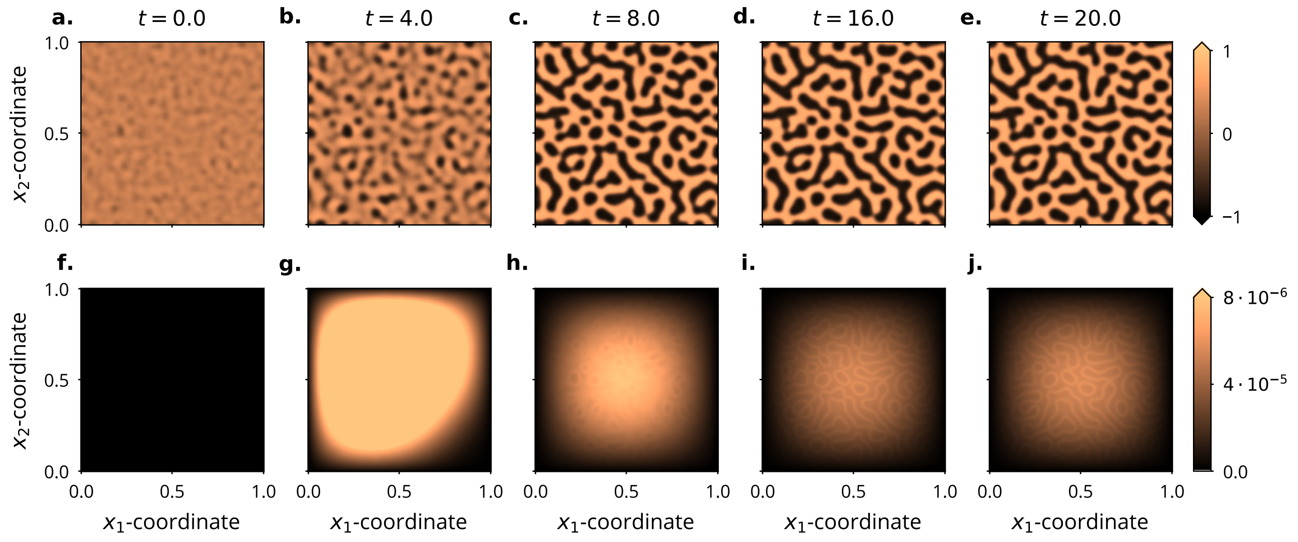

Figure 1. Simulating a high-dimensional ODE: Probabilistic solution of a discretised FitzHugh–Nagumo PDE model (Ambrosio & Françoise, 2009). Means (a-e) and standard-deviations (f-j), $t_0 = 0$ (left) to $t_{\max} = 20$ (right). The patterns in the uncertainties match those in the solution. The simulated ODE is 125k-dimensional. A detailed description of the experiment is provided in Appendix E.1.

each step. The matrices have $O(d^2)$ entries, which leads to $O(d^3)$ complexity for a single solver step and has made the solution of high-dimensional ODEs impossible. ODE filters are essentially nonlinear, approximate Gaussian process inference schemes (with a lot of structure). As in the GP community (e.g. Quiñonero-Candela & Rasmussen, 2005), the path to low computational cost in these models is via factorisation assumptions. But, perhaps surprisingly, the necessary assumptions are minimal, in the sense that they are already fulfilled by existing ODE filtering literature.

Contributions Our main contribution is to prove in which settings ODE filters admit an implementation in $O(d)$ complexity. Thereby, they become a class of algorithms comparable to explicit Runge–Kutta methods not only in estimation performance (error contraction as a function of evaluations of f; Kersting et al., 2020b; Tronarp et al., 2021) but also in computational complexity (cost per evaluation of f). The resulting algorithms deliver uncertainty quantification and other benefits of probabilistic ODE solvers on high-dimensional ODEs (see Figure 1; the ODE from this figure will be explored in more detail in Section 5). The key novelties of the present work are threefold:

1. *Acceleration via independence:* A-priori, ODE filters commonly assume independent ODE dimensions (e.g. Kersting et al., 2020b). We single out those inference schemes that naturally preserve independence. Identification of independence-preserving ODE solvers is helpful because each ODE dimension can be updated separately. The performance implications are that a single matrix-matrix operation with $O(d^2)$ entries is replaced with d matrix-matrix operations with $O(1)$ entries. In other words, $O(d)$ instead of $O(d^3)$ complexity for a single solver step (Proposition 3.3).

2. *Calibration of multivariate output-scales:* A single ODE system often models the interaction between states that occur on different scales. It is useful to acknowledge differing output scales in the "diffusivity" of the prior (details below). We generalise the calibration result by Bosch et al. (2021) to the class of solvers that preserve the independence of the dimensions (Proposition 3.2).

3. *Acceleration via Kronecker structure:* Sometimes, prior independence assumptions may be too restrictive. For instance, one might have prior knowledge of correlations between ODE dimensions (Example 4.1 in Section 4). Fortunately, a subset of probabilistic ODE solvers can exploit *and preserve* Kronecker structure in the system matrices of the state space. Preserving the Kronecker structure brings over the performance gains from above to dependent priors (Proposition 4.3).

To demonstrate how the independence and the Kronecker structure imply extreme scalability of the resulting algorithm, the experiments in Section 5 showcase simulations of ODEs with dimension $d \sim 10^7$.

2. ODE Filter Setup

The following section details the technical setup of an ODE filter, including the prior (Section 2.1), information model (Section 2.2), and practical considerations (Section 2.3).

2.1. Prior Model

The following is standard for probabilistic ODE solvers, and therefore essentially identical to the presentation by e.g. Schober et al. (2019). Herein, however, we place a stronger emphasis on Kronecker- and independence-structures in the system matrices compared to prior work. Both are important for the theoretical statements below. Särkkä (2013) or Särkkä and Solin (2019) provide a comprehensive explanation of the mathematical concepts regarding inference in state-space models.

Stochastic Process Prior on the ODE Solution Let $Y := (Y^i)_{i=1}^d = \left(Y_0^i, \dots, Y_\nu^i\right)_{i=1}^d$ solve the linear, time-invariant stochastic differential equation (SDE)

$$\mathrm{d}Y(t) = AY(t)\,\mathrm{d}t + B\,\mathrm{d}W(t), \tag{2}$$

subject to a Gaussian initial condition

$$Y(t_0) \sim \mathcal{N}(m_0, C_0) \tag{3}$$

for some m_0 and $C_0 := \Gamma \otimes \breve{C}_0$; e.g., $m_0 = 0$, $\breve{C}_0 = I$. The SDE is driven by a d-dimensional Wiener process W with diffusion $\Gamma \in \mathbb{R}^{d\times d}$, and governed by the system matrices

$$A := I_d \otimes \breve{A}, \quad \breve{A} := \sum_{q=0}^{\nu-1} e_q e_{q+1}^\top, \quad B := I_d \otimes e_\nu, \tag{4}$$

where $e_q \in \mathbb{R}^{\nu+1}$ is the q-th basis vector. The zeroth component of Y, $(Y_0^i)_{i=1}^d$, is an integrated Wiener process. With such A and B, the q-th component $(Y_q^i)_{i=1}^d$ models the q-th derivative of the integrated Wiener process. Similar SDEs can be written down for e.g. the integrated Ornstein-Uhlenbeck process or the Matérn process (the only differences would be additional non-zero entries in $\breve{A}$). If Γ were diagonal, the Kronecker structure in A and B would imply prior pairwise independence between Y^i and Y^j, $i \neq j$. Section 3 uses the diagonality assumption to reveal the efficient implementation of a class of ODE filters. Section 4 allows Γ to be any symmetric, positive definite matrix, which is why we do not make strong assumptions on Γ yet.

Discretisation Let $\mathbb{T} = (t_0, ..., t_N)$ be some time-grid with step-size $h_n := t_{n+1} - t_n$. While for the presentation, we assume a fixed grid, practical implementations choose t_n adaptively. Reduced to $\mathbb{T}$, due to the Markov property, the process Y becomes

$$Y(t_{n+1}) \mid Y(t_n) \sim \mathcal{N}(\Phi(h_n)Y_n, \Sigma(h_n)) \tag{5}$$

for matrices

$$\Phi(h_n) := \exp(Ah_n), \tag{6a}$$

$$\Sigma(h_n) := \int_0^{h_n} \Phi(h_n - \tau)B\Gamma B^\top \Phi(h_n-\tau)^\top \,\mathrm{d}\tau. \tag{6b}$$

The definition of $\Phi(h_n)$ uses the matrix exponential. $\Phi(h_n)$ inherits the block diagonal structure from A,

$$\Phi = I_d \otimes \breve{\Phi}(h_n), \quad \breve{\Phi}(h_n) := \exp(\breve{A}h_n), \tag{7}$$

and Σ has a Kronecker factorisation similar to C_0,

$$\Sigma(h_n) = \Gamma \otimes \breve{\Sigma}(h_n), \tag{8a}$$

$$\breve{\Sigma}(h_n) := \int_0^{h_n} \breve{\Phi}(h_n - \tau)e_\nu e_\nu^\top \breve{\Phi}(h_n-\tau)^\top \,\mathrm{d}\tau. \tag{8b}$$

The discretisation allows efficient extrapolation from t_n to t_{n+1}. Let $Y(t_n) \sim \mathcal{N}(m_n, C_n)$. Then,

$$Y(t_{n+1}) \sim \mathcal{N}(m_{n+1}^-, C_{n+1}^-) \tag{9}$$

with mean and covariance

$$m_{n+1}^- = \Phi(h_n)m_n, \tag{10a}$$

$$C_{n+1}^- = \Phi(h_n)C_n\Phi(h_n)^\top + \Sigma(h_n). \tag{10b}$$

For improved numerical stability, probabilistic ODE solvers compute this prediction in square root form, which means that only square root matrices of C_n and C_{n+1}^- are propagated without ever forming full covariance matrices (Krämer & Hennig, 2020; Grewal & Andrews, 2014). Appendix A recalls square root implementations of ODE filters.

