bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

Probabilistic solvers enable a straight-forward exploration of numerical uncertainty in neuroscience models

Jonathan Oesterle[1], Nicholas Krämer[2], Philipp Hennig[2,3], Philipp Berens[1,2]

1 Institute of Ophthalmic Research, University of Tübingen
2 Department of Computer Science, University of Tübingen
3 Max Planck Institute for Intelligent Systems, Tübingen, Germany

* philipp.berens@uni-tuebingen.de

Abstract

Understanding neural computation on the mechanistic level requires models of neurons and neuronal networks. To analyze such models one typically has to solve coupled ordinary differential equations (ODEs), which describe the dynamics of the underlying neural system. These ODEs are solved numerically with deterministic ODE solvers that yield single solutions with either no, or only a global scalar bound on precision. It can therefore be challenging to estimate the effect of numerical uncertainty on quantities of interest, such as spike-times and the number of spikes. To overcome this problem, we propose to use recently developed sampling-based probabilistic solvers, which are able to quantify such numerical uncertainties. They neither require detailed insights into the kinetics of the models, nor are they difficult to implement. We show that numerical uncertainty can affect the outcome of typical neuroscience simulations, e.g. jittering spikes by milliseconds or even adding or removing individual spikes from simulations altogether, and demonstrate that probabilistic solvers reveal these numerical uncertainties with only moderate computational overhead.

1 Introduction

Computational neuroscience is built around computational models of neurons that allow the simulation and analysis of signal processing in the central nervous system. These models can describe neural computations on different levels of abstraction. On the *statistical level*, e.g. generalized linear models have been used to provide a probabilistic model mapping environmental variables to neural activity [1]. For such statistical models, quantifying the uncertainty of the parameters can be achieved using Bayesian approaches [2]. On the *mechanistic level*, the models typically take the form of systems of coupled ordinary differential equations (ODEs), which describe the dynamics of the membrane potential and give rise to the spike-times [3, 4]. Recently, likelihood-free inference approaches have made it possible to perform uncertainty-aware inference even for such complicated mechanistic models [5–7].

However, mechanistic models of neurons are subject to an additional source of uncertainty: the numerical error caused by the solution of the model's ODEs with a concrete algorithm [8]. This arises because all numerical solvers are necessarily run with finite time and limited resources, so their estimate diverges from the true solution of the ODE, even if the problem is well-posed. When simulating neurons, one would like to compute a numerical solution close to the true solution of the ODE, to ensure that conclusions drawn from the simulations are based on the mechanisms described by the model rather than the specific choice, setting and implementation of the ODE solver.

Many of the well-established numerical solvers do report a global error estimate and a corresponding tolerance that can be set by the user [9, Chapter II.4]. This global scalar error, though, does not capture how the numerical error arising from finite step-sizes used in practice affects crucial quantities of interest in the simulation, such as spike-times or the number of spikes. In practice, it can therefore be challenging to select a tolerance that strikes a good balance between run time and accuracy.

For some of the most common mechanistic models in neuroscience like the Hodgkin-Huxley or Izhikevich neuron model, errors in numerical integration have been studied in detail for a range of solvers and different integration step-sizes [10–12]. These studies have shown that standard solvers are often not the best choice in terms of accuracy or the accuracy vs. run time tradeoff. Therefore, the authors of these studies proposed to use specific solvers for the analyzed models, e.g. the Parker-Sochacki method for the Hodgkin-Huxley and Izhikevich neuron [10], an exponential midpoint method [11] or second-order Strang splitting [12] for Hodgkin-Huxley-like models. While improving computations for the specific problems, applying these to other scenarios requires a detailed understanding of the kinetics of the neuron model of interest; and while choosing a "good" solver for a

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

given model is important, it is typically not necessary to choose the "best" ODE solver. In many cases, it can be sufficient to ensure that the computed solution is within a certain accuracy.

As a more general approach to quantify the numerical uncertainty in mechanistic models in neuroscience, we therefore propose to use probabilistic ODE solvers [8, 13, 14]. In contrast to classical ODE solvers, this class of solvers does not only yield a single solution, but instead a distribution over solutions that quantifies the numerical uncertainty.

Several frameworks for probabilistic ODE solvers have been proposed, which differ mostly in the tradeoff between computational cost and flexibility of the posterior, from fast Gaussian filters [15–17] to sampling-based approaches [18–22]. These solvers have been mostly tested for well-behaved systems with well-behaved solutions, but the ODEs used to simulate neural activity model the non-linear membrane dynamics that underlie the all-or-none nature of an action potential. Here, we use two related approaches of probabilistic ODE integration, the state perturbation proposed by Conrad et al. [18] and the step-size perturbation of Abdulle and Garegnani [22]. Both build on existing explicit, iterative ODE solvers and stochastically perturb the numerical integration of individual steps taken by the underlying solvers. These perturbations make the solution of every step probabilistic and therefore of the solution as a whole. The magnitude of the perturbation has to be calibrated, such that the solver's output distribution reflects the numerical uncertainty in the solution.

Here, we explore the potential of probabilistic ODE solvers for neuron models. We show how probabilistic solvers can be used to quantify and reveal numerical uncertainty caused by the numerical ODE integration and demonstrate that the solver outputs are easy to interpret. For this, we simulate typical neuron models, namely the Izhikevich neuron model [23], as a representative of leaky-integrate-and-fire neuron models, single-compartment Hodgkin-Huxley models [24] and a model with three synaptically coupled Hodgkin-Huxley-like neurons [25] as an example of a neuronal network model. Lastly, we discuss practical considerations and limitations of probabilistic solvers such as the calibration of the perturbation and the computational overhead.

Taken together, our results suggest that probabilistic ODE solvers should be considered as a useful tool for the simulation of neuronal systems, to increase the quality and reliability of such simulations over those achieved with classic solvers.

2 Methods and Models

2.1 Probabilistic solvers

Simulating neuron models typically amounts to solving an initial value problem (IVP) based on a set of coupled ODEs. In abstract form, an initial value problem is given by

$$\dot{\mathbf{x}}(t) = f(t, \mathbf{x}(t)), \quad \mathbf{x}(t_0) = \mathbf{x}_0, \tag{1}$$

where f, $\mathbf{x}_0$ and t_0 are known and $\mathbf{x}(t)$ for $t > t_0$ is the quantity of interest. The solution to the initial value problem at time $t + \Delta t$ provided the solution at time t, is given by integrating Eq. (1) from t to $t + \Delta t$:

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \int_t^{t+\Delta t} f(s, \mathbf{x}(s))\, \mathrm{d}s. \tag{2}$$

Except for special cases, this integral has no analytic form and must be solved numerically. For example, the forward Euler method approximates the integral as $\int_t^{t+\Delta t} f(s, \mathbf{x}(s))\, \mathrm{d}s \approx \Delta t \cdot f(t, \mathbf{x}(t))$. To simulate a neuron, Eq. (2) is solved iteratively, which results in a sequence of solutions $X = [x(t_0), x(t_1), x(t_2), ..., x(t_N)]$ for a set of time points with $t_{i+1} > t_i$ and a maximum time point t_N. Standard solvers yield a deterministic solution in every step, and therefore for the solution X as a whole. In contrast, the probabilistic solvers used in this study stochastically perturb the numerical integration used to approximate Eq. (2), which makes the solution of every step—and therefore of the whole solution—probabilistic. For a given IVP and solver, one can therefore generate a sample distribution of solutions X by repeating the iterative numerical integration from t_0 to t_N multiple times. To create these probabilistic solvers, we implemented the state perturbation algorithm of Conrad et al. [18] and the step-size perturbation algorithm of Abdulle and Garegnani [22].

In the state perturbation algorithm [18], in each step of the numerical integration, a small i.i.d. noise term $\boldsymbol{\xi}_t$ is added to the solution $\mathbf{x}_{\mathrm{det}}(t + \Delta t)$ of a corresponding deterministic integration scheme:

$$\mathbf{x}_{\mathrm{prb}}(t + \Delta t) = \mathbf{x}_{\mathrm{det}}(t + \Delta t) + \boldsymbol{\xi}_t, \quad \boldsymbol{\xi}_t \sim \mathcal{N}(\mathbf{0}, \mathrm{diag}(\boldsymbol{\nu}_t)^2), \tag{3}$$

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

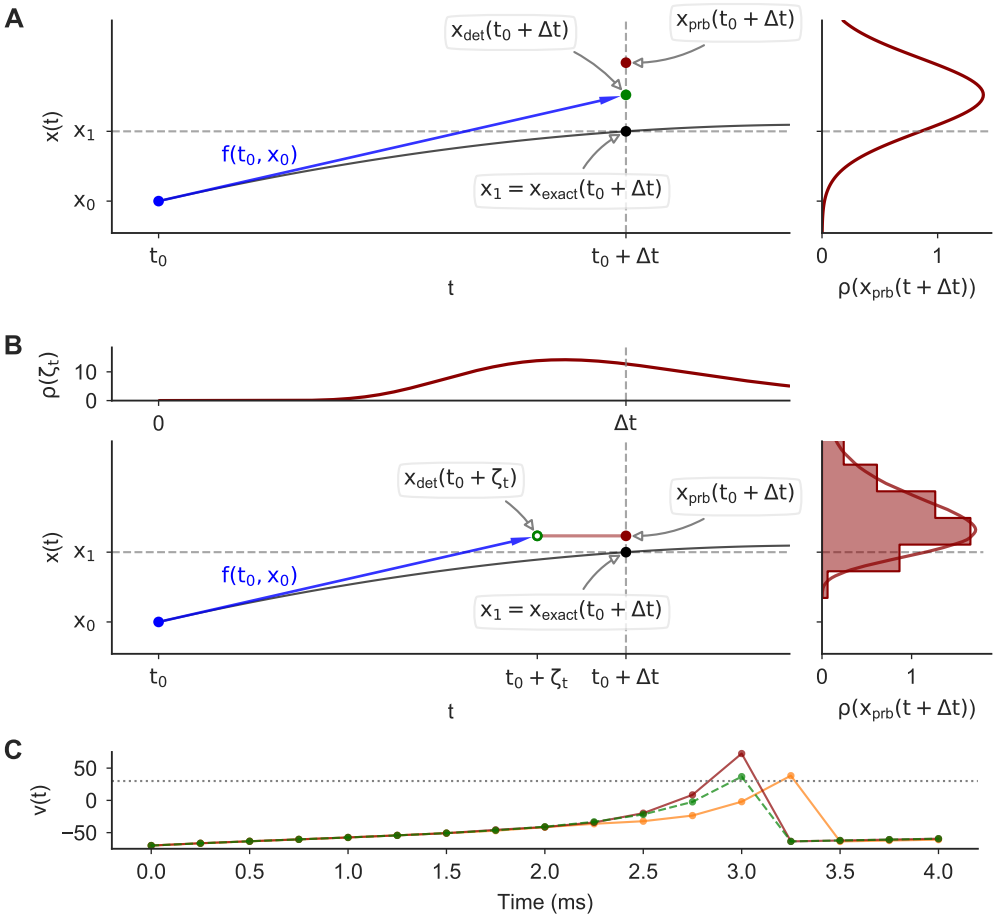

Figure 1: **Illustration of probabilistic ODE solvers. (A)** *Left*: A single integration step with a probabilistic forward Euler method using the state perturbation method [18] for the ODE $f(t, x(t)) = 3 \cdot x(t) \cdot \sin(t + 3)$ and the exact solution $x(t) = \exp(-3 \cdot \cos(t + 3))$ (black curve). We set $t_0 = 0$ and the step-size to Δt=0.1. The exact solution at $t = t_0 + \Delta t$ is highlighted (black dot). A first order solution is computed using forward Euler: $x_{\text{det}}(t + \Delta t) = x_0 + \Delta t \cdot f(t_0, x_0)$ (x_{det}: green dot, f: blue arrow). *Right*: The probability density function ρ of $x_{\text{prb}}(t + \Delta t)$, where $x_{\text{prb}}(t + \Delta t)$ is the output of the probabilistic step. In the state perturbation, ρ is a normal distribution with mean $x_{\text{det}}(t + \Delta t)$ and a standard deviation based on a local error estimator (see Section 2.1). A random sample is shown for illustration (red dot). **(B)** Similar to (A), but for the step-size perturbation method [22] using a log-normal perturbation distribution. Instead of integrating from t_0 to $t_0 + \Delta t$, the ODE is integrated from t_0 to $t_0 + \zeta_t$, where ζ_t is randomly drawn from a log-normal distribution (top panel). The solution of this perturbed integration $x_{\text{det}}(t + \zeta_t)$ (green circle) is then used as the solution $x_{\text{prb}}(t + \Delta t)$ of the probabilistic step (red dot), making $x_{\text{prb}}(t + \Delta t)$ a random variable with a distribution $\rho(x_{\text{prb}}(t + \Delta t))$ (right panel), that has no general analytical form but is dependent on the ODE and the solver. Therefore, $\rho(x_{\text{prb}}(t + \Delta t))$ is shown as an empirical histogram and a kernel density estimate. **(C)** Simulations of an Izhikevich neuron with a deterministic (green dashed line) and a probabilistic forward Euler method using state perturbation (two samples: red and orange).

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

where $\boldsymbol{\nu}_t$ controls the magnitude of the perturbation. The perturbation is only efficient when $\boldsymbol{\nu}_t$ is of the right order: if chosen too small, the uncertainty will be underestimated; if chosen too large, it will render the solver output useless. Conrad et al. [18] suggested calibrating $\boldsymbol{\nu}_t$ to replicate the amount of error introduced by the numerical scheme. We chose $\boldsymbol{\nu}_t = \sigma\boldsymbol{\varepsilon}_t$ using the error estimator $\boldsymbol{\varepsilon}_t$ readily available in methods that were developed for step-size adaptation (see Appendix A), and a scalar perturbation parameter σ that can be adjusted to calibrate the perturbation. If not stated otherwise, we used σ=1. An example of this perturbation method is shown in Fig. 1A for a single integration step and in Fig. 1C for an Izhikevich neuron model.

