Numerical uncertainty can critically affect simulations of mechanistic models in neuroscience

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 biophysically realistic neuron models. To analyze such models one typically has to solve systems of 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. To overcome this problem, we propose to use recently developed probabilistic solvers instead, which are able to reveal and quantify numerical uncertainties, for example as posterior sample paths. Importantly, these solvers neither require detailed insights into the kinetics of the models nor are they difficult to implement. Using these probabilistic solvers, we show that numerical uncertainty strongly affects the outcome of typical neuroscience simulations, in particular due to the non-linearity associated with the generation of action potentials. We quantify this uncertainty in individual single Izhikevich neurons with different dynamics, a large population of coupled Izhikevich neurons, single Hodgkin-Huxley neuron and a small network of Hodgkin-Huxley-like neurons. For commonly used ODE solvers, we find that the numerical uncertainty in these models can be substantial, possibly jittering spikes by milliseconds or even adding or removing individual spikes from the simulation altogether.

Author summary

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 come typically in the form of ordinary differential equations (ODEs). The solution of these ODEs is computed using solvers with finite accuracy and, therefore, the computed solutions deviate from the true solution. If this deviation is too large but goes unnoticed, this can potentially lead to wrong scientific conclusions.

A field in machine learning called probabilistic numerics has recently developed a set of probabilistic solvers for ODEs, which not only produce a single solution of unknown accuracy, but instead yield a distribution over simulations. Therefore, these tools allow one to address the problem state above and quantitatively analyze the numerical uncertainty inherent in the simulation process.

In this study, we demonstrate how such solvers can be used to quantify numerical uncertainty in common neuroscience models. We study both Hodgkin-Huxley and Izhikevich neuron models and show that the numerical uncertainty in these models can be substantial, possibly jittering spikes by milliseconds or even adding or removing individual spikes from the simulation altogether. We discuss the implications of this finding and discuss how our methods can be used to select simulation parameters to trade off accuracy and speed.

Introduction

Neural computations can be described 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 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 it is unique and well-posed.

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 for a range of solvers and different integration step-sizes [9–11]. These studies have shown that standard solvers are often not the best choice in terms of accuracy or the accuracy vs. run time trade-off. 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 [9], an exponential midpoint method [10] or second-order Strang splitting [11] 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 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. In contrast to classical ODE solvers, this class of solvers does not only yield a single solution, but instead a distribution over the solution that quantifies the numerical uncertainty. Several frameworks for probabilistic ODE solvers have been proposed, which differ mostly in the trade-off between computational cost and flexibility of the posterior, from basic Gaussian forms [12, 13] to sampling-based approaches [14–18]. 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 underlie the all-or-none nature of an action potential. In this study, we explore the potential of probabilistic ODE solvers for neuron models, and demonstrate how they can be used to quantify and reveal numerical uncertainty caused by the numerical ODE integration.

Methods or Models

Common ODE models in computational neuroscience

Here we study the effect of numerical uncertainty in the following common neuroscience models involving the simulation of ODEs:

- a single neuron Izhikevich neuron model with a wide range of dynamics,
- a large network of Izhikevich neurons,
- a single Hodgkin-Huxley neuron,
- a small network of Hodgkin-Huxley neurons.

Single Izhikevich neurons

The Izhikevich neuron (IN) model is a simplified non-linear single neuron model in complexity between Hodgkin-Huxley and Integrate & Fire neurons. INs have been used e.g. to build large-scale models of the brain [19] and to understand oscillatory phenomena in the cortex [20, 21] and the olfactory bulb [22]. Its behavior is described by the following pair of ODEs [20]:

$$\begin{aligned} \dot{v}(t, v, u) &= \left(0.04 \cdot v^2 + 5 \cdot v - u + I_{\text{Stim}}(t)\right) / \text{ms}, \\ \dot{u}(t, v, u) &= \left(a(b \cdot v - u)\right) / \text{ms}, \end{aligned} \tag{1}$$

where $v(t)$ is the membrane voltage, $u(t)$ is a recovery variable and $I_{\text{Stim}}(t)$ 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:

$$\begin{aligned} v(t + \Delta t_{\text{spike}}) &= c \\ u(t + \Delta t_{\text{spike}}) &= u(t) + d \end{aligned} \tag{2}$$

where $\Delta t_{\text{spike}} \geq 0$. Typically Δt_{spike} is the set to the solver step-size Δt, which only makes sense for fixed step-size solvers, or to 0. Here we used $\Delta t_{\text{spike}} = 0$ to facilitate the comparison between different step-sizes, and between fixed and adaptive step-sizes. An attractive property of the IN is that a whole range of different response dynamics can be simulated (Fig. 2) depending on the setting of the parameters $\boldsymbol{\theta} = [a, b, c, d]$ [23]. We simulated different parametrizations, with values taken from `https://www.izhikevich.org/publications/figure1.m`.