2.2. Information Model

Information Operator The information operator

$$\mathcal{I}(Y)(t) := Y_1(t) - f(Y_0(t), t) \tag{11}$$

(recall $Y_q \approx y^{(q)}$), known as the local defect (Gustafsson, 1992), captures "how well (a sample from) Y_0 solves the ODE". If this value is large, the current state is an inaccurate approximation, and if it is small, Y_0 provides a good estimate of the truth. Loosely speaking, the goal is to make the defect as small as possible over the entire time domain.

Artificial Data The local defect $\mathcal{I}$ can be kept small by conditioning Y on $\mathcal{I}(Y)(t) \overset{!}{=} 0$ on "many" grid-points. Due to the regular prior and the regularity-preserving information operator $\mathcal{I}$, conditioning the prior on a zero-defect leads to an accurate ODE solution (Tronarp et al., 2021). Altogether, the probabilistic ODE solver targets

$$p\left(Y \,\middle|\, \{\mathcal{I}(Y)(t_n) = 0\}_{n=0}^N, Y_0(t_0) = y_0\right). \tag{12}$$

(Recall from Equation (2) that lower indices in Y refer to the derivative, i.e. Y_0 is the integrated Wiener process, and Y_q its q-th derivative.) We call the posterior in Equation (12) the *probabilistic ODE solution*. Unfortunately, a nonlinear vector field f implies a nonlinear information operator $\mathcal{I}$. Thus, the exact posterior is intractable.

Linearisation A tractable approximation of the probabilistic ODE solution is available through linearisation. Linearising f indirectly linearises $\mathcal{I}$, and the corresponding probabilistic ODE solution arises via Gaussian inference. Let F_y be (an approximation of) the Jacobian of f with respect to y. One can approximate the ODE vector field with a Taylor series

$$f(y) \approx \hat{f}_\xi(y) := f(\xi) + F_y(\xi)(y - \xi) \tag{13}$$

at some $\xi \in \mathbb{R}^d$. Let $E_q := I_d \otimes e_q$ be the projection matrix that extracts the q-th derivative from the full state Y. In other words, $\dot{Y}_0 = E_1 Y$. Equation (13) implies a linearisation of $\mathcal{I}$ at some $\eta \in \mathbb{R}^{d(\nu+1)}$,

$$\mathcal{I}(Y)(t) \approx \hat{\mathcal{I}}_\eta(Y)(t) := H(t)Y(t) + b(t), \tag{14}$$

with linearisation matrices

$$H(t) := E_1 - F_y(E_0\eta, t)E_0, \tag{15a}$$
$$b(t) := F_y(E_0\eta, t)E_0\eta - f(E_0\eta, t). \tag{15b}$$

$\hat{\mathcal{I}}_\eta$ is linear in $Y(t)$. Therefore, the approximate probabilistic ODE solution becomes tractable with Gaussian filtering and smoothing once $\hat{\mathcal{I}}_\eta$ is plugged into Equation (12) (Särkkä, 2013; Tronarp et al., 2019). At time t_n, the linearisation point η is usually chosen as the predicted mean m_n^-, which yields the extended Kalman filter (Särkkä, 2013). For ODE filters, there are three relevant versions of F_y: EK0, EK1, and the diagonal EK1.

EK0 Approximate the Jacobian as $F_y \equiv 0$, which has been a common choice since early work on ODE filters (Schober et al., 2019; Kersting et al., 2020b), and implies a zeroth-order approximation of f (Tronarp et al., 2019).

EK1 Use the full Jacobian $F_y = \nabla_y f$, which amounts to a first-order Taylor approximation of the ODE vector field (Tronarp et al., 2019). It is more stable than the EK0 (Tronarp et al., 2019), but in its general form, the EK1 does not fit the assumptions made below and thus does not immediately scale to high dimensions. Instead, we introduce the diagonal EK1.

Diagonal EK1 Use only the diagonal of the full Jacobian, $F_y = \operatorname{diag}(\nabla_y f)$. This choice conserves the efficiency of the EK0 to a solver that uses Jacobian information. The diagonal EK1 is another minor contribution of the present work. Section 5 empirically investigates how much stability using only the diagonal of the Jacobian provides.

Measurement and Correction A probabilistic ODE solver step consists of an extrapolation, measurement, and correction phase. Extrapolation has been explained in Equations (9) and (10) above. Denote

$$Y_{n+1}^- := Y(t_{n+1}) \sim \mathcal{N}(m_{n+1}^-, C_{n+1}^-). \tag{16}$$

The measurement phase approximates

$$Z_{n+1} := \hat{\mathcal{I}}_{m_{n+1}^-}(Y_{n+1}^-)(t_{n+1}) \approx \mathcal{I}(Y_{n+1}^-)(t_{n+1}) \tag{17}$$

by exploiting the linearisation matrices H and b,

$$Z_{n+1} \sim \mathcal{N}(z_{n+1}, S_{n+1}), \tag{18a}$$
$$z_{n+1} := H(t_{n+1})m_{n+1}^- + b(t_{n+1}), \tag{18b}$$
$$S_{n+1} := H(t_{n+1})C_{n+1}^- H(t_{n+1})^\top. \tag{18c}$$

Z_{n+1} will be used for calibration (details below). The extrapolated random variable is then corrected as

$$Y_{n+1} \sim \mathcal{N}(m_{n+1}, C_{n+1}), \tag{19a}$$
$$m_{n+1} := m_{n+1}^- - C_{n+1}^- H(t_{n+1})^\top S_{n+1}^{-1} z_{n+1}, \tag{19b}$$
$$C_{n+1} := \Xi\, C_{n+1}^-\, \Xi^\top, \tag{19c}$$
$$\Xi := I - C_{n+1}^- H(t_{n+1})^\top S_{n+1}^{-1} H(t_{n+1}). \tag{19d}$$

The update in Equations (19c) and (19d) is the Joseph update (Bar-Shalom et al., 2004). In practice, we never form the full C_{n+1} but compute only the square root matrix by applying Ξ to the square root matrix of C_{n+1}^-. It is not a Cholesky factor (because it is not lower triangular), but generic square root matrices suffice for numerically stable implementation of probabilistic ODE solvers (Krämer & Hennig, 2020).

2.3. Practical Considerations

Let us conclude with brief pointers to further practical considerations that are important for probabilistic ODE solvers.

Initialisation The state space models a stack of a y and the first ν derivatives. The stability of the ODE filter depends on the accurate initialisation of all derivatives (Krämer & Hennig, 2020). We initialise the solver by inferring

$$p(Y(t_0) \mid Y_0(\tau_m) = \hat{y}(\tau_m), m = 0, ..., \nu) \tag{20}$$

on $\nu+1$ small steps $\tau_0, ..., \tau_\nu$ where the $\hat{y}(\tau_m)$ are computed with e.g. a Runge–Kutta method. This is a slight generalisation of the strategy used by Schober et al. (2019) (also refer to Schober et al. (2014); Gear (1980)), in the sense that we formulate this initialisation as probabilistic inference instead of setting the first few means manually. An alternative strategy would be (Taylor-mode) automatic differentiation (Krämer & Hennig, 2020; Griewank & Walther, 2008).

Error Estimation Comprehensive explanation of error estimation and step-size adaptation, and its effect on the estimation quality and calibration of probabilistic ODE solvers, is out of scope for the present work; we refer the reader to Schober et al. (2019) and Bosch et al. (2021).

3. Independence Accelerates ODE Filters

This section establishes the main idea of the present work: *filtering-based probabilistic ODE solvers are fast and efficient when the prior models each dimension independently.*

3.1. Assumptions

Independent dimensions stem from a diagonal Γ.

Assumption 3.1. Assume that the diffusion Γ of the Wiener process in Equation (2) is a diagonal matrix.

Assumption 3.1 implies that the initial covariance C_0 (Equation (3)) is the Kronecker product of a diagonal matrix with another matrix, thus block diagonal. Assumption 3.1 is not very restrictive; in prior work on ODE filters, Γ was always either $\Gamma = \gamma^2 I$ for some $\gamma > 0$ (Schober et al., 2019; Tronarp et al., 2019; Kersting et al., 2020b; Bosch et al., 2021; Tronarp et al., 2021; Krämer & Hennig, 2020), or diagonal (Bosch et al., 2021).