A related approach to stochastically perturbing the numerical integration was proposed by Abdulle and Garegnani [22], where noise is added to the integration step-size (i.e. to the "input" of the solver, rather than the "output", cf. Fig. 1B). The numerical integration is performed using the perturbed step-size ζ_t, but the computed solution is treated as the solution for the original step-size Δt:

$$\mathbf{x}_{\text{prb}}(t + \Delta t) = \mathbf{x}_{\text{det}}(t + \zeta_t), \quad \zeta_t \sim \mathcal{P}, \tag{4}$$

where ζ_t is the i.i.d. perturbed step-size drawn from a distribution $\mathcal{P}$ and $\mathbf{x}_{\text{det}}(\bullet)$ is a deterministic integration scheme that approximates Eq. (2). For example, for the forward Euler method Eq. (4) would be computed as $\mathbf{x}_{\text{prb}}(t + \Delta t) = \mathbf{x}_{\text{det}}(t) + \zeta_t \cdot f(t, \mathbf{x}_{\text{det}}(t))$. Abdulle and Garegnani [22] defined three properties the i.i.d. random variables ζ_t should fulfill:

- $P(\zeta_t > 0) = 1$, where P is the probability,
- there exists Δt such that $\mathbb{E}[\zeta_t] = \Delta t$, and
- there exist $p \geq 0.5$ and $C > 0$ independent of t such that $\mathbb{E}[(\zeta_t - \Delta t)^2] = C \cdot \Delta t^{2p+1}$.

Based on these restrictions, they proposed, as an example, to use a log-normal distribution:

$$\zeta_t \sim \mathcal{LN}_t(m, s^2), \tag{5}$$

where the mean m and standard deviation s of the underlying normal distribution should be chosen such that $\mathbb{E}[\zeta_t] = \Delta t$ and $\mathbb{E}[(\zeta_t - \Delta t)^2] = C \cdot \Delta t^{2p+1}$ hold for some $C > 0$ and $p \geq 0.5$ independent of Δt. m and s can therefore be defined as:

$$\begin{aligned} m &= \ln(\Delta t^2/\phi), \\ s &= \sqrt{2\ln(\phi/\Delta t)}, \\ \phi &= \sqrt{\mathbb{E}^2[\zeta_t] + \text{Var}[\zeta_t]} = \sqrt{\Delta t^2 + C \cdot \Delta t^{2p+1}}. \end{aligned} \tag{6}$$

Using $p = O$, where O is the order of the method, ensures that the mean-squared convergence order of the method is not changed. We therefore used $p = O$ throughout. We further generalized the example provided by Abdulle and Garegnani [22] in which $C = 1$ to a parametrized distribution by setting $C = \sigma^2$, i.e. setting $\phi = \sqrt{\Delta t^2 + \sigma^2 \cdot \Delta t^{2O+1}}$. The introduction of the perturbation parameter σ allows to—similarly to the perturbation parameter used in the state-perturbation—adjust and calibrate the magnitude of perturbation. If not stated otherwise, we used σ=1. The perturbation is illustrated in Fig. 1B for the first order forward Euler scheme.

2.2 Choice of solvers

We used the perturbation methods described above to create probabilistic versions of the solvers listed in Table 1.

Table 1: **Summary of the ODE solvers used in this paper.**

Abbr.	O	O_{e}	**Method & Error estimate**
FE	1	2	*Forward Euler* with Heun's method for error estimation.
EE	1		*Exponential Euler.*
EEMP	2		*Exponential Euler Midpoint* [11].
RKBS	3	2	*Bogacki–Shampine*, an embedded Runge-Kutta method [26].
RKCK	4	5	*Cash–Karp method*, an embedded Runge-Kutta method [27].
RKDP	5	4	*Dormand–Prince*, an embedded Runge-Kutta method [28].

O and O_e are the orders of the solution and the error estimator, respectively.
See Appendix B for details.

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

The usage of fixed (f) and adaptive (a) step-sizes is indicated with subscripts, and the perturbation method is indicated using the superscripts—x for the state perturbation [18] and t for the step-size perturbation [22]—meaning that e.g. $\mathrm{FE}_{\mathrm{f}}^{\mathrm{x}}$ is referring to a forward Euler method using fixed step-sizes and the state perturbation. For the exponential integrators, we chose to only use the step-size perturbation because it preserves the important property of these solvers that the activation and inactivation variables can not leave the interval $[0, 1]$, and also because there are no established methods for local error estimation for these methods.

The second order exponential integrator EEMP was implemented based on the version by Börgers and Nectow [11] (Appendix B), which is a modification of the midpoint method by Oh and French [29]. Computation of Runge-Kutta steps and step-size adaptation were based on the respective `scipy` implementations [30]. To avoid computational overhead, we only computed the local error estimates when necessary, i.e. for adaptive step-sizes or the state perturbation.

2.3 Interpolation

The iterative solvers used in this study yield solutions for $\mathbf{x}(t)$ on either a fixed and equidistant grid of time points T or, in the case of adaptive step-size solvers, on a finite set of time points T automatically chosen by the solver. To interpolate these solutions for example for spike-time estimation (see Section 2.4), we used linear interpolation for FE, EE and EEMP between solutions of single steps. To interpolate the steps of the Runge-Kutta methods we utilized the "dense output" implemented in the respective `scipy` methods [30]. These "dense outputs" allow to evaluate the solution between two steps $\mathbf{x}(t_i)$ and $\mathbf{x}(t_{i+1})$ for any t with $t_i \leq t \leq t_{i+1}$ without any additional ODE evaluation. To not discard the effect of the state perturbation during interpolation, we defined the dense output $\hat{d}_{\mathrm{RK}}(t, t_i, t_{i+1})$ for a state perturbed Runge-Kutta step from time t_i to t_{i+1} as:

$$\hat{d}_{\mathrm{RK}}(t, t_i, t_{i+1}) = d_{\mathrm{RK}}(t, t_i, t_{i+1}) + \frac{t - t_i}{t_{i+1} - t_i} \boldsymbol{\xi}_{t_i}, \tag{7}$$

where $d_{\mathrm{RK}}(t, t_i, t_{i+1})$ is the dense output of the respective deterministic Runge-Kutta step and $\boldsymbol{\xi}_{t_i}$ is the perturbation noise that was added to this step to compute $\mathbf{x}(t_{i+1})$ (see Eq. (3). This is a simplified version of the continuous-time output proposed by Conrad et al. [18].

2.4 Spike-time estimation

To determine spike-times based on simulated voltage traces $v(t)$, we interpolated the ODE solutions for all steps where $v(t)$ started from below and ended above a certain threshold voltage v_{th}. For lineally interpolated solutions (Section 2.3) we computed spike-times as follows. For every step from a time t_i to t_{i+1} with $v(t_i) < v_{\mathrm{th}} \leq v(t_{i+1})$ we estimated the respective spike-time t_{spike} as:

$$t_{\mathrm{spike}} = \frac{v_{\mathrm{th}} - v(t_i)}{t_{i+1} - t_i}. \tag{8}$$

To estimate spike-times for Runge-Kutta methods with "dense-outputs", we utilized `scipy`'s "brentq" root finding algorithm to determine the time point t_{spike} when the threshold is reached, i.e. $|v(t_{\mathrm{spike}}) - v_{\mathrm{th}}| < \epsilon$, with $\epsilon = 1e{-}12$.

2.5 Common ODE models in computational neuroscience

In this study, we use probabilistic ODE solvers to analyze the effect of numerical uncertainty in the following neuroscience models:

- The Izhikevich neuron model with a wide range of dynamics,
- the Hodgkin-Huxley neuron model,
- and a small network of Hodgkin-Huxley neurons.

We picked these models to cover both single neuron models and models of neuronal networks.

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

2.5.1 Single Izhikevich neurons

The Izhikevich neuron (IN) model is a simplified non-linear single neuron model that has been used e.g. to build large-scale models of the brain [31] and to understand oscillatory phenomena in the cortex [32,33] and the olfactory bulb [34]. An attractive property of the IN is that a whole range of different response dynamics can be simulated (Fig. S1) depending on the setting of the parameters $\boldsymbol{\theta} = [a, b, c, d]$ [23]. The IN is described by the following pair of ODEs [32]:

$$\begin{aligned} \dot{v}(t, v, u) &= 0.04 \cdot v^2 + 5 \cdot v - u + I_{\text{Stim}}(t), \\ \dot{u}(t, v, u) &= a(b \cdot v - u), \end{aligned} \tag{9}$$

where v is the membrane potential, u is a recovery variable and I_{Stim} is a given input current. Whenever the threshold is reached, i.e. $v(t) \geq 30$, a "spike" is triggered and the neuron is reset in the next time step of the simulation:

$$\begin{aligned} v(t + \Delta t_{\text{spike}}) &= c, \\ u(t + \Delta t_{\text{spike}}) &= u(t) + d, \end{aligned} \tag{10}$$

where $\Delta t_{\text{spike}} \geq 0$. Typically $\Delta t_{\text{spike}} = \Delta t$ is used, but to facilitate the comparison between different step-sizes we used $\Delta t_{\text{spike}} = 0$ instead. The reset is problematic, because it introduces an error of order $O(\Delta t)$ [10], independent of the solver scheme. This is because spikes can only occur after a full step of integration and the value $u(t + \Delta t_{\text{spike}})$ in Eq. (10) is dependent on the previous value $u(t)$.

To address this problem, we implemented two complementary strategies. Fist, we adapted Eq. (9) such that whenever $\dot{v}$ and $\dot{u}$ would have been evaluated for $v(t) \geq 30$—which can only happen for multi-stage methods—the derivatives were evaluated for $v(t) = 30$ instead. Second, we implemented the strategy suggested by Stewart et al. [10]: Every step resulting in a reset is split into two intermediate steps, a step until the threshold is reached, and a step after the reset. For this, the spike-time t_{spike} during such as step was estimated as described in Section 2.4 with a threshold of $v_{\text{th}} = 30$. Then, the pre-reset step solution $\mathbf{x}(t_{\text{spike}})$ was approximated based on the interpolation strategies described in Section 2.3. And finally, the post-reset step solution $\mathbf{x}(t_{t_i})$ was computed by resetting (see Eq. (10)) and integrating $\mathbf{x}$ from t_{spike} to t_{t_i}.

2.5.2 Single Hodgkin-Huxley neurons

Hodgkin-Huxley (HH) models [24] are widely used to simulate single and multi-compartment neurons. We study both the classical HH neuron [24] and a single compartment HH-like neuron model [35] prominently used to study the stomatogastric ganglion (STG) [25]. Both models are described by ODEs including the membrane potential $v(t)$ described by:

$$\dot{v}(t) = \left(I_{\text{Stim}}(t) - \textstyle\sum_i I_i(\mathbf{x})\right)/C, \tag{11}$$

where C is the membrane capacitance, I_{Stim} is the stimulation current and I_i are membrane currents. These membrane currents are described by the following equation:

$$I_i(\mathbf{x}) = \bar{g}_i \cdot m_i(\mathbf{x})^{p_i} \cdot h_i(\mathbf{x}) \cdot (v - E_i), \tag{12}$$

where E_i is the reversal potential of the current, $\bar{g}_i$ is the maximum channel conductance, p_i are integer exponents, and m_i and h_i are activation and inactivation functions. m_i and h_i were modeled by the following differential equations:

$$\dot{m}(v) = (m_\infty(v) - m)/\tau_m(v), \quad \dot{h}(v) = (h_\infty(v) - h)/\tau_h(v), \tag{13}$$

where m_∞, τ_m, h_∞, and τ_h are voltage dependent functions defining the channel's kinetics. For non-inactivating channels, h_i is removed from Eq. (12). In the classical HH model, this amounts to a 4-dimensional ODE [36]. For the STG neuron, which has eight instead of two membrane currents and also implements a model for the intracellular calcium concentration, the ODE is 13-dimensional [35]. The respective parametrizations can be found in Appendix C.

We simulated the HH neuron's response to two different input current I_{Stim}, a step and a noisy step stimulus. Both stimuli were 200 ms long, with $I_{\text{Stim}}(t) = 0$ for $t < t_{\text{onset}}$ and $t \geq t_{\text{offset}}$, where $t_{\text{onset}} = 10\,\text{ms}$ and $t_{\text{offset}} = 190\,\text{ms}$. The amplitude of the step stimulus for $t_{\text{onset}} \geq t < t_{\text{offset}}$ was $I_{\text{Stim}}(t) = 0.2\,\text{mA}$. The amplitude of the noisy step stimulus were created by drawing 99 values from a uniform distribution between 0.0 mA and 0.4 mA that were spaced equidistantly between t_{onset} and t_{offset}. These points were interpolated using a cubic spline with endpoints at t_{onset} and t_{offset}. At the endpoints both I_{Stim} and its derivative were set to zero. The single STG neuron was simulated for 3 s using a step stimulus starting at $t_{\text{onset}} = 0.9\,\text{s}$ with an amplitude of $I_{\text{Stim}}(t) = 3\,\text{nA}$.

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

2.5.3 STG model

The STG neuron model described above was used by Prinz et al. [25] in a network of three synaptically coupled neurons, ABPD, LP and PY, to study their firing patterns in dependence of the synaptic and neuronal parametrizations. In the model, there are seven synapses connecting the neurons, that are either modeled as slow or fast synapses. The postsynaptic input current I_i to a neuron is described by:

$$I_i(\mathbf{x}) = \bar{g}_i \cdot s_i(\mathbf{x}) \cdot (v - E_i), \tag{14}$$

where, similarly to Eq. (12), E_i is the reversal potential of the current, $\bar{g}_i$ is the synapse's maximum conductance, v is the membrane potential of the postsynaptic neuron and s is the activation function of the synapse. s is described by the following differential equation:

$$\begin{aligned} \dot{s} &= (\bar{s} - s)/\tau_s, \\ \bar{s} &= \left(1 + \exp((-35\,\mathrm{mV} - v_{\mathrm{pre}})/5\,\mathrm{mV})\right)^{-1}, \\ \tau_s &= (1 - \bar{s})/f_s, \end{aligned} \tag{15}$$

where v_{pre} is the membrane potential of the presynaptic neuron and τ_s and f_s are constants (see Appendix C).