A network of Izhikevich neurons

INs can also be coupled to simulate the activity in neural networks, based on the same equations used for the single neurons. Here, we study a network of 800 excitatory and 200 inhibitory neurons, which has been used to simulate gamma oscillations [20]. In this network, every neuron n_i is parametrized randomly:

$$\boldsymbol{\theta}_i = \begin{cases} [0.02, 0.2, -65 + \rho_i^2, 8 - 6\rho_i^2], & \text{if } n_i \text{ is excitatory}, \\ [0.02 + 0.08\rho_i, 0.25 - 0.05\rho_i, -65, 2], & \text{otherwise}, \end{cases} \tag{3}$$

where $\rho_i \sim \mathcal{U}(0, 1)$, and where $\mathcal{U}(a, b)$ denotes a uniform distribution between a and b. Here, the input current $I_{\text{Stim}}(t)$ in Eq. (1) to a neuron n_i is replaced by an input current $I_i(t)$, which depends on all other neurons n_j, and the randomly initialized weights w_{ij} that do not change over time between them. Weights from inhibitory neurons n_j to any other neuron n_i are drawn from $\mathcal{U}(-1, 0)$ and are therefore negative, whereas weights w_{ij} from excitatory neurons n_j are drawn from $\mathcal{U}(0, 0.5)$ and positive. Input currents are computed as:

$$I_i(t) = I_i^{\text{Stim}}(t) + \sum_{j=1}^{1000} w_{ij} s_j(t), \tag{4}$$

where $s_j(t)$ is a function that depends on the spikes of neuron n_j. $I_i^{\text{Stim}}(t)$ is an additional zero-centered Gaussian noise stimulus that was different for every neuron, with a standard deviation of 5 and 2 for excitatory and inhibitory neurons, respectively. We simulated two versions of these Gaussian noise stimulus currents. In the first, the currents changes every full millisecond and are constant in between, as it was in the original implementation. We refer to this stimulus as the step stimulus. Second, we used a smooth stimulus, for which we fitted a cubic spline to the noise stimulus currents,

such that at full milliseconds the stimuli were identical, but the transitions between those points were smooth.

Originally, the network was simulated with a fixed step-size of 1 ms, and every neuron n_i received input from all neurons n_j that spiked in the previous step of the simulation, i.e. $s_j(t) = 1$ if the input neuron n_j spiked 1 ms before, and $s_j(t) = 0$ otherwise. To extend this model to smaller step-sizes, we defined s_j similar to the probability density function of a log-normal distribution as:

$$s_j(t) = \frac{\exp\left(-c_1 \left(\log\left(\beta_j(t) - c_2\right) - c_3\right)^2\right)}{\beta_j(t) - c_2}, \tag{5}$$

where β_j is the time that passed since the last spike of neuron n_j, and c_1, c_2 and c_3 were chosen such that the integral for s over β_j between zero and infinity sums up to one and that $s_j(t = 1) = 1$, i.e. $c_1 = 3.125$, $c_2 = 0.0775$ and $c_3 = 0.08$.

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 single compartment HH-like neuron model [25] prominently used to study the stomatogastric ganglion (STG) [26]. Both models are described by ODEs including the membrane voltage $v(t)$ described by:

$$\dot{v}(t) = \left(I_{\text{Stim}}(t) - \textstyle\sum_i I_i(\mathbf{x})\right)/C, \tag{6}$$

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:

$$I_i(\mathbf{x}) = \bar{g}_i \cdot m_i(\mathbf{x})^{p_i} \cdot h_i(\mathbf{x}) \cdot (v - E_i), \tag{7}$$

where E_i is the channel's reversal potential, $\bar{g}_i$ 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:

$$\dot{m}(v) = \left(m_\infty(v) - m\right)/\tau_m(v), \quad \dot{h}(v) = \left(h_\infty(v) - h\right)/\tau_h(v), \tag{8}$$

where m_∞, τ_m, h_∞, and τ_h are voltage dependent functions defining the channel's kinetics. In the classical HH model, this amounts to a 4-dimensional ODE [27]. 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 [25].

The classical HH neuron is parametrized by its three maximum conductances $\boldsymbol{\theta}_{\text{HH}} = [\bar{g}_{\text{Na}}, \bar{g}_{\text{K}}, \bar{g}_{\text{leak}}]$, and the stimulus current I_{Stim}. Here we used $\boldsymbol{\theta}_{\text{HH}} = A \cdot [120, 36, 0.3]\,\text{mS/cm}^2$, where the membrane area was $A = 0.01\,\text{cm}^2$. We simulated four different stimuli I_{Stim}: a constant stimulus ("constant"), and step stimulus ("step"), a noisy step stimulus ("noisy") and a noisy subthreshold stimulus ("subthreshold"). For the constant and step stimulus, the amplitude was set to 0.15 µA. All simulations with the HH neuron were set to $t \in [0\,\text{ms}, 100\,\text{ms}]$, and the on- and offset of all stimuli, except the constant one, were set to $t_{\text{onset}} = 10\,\text{ms}$ and $t_{\text{offset}} = 90\,\text{ms}$, respectively. The amplitude of the noisy step stimulus was computed by drawing 51 amplitudes from a uniform distribution between 0 µA and 0.4 µA that were placed equidistant in time between stimulus on- and offset, and interpolated using a cubic spline. The noisy subthreshold stimulus was computed analogously, except that the amplitudes were drawn from a zero-centered uniform distribution between −0.01 µA and 0.01 µA, which was too small in amplitude to induce any spiking.