3.2. Calibration

Tuning the diffusion Γ is crucial to obtain accurate posterior uncertainties. As announced in Section 2, the mathematical assumptions for calibrating Γ coincide with those that lead to an efficient ODE filter. Thus, we discuss Γ before proving the linear complexity of ODE filters under Assumption 3.1.

Four Approaches Recall the observed random variable Z_n (Equation (17)). ODE filters calibrate Γ with quasi-maximum-likelihood-estimation (quasi-MLE): Consider the prediction error decomposition (Schweppe, 1965),

$$p(\{Z_n\}_{n=0}^N) = p(Z_0) \prod_{n=0}^{N-1} p(Z_{n+1} \mid Z_n) \tag{21a}$$

$$\approx \mathcal{N}(z_0, S_0) \prod_{n=0}^{N-1} \mathcal{N}(z_{n+1}, S_{n+1}). \tag{21b}$$

Γ is a quasi-MLE if it maximises Equation (21b). The specific choice of calibration depends on the respective model for Γ, and reduces to one of four approaches: on the one hand, fixing and calibrating a *time-constant* Γ versus allowing a *time-varying* Γ; on the other hand, choosing a *scalar* diffusion $\Gamma = \gamma^2 I$ versus choosing a *vector-valued* diffusion $\Gamma = \operatorname{diag}(\gamma^1, ..., \gamma^d)$. Roughly speaking, a time-varying, vector-valued diffusion allows for the greatest flexibility in the probabilistic model. One contribution of the present work is to extend the vector-valued diffusion results by Bosch et al. (2021) to a slightly broader class of solvers (Proposition 3.2 below). Scalar diffusion will reappear in Section 4. Appendix B addresses time-constant diffusion.

Time-Varying Diffusion Allowing Γ to change over time, all measurements before time t_n are independent of Γ_n. Under the assumption of an error-free previous state (which is common for hyperparameter calibration in ODE solvers), a local quasi-MLE for $\Gamma_n = \gamma_n^2 I$ is (Schober et al., 2019)

$$\hat{\gamma}_n^2 := \frac{1}{d} z_n \left[H(t_n)\Sigma(h_n)H(t_n)^\top\right]^{-1} z_n. \tag{22}$$

This can be extended to a quasi-MLE for the EK0 with vector-valued $\Gamma_n = \operatorname{diag}(\gamma_n^1, ..., \gamma_n^d)$ (Bosch et al., 2021)

$$(\hat{\gamma}_n^i)^2 := (z_n^i)^2 / [H(t_n)\Sigma(h_n)H(t_n)^\top]_{ii}, \tag{23}$$

for all $i = 1, ..., d$. In this work, we generalise the EK0 requirement to Assumption 3.1 and a diagonal Jacobian.

Proposition 3.2. *Under Assumption 3.1 and for diagonal F_y, the estimators $(\hat{\gamma}_n^i)_i$ in Equation (23) are quasi-MLEs.*

Sketch of the proof. Two ideas are relevant: (i) a diagonal Jacobian implies a block diagonal $H(t_n)$ and a diagonal $H(t_n)\Sigma(h_n)H(t_n)^\top$ (which will be proved formally in Proposition 3.3 below); (ii) the local evidence, i.e. the probability of $\mathcal{N}(z_n, S_n)$ being zero, decomposes into a sum over the coordinates. Maximising each summand with respect to γ_n^i yields the claim. □

A very similar case can be made for time-constant diffusion (see Appendix B). Proposition 3.2 is a generalisation of the results by Bosch et al. (2021) in the sense that Proposition 3.2 is not restricted to the EK0.

3.3. Complexity

Now, with calibration in place, we can discuss the computational complexity of ODE filters under Assumption 3.1. The following proposition establishes that for diagonal Jacobians, a single solver step costs $O(d)$.

Proposition 3.3. *Suppose that Assumption 3.1 is in place. If the Jacobian of the ODE is (approximated as) a diagonal matrix, then a single step with a filtering-based probabilistic ODE solver costs $O(d\nu^3)$ in floating-point operations, and $O(d\nu^2)$ in memory.*

Proof. Let $Y_n \sim \mathcal{N}(m_n, C_n)$. Assume that C_n is block diagonal. We show that block diagonality is preserved through a step, and since by Assumption 3.1, C_0 is block diagonal, we do not lose generality. Recall $\Phi(h_n)$ and $\Sigma(h_n)$ from Equations (5) and (6). $\Phi(h_n)$ is block diagonal, and since Γ_n is diagonal, $\Sigma(h_n)$ is block diagonal.

(i) Extrapolate mean: The mean is extrapolated according to Equation (10a), which costs $O(d\nu^2)$ due to the block diagonal $\Phi(h_n)$. Each dimension is extrapolated independently.

(ii) Evaluate ODE: Next, $H = H(t_{n+1})$ and $b = b(t_{n+1})$ from Equation (15) are assembled, which involves evaluating f and F_y at $\xi := E_0 m^-_{n+1}$ (E_0 is a projection matrix and can be implemented as array indexing, so ξ comes at negligible cost). $F_y = \mathrm{diag}(F^1_y, ...F^d_y)$ is diagonal, thus

$$H = \mathrm{blockdiag}(H^1, ..., H^d) \tag{24}$$

is block diagonal with blocks

$$H^i := e_1 - e_0 F^i_y, \quad i = 1, ..., d \tag{25}$$

(recall the basis vectors e_q from Equation (4)). The block diagonal H has been pre-empted in Proposition 3.2 above.

(iii) Calibrate Γ: The cost of assembling the quasi-MLE for Γ_{n+1} according to Equation (22) or Equation (23) is $O(d)$, because the matrix to be inverted is diagonal.

(iv) Extrapolate the covariance: The covariance can be extrapolated dimension-by-dimension as well, because C_n, $\Phi(h_n)$, and $\Sigma(h_n)$ are all block diagonal with the same block structure: d square blocks with $\nu + 1$ rows and columns; recall Equation (10b). In reality, the matrix-matrix multiplication is replaced by a QR decomposition; we refer to Appendix A.1 for details on square root implementation. Using either strategy—square root or traditional implementation—extrapolating the covariance costs $O(d\nu^3)$ and C^-_{n+1} is block diagonal.

(v) Measure: Computing the mean of Z_{n+1} (recall Equations (17) and (18)) costs $O(d)$. The covariance S_{n+1} of Z_{n+1} is diagonal, since H and C^-_{n+1} are block diagonal. Thus, assembling *and inverting* S_{n+1} costs $O(d)$.

(vi) Correct mean and covariance: The mean is corrected according to Equation (19b), which—since S_{n+1} is diagonal—costs $O(d\nu)$. The covariance is corrected according to Equations (19c) and (19d), the complexity of which hinges on the structure of Ξ (Equation (19d)): due to the block diagonal C^-_{n+1}, H, and S_{n+1}, Ξ is block diagonal again, and correcting the covariance costs $O(d\nu^3)$. The square root of C_{n+1} arises by multiplying Ξ with the "left" square root matrix of C^-_{n+1}. The complexity does not change (asymptotically; QR decompositions cost more than matrix multiplications).

All in all, ODE filter steps preserve block-diagonal structure in the covariances. The expensive phases are the covariance extrapolation and correction in $O(d\nu^3)$ floating-point operations. The maximum memory demand is $O(d\nu^2)$ for the block diagonal covariances. □

While it may seem restrictive to use only the diagonal of the Jacobian, Proposition 3.3 includes the EK0, one of the central ODE filters. The $O(d)$ complexity puts the EK0 and the diagonal EK1 into the complexity class of explicit Runge–Kutta methods. Usually, $\nu < 12$ holds, so ν can be treated as some constant (Krämer & Hennig, 2020).

4. EK0 Preserves Kronecker Structure

As hinted in Section 1, scalar or diagonal diffusion may be too restrictive in certain situations (Krämer et al., 2022).

Example 4.1. Consider a spatio-temporal Gaussian process model $u(t, x) \sim \mathrm{GP}(0, \gamma^2 k_t \otimes k_x)$, where k_t is the covariance kernel that directly corresponds to an integrated Wiener process prior (Särkkä & Solin, 2019). Such a spatio-temporal model could be a useful prior distribution for applying an ODE solver to problems that are discretised PDEs, because k_x encodes spatial dependency structures. Restricted to a spatial grid $\mathcal{X} := \{x_1, ..., x_G\}$, $y := u(t, \mathcal{X})$ satisfies the prior model in Equations (2) to (3)[1], but with $\Gamma = \gamma^2 k_x(\mathcal{X}, \mathcal{X})$ (Solin, 2016), which is usually dense.

4.1. Assumptions

Despite the lack of independence in Example 4.1, fast ODE solutions remain possible with the EK0. In the remainder of this section, let $\Gamma = \gamma^2 \breve{\Gamma}$ for some matrix $\breve{\Gamma}$ and some scalar γ. Calibrating the scalar γ allows preserving Kronecker structure in the system matrices that appear in an ODE filter step (Proposition 4.3 below). Tronarp et al. (2019) show how for $\Gamma = \gamma^2 \breve{\Gamma}$, a time-constant quasi-MLE $\hat{\gamma}$ arises in closed form and also, that the posterior covariances all look like $C_n = \gamma^2 \breve{C}_n$: calibration can happen entirely post-hoc.