2.6 Quantifying numerical uncertainty

2.6.1 Reference solutions

None of the aforementioned neuron models has an analytical solution. It is therefore not possible to compare simulations to the true solutions of the respective IVPs. As a substitute, we computed reference solutions using a deterministic $\mathrm{RKDP_a}$ solver with a tolerance of κ=1e−12 and a maximum step-size dependent on the model investigated (0.01 ms for IN and HH; 0.1 ms for the STG model). To obtain a reference solution at the same time points of a given fixed step-size solution $X = [\mathbf{x}(t_0), ..., \mathbf{x}(t_M)]$, we forced the reference solver to evaluate $\mathbf{x}(t)$ at least at all time points $T = [t_0, ..., t_M]$ of the given solution. For this, in every step in which the adaptive reference solver automatically picked a step-size that would skip any t_i in T by taking a too large step-size Δt_{i-1}, the step-size Δt_{i-1} was clipped such that the step was evaluated exactly at $\mathbf{x}(t_i)$. All solutions $\mathbf{x}(t)$ for t not in T were dropped before the comparison. To compare adaptive step-size solvers to reference solutions, we also forced these solvers to evaluate time points on a grid $T = [t_0, ..., t_M]$ with time points space equidistantly using a distance of 1 ms.

2.6.2 Distance metrics

To estimate the uncertainty for a given neuron model and solver, we computed multiple solutions (samples) with the same probabilistic solver to obtain a distribution of solutions. Based on these sample distributions and the respective reference solutions, we evaluated the distributions of sample-sample distances and sample-reference distances. For this, we computed the Mean Absolute Error (MAE) between single traces of the ODE solutions as a distance measure. If not stated otherwise, MAEs were computed on the simulated membrane potentials, because this is typically the quantity of interest. For two traces of equal size $\mathbf{a} = [a_0, ..., a_M]$ and $\mathbf{b} = [b_0, ..., b_M]$ the MAE was defined as:

$$\mathrm{MAE} = \frac{1}{M} \sum_{i=0}^{M} |a_i - b_i|. \tag{16}$$

For n samples from a probabilistic solver, we computed the sample-sample distance distribution $\mathrm{MAE_{SM}}$ as the n MAEs between single samples and the mean trace of the other $n - 1$ samples. Sample-reference distance distributions $\mathrm{MAE_{SR}}$ were computed as the n MAEs between single samples and the reference solution. In some cases, we also computed the distance between the solution of a corresponding deterministic solver to the reference solution, abbreviated as $\mathrm{MAE_{DR}}$.

2.7 Code and availability

The probabilistic solvers and models were implemented in Python and Cython. The code is available at `https://github.com/berenslab/neuroprobnum`.

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

3 Results

In this study, we explored the potential of probabilistic ODE solvers in computational neuroscience. First, we study the effect of numerical uncertainty on simulations of neuron models and qualitatively show that probabilistic solvers can reveal this uncertainty in a way that is easy to interpret. Second, we provide examples and guidelines were probabilistic solvers can be useful when conducting a new study. Third, we analyze potential drawbacks of probabilistic solvers, such as computational overhead.

3.1 Probabilistic solvers can reveal numerical uncertainty in neuron models

To demonstrate the effect of numerical uncertainty on simulations of single neuron models, we first simulated the classical HH neuron with the step stimulus (Fig. 2A). We computed solutions with a deterministic and probabilistic EE solver for a step-size of Δt=0.25 ms. Additionally, we computed a reference solution. We found that the exact spike-times of the deterministic EE solver differed substantially from the reference solution (spike-time difference $t_{\text{spike}}^{\text{det}} - t_{\text{spike}}^{\text{ref.}}$ of the first three spikes: 0.7 ms, 2.8 ms, 4.5 ms). The probabilistic solver revealed this numerical uncertainty with spike-times varying substantially between samples (standard deviation (SD) of the spike-time t_{spike} for the first three spikes over all 20 samples: 0.2 ms, 0.9 ms, 1.2 ms).

Next we simulated single INs with different parametrizations θ_i and response dynamics [23]. Using the original step-sizes Δt and input currents I_i, we compute solutions with the original solver scheme—which is related to a FE_f solver (Appendix B)—a deterministic FE scheme and a probabilistic $\text{FE}_\text{f}^\text{t}$ solver. We found, that for the "Inhibition-induced spiking" neuron all solvers produced similar spiking patterns in response to a negative current step (Fig. 2B). However, the original solver produced longer intervals between the spikes compared to the reference, resulting in only three instead of four spikes. The deterministic FE solution matched the reference better (e.g. both had four spikes), but the spike-times were still off by several milliseconds (spike-time difference $t_{\text{spike}}^{\text{det}} - t_{\text{spike}}^{\text{ref.}}$ of the last two spikes: 8.2 ms, 3.9 ms). The probabilistic solver revealed this numerical uncertainty (SD of the spike-time t_{spike} of the four spikes: 3.7 ms, 5.0 ms, 6.6 ms, 4.1 ms).

Similarly, for the "Inhibition-induced bursting" neuron the solution from the original solver and the deterministic FE solver were qualitatively broadly consistent with the reference solution (Fig. 2C). In all simulations, the neuron responded with spike bursts to a negative stimulus current step. The spike-times and the number of spikes of the original solution ($n_{\text{spikes}} = 11$) and the deterministic FE solution ($n_{\text{spikes}} = 14$) differed substantially from the reference ($n_{\text{spikes}} = 33$) though, with the FE solution having only two bursts instead of three during the simulated period. Here, the probabilistic solver revealed the substantial uncertainty in the spike-times and number of spikes ($\overline{n}_{\text{spikes}} = 14.4$ (SD 2.5), where $\overline{n}$ denotes the sample mean), with more than half of the samples having a third burst (Fig. 2C, bottom). All 16 simulated parametrizations are shown in Fig. S1.

To provide an example of a neuronal network, we simulated the STG model for two parametrizations (Figs. 2D and 2E) that only differ in their synaptic conductances (see Section 2.5.3). We computed solutions with a reference solver, a deterministic and a probabilistic EE solver. We focused the analysis on the LP neuron for simplicity. For the first parametrization (Fig. 2D), the LP neuron showed continuous spiking in all simulations. Similar to the HH neuron, we found differences in the exact spike-times and number of spikes between the reference ($n_{\text{spikes}} = 17$) and the deterministic EE solution ($n_{\text{spikes}} = 13$). The uncertainty was again revealed by the probabilistic solver ($\overline{n}_{\text{spikes}} = 14.6$ (SD 1.3); Fig. 2D). The second parametrization resulted in a different spiking behavior of the LP neuron. Here, the neuron started to fire at a high frequency for a prolonged time after approximately two seconds. In the reference solution, the neuron continued to fire. In contrast, in the deterministic solution, the neuron stopped after about another two seconds to then start another burst shortly later. While this also happened in all generated samples from the probabilistic solvers, the sample distribution still indicated a high uncertainty about the duration of the firing periods (Fig. 2C). Simulations of all five synaptic parametrizations from the original paper [25] are shown in Fig. S2.

Finally, we turned to a single STG neuron and stimulated the response to a step stimulus (Fig. 3) based on the original publication [35]. Here, we compared the numerical uncertainty in two different state variables, namely the voltage $v(t)$ (Fig. 3A) and the intracellular calcium $\text{Ca}(t)$ (Fig. 3B). We found that the numerical uncertainty differed strongly between these state variables, and was much higher for $v(t)$ (Fig. 3). While this is expected because of the transient and brief nature of spikes in contrast to the slower changing calcium, it highlights the power of probabilistic ODE solvers, as they can guide the choice of the solver and step-size parameter dependent on the quantity of interest and the desired accuracy without requiring detailed knowledge about the model and its kinetics.

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

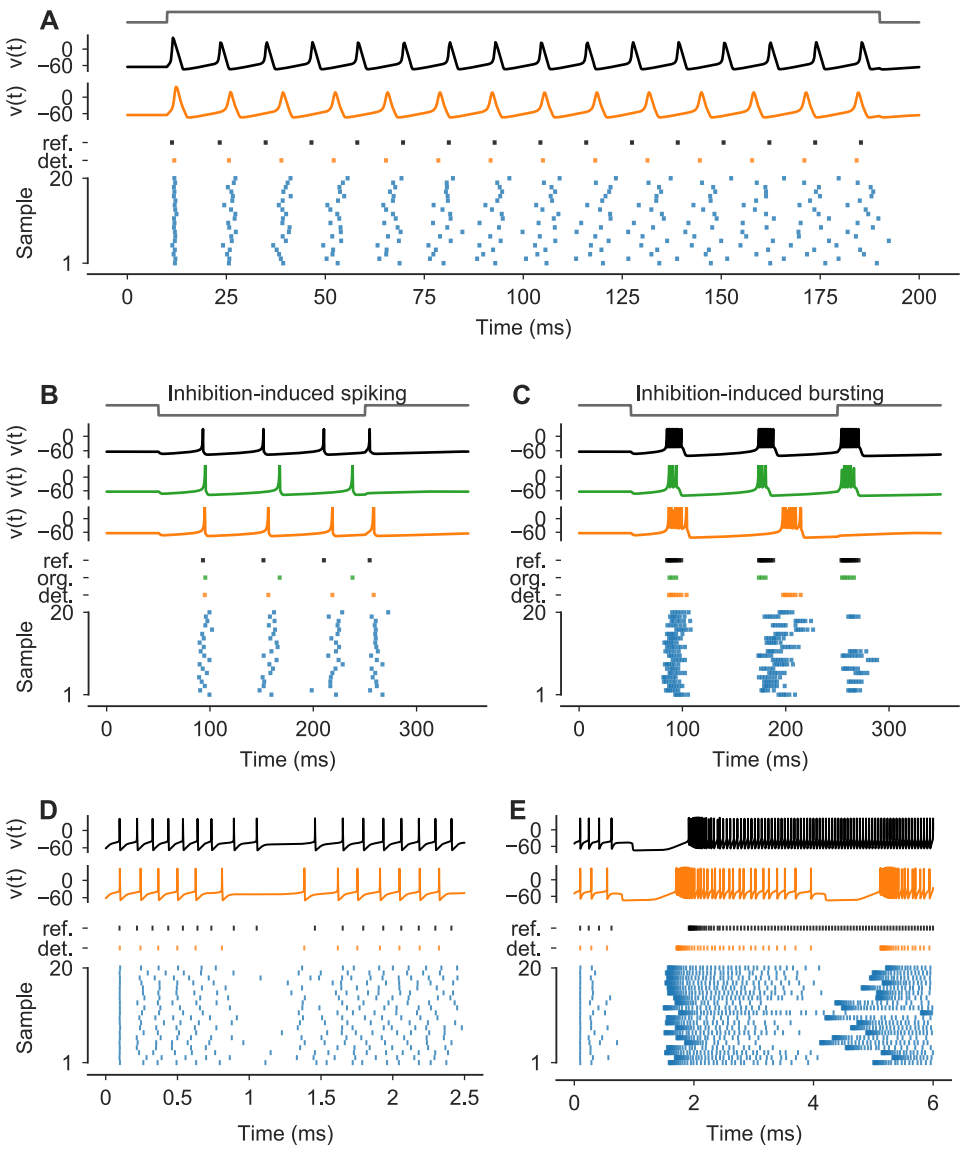

Figure 2: **Neuron simulations can be subject to substantial numerical uncertainty. (A)** Simulations of the classical HH model for the step stimulus I_{Stim} (normalized stimulus in gray). Solutions for $v(t)$ are shown for a reference solver (black) and a deterministic EE solver with Δt=0.25 ms (orange). *Bottom panel*: Spike-times of the reference (black), the deterministic EE solution (orange) and for 20 samples from a probabilistic $\text{EE}_{\text{f}}^{\text{t}}$ solver with Δt=0.25 ms (blue). **(B, C)** Simulations of the IN model for two different parametrizations θ_i (see Appendix C) and stimuli I_{Stim} (normalized stimuli in gray). Solutions for $v(t)$ are shown for a reference solver (black), the original solver scheme (green) and a deterministic FE solver (orange). Based on the original publication, the step-size Δt was set to 0.5 ms for all but the reference solver. For plotting, $v(t)$ were clipped at 30. *Bottom panels*: Spike-times are shown for the reference (black), the original solver solution (green), the deterministic FE solution (orange) and for 20 samples from a probabilistic $\text{FE}_{\text{f}}^{\text{t}}$ solver (blue). Samples were sorted by the number of spikes. **(D, E)** Simulations of the STG model for the two different synaptic parametrizations (see Appendix C) $\boldsymbol{\theta}_{\text{syn}}^{\text{b}}$ and $\boldsymbol{\theta}_{\text{syn}}^{\text{d}}$, respectively. Solutions for the membrane potential $v(t)$ of the LP neuron are shown for a reference solver (black) and a deterministic EE solver with Δt=0.1 ms (orange). *Bottom panels*: Spike-times of the LP neuron are shown for the reference (black), the deterministic EE solution (orange) and for 20 samples from a probabilistic $\text{EE}_{\text{f}}^{\text{t}}$ solver with Δt=0.1 ms (blue). Samples were sorted by the number of spikes.