The STG neuron is parametrized by its eight maximum conductances:

$$\boldsymbol{\theta}_{\mathrm{STGN}} = [\bar{g}_{\mathrm{Na}}, \bar{g}_{\mathrm{CaT}}, \bar{g}_{\mathrm{CaS}}, \bar{g}_{\mathrm{A}}, \bar{g}_{\mathrm{KCa}}, \bar{g}_{\mathrm{Kd}}, \bar{g}_{\mathrm{H}}, \bar{g}_{\mathrm{leak}}]. \tag{9}$$

For the single STG neuron, we set $\boldsymbol{\theta} = A \cdot [400, 2.5, 10, 50, 20, 0, 0.04, 0]\,\mathrm{mS/cm^2}$, where $A = 0.628 \times 10^{-3}\,\mathrm{cm^2}$ and $I_{\mathrm{Stim}}(t)$ was a current step of $6\,\mathrm{nA}$ starting at $t_{\mathrm{onset}} = 0.9\,\mathrm{s}$.

STG model

The HH-like STG neuron model described above was used by Prinz et al. [26] in a network of three syntactically coupled neurons to study their firing patterns. In the model there are two types of synapses, slow and fast ones, connecting the neurons AB (short for ABPD), LP and PY with a total count of seven synapses. The synaptic input current to a neuron is described by the equation in Eq. (7), except that the activation m and inactivation h variables were replaced by a single function s that is described by:

$$\begin{aligned} \dot{s} &= (\bar{s} - s)/\tau_s, \\ \bar{s} &= \left(1 + \exp((V_{th} - V_{pre})/5\,\mathrm{mV})\right)^{-1}, \\ \tau_s &= (1 - \bar{s})/f_s \end{aligned} \tag{10}$$

where V_{pre} is the membrane voltage of the presynaptic neuron. The frequencies f_s for the fast and slow synapses were $25\,\mathrm{Hz}$ and $10\,\mathrm{Hz}$, and the reversal potentials E_i (see Eq. (7)) were $-70\,\mathrm{mV}$ and $-80\,\mathrm{mV}$, respectively. The network is parametrized by both the conductances of the neurons $\boldsymbol{\theta}_{\mathrm{STGN}}$ and synaptic conductances $\boldsymbol{\theta}_{\mathrm{Syn.}}$:

$$\boldsymbol{\theta}_{\mathrm{Syn.}} = [\bar{g}^{\mathrm{fast}}_{\text{AB-LP}}, \bar{g}^{\mathrm{slow}}_{\text{AB-LP}}, \bar{g}^{\mathrm{fast}}_{\text{AB-PY}}, \bar{g}^{\mathrm{slow}}_{\text{AB-PY}}, \bar{g}^{\mathrm{fast}}_{\text{AB-LP}}, \bar{g}^{\mathrm{fast}}_{\text{LP-AB}}, \bar{g}^{\mathrm{fast}}_{\text{LP-PY}}, \bar{g}^{\mathrm{fast}}_{\text{PY-LP}}], \tag{11}$$

where for example $\bar{g}^{\mathrm{fast}}_{\text{AB-LP}}$ is the maximum conductance of the fast synapse connecting neuron AB (presynaptic) to neuron LP (postsynaptic). We simulated the network for five different parametrizations taken from the original publication [26]. In the chosen parametrizations the neuron conductances are fixed and set to:

$$\begin{aligned} \boldsymbol{\theta}_{\mathrm{AB}} &= A \cdot [100, 2.5, 6, 50, 5, 100, 0.01, 0.0]\,\mathrm{mS/cm^2}, \\ \boldsymbol{\theta}_{\mathrm{LP}} &= A \cdot [100, 0.0, 4, 20, 0, 25, 0.05, 0.03]\,\mathrm{mS/cm^2}, \\ \boldsymbol{\theta}_{\mathrm{PY}} &= A \cdot [100, 2.5, 2, 50, 0, 125, 0.05, 0.01]\,\mathrm{mS/cm^2}, \end{aligned} \tag{12}$$

with $A = 0.628 \times 10^{-3}\,\mathrm{cm^2}$ for the AB, LP and PY neuron, respectively. The five synaptic parametrizations analyzed are:

$$\begin{aligned} \boldsymbol{\theta}^{\mathrm{a}}_{\mathrm{Syn.}} &= [10, 100, 10, 3, 30, 1, 3]\,\mathrm{nS}, \\ \boldsymbol{\theta}^{\mathrm{b}}_{\mathrm{Syn.}} &= [3, 0, 0, 30, 3, 3, 0]\,\mathrm{nS}, \\ \boldsymbol{\theta}^{\mathrm{c}}_{\mathrm{Syn.}} &= [100, 0, 30, 1, 0, 3, 0]\,\mathrm{nS}, \\ \boldsymbol{\theta}^{\mathrm{d}}_{\mathrm{Syn.}} &= [3, 100, 10, 1, 10, 3, 10]\,\mathrm{nS}, \\ \boldsymbol{\theta}^{\mathrm{e}}_{\mathrm{Syn.}} &= [30, 30, 10, 3, 30, 1, 30]\,\mathrm{nS}. \end{aligned} \tag{13}$$