Constraints The following statement about linear complexity of ODE filters is only valid under two constraints: one can ignore (i) the quadratic costs of multiplying the posterior covariances with the quasi-MLE, and (ii) the cubic costs of solving a linear system involving Γ. The system matrices Φ, Σ, C_0 are all Kronecker products of a $\mathbb{R}^{d \times d}$ ("left") and a $\mathbb{R}^{\nu \times \nu}$ factor ("right"). The first constraint is thus avoided by scaling the "right" Kronecker factor of the covariances with γ^2 in $O(\nu^2)$. (ODE filters preserve Kronecker structure; see below.) The second one becomes the following assumption.

Assumption 4.2. Assume that the inverse of Γ is readily available and cheap to apply; that is, the quantity $x^\top \Gamma^{-1} x$ can be computed in $O(d)$.

Naturally, Assumption 4.2 holds for diagonal or at least sufficiently sparse matrices Γ. There are also settings in which Assumption 4.2 holds even if Γ is dense. For instance, if Γ is the covariance of a Gauss–Markov random field, the sparsity structure in Γ^{-1} implies adjacency of grid nodes (Lindgren et al., 2011; Sidén & Lindsten, 2020). In Example 4.1 with

[1]Technically, the stack of y and its ν derivatives does.

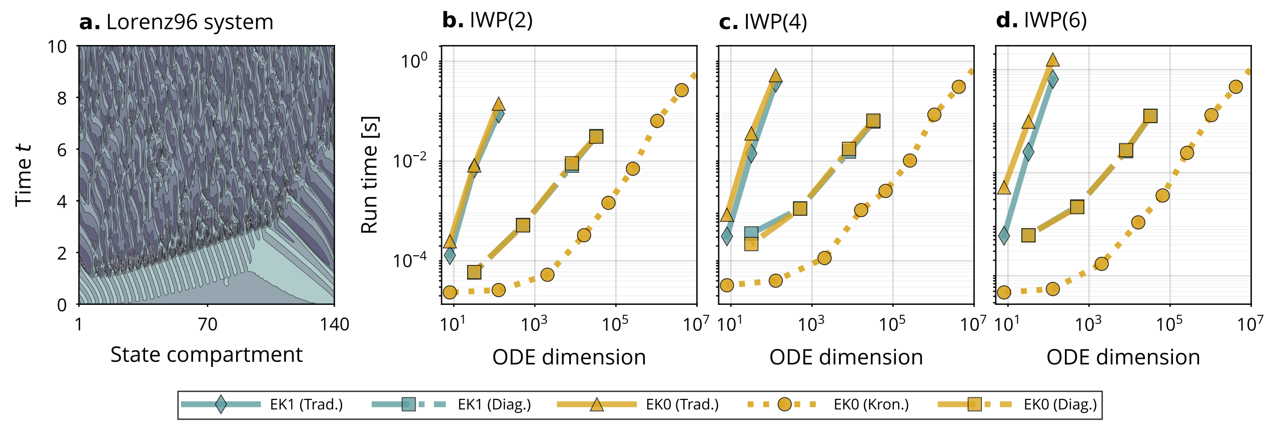

Figure 2. Runtime of a single ODE filter step: Run time (wall-clock) of a single step of ODE filter variations on the Lorenz96 problem (a) for increasing ODE dimension and $\nu = 2, 4, 6$ (b-d). The traditional implementations cost $O(d^3)$ per step, the diagonal EK1 and diagonal EK0 are $O(d)$ per step, just like the Kronecker version of the EK0. The Kronecker EK0 is faster than the diagonal solvers, because covariance operations involve a single $(\nu + 1) \times (\nu + 1)$ matrix instead of a batch of d such matrices (further details: Appendix C).

a spatial Matern kernel, for example, inverse Gram matrices can be approximated efficiently using the stochastic partial differential equation formulation (Lindgren et al., 2011).

4.2. Computational Complexity

Under Assumption 4.2, a single EK0 step costs $O(d)$:

Proposition 4.3. *Under Assumption 4.2, and if a time-constant diffusion model* $\Gamma = \gamma^2\breve{\Gamma}$ *is calibrated via* γ*, a single step of the EK0 costs* $O(\nu^3 + d\nu^2)$ *floating point operations, and* $O(d\nu + d^2 + \nu^2)$ *memory.*

The proof parallels that of Proposition 3.3 (details are in Appendix C). It hinges on computing everything only in the "right" factor of each Kronecker matrix. The proposition can be extended to time-varying diffusion if one tracks γ in the "right" Kronecker factor instead of the "left" one. Since this obfuscates the notation, we prove the claim in Appendix D. The $O(d^2)$ memory requirement is entirely due to the cost of storing Γ. If Γ or Γ^{-1} are sparse, it reduces to $O(d\nu + \nu^2)$.

5. Empirical Evaluation

The remainder evaluates the efficiency of the proposals. Next to everything detailed above, our implementation uses the preconditioner suggested by Krämer & Hennig (2020). Its complexity is neglibible in light of Propositions 3.3 and 4.3, because all preconditioners are diagonal matrices. The full code for the solver implementation and experiments is publicly available on GitHub.[2]

[2] https://github.com/pnkraemer/million-dimension-prob-ode-solver-experiments

Single ODE Filter Step We begin by evaluating the cost of a single step of the ODE filter variations on the Lorenz96 problem (Lorenz, 1996). This is a chaotic dynamical system and recommends itself for the first experiment, as its dimension can be increased freely. Appendix E.2 contains a more detailed description of the ODE model. We time a single ODE filter step for increasing ODE dimension d and different solver orders $\nu \in \{2, 4, 6\}$. The results are depicted in Figure 2. The traditional EK0 and EK1 become infeasible due to their cubic complexity in the dimension. The diagonal EK1 and the diagonal EK0 exhibit their $O(d)$ cost. The Kronecker EK0 is cheaper than the independence-based solvers. A step with the Kronecker EK0 takes $\sim$1 second for a 16 million-dimensional ODE on a consumer-level CPU. Figure 2 confirms Propositions 3.3 and 4.3.

Full Simulation Next, we evaluate whether the performance gains for a single ODE filter step translate into a reduced overall runtime (including initialisation, step-size adaptation and calibration) on a medium-dimensional problem: the Pleiades problem (Hairer et al., 1993). It describes the motion of seven stars in a plane and is commonly solved as a system of 28 first-order ODEs. Appendix E.3 describes the ODE dynamics and the experimental setup in more detail. The results are in Figure 3. Pleiades reveals the increased efficiency of the ODE filters. The probabilistic solvers are as fast as Radau, only by a factor $\sim$10 slower than SciPy's RK45 (Virtanen et al., 2020), but 100 times faster than their reference implementations. (It should be noted that the ODE filters use just-in-time compilation for some components, whereas SciPy does not.) These findings confirm how the practical considerations (initialisation, step-size adaptation, etc.) scale sufficiently well to higher-dimensional settings.

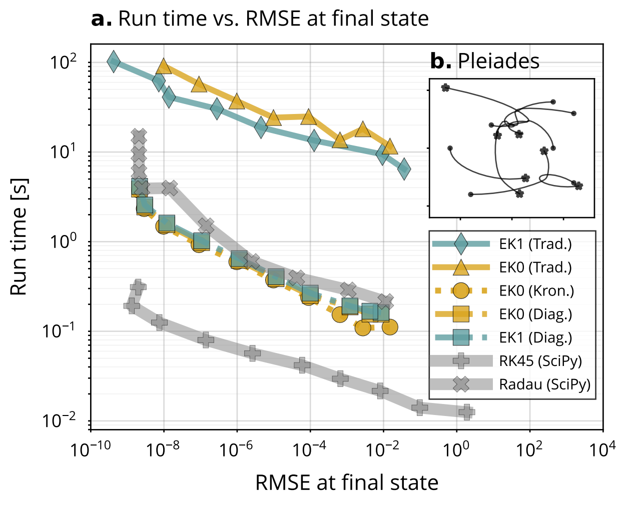

Figure 3. Runtime efficiency of fast ODE filters: Run time per root mean-square error of the ODE filters (a) on the Pleiades problem (b). The figure also shows two reference ODE filters, EK0 and EK1 in the traditional implementation, and Scipy's RK45 (explicit) and Radau (implicit). On the 28-dimensional Pleiades problem, the improved implementation accelerates the ODE filter implementations significantly.

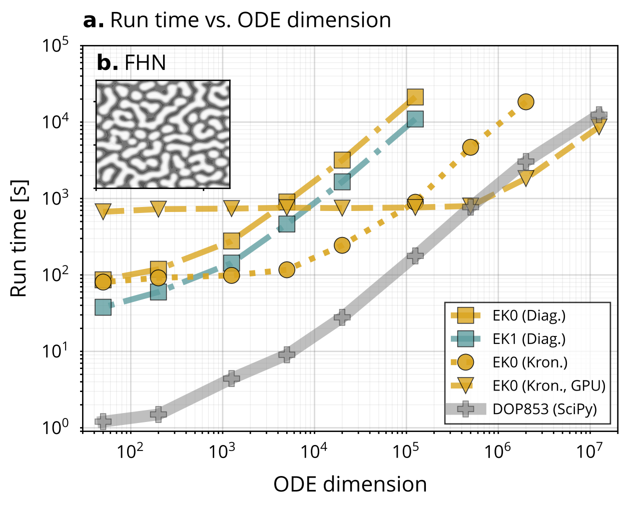

Figure 4. High-dimensional PDE discretisation: Run-time of ODE filters on the discretised FitzHugh–Nagumo model for increasing ODE dimension (i.e. increasing spatial resolution) including calibration and adaptive time-steps. SciPy's DOP853 for reference. Simulating $\gg 10^6$-dimensional ODEs takes $\approx 3h$.