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

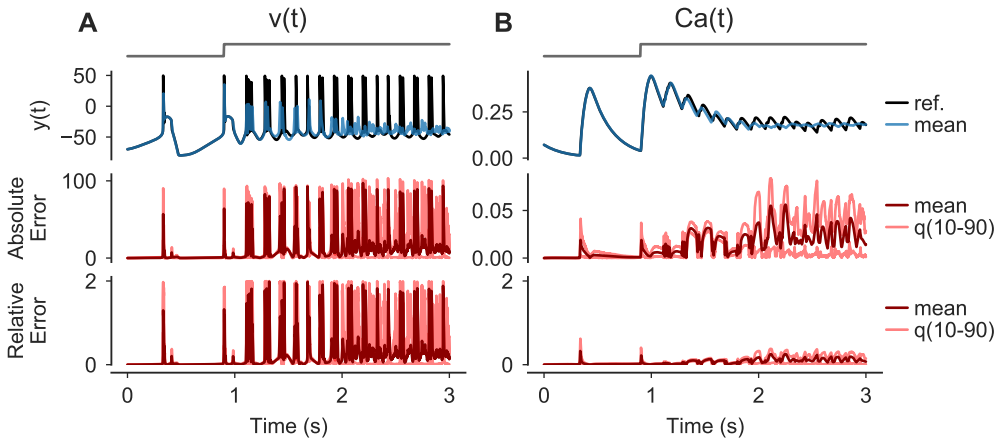

Figure 3: **Numerical uncertainty can vary between state variables. (A, B)** Simulations of a single STG neuron in response to a step-stimulus (gray, top row). Solutions were computed using a reference solver and by drawing 100 samples from a $\mathrm{EE}_1^{\natural}$ solver with Δt=0.1 ms. The reference solution (black) and the mean over the samples (blue) is shown for two state variables: the membrane potential $v(t)$ (A, second row) and the intracellular calcium $\mathrm{Ca}(t)$ (B, second row). For both state variables, the absolute error $\mathrm{AE}(t) = |y_{\mathrm{sample}}(t) - y_{\mathrm{ref.}}(t)|$ (third row) and the relative error $\mathrm{RE}(t) = |y_{\mathrm{sample}}(t) - y_{\mathrm{ref.}}(t)| / \max(|y_{\mathrm{sample}}(t)|, |y_{\mathrm{ref.}}(t)|)$ (bottom row) between sample and reference traces are shown as means and the 10th and 90th percentiles over all samples, respectively.

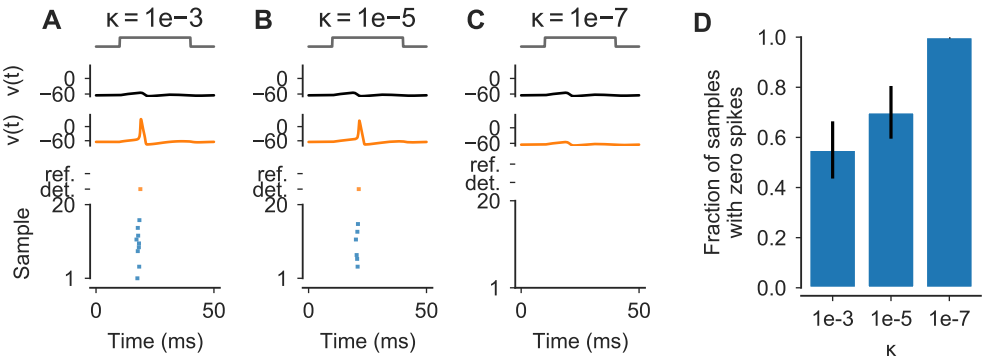

Figure 4: **Numerical uncertainty affects also higher order methods. (A-C)** Simulations of the classical HH model for the step stimulus I_{Stim} with an amplitude of 0.022 523 5 mA (normalized stimulus in gray). Solutions for $v(t)$ are shown for a reference solver (black) and a deterministic $\mathrm{RKDP_a}$ solver with κ=1e−3, κ=1e−5 and κ=1e−7, respectively (orange). *Bottom panels*: Spike-times of the reference (black), the deterministic solutions (orange) and for 20 samples from probabilistic $\mathrm{RKDP_a^x}$ solvers with κ=1e−3, κ=1e−5 and κ=1e−7, respectively (blue). **(D)** Fraction of samples (n = 20) from the probabilistic solvers in (A-C) that had no spike, shown as mean and standard error. All other samples had exactly one spike.

All the examples in Fig. 2 used first order methods. To also provide an example where higher order solvers with low tolerances yield solutions qualitatively different from the reference solution, we simulated the classical HH neuron's response for 50 ms to a step stimulus with an amplitude of 0.022 406 mA and t_{onset} = 10 ms and t_{offset} = 40 ms. This amplitude did not evoke a single spike in the reference solver (Fig. 4A), but was very close to the threshold, i.e. slightly larger amplitudes (e.g. 0.022 410 mA) did produce a spike for the reference solver. When simulating this model with a $\mathrm{RKDP_a}$ solver, we found that for tolerances of κ=1e−3 and κ=1e−5 the solutions did contain a spike (Figs. 4A and 4B). Only a tolerance as small as κ=1e−7 yielded a solution with no spike for this solver (Fig. 4C). Simulating the model with probabilistic solvers revealed this numerical uncertainty for both κ=1e−3 and κ=1e−5, with a fraction of samples containing one and a fraction containing zero spikes in both cases (Fig. 4D).

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

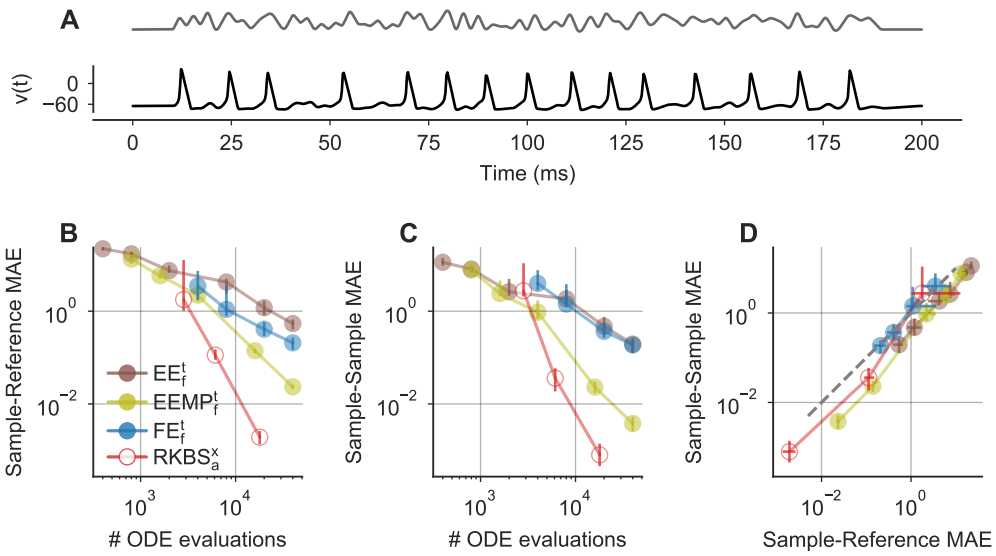

Figure 5: **Probabilistic solvers can be used to compare different solver schemes.** **(A)** Reference solutions of $v(t)$ (black) for the Hodgkin-Huxley model stimulated with the noisy step stimulus (normalized stimulus in gray). **(B)** Mean Absolute Errors $\mathrm{MAE}_{\mathrm{SR}}$ between sample traces of $v(t)$ and the respective reference solutions for different solver schemes (legend) and step-sizes / tolerances. Mean Absolute Errors from 100 samples are shown as medians (dots) and 10th to 90th percentiles (vertical lines) as a function of the number of ODE evaluations of a corresponding deterministic solver (x-axis). **(C)** As in (B), but for sample-sample Mean Absolute Errors $\mathrm{MAE}_{\mathrm{SM}}$. **(D)** As in (B), but for sample-sample Mean Absolute Errors $\mathrm{MAE}_{\mathrm{SM}}$ vs. sample-reference Mean Absolute Errors $\mathrm{MAE}_{\mathrm{SR}}$.

3.2 Probabilistic solvers can guide solver selection

To demonstrate how probabilistic ODE solvers can be used to compare the accuracy vs. run time tradeoff between different solver schemes, we simulated the HH neuron's response to the noisy step stimulus (Fig. 5A) using the following probabilistic solvers: $\mathrm{EE}_{\mathrm{f}}^{\mathrm{t}}$, $\mathrm{EEMP}_{\mathrm{f}}^{\mathrm{t}}$, $\mathrm{FE}_{\mathrm{f}}^{\mathrm{t}}$ and $\mathrm{RKBS}_{\mathrm{a}}^{\mathrm{x}}$. To this end, we computed the Mean Absolute Error between voltage trace samples $v(t)$ and the reference ($\mathrm{MAE}_{\mathrm{SR}}$) for each solver as an estimate of the numerical error induced. We compared these errors to the number of ODE evaluations a corresponding deterministic solver would need. We found that the exponential integrators EE and EEMP allowed computing solutions with the fewest ODE evaluations, as they terminated successfully even for the relatively large step-size Δt=0.5 ms (Fig. 5B). In contrast, when using the FE solver, all step-sizes $\Delta t \gg 0.05$ ms resulted in floating-point overflow errors and therefore in both useless and incomplete solutions. However, when choosing a sufficiently small step-size of $\Delta t \leq 0.05$ ms the samples obtained with the FE method had on average a smaller error compared to the EE method (Fig. 5B). From the methods tested, the adaptive RKBS method was the most efficient one, i.e. it produced the most accurate solutions for the fewest number of ODE solutions, but it also required a substantially higher number of minimum ODE evaluations to successfully terminate compared to the exponential integrators (Fig. 5B).

In principle, a very similar analysis could also have been done with deterministic solvers. However, probabilistic solvers have two advantages. First, they yield sample distributions instead of single solutions which make it possible to compute confidence intervals etc. when comparing different solver outputs. Second, and more crucially, probabilistic solvers do not require a reference solution to estimate the numerical error in a solution. For a sufficiently calibrated probabilistic solver, the sample distribution, i.e. the solver's output, can be used to estimate the numerical error of the solver. In Fig. 5C we computed the sample-sample distances $\mathrm{MAE}_{\mathrm{SM}}$, which are independent of the reference, for the same samples used in Fig. 5B. We found that the mean sample-reference distances $\overline{\mathrm{MAE}}_{\mathrm{SR}}$ were highly similar to the respective mean sample-sample distances $\overline{\mathrm{MAE}}_{\mathrm{SM}}$ for all solvers (Figs. 5B–5D). Therefore, the solver comparison described above could have also been based on $\mathrm{MAE}_{\mathrm{SM}}$ instead of $\mathrm{MAE}_{\mathrm{SR}}$, and therefore would not have required a reference solution.

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

3.3 Calibration of probabilistic solvers

The mean sample-sample distance $\overline{\text{MAE}}_{\text{SM}}$ is only then a good approximation to the mean sample-reference distance $\overline{\text{MAE}}_{\text{SM}}$ (as for example in Fig. 5) when the probabilistic solver is well calibrated. Ideally, the magnitude of the perturbation is large enough to capture the numerical uncertainty of the underlying numerical integration, but it is not too large to severely reduce the accuracy of the integration scheme. To quantify the calibration of different solvers, we therefore defined two metrics, the ratio $R_N = \overline{\text{MAE}}_{\text{SM}}/\overline{\text{MAE}}_{\text{SR}}$ and the ratio $R_D = \text{MAE}_{\text{DR}}/\overline{\text{MAE}}_{\text{SR}}$, where MAE_{DR} is the distance between a corresponding deterministic solution and the reference. R_N is close to zero if the perturbation is too small (i.e. the sample-sample distance is much smaller than the sample-reference distance) and close to one if the perturbation is sufficiently large to not underestimate the numerical uncertainty (i.e. the sample-sample distance can be used as an approximate measure of the sample-reference distance). R_D is close to one if the perturbation is either too small to affect the model output (i.e. all samples are approximately equal to the deterministic solution) or if samples are on average approximately equally close to the reference than the deterministic solution. R_D converges to zero, if the perturbation is too large and the perturbation severely reduces the solver accuracy. Note that R_D can also take values larger than one, which happens when the perturbation increases the solver accuracy on average (e.g. see Fig. 2C). This happens for example when the deterministic solution is missing a spike but almost reaches the model's spike threshold, and the perturbation is strong enough to generate the missing spike in some samples.

For a well calibrated solver, R_N is close to one, such that the sample-reference distance $\overline{\text{MAE}}_{\text{SR}}$ can be estimated from sample-sample distance $\overline{\text{MAE}}_{\text{SM}}$ while R_D is close to or larger than one, such that the perturbation does not decrease the solver accuracy.

The magnitude of the perturbation can be adjusted with the perturbation parameter σ that we defined for both the state and step-size perturbation (see Section 2.1). To analyze how the parameter σ affects the calibration of the perturbation and to test for which σ the solvers are well calibrated, we simulated the classical HH neuron in response to the noisy step stimulus with probabilistic solvers for a range of perturbation parameters (Fig. 6). First, we used a probabilistic $\text{EE}_{\text{f}}^{\text{t}}$ solver and computed MAE_{SM}, MAE_{SR} and MAE_{DR} and the ratios of the distribution means R_N and R_D for five different σ ranging from 0.0625 to 16 (Figs. 6A–6C). We found that with increasing σ, the mean sample-sample distance $\overline{\text{MAE}}_{\text{SM}}$ converged to the mean sample-reference distance $\overline{\text{MAE}}_{\text{SR}}$, and for sufficiently large σ the mean sample-sample distance $\overline{\text{MAE}}_{\text{SM}}$ could therefore be used as an approximate measure of mean sample-reference distance $\overline{\text{MAE}}_{\text{SR}}$ (Figs. 6A and 6B). For example, for σ=0.25, the perturbation magnitude was too small and the solver was underestimating the numerical uncertainty: Here, the mean sample-sample distances was much smaller $\overline{\text{MAE}}_{\text{SM}}$ (0.33) than mean the sample-reference distance $\overline{\text{MAE}}_{\text{SR}}$ (4.35) (Fig. 6A) with all sample-reference distances distributed narrowly (MAE_{SR} 10th to 90th percentiles: $[4.14, 4.56]$) around the sample-deterministic distance MAE_{DR} (4.36), indicating that all samples were very close to the deterministic solution, despite the numerical error. When using $\sigma \geq 4$, the mean sample-reference distance was higher than the deterministic-reference distance (Figs. 6A and 6C), indicating a loss of solver accuracy caused by the perturbation (e.g. $R_D = 0.51$ for σ=8). The best calibration was achieved with $1 \leq \sigma \leq 4$, with distributions of MAE_{SM} close to MAE_{SR} (R_N: 0.51, 0.83 and 0.97 for σ=1, σ=2 and σ=4, respectively; Figs. 6A and 6B) and with the mean sample accuracy close to the accuracy of the deterministic solution (R_D: 1.11, 1.08 and 0.74 for σ=1, σ=2 and σ=4, respectively; Figs. 6A and 6C).