Neuron simulations as numerical solutions of initial value problems

Simulating the activity of any of these neuron models amounts to solving an initial value problem based on a set of coupled ODEs. In abstract form, an initial value problem is given by

$$\dot{\mathbf{x}}(t) = f(t, \mathbf{x}(t)), \quad \mathbf{x}(t_0) = \mathbf{x}_0, \tag{14}$$

where f, $\mathbf{x}_0$ and t_0 are known and $\mathbf{x}(t)$, $t > t_0$ is the quantity of interest. All the aforementioned neuron models can be cast in that form. The solution to the initial value problem at time $t + \Delta t$ provided the solution at time t, is given by integrating Eq. (14) from t to $t + \Delta t$:

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \int_t^{t+\Delta t} f(s, \mathbf{x}(s))\,\mathrm{d}s. \tag{15}$$

Except for special cases, this integral has no analytic form and must be solved numerically. Among the most basic approximations for this purpose is the forward Euler method, which is also frequently used in simulations of neural systems. This method approximates the integral as $\int_t^{t+\Delta t} f(s, \mathbf{x}(s))\,\mathrm{d}s \approx \Delta t f(t, \mathbf{x}(t))$. Several families of more advanced methods are available in numerical analysis. They include in particular the Runge-Kutta family, which rests on a deep body of theoretical analysis [28, Chapter 2] and has become a popular standard. For example, the *Dormand-Prince pair* [29] of two embedded explicit Runge-Kutta methods of order 4 and 5, respectively, has become an industry standard, available with adaptive step-size selection as Matlab's `ode45` or scipy's `solve_ivp`.

Even with advanced solvers, some equations of type (14) require extremely small step-sizes Δt in some regions to be numerically stable and the scheme to be successful, while in other regions much larger step-sizes would yield accurate solutions. To alleviate this issue, fast step-size adaptation schemes have been devised. For example, for methods of Runge-Kutta type, one usually runs two different Runge-Kutta methods simultaneously and uses their discrepancy as a local error estimate on the base of which step-sizes are adapted. In the context of simulating the neuron models above, adaptive step-sizes can be especially useful when a neuron alternates between spiking and silence; when the neuron spikes, the solver takes small steps to accurately capture the steep slope of the action potential, when it does not, the steps are larger.

Probabilistic solvers for quantifying uncertainty in neural simulations

When simulating neurons or neural populations, one would like to know how close the numerically computed solution is to the true solution. Most of the classic theory studies how the well-established solvers behave in the asymptotic regime of infinitesimally small step-sizes. Most widely available numerical solvers do report a global error estimate and a corresponding tolerance that can be set by the user [28, Chapter II.4]. This global scalar error, though, does not capture how the finite error arising from finite step-sizes used in practice affects crucial quantities of interest in the simulation, such as spike-times or number.

Addressing this issue, probabilistic numerics provides algorithms to quantify this kind of numerical uncertainty [8, 30, 31]. For sufficiently regular ODE solutions, randomized solvers have the same mean-square convergence rates as their deterministic counterparts, provided the perturbation variance is at most of the same order as the local error [14, 32]. One example is the state perturbation algorithm of Conrad et al. [14], which uses a stochastically perturbed version of the integration scheme. In each step of the numerical integration, a small i.i.d. noise term $\boldsymbol{\xi}_t^{\Delta t}$ is added to the solution $\hat{\mathbf{x}}(t + \Delta t)$ of a corresponding deterministic integration scheme:

$$\mathbf{x}(t + \Delta t) = \hat{\mathbf{x}}(t + \Delta t) + \boldsymbol{\xi}_t^{\Delta t}, \quad \boldsymbol{\xi}_t^{\Delta t} \sim \mathcal{N}(\mathbf{0}, \mathrm{diag}(\boldsymbol{\nu}_t)^2). \tag{16}$$

The perturbation is only efficient if the noise $\boldsymbol{\nu}_t$ is of the right order of magnitude: if chosen too small, the uncertainty will be underestimated; if chosen too large, it will render the solver output useless. Conrad et al. [14] suggested calibrating $\boldsymbol{\nu}_t$ to replicate