High-Dimensional Setting To evaluate how well the improved efficiency translates to extremely high dimensions, we solve the discretised FitzHugh–Nagumo PDE model on high spatial resolution (which translates to high-dimensional ODEs); details on the experiment setup can be found in Appendix E.4. The results are in Figure 4. The main takeaway

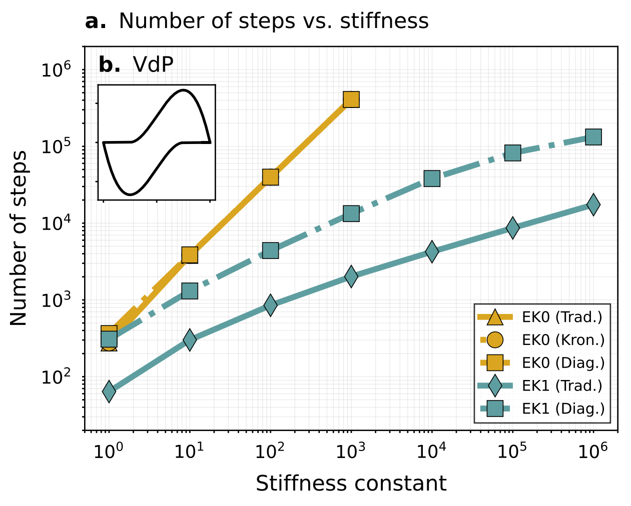

Figure 5. Stability: Number of steps taken by an ODE filter (a) for an increasingly stiff Van der Pol system (b). The diagonal EK1 is more stable than the EK0, but less stable than the EK1 (which is expected because it uses strictly less Jacobian information).

is that ODEs with millions of dimensions can now be solved *probabilistically* within a realistic time frame (hours), which has not been possible before. GPUs improve the runtime for extremely high-dimensional problems ($d \gg 10^5$).

Stability of the Diagonal EK1 How much do we lose by ignoring off-diagonal elements in the Jacobian? To evaluate the loss (or preservation) of stability against the A-stable EK1 (Tronarp et al., 2019), we solve the Van der Pol system (Guckenheimer, 1980). (The reference solver requires a low-dimensional ODE.) It includes a free parameter $\mu > 0$, whose magnitude governs the stiffness of the problem: the larger μ, the stiffer the problem, and for e.g. $\mu = 10^6$, Van der Pol is a famously stiff equation. For further details see Appendix E.5. The results are in Figure 5. We observe how the diagonal EK1 is less stable than the traditional EK1 for an increasing stiffness constant, but also that it is significantly more stable than the EK0. It is a success that the diagonal EK1 solves the Van der Pol equation for large μ.

Uncertainty Calibration of the Diagonal EK1 It has previously been shown by Bosch et al. (2021) that the EK0, diagonal EK0 and EK1 are similarly-well calibrated, with the EK1 having a tendency to be underconfident. Here, we investigate the calibration of the diagonal EK1. We evaluate the chi-squared statistic

$$\chi^2(t) := \|m(t) - y(t)\|^2_{C(t)^{-1}}, \tag{26}$$

which measures the ratio of the error in the mean $m \in \mathbb{R}^d$ of Y_0, which approximates the ODE solution y, normalised

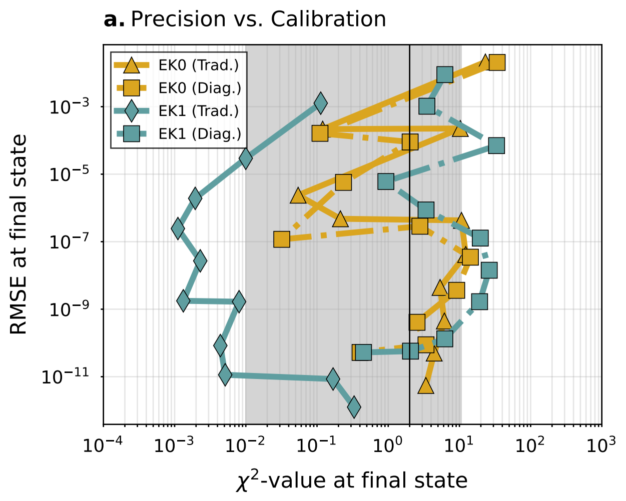

Figure 6. Uncertainty calibration: Chi-squared statistics and root-mean-square errors (RMSEs) over a range of tolerance levels. The gray area shows the 99% confidence intervals for the chi-squared statistic and the black horizontal line denotes perfect calibration ($\chi^2 = d$). While the traditional EK1 shows underconfidence ($\chi^2 \ll d$), the diagonal EK1 is more similar in calibration to the EK0 variants.

by the covariance $C \in \mathbb{R}^{d\times d}$ of Y_0, for a range of error-tolerance levels, on a non-stiff Van der Pol system. The results are in Figure 6. While the traditional EK1 tends to be underconfident, the calibration of the diagonal EK1 is more comparable to that of the EK0 variants. In particular, the diagonal approximation of the Jacobian does not seem to have a negative impact on the uncertainty calibration in terms of what is measured by the chi-squared statistic.

6. Conclusion

For probabilistic ODE solvers to capitalize on their theoretical advantages, their computational cost has to come close to that of their non-probabilistic, point-estimate counterparts (which benefit from decades of optimization). High-dimensional problems are one obstacle on this path, which we cleared here. We showed that independence assumptions in the underlying state-space model, or preservation of Kronecker structures, can bring the computational complexity of a large subset of known ODE filters close to non-probabilistic, explicit Runge–Kutta methods. As a result, probabilistic simulation of extremely large systems of ODEs is now possible, opening up opportunities to exploit the advantages of probabilistic ODE solvers on challenging real-world problems.

Acknowledgements

The authors gratefully acknowledge financial support by the German Federal Ministry of Education and Research (BMBF) through Project ADIMEM (FKZ 01IS18052B), and financial support by the European Research Council through ERC StG Action 757275 / PANAMA; the DFG Cluster of Excellence “Machine Learning - New Perspectives for Science”, EXC 2064/1, project number 390727645; the German Federal Ministry of Education and Research (BMBF) through the Tübingen AI Center (FKZ: 01IS18039A); and funds from the Ministry of Science, Research and Arts of the State of Baden-Württemberg. The authors also thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Nicholas Krämer, Nathanael Bosch, and Jonathan Schmidt. The authors thank Katharina Ott for helpful feedback on the manuscript.

References

Ambrosio, B. and Françoise, J.-P. Propagation of bursting oscillations. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1908):4863–4875, 2009.

Bar-Shalom, Y., Li, X. R., and Kirubarajan, T. *Estimation With Applications to Tracking and Navigation: Theory, Algorithms and Software*. John Wiley & Sons, 2004.

Bosch, N., Hennig, P., and Tronarp, F. Calibrated adaptive probabilistic ODE solvers. In *AISTATS 2021*, 2021.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural ordinary differential equations. In *NeurIPS 2018*, 2018.

Gear, C. W. Runge–Kutta starters for multistep methods. *ACM Transactions on Mathematical Software (TOMS)*, 6 (3):263–279, 1980.

Grewal, M. S. and Andrews, A. P. *Kalman Filtering: Theory and Practice with MATLAB*. John Wiley & Sons, 2014.

Griewank, A. and Walther, A. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, 2008.

Guckenheimer, J. Dynamics of the Van der Pol equation. *IEEE Transactions on Circuits and Systems*, 27(11):983–989, 1980.

Gustafsson, K. *Control of Error and Convergence in ODE solvers*. PhD thesis, Lund University, 1992.

Hairer, E., Nørsett, S. P., and Wanner, G. *Solving Ordinary Differential Equations I, Nonstiff Problems*. Springer, 1993.

Kersting, H., Krämer, N., Schiegg, M., Daniel, C., Tiemann, M., and Hennig, P. Differentiable likelihoods for fast inversion of "likelihood-free" dynamical systems. In *ICML 2020*, 2020a.

Kersting, H., Sullivan, T. J., and Hennig, P. Convergence rates of Gaussian ODE filters. *Statistics and Computing*, 30(6):1791–1816, 2020b.

Krämer, N. and Hennig, P. Stable implementation of probabilistic ODE solvers. *arXiv:2012.10106*, 2020.

Krämer, N., Schmidt, J., and Hennig, P. Probabilistic numerical method of lines for time-dependent partial differential equations. In *AISTATS 2022*, 2022.

Lindgren, F., Rue, H., and Lindström, J. An explicit link between Gaussian fields and Gaussian Markov random fields: The stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(4):423–498, 2011.