To provide an overview of the calibration for different solvers settings, we defined the clipped ratio product $R_S^c R_D^c = \min(R_N, 1) \cdot \min(R_D, 1)$, which is ideally close to one. We used clipped values, because in some cases R_S and R_D took values larger than one which makes their product more difficult to interpret (Fig. S3). $R_S^c R_D^c$ is close to zero for either an underestimation of the numerical uncertainty ($R_S^c \approx 0$) or for a too strong perturbation that renders the solver output useless ($R_D^c \approx 0$).

We computed $R_S^c R_D^c$ for different probabilistic solvers and step-sizes—including the $\text{EE}_{\text{f}}^{\text{t}}$ solver with Δt=0.025 ms used above—for the HH neuron stimulated with the step and noisy step stimulus (Fig. 6D). The respective values for R_N and R_D are shown in Fig. S3. We found that the default perturbation parameter σ produced reasonably calibrated solutions in all cases. However, in most cases σ=1 was also not ideal. For $\text{EE}_{\text{f}}^{\text{t}}$, larger values (e.g. σ=2 or σ=4) resulted in better calibration, whereas for $\text{FE}_{\text{f}}^{\text{t}}$ the calibration was improved using smaller values (e.g. σ=0.5 or σ=0.25). The adaptive Runge-Kutta methods RKBS and RKDP were well calibrated for a wide range of perturbation parameters σ, including very small ones (e.g. σ=0.0625), especially in the high tolerance case (κ=$1e-2$). This is likely because even small perturbations cause the solvers to take different step-sizes and therefore to evaluate the ODE at different time points.

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

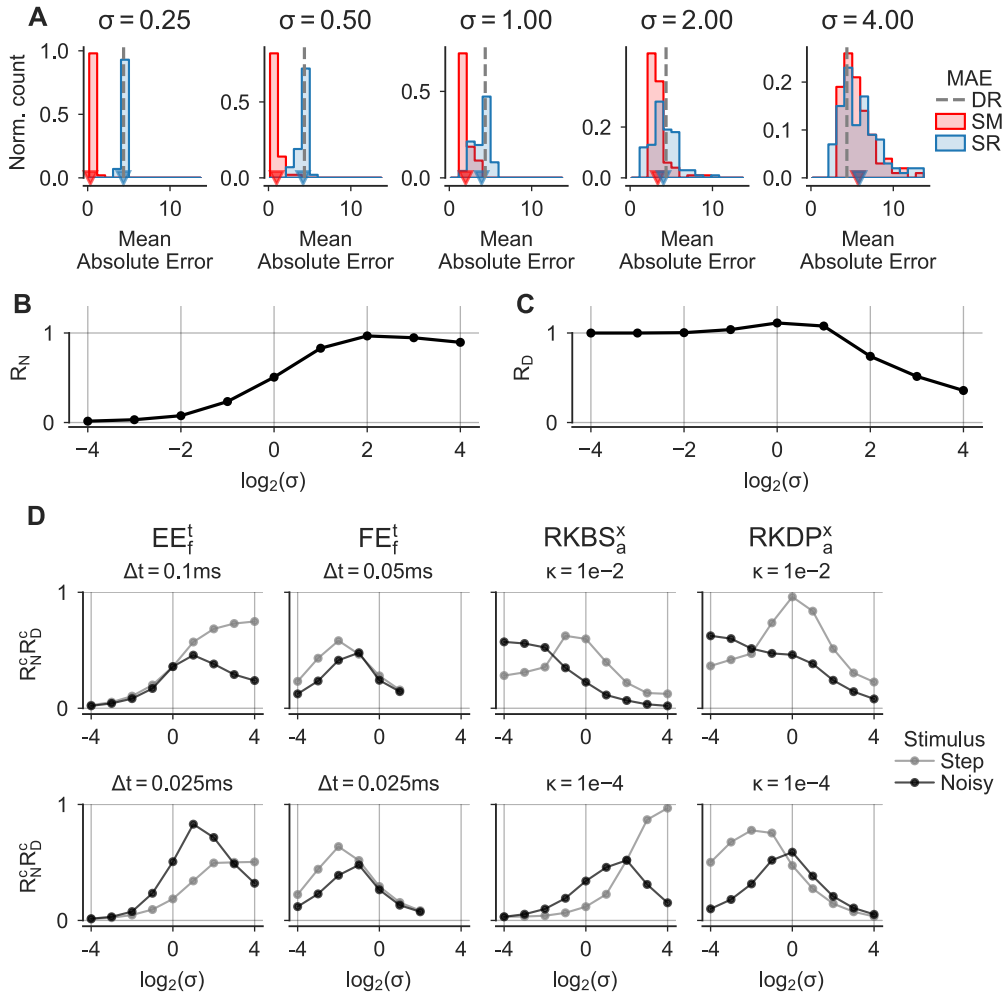

Figure 6: **Default calibration of probabilistic solvers is good but not optimal.** Simulations of the Hodgkin-Huxley model with different probabilistic solvers and perturbation parameters σ ranging from 0.0625 to 16. **(A)** Distributions of the Mean Absolute Error between voltage traces $v(t)$ computed with a probabilistic $\mathrm{EE}_\mathrm{f}^\mathrm{t}$ solver with Δt=0.025 ms, a deterministic version of this solver, and a reference solver for five perturbation parameters (titles). Solutions were computed for the noisy step stimulus. In each panel: Mean Absolute Error distributions were computed between 100 samples and the reference as MAE_SR (blue histograms), between the samples and the sample mean MAE_SM (red histograms) and between the deterministic and reference solution as MAE_DR (dashed grey line). Means of the distributions are highlighted (triangles). **(B)** Ratios of the Mean Absolute Error distribution means $R_N = \overline{\mathrm{MAE}}_\mathrm{SM}/\overline{\mathrm{MAE}}_\mathrm{SR}$ as a function of the perturbation parameter. **(C)** As in (B), but for ratio $R_D = \mathrm{MAE}_\mathrm{DR}/\overline{\mathrm{MAE}}_\mathrm{SR}$. **(D)** Clipped ratio products $R_S^c R_D^c = \min(R_N, 1) \cdot \min(R_D, 1)$ for different solvers (column titles) and step-sizes Δt / tolerances κ (panel titles) for the step (grey) and the noisy step (black) stimulus as a function of the perturbation parameter σ (x-axis).

3.4 Computational overhead

Probabilistic solvers based on state or step-size perturbation increase the computational costs for two reasons. First, they are sampling based and require computing multiple solutions for a single IVP. While this process can be parallelized, it may nevertheless come with a computational overhead, especially if it conflicts with other computations using parallelized model evaluation, e.g. in simulation based inference where the same model is evaluated for different model parameters [37, 38]. Second, probabilistic solvers induce a computational overhead per solution computed relative to their deterministic counterparts. We analyzed both aspects in the following.

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

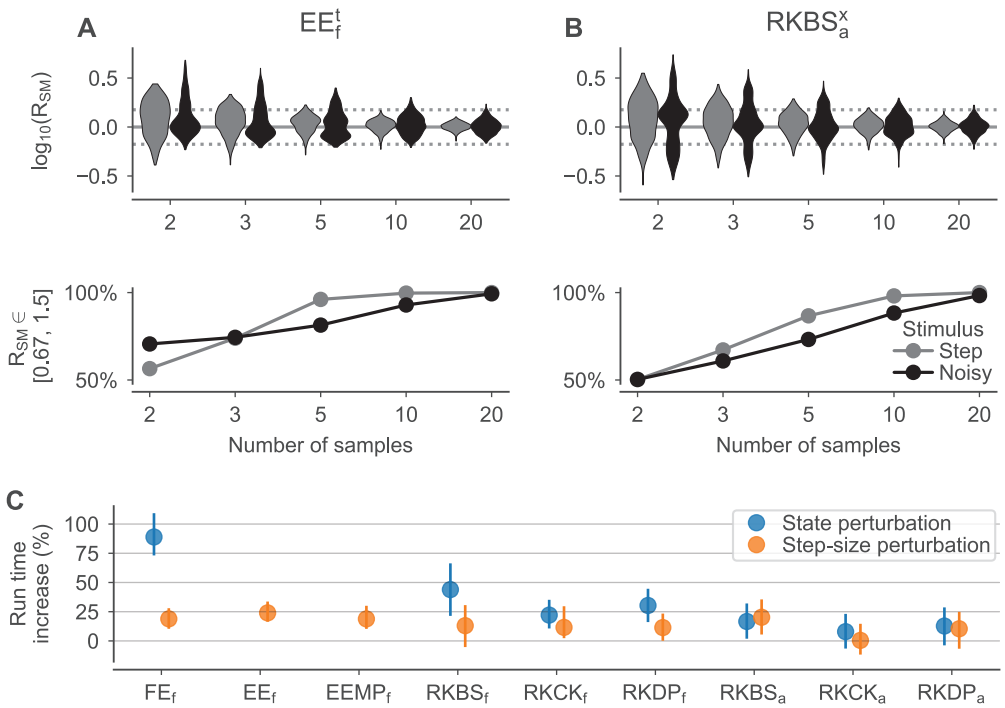

Figure 7: **The computational overhead of probabilistic solver is moderate. (A,B)** Bootstrapped distributions of normalized sample-sample distances $R_{\mathrm{SM}} = \overline{\mathrm{MAE}}_{\text{SM-n}}/\overline{\mathrm{MAE}}_{\text{SM-300}}$ as a function of the numbers of samples n. $R_{\text{SM-n}}$ were computed for the classical HH neuron simulated with a $\mathrm{EE}_\mathrm{f}^\mathrm{t}$ with Δt=0.1 ms (A) and a $\mathrm{RKBS}_\mathrm{a}^\mathrm{x}$ solver with κ=1e−2 (B) for the step (grey) and noisy step (black) stimulus, respectively. *Top:* Distributions over $\log_{10}(R_{\mathrm{SM}})$ computed by 1000 times repeated random sub-sampling of $\overline{\mathrm{MAE}}_{\text{SM-n}}$ from all 300 generated samples. Vertical lines highlight R_{SM} values of 0.67, 1 and 1.5. *Bottom:* Percentages of the bootstrapped R_{SM} distributions in the interval $[0.67, 1.5]$. **(C)** Relative run times for different solver schemes measured for the classical HH neuron with the noisy step stimulus. For every solver, 100 samples were simulated for both a probabilistic and a corresponding deterministic solver. Relative run times were computed by dividing the run times of the probabilistic samples by the run times of the respective deterministic samples. Distributions were computed by bootstrapping 10000 ratios and are shown as medians and the 10th to 90th percentiles. The step-size of fixed step-size methods was $\Delta t = 0.05$ ms and the tolerance of adaptive methods was κ=1e−4. For EE and EEMP, only the step-size perturbation was used.

3.4.1 Required number of samples

To empirically determine the number of samples necessary to obtain a reliable measure of numerical uncertainty, we simulated the classical HH neuron with probabilistic solvers for the step and noisy step stimulus. To this end, we computed mean sample-sample distances $\overline{\mathrm{MAE}}_{\text{SM-n}}$ for small numbers of samples n, and divided them by the mean sample-sample distances for a much larger number of samples (300) to obtain normalized sample-sample distances $R_{\mathrm{SM}}(n) = \overline{\mathrm{MAE}}_{\text{SM-n}}/\overline{\mathrm{MAE}}_{\text{SM-300}}$ (Figs. 7A and 7B). We found that often already two samples were sufficient to get a good estimate of the sample-sample distance. e.g. for the step stimulus and $n = 2$, more than half of the R_{SM} were in $[0.67, 1.5]$ with little difference between the solvers $\mathrm{EE}_\mathrm{f}^\mathrm{t}$ (Fig. 7A) and $\mathrm{RKBS}_\mathrm{a}^\mathrm{x}$ (Fig. 7B).

3.4.2 Overhead per sample

In addition to in the computational overhead caused by the computation of multiple samples, probabilistic methods also come with a computational overhead per solution. For the state perturbation [18] this overhead has three components. First, one needs to compute the local error estimator, which only causes overhead for fixed step-sizes since for adaptive methods the local error estimator needs to be computed anyway. The second potential source of overhead is that the "First Same As Last" property—i.e. that the last stage in one step can be used as the first stage of the next step, which is used in RKBS and RKDP—is not applicable. This is because the last stage is computed before the perturbation, and after the perturbation the evaluation of the ODE is not valid anymore.