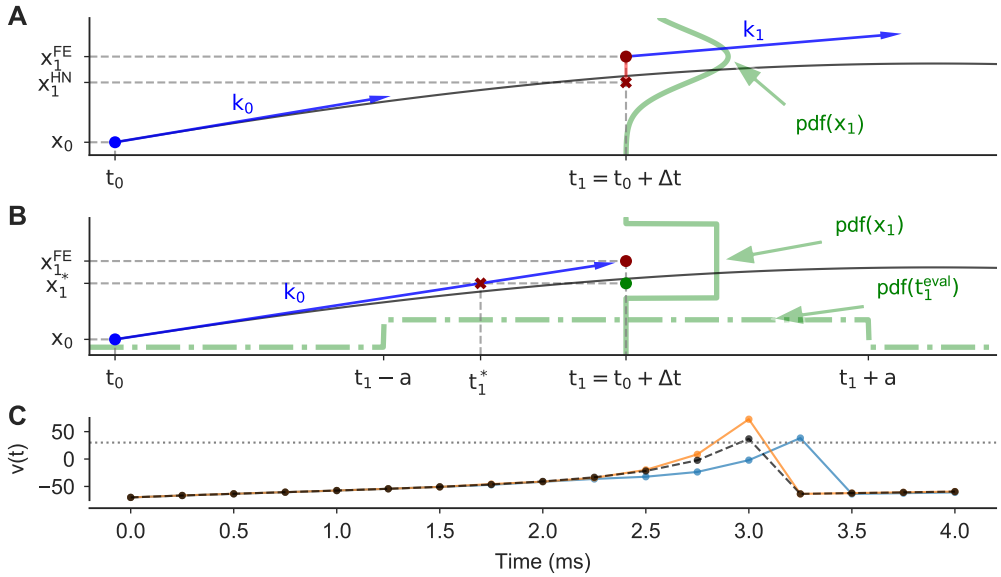

Fig 1. Illustration of probabilistic ODE solvers. (A) A single integration step with a probabilistic forward Euler method using the state perturbation method [14] for the ODE $f(t, x(t)) = x(t) \cdot \sin(t)$ and the exact solution $x(t) = \exp(\cos(t))$ (black line). A first order solution is computed using forward Euler: $x_1^{\mathrm{FE}} = x_0 + \Delta t k_0$ (red dot), where $k_0 = f(t_0, x_0)$. A second order solution is computed using Heun's method: $x_1^{\mathrm{HN}} = x_0 + 0.5\Delta t(k_0 + k_1)$ (red x), where $k_1 = f(t_0 + \Delta t, x_1^{\mathrm{FE}})$. The error estimate $\varepsilon = |x_1^{\mathrm{FE}} - x_1^{\mathrm{HN}}|$ can be used to calibrate ν in Eq. (16) to compute a stochastic solution. For $\nu = \varepsilon$ the probability density function of the solver solution for this step is given by $\mathrm{pdf}(x_1) = \mathcal{N}(x_1^{\mathrm{FE}}, \varepsilon^2)$ (green). **(B)** Similar to (A), but for the step-size perturbation method [18] using a uniform perturbation distribution. Instead of integrating from t_0 to t_1 (red dot), the ODE is integrated from t_0 to t_1^*, where t_1^* is randomly drawn from a uniform distribution $\mathrm{pdf}(t_1^{\mathrm{eval}})$ (green dashed line). The solution of this perturbed integration x_1^* (red cross) is then used as the solution x_1 at t_1 (green dot), making x_1 a random variable with a distribution $\mathrm{pdf}(x_1)$. Here, $\mathrm{pdf}(x_1)$ is a also a uniform distribution (green solid line). **(C)** Two runs of a probabilistic forward Euler method (orange/blue) and the solution of a deterministic forward Euler method (black) for an Izhikevich neuron.

the amount of error introduced by the numerical scheme. We chose $\boldsymbol{\nu}_t = \sigma \boldsymbol{\varepsilon}_t$ using the aforementioned error estimator $\boldsymbol{\varepsilon}_t \approx |\mathbf{x}_{\mathrm{exact}}(t) - \mathbf{x}_{\mathrm{numerical}}(t)|$ readily available in methods that were developed for step-size adaptation, and a perturbation parameter σ that can be adjusted to calibrate the perturbation. If not stated otherwise, we used $\sigma = 1$. An example of this perturbation method is shown in Fig. 1A.

A related approach to stochastically perturbing the numerical integration was proposed by Abdulle and Garegnani [18], 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 $\zeta_t^{\Delta t}$, but the computed solution is treated as the solution for the original step-size Δt:

$$\mathbf{x}(t + \Delta t) = \hat{\mathbf{x}}(t + \zeta_t^{\Delta t}), \quad \zeta_t^{\Delta t} \sim \mathcal{P}, \tag{17}$$

where $\zeta_t^{\Delta t}$ is the i.i.d. perturbed step-size drawn from a distribution $\mathcal{P}$ with mean $\mathbb{E}[\zeta_t^{\Delta t}] = \Delta t$ and $\hat{\mathbf{x}}(\bullet)$ is a deterministic integration scheme that approximates Eq. (15). For example, for the forward Euler method Eq. (17) would be computed as $\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \zeta_t^{\Delta t} f(t, \mathbf{x}(t))$.

To neither over- nor underestimate the uncertainty in the numerical integration, the distribution $\mathcal{P}$ must be calibrated with respect to the order of the underlying integration scheme and the step-size. Abdulle and Garegnani [18] proposed two examples from which one is a zero-centered uniform-distributions $\mathcal{P} = \mathcal{U}(\Delta t - a, \Delta t + a)$ with bounds dependent on the step-size $a = \Delta t^{p+0.5}$. The bounds have to be chosen such that $a > \Delta t$ holds to avoid negative step-sizes. 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 found that using such a distribution is not robust to changes of the step-size unit and—more generally—is prone to poor calibration. Assume for example that for a given model the step-size Δt is changed from $\Delta t_1 = 0.0001\,\mathrm{s}$ to $\Delta t_2 = 0.1\,\mathrm{ms}$, i.e. the step-size is numerically changed from 0.0001 to 0.1. If the model treats this unit change appropriately, this change should not alter the model output as both step-sizes Δt_1 and Δt_2 are physically identical. Both, deterministic and probabilistic solvers using state perturbation would indeed be unaffected by such a change (except for potentially different rounding errors). However, a probabilistic solver using step-size perturbation (Eq. (17)) with $\mathcal{P} = \mathcal{U}(\Delta t - a, \Delta t + a)$ and $a = \Delta t^{p+0.5}$, would yield very different solutions for the two step-sizes. For example setting $p = 2$, yields either $a = 0.0001^{2+0.5} = 1 \times 10^{-10}$ or $a = 0.1^{2+0.5} \approx 0.003$ for Δt_1 and Δt_2, respectively. Since a implicitly has the same unit as Δt, the bounds in the two cases would be $a = 1 \times 10^{-7}\,\mathrm{ms}$ and $a \approx 3 \times 10^{-3}\,\mathrm{ms}$, respectively, resulting in different amplitudes of perturbation and therefore different uncertainty estimates. To be able to account for such unit changes and more generally to be able to calibrate the perturbation method, we defined the bounds of the uniform distribution $\mathcal{P} = \mathcal{U}(\Delta t - a, \Delta t + a)$ instead as:

$$a = \sigma \cdot \Delta t^{O+0.5}, \quad \sigma > 0, \quad a > \Delta t, \tag{18}$$

where σ is a perturbation parameter that scales the standard deviation linearly and has to be calibrated dependent on the model. The perturbation is illustrated in Fig. 1B for the first order forward Euler scheme with $\sigma = 0.75$.

The second step-size perturbation distribution they proposed is a log-normal distribution $\zeta_t^{\Delta t} \sim \mathcal{LN}_t^{\Delta t}(m, s^2)$. Here, we also added a perturbation parameter σ that scales the standard deviation linearly. Setting $\mathbb{E}[\zeta_t^{\Delta t}] = \Delta t$ and $\mathrm{Var}[\zeta_t^{\Delta t} - \Delta t] = \sigma^2 \cdot \Delta t^{2p+1}$ (see [18]) gives the following mean m and standard deviation s for the underlying normal distribution:

$$\begin{aligned} m &= \ln(\Delta t^2/\phi), \\ s &= \sqrt{2\ln(\phi/\Delta t)}, \\ \phi &= \sqrt{\mathbb{E}^2[\zeta_t^{\Delta t}] + \mathrm{Var}[\zeta_t^{\Delta t}]} = \sqrt{\Delta t^2 + \sigma^2 \cdot \Delta t^{2p+1}}. \end{aligned} \tag{19}$$

In contrast to the uniform distribution, using the log-normal distribution can not result in negative step-sizes independent of the step-size and perturbation parameter.

Choice of solvers and step-size adaptation

Here we used the perturbation methods presented by Conrad et al. [14] and Abdulle and Garegnani [18] 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 HN for error estimation.
HN	2	1	*Heun's method* with FE for error estimation.
EE	1	2	*Exponential Euler* with EEMP for error estimation.
EEMP	2	1	*Exponential Euler Midpoint* [10] with EE for error estimation.
RKBS	3	2	*Bogacki–Shampine*, an embedded Runge-Kutta method [33].
RKCK	4	5	*Cash–Karp method*, an embedded Runge-Kutta method [34].
RKDP	5	4	*Dormand–Prince*, an embedded Runge-Kutta method [29].

O and O_e are the orders of the solution and the error estimator, respectively.

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 [14] and t for the step-size perturbation [18]—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.

The probabilistic solvers were implemented in Python. The respective code and the code for the models and figures is available at `https://github.com/berenslab/neuroprobnum` as python code and jupyter notebooks. The second order exponential integrator EEMP was implemented based on the version by Börgers and Nectow [10], which is a modification of the midpoint method by Oh and French [35]. Computation of Runge-Kutta steps and step-size adaptation were based on the respective scipy implementations [36]. To avoid computational overhead, we only computed the local error estimates when necessary, i.e. for adaptive step-sizes or the state perturbation.

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

$$e_i = \frac{\varepsilon_t^i}{\kappa_a + \kappa_r \cdot \max(|x_i(t)|, |x_i(t + \Delta t)|)}, \tag{20}$$

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.

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

$$||\mathbf{e}|| = \sqrt{\frac{1}{d} \sum_i e_i^2}. \tag{21}$$

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

$$\Delta t_{\mathrm{next}} = 0.9 \cdot \Delta t \cdot \min(\max(||\mathbf{e}||^{-1/k_{\mathrm{exp}}}, k_{\mathrm{min}}), k_{\mathrm{max}}), \tag{22}$$

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 [28]. k_{exp} was 2 for FE, HN, EE and EEMP, 3 for RKBS, 4 for RKCK, and 5 for RKDP. Furthermore, we limited the step-sizes to be always smaller or equal to a maximum step-size Δt_{max}, which we set to $\Delta t_{\mathrm{min}} = 1\,\mathrm{ms}$ for all models if not stated otherwise.

Computational overhead of probabilistic methods

The methods used in this study differ in their computational costs per integration step. Higher order methods like RKBS and RKDP use multiple stages—i.e. multiple evaluations of the ODE per step—whereas FE requires only a single stage.