Lorenz, E. N. Predictability: A problem partly solved. In *Proceedings of the Seminar on Predictability*, volume 1, 1996.

Quiñonero-Candela, J. and Rasmussen, C. E. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6(65):1939–1959, 2005.

Rackauckas, C. and Nie, Q. DifferentialEquations.jl—a performant and feature-rich ecosystem for solving differential equations in Julia. *Journal of Open Research Software*, 5(1), 2017.

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

Särkkä, S. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.

Särkkä, S. and Solin, A. *Applied Stochastic Differential Equations*. Cambridge University Press, 2019.

Schiesser, W. E. *The Numerical Method of Lines: Integration of Partial Differential Equations*. Elsevier, 2012.

Schmidt, J., Krämer, N., and Hennig, P. A probabilistic state space model for joint inference from differential equations and data. *NeurIPS 2021*, 2021.

Schober, M., Duvenaud, D. K., and Hennig, P. Probabilistic ODE solvers with Runge–Kutta means. *NeurIPS 2014*, 2014.

Schober, M., Särkkä, S., and Hennig, P. A probabilistic model for the numerical solution of initial value problems. *Statistics and Computing*, 29(1):99–122, 2019.

Schweppe, F. Evaluation of likelihood functions for Gaussian signals. *IEEE Transactions on Information Theory*, 11(1):61–70, 1965.

Shampine, L. F. and Reichelt, M. W. The MATLAB ODE suite. *SIAM Journal on Scientific Computing*, 18(1):1–22, 1997.

Sidén, P. and Lindsten, F. Deep Gaussian Markov random fields. In *ICML 2020*, 2020.

Solin, A. *Stochastic Differential Equation Methods for Spatio-Temporal Gaussian Process Regression*. PhD thesis, 2016.

Tronarp, F., Kersting, H., Särkkä, S., and Hennig, P. Probabilistic solutions to ordinary differential equations as nonlinear Bayesian filtering: a new perspective. *Statistics and Computing*, 29(6):1297–1315, 2019.

Tronarp, F., Särkkä, S., and Hennig, P. Bayesian ODE solvers: The maximum a posteriori estimate. *Statistics and Computing*, 31(3):1–18, 2021.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

Wanner, G. and Hairer, E. *Solving Ordinary Differential Equations II, Stiff Problems*, volume 375. Springer, 1996.

A. Square-Root Implementation of Probabilistic ODE Solvers

The following two sections detail the square-root implementation of the transitions underlying the probabilistic ODE solver. The whole section is a synopsis of the explanations by Krämer & Hennig (2020). See also the monograph by Grewal & Andrews (2014) for additional details.

A.1. Extrapolation

The extrapolation step

$$C_{n+1}^{-} = \Phi(h_n) C_n \Phi(h_n)^\top + \Sigma(h_n) \tag{27}$$

does not lead to stability issues further down the line (i.e. in calibration/correction/smoothing steps) if carried out in square root form. Square-root form means that instead of tracking and propagating covariance matrices C, only square root matrices $C = \sqrt{C}\sqrt{C}^\top$ are used for extrapolation and correction steps without forming the full covariance.

This is possible by means of QR decompositions. The matrix square root $\sqrt{C_{n+1}^{-}}$ arises from $\sqrt{C_n}$ through the QR decomposition of

$$Q \begin{pmatrix} R \\ 0 \end{pmatrix} \leftarrow \begin{pmatrix} \sqrt{C_n}\Phi(h_n)^\top \\ \sqrt{\Sigma(h_n)}^\top \end{pmatrix}, \quad \sqrt{C_{n+1}^{-}} \leftarrow R^\top \tag{28}$$

because

$$R^\top R = \begin{pmatrix} R \\ 0 \end{pmatrix}^\top Q^\top Q \begin{pmatrix} R \\ 0 \end{pmatrix} = \begin{pmatrix} \sqrt{C_n}\Phi(h_n)^\top \\ \sqrt{\Sigma(h_n)}^\top \end{pmatrix}^\top \begin{pmatrix} \sqrt{C_n}\Phi(h_n)^\top \\ \sqrt{\Sigma(h_n)}^\top \end{pmatrix} = \Phi(h_n) C_n \Phi(h_n)^\top + \Sigma(h_n). \tag{29}$$

QR decompositions of a rectangular matrix $M \in \mathbb{R}^{m \times n}$, $m > n$ costs $O(mn^2)$, which implies that the covariance square root correction costs $O(d^3\nu^3)$. The QR decomposition is unique up to orthogonal row-operations (e.g. multiplying with ± 1). Probabilistic ODE solvers require only *any* square root matrix, so this equivalence relation can be safely ignored – they all imply the same covariance.

A.2. Correction

The correction follows a similar pattern. Recall the linearised observation model

$$\mathcal{I}(y) \approx \hat{\mathcal{I}}(y) = Hy + b \tag{30}$$

where H contains the vector field information and the Jacobian information (potentially, depending on linearisation style). There are two ways of performing square root corrections: the conventional way, and the way that is tailored to probabilistic ODE solvers, which builds on Joseph form corrections.

Conventional Way Let $\sqrt{C^-}$ be a matrix square root of the current extrapolated covariance (we drop the $n+1$ index for improved readability). Let 0_n be the $n \times n$ zero matrix, and $0_{n \times m}$ the $n \times m$ zero matrix. The heart of the square root correction is another QR decomposition of the matrix

$$Q \begin{pmatrix} R_{11} & R_{12} \\ 0_{d(\nu+1)\times d} & R_{22} \end{pmatrix} \leftarrow \begin{pmatrix} \sqrt{C^-}^\top H^\top & \sqrt{C^-}^\top \\ 0_d & 0_{d \times d(\nu+1)} \end{pmatrix} \tag{31}$$

with $R_{11} \in \mathbb{R}^{d \times d}$, $R_{12} \in \mathbb{R}^{d \times d(\nu+1)}$, and $R_{22} \in \mathbb{R}^{d(\nu+1) \times d(\nu+1)}$. The R_{ij} matrices contain the relevant information about the covariance matrices involved in the correction:

- ◇ $\sqrt{S} := R_{11}^\top$ is the matrix square root of the innovation covariance
- ◇ $\sqrt{C} := R_{22}^\top$ is the matrix square root of the posterior covariance
- ◇ $K := R_{12}(R_{11})^{-1}$ is the Kalman gain and can be used to correct the mean

This QR decomposition costs $O(d^3\nu^3)$ again, but the matrix involved is larger than the stack of matrices in the extrapolation step (it has d more columns), so for high-dimensional problems, the increased overhead becomes significant. However, if only *any* square root matrix is desired, this step can be circumvented.

Joseph Way Again, let $\sqrt{C^-}$ be the square root matrix of the current extrapolated covariance which results from the extrapolation step in square root form. Next, the full covariance is assembled (which goes against the usual grain of avoiding full covariance matrices, but works well in the present case) as $C^- = \sqrt{C^-}\sqrt{C^-}^\top$. Since in the probabilistic solver, C^- is either a Kronecker matrix $I_d \otimes \breve{C}^-$ or a block diagonal matrix $\mathrm{blockdiag}((C^-)^1, ..., (C^-)^d)$, this is sufficiently cheap. The innovation covariance itself then becomes

$$\begin{aligned} S &= HC^-H^\top && (32a)\\ &= (E_1 - F_x E_0)C^-(E_1 - F_x E_0)^\top && (32b)\\ &= E_1 C^- E_1^\top - F_x E_0 C^- E_1^\top - E_1 C^- E_0^\top F_x^\top + F_x E_0 C^- E_0^\top F_x^\top && (32c) \end{aligned}$$

which can be computed rather efficiently because $E_i C^- E_j^\top$ only involves accessing elements, not matrix multiplication. The only non-negligible cost here is multiplication with the Jacobian of the ODE vector field (which is often sparse in high-dimensional problems). Then, the Kalman gain

$$K = C^- H^\top S^{-1} \tag{33}$$

can be computed from S which implies that the covariance correction reduces to

$$\sqrt{C} = (I - KH)\sqrt{C^-} \tag{34}$$

which is the "left half" of the Joseph correction. The resulting matrix is square, and a matrix square root of the posterior covariance, but not triangular thus no valid Cholesky factor. If the sole purpose of the square root matrices is improved numerical stability, generic square root matrices suffice.

B. Independence and Fast ODE Filters for Time-Constant Diffusion

A similar result to Proposition 3.2 can be formulated for vector-valued, time-constant diffusion models.

Proposition B.1. *Under Assumption 3.1 and for diagonal F_y, a quasi-maximum likelihood estimate (MLE) for a vector-valued, time-constant diffusion model $\Gamma = \mathrm{diag}((\gamma^1)^2, ..., (\gamma^d)^2)$ is given by the estimator*

$$(\hat{\gamma}^i)^2 := \frac{1}{N}\sum_{i=1}^{N} \frac{(z_n^i)^2}{[S_n]_{ii}}, \qquad i = 1, \dots, d, \tag{35}$$

where $S_n := H(t_n)\Sigma(h_n)H(t_n)^\top$ is the diagonal covariance matrix of the measurement Z_n (recall Section 2.2).