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

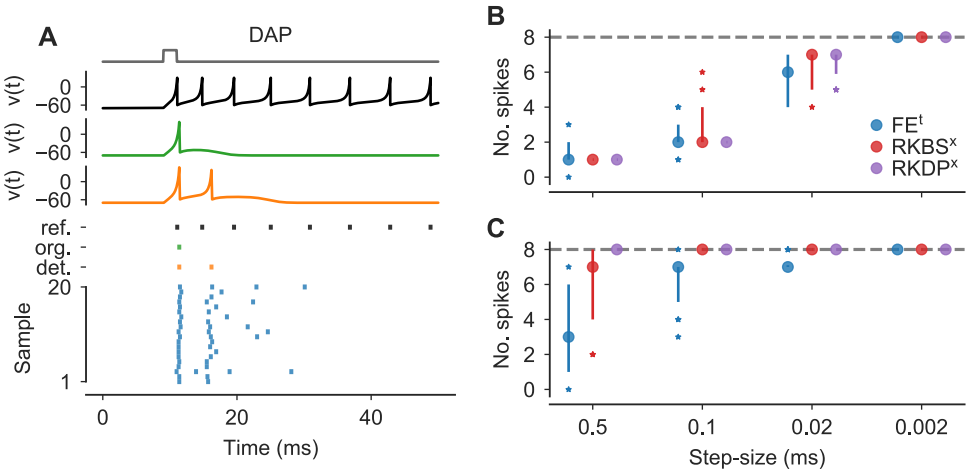

Figure 8: **Probabilistic solver only account for errors in numerical integration. (A)** Simulations of the "DAP" IN model with a pulse stimulus I_{Stim} (normalized stimuli in gray). Solutions for $v(t)$ are shown for a reference solver (black), the original solver scheme (green) and a deterministic FE solver (orange). Based on the original publication, the step-size Δt was set to 0.1 ms. For plotting, $v(t)$ were clipped at 30. *Bottom panel*: Spike-times are shown for the reference (black), the original solver solution (green), the deterministic FE solution (orange) and for 20 samples from a probabilistic $\text{FE}_\text{f}^\text{t}$ solver (blue). Samples were sorted by the number of spikes. **(B,C)** Number of spikes for the "DAP" IN model (see Fig. 2C) dependent on the step-size (x-axis) for different solver methods (legend) for fixed (**A**) and pseudo-fixed (**B**) step-sizes, respectively. Number of spikes are shown for 40 samples as medians, 10th to 90th percentiles (vertical lines) and outliers (stars). The dashed horizontal line refers to the reference solution (see Fig. 2C).

Lastly, the perturbation itself, which includes sampling form a Gaussian, needs to be computed.

In total, this overhead is relatively small for higher order methods optimized for step-size adaptation like RKBS, RKCK and RKDP. For example, the state perturbation for RKDP_a increases the number of ODE evaluations per step from six to seven (+16%) due to the loss of the First Same As Last property, and for RKCK_a—which does not make use of this property—no additional ODE evaluation is required. However, for first order methods like FE this overhead severely reduces the computational efficiency because instead of a single ODE evaluation per step, a state perturbed version needs two (+100%). Additionally, lower order methods typically require more steps in total compared to higher order methods, because they are typically used in combination with smaller step-sizes. This increases the total computational costs of the perturbation itself, which is done once per step. For the step-size perturbation, the overhead is reduced to the perturbation and, for adaptive step-size methods, the loss of the First Same As Last property.

To quantify this overhead empirically, we simulated the HH neuron with different probabilistic solvers and their deterministic counterparts and compared the run times relative to each other. As expected, for the state perturbation, the computational overhead was larger for the lower order methods (Fig. 7C; on average for $\text{FE}_\text{f}^\text{x}$: +91%, $\text{RKBS}_\text{f}^\text{x}$: +44%, $\text{RKCK}_\text{f}^\text{x}$: +22%, $\text{RKDP}_\text{f}^\text{x}$: +30%). The adaptive methods—where the local error estimates were computed not only for the probabilistic, but also for the deterministic methods—showed the smallest increase in run times (+12% on average across all adaptive methods), with $\text{RKCK}_\text{a}^\text{x}$, not using the First Same As Last property, having the least overhead (+8%). For the step-size perturbation, the increase in run times was on average smaller (+15% on average across all methods) and without large differences between the solver schemes and the usage of adaptive or fixed step-sizes.

3.5 Limitations

Finally, we turned back to the "DAP" IN model, to illustrate limitations of the approach. For this neuron model we had found a large difference in the number of spikes for the fixed step-size methods, like the original solver, compared to the reference solution (Fig. 8A). While the reference solution had eight spikes during the simulated period, the original solution had only one and a deterministic FE solver had two. While the probabilistic solver $\text{FE}_\text{f}^\text{t}$ arguably indicated some numerical uncertainty ($\overline{n}_{\text{spikes}} = 2.4$ (SD 0.7)), the number of spikes was still much lower compared to the reference. To better understand the source of this numerical uncertainty, we simulated the

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

"DAP" neuron model with different probabilistic solvers, $\mathrm{FE}_{\mathrm{f}}^{\mathrm{t}}$, $\mathrm{RKBS}_{\mathrm{f}}^{\mathrm{x}}$ and $\mathrm{RKDP}_{\mathrm{f}}^{\mathrm{x}}$.

First, we simulated the neuron for different fixed step-sizes. We found that all probabilistic solvers underestimated the true number of spikes when using relatively large fixed step-sizes (Fig. 8B). For the largest step-size tested, Δt=0.5 ms, only the $\mathrm{FE}_{\mathrm{f}}^{\mathrm{t}}$ solver indicated uncertainty in the number of spikes, whereas for $\mathrm{RKBS}_{\mathrm{f}}^{\mathrm{x}}$ and $\mathrm{RKDP}_{\mathrm{f}}^{\mathrm{x}}$ all samples had only a single spike. When using smaller step-sizes, the probabilistic solvers' outputs were more indicative of the numerical uncertainty. For Δt=0.02 ms, all solvers produced outputs that were closer ($\overline{n}_{\mathrm{spikes}} = 5.9$ for FE, $\overline{n}_{\mathrm{spikes}} = 6.5$ for RKBS and $\overline{n}_{\mathrm{spikes}} = 6.7$ for RKDP) to the reference solution and all methods indicated uncertainty in the number of spikes. With the very small step-size Δt=0.002 ms, all samples from all solvers showed the same number of spikes as the reference and the probabilistic solvers indicated no remaining uncertainty about the number of spikes here.

While these results may be unsatisfactory at first glance, they are not necessarily unexpected. The probabilistic solvers used here can only capture the uncertainty arising through the numerical integration; they can not capture the error that is introduced by restricting spikes to occur only on a fixed time grid, which is the case for the fixed step-size solvers. We therefore simulated the neuron for the same solvers and step-sizes again, but allowed the solver to take intermediate steps (see Eq. (10)) every time a reset occurred. When using these "pseudo-fixed" step-sizes, we found that $\mathrm{RKDP}_{\mathrm{f}}^{\mathrm{x}}$ still did not indicate uncertainty in the number of spikes for any step-size tested, but now all samples had the same number of spikes as the reference (Fig. 8C). And while $\mathrm{FE}_{\mathrm{f}}^{\mathrm{t}}$ and $\mathrm{RKBS}_{\mathrm{f}}^{\mathrm{x}}$ still underestimated the number of spikes for larger step-sizes on average (e.g. for Δt=0.5 ms: $\overline{n}_{\mathrm{spikes}} = 3.5$ for FE and $\overline{n}_{\mathrm{spikes}} = 6.3$ for RKBS), both indicated high numerical uncertainty (e.g. for Δt=0.5 ms: $q_{90}(n_{\mathrm{spikes}}) = 6$ for FE and $q_{90}(n_{\mathrm{spikes}}) = 8$ for RKBS, where q_{90} is the 90th percentile).

4 Discussion

The outcome of neuron simulations is affected by numerical uncertainty arising from the inevitably finite step-sizes used in numerical ODE integration. With standard solvers there is no straightforward way to quantify how this uncertainty affects quantities of interest such as spike-times and the number of spikes.

In this study, we demonstrated how probabilistic solvers can be used to quantify and reveal numerical uncertainty in commonly used neuron models. Crucially, these solvers can be easily implemented and do not require a detailed understanding of the underlying kinetics of the neuron model of interest.

Further, we showed that numerical uncertainty can affect the precise timing and sometimes even the number of the spikes in simulations of neuron models commonly used in neuroscience. We also found that some models and parametrizations are more susceptible to numerical uncertainty than others, and that some solvers employed in the neuroscience literature yield rather large uncertainties. These findings highlight the need for a thorough quantification of numerical uncertainty in neuroscience simulations to strike an informed balance between simulation time and tolerated uncertainty.

The idea to quantify the accuracy or numerical errors of different solvers for mechanistic models in neuroscience is not new. For example, Butera and McCarthy [39] showed that for small step-sizes, the forward Euler method produces more accurate solutions than the exponential Euler method, which is in agreement with our findings. Börgers and Nectow [11] on the other hand argued that for Hodgkin-Huxley-like systems exponential integrators—such as exponential Euler and the exponential midpoint Euler—are often the best choice, as they allow for much larger step-sizes especially when high accuracy is not necessary, which is again what we observed. Stewart and Bair [10] argued in favor of the Parker-Sochacki integration method and showed that it can be used to generate highly accurate solutions for both the Izhikevich and Hodgkin-Huxley model. However, this method has the disadvantage that the ODE system at hand has to be put into the proper form and therefore requires specific knowledge about the model and solver. In a more recent study, Chen et al. [12] recommended to use splitting methods, such as second-order Strang splitting, instead of exponential integrators.

In contrast to these studies, probabilistic solvers offer a more general approach to tackle the problem of numerical uncertainty. Instead of finding the "best" solver for a specific problem, they produce an easy-to-interpret uncertainty measure that can be analyzed without specific knowledge about the solver or solved neuron model. This allows to easily assess if a solver is sufficiently accurate for a given research question. It can therefore facilitate both the choice of the solver and choice of solver settings such as the step-size.

In this study, we used two simple probabilistic solvers that build on deterministic solver and stochastically perturb the numerical integration. For both, the state [18] and the step-size perturbation [22] method, it is crucial that the perturbation is of the right order to neither underestimate the numerical uncertainty nor to reduce the solver accuracy unnecessarily. To be able to adjust the perturbation we introduced a perturbation parameter for

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

both the state and step-size perturbation. We found that using the default value for this parameter yielded good calibration in most cases, but slight adjustments often improved the calibration further.

The state and step-size perturbation are conceptually quite similar, but the two methods have some clear advantages and disadvantages with respect to each other. The step-size perturbation requires a local error estimator to be calibrated. This is disadvantageous because it requires a method for local error estimation which can introduce a relatively large computational overhead per solution for lower order methods like FE. The step-size perturbation may therefore a better choice for lower order methods. However, for higher order methods like RKDP this difference vanishes and both approaches require an equally small computational overhead per solution. Another advantage of the step-size perturbation is that it preserves desirable properties of the underlying solver schemes [22]. For example, when Hodgkin-Huxley-like models are solved with exponential integrators like EE or EEMP, the state variables of the activation and inactivation can not leave their domain $[0, 1]$ by design of the solvers, a property preserved by the step-size but not the state perturbation. A downside of the step-size perturbation is that the calibration can be slightly more challenging because the perturbation is influenced by linear scaling of the simulated time, which happens for example if the time unit of the model is changed.

Beyond the two perturbation methods used and discussed in this study, there are probabilistic ODE solvers constructed using techniques from (nonlinear) Gaussian filtering and smoothing [15, 16, 40]. These methods have the advantage that instead of repeatedly integrating the initial value problem, they only require a single forward integration and return local uncertainty estimates that are proportional to the local truncation error. The disadvantage of Gaussian ODE filters and smoothers is that the uncertainty estimates are Gaussian. This restriction can be lifted by replacing Gaussian filters and smoothers with particle filters and smoothers [16]. In particular for large neural network simulations, such efficient methods will be key in quantifying uncertainty.

To further extent the applicability of probabilistic solvers in neuroscience, it will be crucial to also develop and test implicit probabilistic solvers for neuron models. For example, the ODEs of multi-compartment neuron models are typically stiff which makes implicit solvers the better choice for such models [41]. A priori, it is often not easy to judge whether a ODE system is stiff or not. A noteworthy attempt to tackle this problem is the algorithm by Blundell et al. [42] that automatically determines whether an implicit or an explicit solver should be used.

5 Acknowledgements

This research was funded by the Deutsche Forschungsgemeinschaft through a Heisenberg Professorship (BE5601/8-1, PB), the Excellence Cluster 2064 “Machine Learning — New Perspectives for Science” (ref number 390727645, PB and PH), ADIMEM (01IS18052C and 01IS18052B to PB and PH) and the Tübingen AI Center (FKZ: 01IS18039A, PB and PH). NK and PH gratefully acknowledge financial support by the German Federal Ministry of Education and Research (BMBF) through Project ADIMEM (FKZ 01IS18052B), as well as by the European Research Council through ERC StG Action 757275 / PANAMA, and funds from the Ministry of Science, Research and Arts of the State of Baden-Württemberg. The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Nicholas Krämer.

6 Competing interests

The authors have no competing interest to declare.

References

[1] Jonathan W Pillow, Jonathon Shlens, Liam Paninski, Alexander Sher, Alan M Litke, EJ Chichilnisky, and Eero P Simoncelli. Spatio-temporal correlations and visual signalling in a complete neuronal population. *Nature*, 454(7207):995–999, 2008.

[2] Sebastian Gerwinn, Matthias Bethge, Jakob H Macke, and Matthias Seeger. Bayesian inference for spiking neuron models with a sparsity prior. In *Advances in Neural Information Processing Systems*, pages 529–536, 2008.

[3] Wulfram Gerstner and Werner M Kistler. *Spiking neuron models: Single neurons, populations, plasticity.* Cambridge University Press, 2002.

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

[4] Eugene M Izhikevich. *Dynamical systems in neuroscience*. MIT Press, 2007.

[5] Pedro J Gonçalves, Jan-Matthis Lueckmann, Michael Deistler, Marcel Nonnenmacher, Kaan Öcal, Giacomo Bassetto, Chaitanya Chintaluri, William F Podlaski, Sara A Haddad, Tim P Vogels, et al. Training deep neural density estimators to identify mechanistic models of neural dynamics. *bioRxiv doi: 10.1101/838383*, 2019.