Another factor that needs to be considered when looking at computational costs is that computing a sample with either of the two probabilistic methods used here comes with a computational overhead compared to their deterministic counterparts. For the state perturbation [14] this overhead is three-fold. 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. 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 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—which does not make use of this property—no additional ODE evaluation is required. However, for first order methods like FE and EE this overhead severely reduces the computational efficiency because instead of a single ODE evaluation per step, a probabilistic version needs two (+100%). Additionally, as typically more steps are computed compared to higher order method, also the sampling from the Gaussian becomes more expensive in total. For the step-size perturbation, the overhead is reduced to the sampling from the perturbation distribution, and—for adaptive step-size methods—the loss of the First Same As Last property. To estimate the computational overhead empirically, we compare run times of samples from probabilistic and deterministic solvers.

Resets in Izhikevich neurons

INs are reset when their voltage $v(t)$ reaches the threshold of 30. During a single integration step, this threshold can be exceeded, which can lead to large overshoots that—in the worst case—cause numerical instabilities. We therefore implemented Eq. (1) such that whenever $\dot{v}(t, v, u)$ or $\dot{u}(t, v, u)$ were evaluated for $v(t)$ greater than the threshold 30, we used the threshold value instead, i.e.:

$$\begin{aligned} \dot{v}(t, v, u) &= \left(0.04 \cdot \min(v, 30)^2 + 5 \cdot \min(v, 30) - u + I(t)\right) / \text{ms}, \\ \dot{u}(t, v, u) &= \left(a(b \cdot \min(v, 30) - u)\right) / \text{ms}. \end{aligned} \tag{23}$$

Despite this modification, for fixed step-sizes these overshoots can cause large errors in the solution which are of order $O(\Delta t)$ [9]. For adaptive step-sizes, these resets—if not addressed properly—can be even more problematic, as the local errors controlling the step-size can be small during a step exceeding the threshold, leading to a large overshoot and thereby to a large global error. Stewart et al. [9] suggested to use intermediate steps whenever the threshold is exceeded. For this, first the time when the threshold was reached during the step is estimated, then the step is split into a step up to the threshold and a step after the reset. We adapted this strategy for Runge-Kutta methods and more specifically state perturbed Runge-Kutta methods as follows. To determine the spike-times we used the dense output $d_{\text{RK}}(t, t_i, t_{i+1})$ of Runge-Kutta methods that interpolate the solution of a single integration step from t_i to t_{i+1} and return the evaluation of this interpolation at time t, where $t_i \leq t \leq t_{i+1}$. For FE this

interpolation is linear. For RKBS we used the cubic and for RKDP the quatric interpolation implemented in scipy [36]. For all but the linear interpolation, where this step is trivial, we utilized scipy's "brentq" root finding algorithm to determine the time point t_{spike} when the threshold is reached. We set the relative tolerance of the algorithm to 1e−12 to ensure that the remaining error in the spike-time estimate is mostly driven by the uncertainty of the integration scheme and the interpolation rather than the root finding algorithm. The overhead introduced by the root finding was still relatively small ($\leq 30\,\mu\text{s}$) compared to the computational costs of a single ODE evaluation of the Izhikevich neural network ($\geq 200\,\mu\text{s}$). The spike-time t_{spike} and the evaluation of the dense output at t_{spike} were then used as the solution of this step. For the state perturbation method [14], this required an additional step to maintain the stochasticity also during steps resulting in a reset. For this, we used a modification $\hat{d}_{\text{RK}}$ of the dense outputs d_{RK} by adding the perturbation noise term $\boldsymbol{\xi}_{t_i}$ (see Eq. (16)) linearly weighted with the distance to the time before the step t_i:

$$\hat{d}_{\text{RK}}(t, t_i, t_{i+1}) = d_{\text{RK}}(t, t_i, t_{i+1}) + \frac{t - t_i}{t_{i+1} - t_i} \boldsymbol{\xi}_{t_i}, \tag{24}$$

which is a simplified version of the continuous-time output proposed by Conrad et al. [14].

We implemented the spike time estimation with intermediate steps both for adaptive and fixed step-sizes. We refer to the latter case as "pseudo-fixed" step-sizes, because at spike resets the step-size can vary, but it is otherwise fixed.

Neuron metrics and reference solutions

We quantify the numerical uncertainty in the neuron models with different metrics. In most cases, we compare the sample distributions from probabilistic solvers to a reference solution. These reference solutions were computed using a deterministic RKDP_a solver with a tolerance of $\kappa = 1\text{e}{-12}$ and a maximum step-size dependent on the model investigated (0.001 ms for HH, 0.01 ms for IN and the IN network and 0.1 ms for the STG model). Based on the computed samples and the respective reference solution, we evaluated the distributions of inter-sample distances and sample-reference distances. In some cases, we also computed the deterministic-reference distance, which is the distance between the solution from a deterministic solver and the reference solution. As a distance measure, we computed the Mean Absolute Error (MAE) between single traces. If not stated otherwise, MAEs were computed on the simulated membrane voltages, because this is typically the quantity of interest. We refer to the pairwise inter-sample distances as MAE_{SS}, the sample-reference distances as MAE_{SR}, and the deterministic-reference distance as MAE_{DR}. For N samples from a probabilistic solver, this resulted in $N(N-1)/2$ values for MAE_{SS}, N values for MAE_{SR}, and 1 value for MAE_{DR}.