Proof. The proof is structured as follows. First, we show that an initial covariance

$$C_0 = \mathrm{blockdiag}((\gamma^1)^2\breve{C}_0, \dots, (\gamma^d)^2\breve{C}_0) \tag{36}$$

implies covariances

$$\begin{aligned} C_n^- &= \mathrm{blockdiag}\left((\gamma^1)^2 (C_n^1)^-, \dots, (\gamma^d)^2 (C_n^d)^-\right), && (37a)\\ C_n &= \mathrm{blockdiag}\left((\gamma^1)^2 C_n^1, \dots, (\gamma^d)^2 C_n^d\right), && (37b)\\ S_n &= \mathrm{diag}\left((\gamma^1)^2 s_n^1, \dots, (\gamma^d)^2 s_n^d\right). && (37c) \end{aligned}$$

Then, for measurement covariances S_n of such form, we can compute the (quasi) maximum likelihood estimate $\hat{\Gamma}$. Because every covariance depends multiplicatively on γ, calibration can happen entirely post-hoc.

Block-Wise Scalar Diffusion Recall from Section 2.1 that the transition matrix and the process noise covariance are of the form $\Phi(h_n) = I_d \otimes \breve{\Phi}(h_n)$ and $\Sigma(h_n) = \Gamma \otimes \breve{\Sigma}(h_n)$. Thus, for a diagonal diffusion $\Gamma = \mathrm{diag}((\gamma^1)^2, ..., (\gamma^d)^2)$, both $\Phi(h_n)$ and $\Sigma(h_n)$ are block diagonal. Assuming a block diagonal covariance matrix that depends multiplicatively on γ,

$$C_{n-1} = \mathrm{blockdiag}\left((\gamma^1)^2 C_{n-1}^1, \dots, (\gamma^d)^2 C_{n-1}^d\right), \tag{38}$$

the extrapolated covariance is also of the form

$$C_n^- = \operatorname{blockdiag}\left((\gamma^1)^2 (C_n^1)^-, \dots, (\gamma^d)^2 (C_n^d)^-\right), \tag{39a}$$

$$(C_n^i)^- := \breve{\Phi}(h_n) C_{n-1}^i \breve{\Phi}(h_n)^\top + \breve{\Sigma}(h_n), \qquad i = 1, \dots, d. \tag{39b}$$

The diagonal Jacobian F_y implies a block diagonal linearisation matrix

$$H_n = E_1 - F_y E_0 = \operatorname{blockdiag}\left(H_n^1, \dots, H_n^d\right), \tag{40a}$$

$$H_n^i := e_1 - [F_y]_{i,i} e_0, \qquad i = 1, \dots, d. \tag{40b}$$

The measurement covariance S_n is therefore given by a *diagonal* matrix and depends multiplicatively on γ, as

$$S_n = H_n C_n^- H_n^\top = \operatorname{diag}\left((\gamma^1)^2 s_n^1, \dots, (\gamma^d)^2 s_n^d\right), \tag{41a}$$

$$s_n^i := H_n^i (C_n^i)^- (H_n^i)^\top, \qquad i = 1, \dots, d. \tag{41b}$$

This implies a block diagonal Kalman gain

$$\Xi_n = I - C_n^- H(t_n)^\top S_n^{-1} H(t_n) = \operatorname{blockdiag}\left(\Xi_n^1, \dots \Xi_n^d\right), \tag{42a}$$

$$\Xi_n^i := I_{\nu+1} - (C_n^-)^i (H_n^i)^\top H_n^i / s_n^i \qquad i = 1, \dots, d. \tag{42b}$$

Finally, we obtain the corrected covariance

$$C_n = \operatorname{blockdiag}\left((\gamma^1)^2 C_n^1, \dots, (\gamma^d)^2 C_n^d\right), \tag{43a}$$

$$C_n^i := \Xi_n^i (C_n^i)^- (\Xi_n^i)^\top \qquad i = 1, \dots, d. \tag{43b}$$

This concludes the first part of the proof.

Computing The Quasi-MLE It is left to compute the (quasi) MLE $\hat{\Gamma} = \operatorname{diag}((\hat{\gamma}^1)^2, ..., (\hat{\gamma}^d)^2)$ by maximizing the log-likelihood $\log p(z_{1:N}) = \log \prod_{n=1}^N \mathcal{N}(0; z_n, S_n)$. Since $S_n = \operatorname{diag}\left((\gamma^1)^2 s_n^1, \dots, (\gamma^d)^2 s_n^d\right)$ is a diagonal matrix, we obtain

$$\hat{\Gamma} = \arg\max_{\Gamma} \sum_{n=1}^N \log \mathcal{N}(0; z_n, S_n) \tag{44a}$$

$$= \arg\max_{\Gamma} \sum_{i=1}^d \left(-\frac{N \log (\hat{\gamma}^i)^2}{2} - \sum_{n=1}^N \frac{(z_n)_d^2}{2 s_n^i (\hat{\gamma}^i)^2} \right). \tag{44b}$$

By taking the derivative and setting it to zero, we obtain the quasi-MLE from Equation (35).

□

C. Proof of Proposition 4.3

Proof. Let $Y_n \sim \mathcal{N}(m_n, C_n)$. Assume $C_n = \Gamma \otimes \breve{C}_n$ which is no loss of generality, because such a Kronecker structure is preserved through the ODE filter step as shown below.

(i) Extrapolate mean: The mean extrapolation costs $O(d\nu^2)$ like in the proof of Proposition 3.3.

(ii) Evaluate the ODE: Evaluation of H and b is essentially free—recall that we only consider the EK0 in this setting, which uses the projection $H(t_n) = E_1$. Matrix multiplication with H consists of a projection, which costs $O(1)$.

(iii) Calibrate: Calibration of a time-constant γ^2 costs $O(d)$ under Assumption 4.2.

(iv) Extrapolate covariance: In the time-constant diffusion model, $\Sigma(h_n)$ and C_n are both Kronecker matrices and share the left Kronecker factor: Γ. Thus, the extrapolation of the covariance can be carried out "in the right Kronecker factor", which costs $O(\nu^3)$ in traditional as well as square root implementation. Denote the extrapolated covariance by $C_{n+1}^- := \Gamma \otimes \breve{C}_{n+1}^-$.

(v) Measure: Recall $H(t_n) = E_1 = I \otimes e_1$. The mean of the measured random variable $Z_n \sim \mathcal{N}(z_n, S_n)$ comes at negligible cost. The covariance

$$S_{n+1} = H(t_{n+1}) C^-_{n+1} H(t_{n+1})^\top = \Gamma \otimes \left[e_1 \breve{C}^-_{n+1} e_1^\top \right] \tag{45}$$

requires a single element in $\breve{C}^-_{n+1}$. The Kalman gain

$$K := C^-_{n+1} H(t_{n+1})^\top S^\top_{n+1} = I \otimes \breve{K}, \tag{46}$$

with $\breve{K} := e_1 \breve{C}^-_{n+1} / \left[e_1 \breve{C}^-_{n+1} e_1^\top \right]$ involves dividing the first row of $\breve{C}^-_{n+1}$ by a scalar. Its cost is $O(\nu + 1)$.

(v) Correct mean and covariance: The mean is corrected in $O(d\nu^2)$ as in the proof of Proposition 3.3. Due to the Kronecker structure in K, the "left" Kronecker factor of C_{n+1} must be Γ again. Therefore, we need to correct only the "right" Kronecker factor in $O(\nu^3)$.

All in all, under the assumption of cheap calibration, a single step with the EK0 costs $O(d\nu^2)$ and the expensive steps are (as before) the covariance extrapolation and the covariance correction. The total memory costs are the requirements of storing Γ, the mean(s) in $O(\nu d)$, and the "right" Kronecker factor(s) in $O(\nu^2)$. $\square$

D. Kronecker Structure and Fast ODE Filters for Time-Varying Diffusion

In the following we extend the results of Proposition 4.3 to time-varying diffusion models.

Proposition D.1. *Under Assumption 4.2, and if a time-varying diffusion model $\Gamma_n = \gamma_n^2 \breve{\Gamma}$ is calibrated via γ_n, a single step of the EK0 costs $O(\nu^3 + d\nu^2)$ floating point operations, and $O(d\nu + d^2 + \nu^2)$ memory.*

Proof. The proof of Proposition 4.3 shown in Appendix C depends on the specific time-fixed diffusion model only in the calibration (iii) and the extrapolation of the covariance (iv). In the following, we discuss these two steps for a time-varying diffusion $\Gamma_n = \gamma_n^2 \breve{\Gamma}$. We show that Kronecker structure is preserved and we obtain the same complexities as in Proposition 4.3.

(iii) Calibrate: Calibration of a time-varying γ^2_{n+1} is done with

$$\hat{\gamma}^2_{n+1} := \frac{1}{d} z^\top_{n+1} \left[H(t_{n+1}) \Sigma(h_{n+1}) H(t_{n+1})^\top \right]^{-1} z_{n+1} \tag{47a}$$

$$= \frac{1}{d} z^\top_{n+1} \left(\breve{\Gamma}_{n+1} \cdot \left[\breve{\Sigma}^-_{n+1} \right]_{11} \right)^{-1} z_{n+1} \tag{47b}$$

$$= \frac{1}{d} z^\top_{n+1} \left(\breve{\Gamma}_{n+1} \right)^{-1} z_{n+1} \Big/ \left[\breve{\Sigma}^-_{n+1} \right]_{11}, \tag{47c}$$

where we used that $H(t_n) = I \otimes e_1$. With Assumption 4.2, this computation costs $O(d)$.