[6] Jonathan Oesterle, Christian Behrens, Cornelius Schroeder, Thoralf Herrmann, Thomas Euler, Katrin Franke, Robert G Smith, Guenther Zeck, and Philipp Berens. Bayesian inference for biophysical neuron models enables stimulus optimization for retinal neuroprosthetics. *bioRxiv doi: 10.1101/2020.01.08.898759*, 2020.

[7] George Papamakarios, David C Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. *arXiv:1805.07226*, 2018.

[8] Philipp Hennig, Michael A Osborne, and Mark Girolami. Probabilistic numerics and uncertainty in computations. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 471(2179), 2015.

[9] Ernst Hairer, Syvert P Nørsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I – Nonstiff Problems*. Springer, 1993.

[10] Robert D Stewart and Wyeth Bair. Spiking neural network simulation: numerical integration with the parker-sochacki method. *Journal of Computational Neuroscience*, 27(1):115–133, 2009.

[11] Christoph Borgers and Alexander R Nectow. Exponential time differencing for hodgkin–huxley-like odes. *SIAM Journal on Scientific Computing*, 35(3):B623–B643, 2013.

[12] Zhengdao Chen, Baranidharan Raman, and Ari Stern. Structure-preserving numerical integrators for hodgkin–huxley-type systems. *SIAM Journal on Scientific Computing*, 42(1):B273–B298, 2020.

[13] Chris J Oates and Tim J Sullivan. A modern retrospective on probabilistic numerics. *Statistics and Computing*, 29:1335–1351, 2019.

[14] John Cockayne, Chris J Oates, Tim J Sullivan, and Mark Girolami. Bayesian probabilistic numerical methods. *SIAM Review*, 64:756–789, 2019.

[15] Michael Schober, Simo Särkkä, and Philipp Hennig. A probabilistic model for the numerical solution of initial value problems. *Statistics and Computing*, 29:99–122, 2019.

[16] Filip Tronarp, Hans Kersting, Simo Särkkä, and Philipp Hennig. Probabilistic solutions to ordinary differential equations as nonlinear Bayesian filtering: a new perspective. *Statistics and Computing*, 29:1297–1315, 2019.

[17] Nicholas Krämer, Nathanael Bosch, Jonathan Schmidt, and Philipp Hennig. Probabilistic ODE solutions in millions of dimensions. *arXiv preprint arXiv:2110.11812*, 2021.

[18] Patrick R Conrad, Mark Girolami, Simo Särkkä, Andrew Stuart, and Konstantinos Zygalakis. Statistical analysis of differential equations: introducing probability measures on numerical solutions. *Statistics and Computing*, 27:1065–1082, 2017.

[19] Oksana A Chkrebtii, David A Campbell, Ben Calderhead, and Mark A Girolami. Bayesian solution uncertainty quantification for differential equations. *Bayesian Analysis*, 11:1239–1267, 2016.

[20] Onur Teymur, Konstantinos Zygalakis, and Ben Calderhead. Probabilistic linear multistep methods. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, pages 4314–4321. Curran Associates, Inc., 2016.

[21] Onur Teymur, Han Cheng Lie, Tim Sullivan, and Ben Calderhead. Implicit probabilistic integrators for odes. In *Advances in Neural Information Processing Systems*, pages 7244–7253, 2018.

[22] Assyr Abdulle and Giacomo Garegnani. Random time step probabilistic methods for uncertainty quantification in chaotic and geometric numerical integration. *Statistics and Computing*, 2020.

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

[23] Eugene M Izhikevich. Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks*, 15(5):1063–1070, 2004.

[24] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500–544, 1952.

[25] Astrid A Prinz, Dirk Bucher, and Eve Marder. Similar network activity from disparate circuit parameters. *Nature Neuroscience*, 7(12):1345–1352, 2004.

[26] Przemyslaw Bogacki and Lawrence F Shampine. A 3 (2) pair of Runge-Kutta formulas. *Applied Mathematics Letters*, 2(4):321–325, 1989.

[27] Jeff R Cash and Alan H Karp. A variable order runge-kutta method for initial value problems with rapidly varying right-hand sides. *ACM Transactions on Mathematical Software (TOMS)*, 16(3):201–222, 1990.

[28] John R Dormand and Peter J Prince. A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980.

[29] Jiyeon Oh and Donald A French. Error analysis of a specialized numerical method for mathematical models from neuroscience. *Applied mathematics and computation*, 172(1):491–507, 2006.

[30] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[31] Eugene M Izhikevich and Gerald M Edelman. Large-scale model of mammalian thalamocortical systems. *Proceedings of the National Academy of Sciences*, 105(9):3593–3598, 2008.

[32] Eugene M Izhikevich. Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 14(6):1569–1572, 2003.

[33] Justin WM Domhof and Paul HE Tiesinga. Balance between inhibitory cell types is necessary for flexible frequency switching in adult mouse visual cortex. *bioRxiv doi: 10.1101/2020.01.18.911271*, 2020.

[34] Roberto F Galán, Nicolas Fourcaud-Trocmé, G Bard Ermentrout, and Nathaniel N Urban. Correlation-induced synchronization of oscillations in olfactory bulb neurons. *Journal of Neuroscience*, 26(14):3646–3655, 2006.

[35] Astrid A Prinz, Cyrus P Billimoria, and Eve Marder. Alternative to hand-tuning conductance-based models: construction and analysis of databases of model neurons. *Journal of Neurophysiology*, 90(6):3998–4015, 2003.

[36] G Bard Ermentrout and David H Terman. The hodgkin–huxley equations. In *Mathematical foundations of neuroscience*, pages 1–28. Springer, 2010.

[37] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020.

[38] Pedro J Gonçalves, Jan-Matthis Lueckmann, Michael Deistler, Marcel Nonnenmacher, Kaan Öcal, Giacomo Bassetto, Chaitanya Chintaluri, William F Podlaski, Sara A Haddad, Tim P Vogels, et al. Training deep neural density estimators to identify mechanistic models of neural dynamics. *Elife*, 9:e56261, 2020.

[39] Robert J Butera and Maeve L McCarthy. Analysis of real-time numerical integration methods applied to dynamic clamp experiments. *Journal of Neural Engineering*, 1(4):187, 2004.

[40] Hans Kersting, T J Sullivan, and Philipp Hennig. Convergence rates of Gaussian ODE filters. *arXiv:1807.09737v2*, 2019.

[41] Michael V Mascagni, Arthur S Sherman, et al. Numerical methods for neuronal modeling. *Methods in neuronal modeling*, 2, 1989.

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

[42] Inga Blundell, Dimitri Plotnikov, Jochen M Eppler, and Abigail Morrison. Automatically selecting a suitable integration scheme for systems of differential equations in neuron models. *Frontiers in neuroinformatics*, 12:50, 2018.

[43] Peter Dayan and Laurence F Abbott. *Theoretical neuroscience: computational and mathematical modeling of neural systems*. Computational Neuroscience Series, 2001.

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

Appendix

A Local error estimation and step-size adaptation

To compute the local error estimator $\boldsymbol{\varepsilon}_t$ for a single integration step, step solutions were computed with two different numerical methods that were run in parallel to provide two solutions $\mathbf{x}_a(t+\Delta t)$ and $\mathbf{x}_b(t+\Delta t)$ for every step, given t, Δt and $\mathbf{x}(t)$ (e.g. see Dormand and Prince [28]). In every step, the local error estimator was computed as:

$$\boldsymbol{\varepsilon}_t = |\mathbf{x}_a(t+\Delta t) - \mathbf{x}_b(t+\Delta t)|. \tag{17}$$

For adaptive step-size methods, the error estimator $\boldsymbol{\varepsilon}_t = [\varepsilon_t^1, ..., \varepsilon_t^d]^\top$ was used to compute an error norm $||\mathbf{e}||$ on $\mathbf{e} = [e_1, ..., e_d]^\top$, where d was the dimension of the state vector $\mathbf{x}(t) = [x_1(t), ..., x_d(t)]^\top$. For every state variable x_i, e_i was computed as:

$$e_i = \frac{\varepsilon_t^i}{\kappa_a + \kappa_r \cdot \max(|x_i(t)|, |x_i(t+\Delta t)|)}, \tag{18}$$

with κ_a and κ_r being the absolute and relative tolerance. For simplicity, we used $\kappa_a = \kappa_r$ in all simulations and therefore refer to these parameters as the *tolerance* κ.

$||\mathbf{e}||$ was computed as the root-mean-square of $\mathbf{e}$, i.e.:

$$||\mathbf{e}|| = \sqrt{\frac{1}{d}\sum_i e_i^2}. \tag{19}$$

If $||\mathbf{e}|| < 1$, the step was accepted, and rejected otherwise. In both cases, the step-size was adapted and the next step-size Δt_{next} was computed as:

$$\Delta t_{\text{next}} = 0.9 \cdot \Delta t \cdot \min(\max(||\mathbf{e}||^{-1/k_{\text{exp}}}, k_{\min}), k_{\max}), \tag{20}$$

where $k_{\min}$ and $k_{\max}$ are the minimum and maximum allowed change factors, that we set to typical values of 0.1 and 5 respectively [9]. k_{exp} was 2 for FE, 3 for RKBS, 4 for RKCK, and 5 for RKDP, corresponding to the order of the error estimator. Furthermore, we limited the step-sizes to be always smaller or equal to a maximum step-size $\Delta t_{\max}$, which we set to $\Delta t_{\max} = 1\,\text{ms}$ for all simulations.

B Solver details

Runge-Kutta steps were implemented based on the `scipy` implementation [30]. The Butcher tableau for the RKCK method was taken from [27]. Heun's method was used as an error estimator for the FE method and implemented as follows. Given t, Δt, $\mathbf{x}(t)$, $f(t, \mathbf{x}(t))$ and the deterministic FE solution $\mathbf{x}_{\text{det}}^{\text{FE}}(t+\Delta t)$ the solution for Heun's method was computed as:

$$\mathbf{x}(t+\Delta) = \mathbf{x}(t) + \Delta t \left[f(t, \mathbf{x}(t)) + f(t, \mathbf{x}_{\text{det}}^{\text{FE}}(t+\Delta t)) \right] / 2 \tag{21}$$

To use the exponential integrators EE and EEMP, the ODEs were cast into the following form:

$$\dot{z}(t, \mathbf{x}(t)) = [z_\infty(\mathbf{x}) - z] / z_\tau(\mathbf{x}), \tag{22}$$

where z is a state variable of $\mathbf{x}$ (e.g. the membrane potential) and $z_\infty(\mathbf{x})$ and $z_\tau(\mathbf{x})$ are functions depending on $\mathbf{x}$ but not explicitly on t. For a derivation of these functions for the HH model see for example [43]. The EE step was implemented as:

$$z(t+\Delta t) = z_\infty(\mathbf{x}) + [z(t) + z_\infty(\mathbf{x})] \exp(-\Delta t / z_\tau(\mathbf{x})). \tag{23}$$

The second order exponential integrator EEMP proposed by Börgers and Nectow [11] builds on the EE method. Given t, Δt, $\mathbf{x}(t)$, the half-step EE solution $\tilde{\mathbf{x}} = \mathbf{x}_{\text{det}}^{\text{EE}}(t+\Delta t/2)$ and the evaluations of $z_\infty(\tilde{\mathbf{x}})$ and $z_\tau(\tilde{\mathbf{x}})$ at the half-step, the solution for z using the EEMP method was computed as:

$$z(t+\Delta t) = z_\infty(\tilde{\mathbf{x}}) + [z(t) + z_\infty(\tilde{\mathbf{x}})] \exp(-\Delta t / z_\tau(\tilde{\mathbf{x}})). \tag{24}$$

We also implemented the solver used in the original implementation of the IN neurons, where the IVP was solved with a method similar to FE of fixed step-size Δt [23]. The implementation differs from a standard FE scheme in so far, as v and u are updated subsequently:

$$\begin{aligned} v(t+\Delta t) &= v(t) + \Delta t \cdot \dot{v}(t, u(t), v(t)), \\ u(t+\Delta t) &= u(t) + \Delta t \cdot \dot{u}(t, u(t), v(t+\Delta t)). \end{aligned} \tag{25}$$

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

C Neuron model parameters

The neuron models simulated in this study were parametrized as follows. The parameters $\boldsymbol{\theta} = [a, b, c, d]$ and I_{Stim} of the IN model and the respective original step-sizes Δt were taken from `https://www.izhikevich.org/publications/figure1.m` [23].

The three maximum conductances for the classical HH neuron (see Eq. (12)) were set to $\bar{g}_{\text{Na}} = 1.2\,\text{mS}$, $\bar{g}_{\text{K}} = 0.36\,\text{mS}$ and $\bar{g}_{\text{leak}} = 0.003\,\text{mS}$. The membrane capacitance was set to $C = 0.01\,\mu\text{F}$ (see Eq. (11)). For all STG neurons, we set the membrane area to $A = 0.628 \times 10^{-3}\,\text{cm}^2$ and the membrane capacitance to $C = A \cdot 1\,\mu\text{F/cm}^2$ (see Eq. (11)). An STG neuron has eight maximum channel conductances (see Eq. (12)):

$$\boldsymbol{\theta}_{\text{STG-neuron}} = [\bar{g}_{\text{Na}}, \bar{g}_{\text{CaT}}, \bar{g}_{\text{CaS}}, \bar{g}_{\text{A}}, \bar{g}_{\text{KCa}}, \bar{g}_{\text{Kd}}, \bar{g}_{\text{H}}, \bar{g}_{\text{leak}}]. \tag{26}$$

For the single STG neuron, we set $\boldsymbol{\theta}_{\text{STG-neuron}} = A \cdot [400, 2.5, 10, 50, 20, 0, 0.04, 0]\,\text{mS/cm}^2$, taken from an example in [35]. The STG neuronal network consists of three neuron models ABPD, LP and PY. The network is parametrized by the three neurons' conductances:

$$\begin{aligned} \boldsymbol{\theta}_{\text{ABPD}} &= A \cdot [100, 2.5, 6, 50, 5, 100, 0.01, 0.0]\,\text{mS/cm}^2, \\ \boldsymbol{\theta}_{\text{LP}} &= A \cdot [100, 0.0, 4, 20, 0, 25, 0.05, 0.03]\,\text{mS/cm}^2, \\ \boldsymbol{\theta}_{\text{PY}} &= A \cdot [100, 2.5, 2, 50, 0, 125, 0.05, 0.01]\,\text{mS/cm}^2, \end{aligned} \tag{27}$$

where $A = 0.628 \times 10^{-3}\,\text{cm}^2$ and the synaptic conductances $\boldsymbol{\theta}_{\text{syn}}$:

$$\boldsymbol{\theta}_{\text{syn}} = [\bar{g}^{\text{fast}}_{\text{ABPD-LP}}, \bar{g}^{\text{slow}}_{\text{ABPD-LP}}, \bar{g}^{\text{fast}}_{\text{ABPD-PY}}, \bar{g}^{\text{slow}}_{\text{ABPD-PY}}, \bar{g}^{\text{fast}}_{\text{ABPD-LP}}, \bar{g}^{\text{fast}}_{\text{LP-ABPD}}, \bar{g}^{\text{fast}}_{\text{LP-PY}}, \bar{g}^{\text{fast}}_{\text{PY-LP}}], \tag{28}$$

where for example $\bar{g}^{\text{fast}}_{\text{ABPD-LP}}$ is the maximum conductance of the fast synapse connecting neuron ABPD (presynaptic) to neuron LP (postsynaptic). We simulated the network for five different synaptic parametrizations taken from the original publication [25]:

$$\begin{aligned} \boldsymbol{\theta}^{\text{a}}_{\text{syn}} &= [10, 100, 10, 3, 30, 1, 3]\,\text{nS}, \\ \boldsymbol{\theta}^{\text{b}}_{\text{syn}} &= [3, 0, 0, 30, 3, 3, 0]\,\text{nS}, \\ \boldsymbol{\theta}^{\text{c}}_{\text{syn}} &= [100, 0, 30, 1, 0, 3, 0]\,\text{nS}, \\ \boldsymbol{\theta}^{\text{d}}_{\text{syn}} &= [3, 100, 10, 1, 10, 3, 10]\,\text{nS}, \\ \boldsymbol{\theta}^{\text{e}}_{\text{syn}} &= [30, 30, 10, 3, 30, 1, 30]\,\text{nS}. \end{aligned} \tag{29}$$

The frequencies f_s (Eq. (15)) for the fast and slow synapses were 25 Hz and 10 Hz, and the reversal potentials E_i (Eq. (14)) were $-70\,\text{mV}$ and $-80\,\text{mV}$, respectively [25].

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

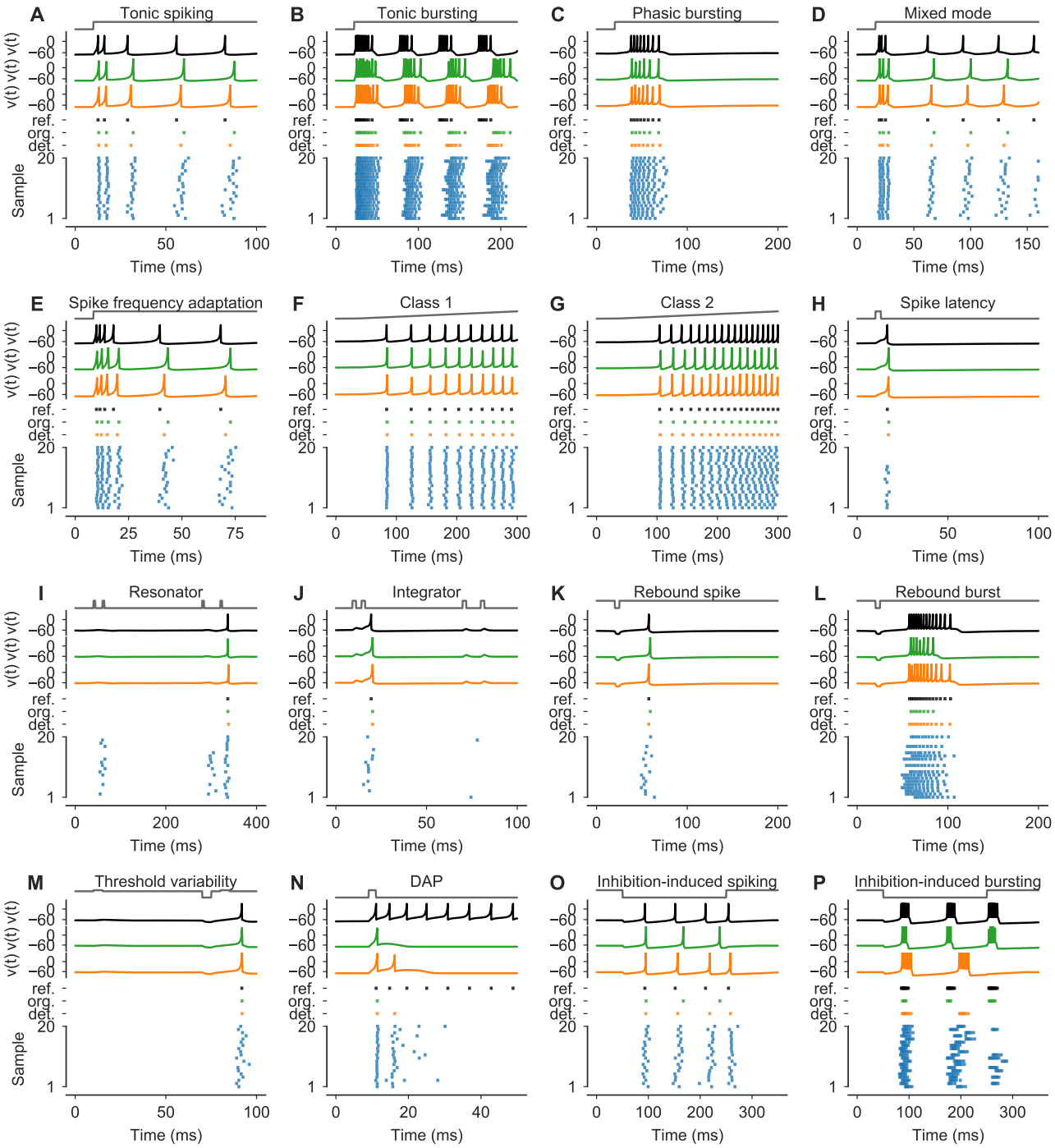

Figure S1: **(A-P)** As in Figs. 2B and 2C. Simulations of the IN model for different parametrizations θ_i with stimuli I_{Stim} (normalized in gray). Solutions for $v(t)$ and the respective spike-times for a reference solver (black) and the original solver scheme (orange). For plotting, $v(t)$ were clipped at 30. For both solutions, spike-times are shows in a raster plot (bottom) together with spike-times of 20 samples from a $\text{FE}_{\text{f}}^{\text{t}}$ solver (blue).

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

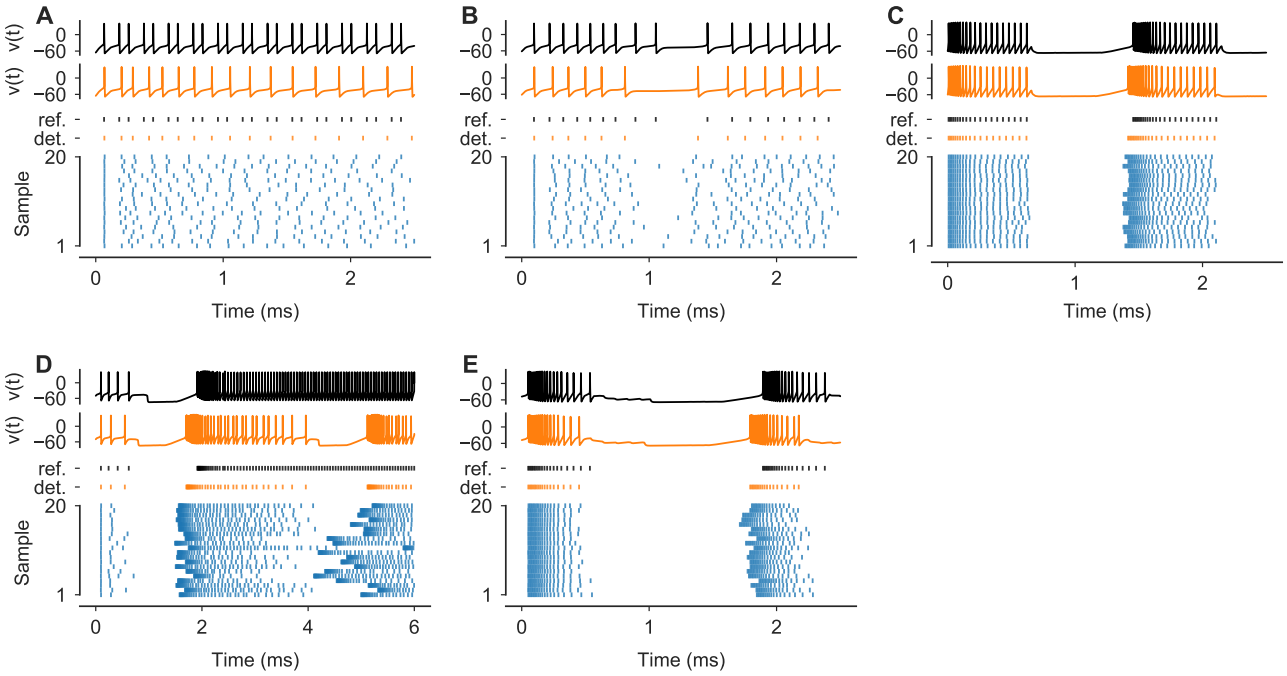

Figure S2: **(A-E)** As in Figs. 2D and 2E. Simulations of the STG model for all five synaptic parametrizations (see Appendix C) from $\boldsymbol{\theta}^{\mathrm{a}}_{\mathrm{syn}}$ to $\boldsymbol{\theta}^{\mathrm{e}}_{\mathrm{syn}}$, respectively. Solutions for the membrane potential $v(t)$ of the LP neuron are shown for a reference solver (black) and a deterministic EE solver with Δt=0.1 ms (orange). *Bottom panels*: Spike-times of the LP neuron are shown for the reference (black), the deterministic EE solution (orange) and for 20 samples from a probabilistic EE$^{\natural}_{\natural}$ with Δt=0.1 ms (blue).

bioRxiv preprint doi: https://doi.org/10.1101/2021.04.27.441605; this version posted December 1, 2021. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

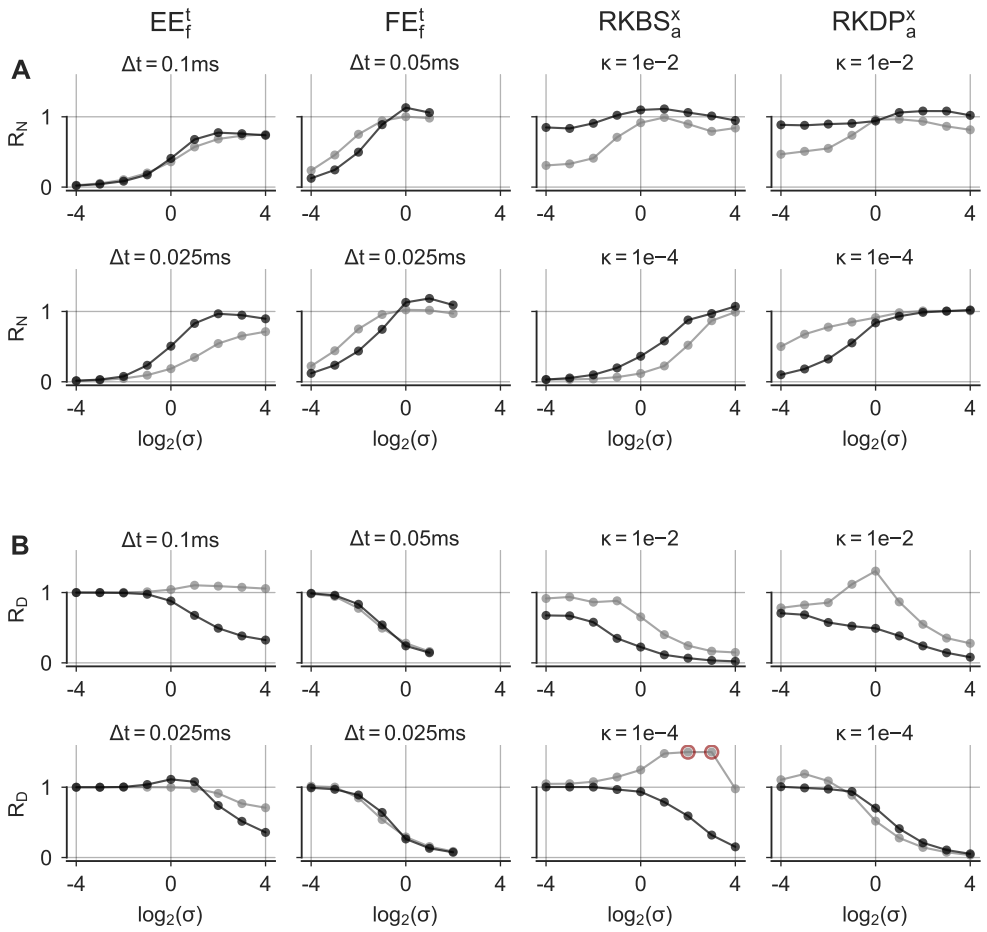

Figure S3: **(A, B)** MAE ratios R_N and R_D for different solvers (column titles) and step-sizes Δt / tolerances κ (panel titles) for the step (grey) and the noisy step (black) stimulus as a function of the perturbation parameter σ (x-axis), respectively. See also Fig. 6D. Values larger than 1.5 were clipped (indicated by red circles).