(iv) Extrapolate covariance: Assume a covariance of the form $C_n = \left(\gamma_n^2 \breve{\Gamma} \right) \otimes \breve{C}_n$. Since scalars can be moved between the Kronecker factors, the covariance matrix can be written with the diffusion matrix $\Gamma_{n+1} = \gamma^2_{n+1} \breve{\Gamma}$, as

$$C_n = \left(\gamma^2_{n+1} \breve{\Gamma} \right) \otimes \left(\frac{\gamma_n^2}{\gamma^2_{n+1}} \breve{C}_n \right). \tag{48}$$

Then, since $\Sigma_{n+1} = \left(\gamma^2_{n+1} \breve{\Gamma} \right) \otimes \breve{\Sigma}_{n+1}$, the prediction step can be written as

$$C^-_{n+1} = \Phi(h_{n+1}) C_n \Phi(h_{n+1})^\top + \Sigma(h_{n+1}) \tag{49a}$$

$$= \left(I_d \otimes \breve{\Phi}(h_{n+1}) \right) \left(\left(\gamma^2_{n+1} \breve{\Gamma} \right) \otimes \left(\frac{\gamma_n^2}{\gamma^2_{n+1}} \breve{C}_n \right) \right) \left(I_d \otimes \breve{\Phi}(h_{n+1}) \right)^\top + \left(\left(\gamma^2_{n+1} \breve{\Gamma} \right) \otimes \breve{\Sigma}_{n+1} \right) \tag{49b}$$

$$= \left(\gamma^2_{n+1} \breve{\Gamma} \right) \otimes \left(\frac{\gamma_n^2}{\gamma^2_{n+1}} \breve{\Phi}(h_{n+1}) \breve{C}_n \breve{\Phi}(h_{n+1})^\top + \breve{\Sigma}_{n+1} \right) \tag{49c}$$

$$=: \left(\gamma^2_{n+1} \breve{\Gamma} \right) \otimes \breve{C}^-_{n+1}. \tag{49d}$$

With a Kalman gain of the form $K = I \otimes \breve{K}$, the corrected covariance can be written as $C_{n+1} = (\gamma_{n+1}^2 \breve{\Gamma}) \otimes \breve{C}_{n+1}$, thus confirming our assumption on the Kronecker structure of covariance matrices. Since all matrix multiplications happen only "in the right Kronecker factor", extrapolating the covariance costs $O(\nu^3)$.

All other parts of the proof can be reproduced as in Appendix C to obtain the specified complexities. □

E. Additional Details on the Empirical Evaluation

E.1. Figure 1: Simulating a high-dimensional ODE

The considered FitzHugh–Nagumo PDE is provided in Appendix E.4. For this first visualization, the PDE solved on the time span $t \in [0, 20]$ and in a spatial domain $x \in [-1.25, 1.25]^2$, discretized in intervals $\Delta_x = 0.01$. The probabilistic numerical solution is computed with a diagonal EK1 solver with a 2-times integrated Wiener process (IWP(2)) prior and a time-varying scalar diffusion model. Adaptive steps are chosen with tolerance levels $\tau_{\text{abs}} = 10^{-3}$, $\tau_{\text{rel}} = 10^{-1}$.

E.2. Single ODE Filter Step

Lorenz96 The Lorenz96 model describes a chaotic dynamical system for which the dimension can be chosen freely (Lorenz, 1996). It is given by a system of $N \geq 4$ ODEs

$$\dot{y}_1 = (y_2 - y_{N-1}) y_N - y_1 + F, \tag{50a}$$
$$\dot{y}_2 = (y_3 - y_N) y_1 - y_2 + F, \tag{50b}$$
$$\dot{y}_i = (y_{i+1} - y_{i-2}) y_{i-1} - y_i + F \qquad i = 3, \dots, N-1, \tag{50c}$$
$$\dot{y}_N = (y_1 - y_{N-2}) y_{N-1} - y_N + F, \tag{50d}$$

with forcing term $F = 8$, initial values $y_1(0) = F + 0.01$ and $y_{>1}(0) = F$, and time span $t \in [0, 30]$.

E.3. Full Simulation

Pleiades The Pleiades system describes the motion of seven stars in a plane, with coordinates (x_i, y_i) and masses $m_i = i$, $i = 1, \dots, 7$ (Hairer et al., 1993, Section II.10). It can be described with a system of 28 ODEs

$$\dot{x}_i = v_i \tag{51a}$$
$$\dot{y}_i = w_i \tag{51b}$$
$$\dot{v}_i = \sum_{j \neq i} m_j (x_j - x_i) / r_{ij}, \tag{51c}$$
$$\dot{w}_i = \sum_{j \neq i} m_j (y_j - y_i) / r_{ij}, \tag{51d}$$

where $r_{ij} = \left((x_i - x_j)^2 + (y_i - y_j)^2 \right)^{3/2}$, for $i, j = 1, \dots, 7$. It is commonly solved on the time span $t \in [0, 3]$ and with initial locations

$$x(0) = [3, 3, -1, -3, 2, -2, 2], \tag{52a}$$
$$y(0) = [3, -3, 2, 0, 0, -4, 4], \tag{52b}$$
$$v(0) = [0, 0, 0, 0, 0, 1.75, -1.5], \tag{52c}$$
$$w(0) = [0, 0, 0, -1.25, 1, 0, 0]. \tag{52d}$$

Further Details All considered probabilistic numerical solvers use a 4-times integrated Wiener process (IWP(4)) prior, as well as a time-varying scalar diffusion. The SciPy solutions and the fast PN solutions shown in Figure 3 correspond to absolute and relative tolerance levels $\tau_{\text{abs}}, \tau_{\text{rel}} \in \{10^{-i}\}_{i=3}^{12}$; PN solutions in their traditional implementation are only computed for tolerances $\tau_{\text{abs}}, \tau_{\text{rel}} \in \{10^{-i}\}_{i=3}^{10}$. The references solution is computed with the LSODA solver and with absolute and relative tolerances of $\tau_{\text{abs}} = 10^{-12}$, $\tau_{\text{rel}} = 10^{-12}$.

E.4. High-Dimensional Setting

FitzHugh–Nagumo PDE Let $\Delta = \sum_{i=1}^{d} \frac{\partial^2}{\partial x_i^2}$ be the Laplacian. The FitzHugh–Nagumo partial differential equation (PDE) is (Ambrosio & Françoise, 2009)

$$\frac{\partial}{\partial t} u(t,x) = a\Delta u(t,x) + u(t,x) - u(t,x)^3 - v(t,x) + k, \tag{53a}$$

$$\frac{\partial}{\partial t} v(t,x) = \frac{1}{\tau}(b\Delta v(t,x) + u(t,x) - v(t,x)), \tag{53b}$$

for some parameters a, b, k, τ, and initial values $u(t_0, x) = h_0(x)$, $v(t_0, x) = h_1(x)$. In our experiments, we chose $a = 208 \cdot 10^{-4}$, $b = 5 \cdot 10^{-3}$, $k = -5 \cdot 10^{-3}$, $\tau = 0.1$. As initial values, we used random samples from the uniform distribution on $(0, 1)$. We solve the PDE from $t_0 = 0$ to $t_{\text{max}} = 20$ on a range of spatial domains $x \in [0, W] \times [0, W] \subseteq \mathbb{R}^2$, with $W \in \{0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50\}$. To turn the PDE into a system of ODEs, we discretised the Laplacian with central, second-order finite differences schemes on a uniform grid. The mesh size of the grid determines the number of grid points, which controls the dimension of the ODE problem.

Further Details PN solutions are computed with a 3-times integrated Wiener process (IWP(3)) prior, a time-varying scalar diffusion, and with step-size adaptation for chosen tolerances $\tau_{\text{abs}} = 10^{-3}$, $\tau_{\text{rel}} = 10^{-1}$. The DOP853 solutions are computed with tolerance levels $\tau_{\text{abs}} = 10^{-6}$, $\tau_{\text{rel}} = 10^{-3}$.

E.5. Stability of the Diagonal EK1

Van der Pol The Van der Pol system is often employed to evaluate the stability of stiff ODE solvers (Wanner & Hairer, 1996). It is given by a system of ODEs

$$\dot{y}_1(t) = y_2(t), \qquad \dot{y}_2(t) = \mu\left(\left(1 - y_1^2(t)\right) y_2(t) - y_1(t)\right), \tag{54}$$

with stiffness constant $\mu > 0$, time span $t \in [0, 6.3]$, and initial value $y(0) = [2, 0]$.

Further Details All considered probabilistic numerical solutions are computed with a 5-times integrated Wiener process (IWP(5)) prior, a time-varying scalar diffusion model, and with step-size adaptation for tolerances $\tau_{\text{abs}} = 10^{-6}$, $\tau_{\text{rel}} = 10^{-3}$.