To quantify the calibration of the perturbation methods, we computed MAE ratios R defined as $R = \overline{\text{MAE}}_{\text{SS}}/\overline{\text{MAE}}_{\text{SR}}$, where $\bar{\bullet}$ denotes the distribution mean. If the perturbation is too small, the average inter-sample distance $\overline{\text{MAE}}_{\text{SS}}$ is much smaller the average sample-reference distance $\overline{\text{MAE}}_{\text{SR}}$, and R is typically close to zero. If however, the ODE is evaluated exactly at a discontinuity (e.g. a sharp stimulus onset), arbitrarily small but greater-than-zero step-size perturbations may result in non-zero values for R. This is because the evaluation of the ODE happens just before or directly after the discontinuity even for very small perturbations. In the extreme case, where the perturbation parameter is zero, all samples are equal to the deterministic solution, and $R = 0$.

When the perturbation is increased, the average inter-sample distance $\overline{\text{MAE}}_{\text{SS}}$ is increased and $R > 0$. Empirically, we found that R typically converges to $\sqrt{2}$, when the

perturbation parameter is set to large values. We therefore defined the normalized ratio as $R_N = R/\sqrt{2}$. The convergence of R to $\sqrt{2}$ is expected, if the perturbation in every step is i.i.d. Gaussian noise and the standard deviation of the noise is much greater than the difference between the sample mean and true solution.

To compare the samples to the deterministic solution, we defined the ratio $R_D = \mathrm{MAE}_{\mathrm{DR}}/\overline{\mathrm{MAE}}_{\mathrm{SR}}$ which is close to 1 if the perturbation is too small to affect the model output (i.e. all samples are approximately equal to the deterministic solution) and converges to zero, if the perturbation is too large (i.e. the perturbation severely reduces the solver accuracy). If the perturbation increases the average solution accuracy, R_D can also take values greater than 1 for non-zero perturbation parameters.

We also looked at spike times and spike numbers. For INs, spike times were simply the time of the reset. For the other models, we utilized the dense outputs of the solvers to estimate spike times by estimating the times when the threshold, which we set to $v(t) = 0\,\mathrm{mV}$, was reached. The exponential integrators EE and EEMP were linearly interpolated for this purpose.

For the simulations of neuron networks, we also estimated the firing rates for either single neurons or—in the case of the IN network—for the network as a whole. Firing rates were estimated using a Gaussian kernel density estimate with a bandwidth of 20 ms and 0.1 ms for the STG neurons and the IN network, respectively.

We quantified the computational costs of a solution by counting the number of ODE evaluations $n(\mathrm{ODE})$. To separate the computational overhead of the perturbation from the comparison of different solver schemes, we subtracted the evaluations only necessary for the perturbation $n_{\mathrm{pert.}}(\mathrm{ODE})$ to obtain the number of evaluations a corresponding deterministic method $n_{\mathrm{det.}}(\mathrm{ODE}) = n(\mathrm{ODE}) - n_{\mathrm{pert.}}(\mathrm{ODE})$ would require. To normalize this, we divided by the simulated time T, which gives the number of evaluations per simulated millisecond $N_{\mathrm{det.}}(\mathrm{ODE}) = n_{\mathrm{det.}}(\mathrm{ODE})/T$. For example, for a step-size $\Delta t = 0.1\,\mathrm{ms}$ this would be $N_{\mathrm{det.}}(\mathrm{ODE}) = 10$ for $\mathrm{FE_f}$, while for $\mathrm{EEMP_f}$—which requires two ODE evaluations per step—it would be $N_{\mathrm{det.}}(\mathrm{ODE}) = 20$. It should be noted that $n(\mathrm{ODE})$ can be different for individual samples if the step-size is not fixed, and therefore we report $N_{\mathrm{det.}}(\mathrm{ODE})$ as empirical distributions in the following. The computational costs caused by the perturbation itself is discussed theoretically above, and was empirically quantified by measuring run times of probabilistic solvers and comparing them to the run times of the respective deterministic solvers.

Results

Numerical uncertainty in Izhikevich neurons with different dynamics

To demonstrate the effect of numerical uncertainty on simulations of mechanistic neuron models, we first simulated single INs with 18 different parametrizations θ_i and I_i, covering a wide range of response dynamics [23], using the original step-sizes Δt (Fig. 2). We computed the solution with the original solver, which is similar to a $\mathrm{FE_f}$ solver, and generated 40 samples with a probabilistic (state perturbation) $\mathrm{FE_f^x}$ solver. We found that for some parametrizations, the model solution was highly uncertain (e.g. Figs. 2P–2R) while it was relatively stable for others (e.g. Figs. 2A and 2K).

For a more quantitative analysis, we studied two IN models (Figs. 2N and 2P) in more detail and quantified the solution uncertainty for different solver settings. For the "rebound burst" IN (Figs. 3A–3F), we found that not only the precise spike times but also the number of spikes was uncertain when using a $\mathrm{FE_f^x}$ solver with a step-size equal to the original step-size $\Delta t = 0.2\,\mathrm{ms}$ (Fig. 3A) and even more so for a step-size of $\Delta t = 0.5\,\mathrm{ms}$ (Figs. 3D and 3E). Also the higher order solvers $\mathrm{RKBS^x}$ (3rd order) and

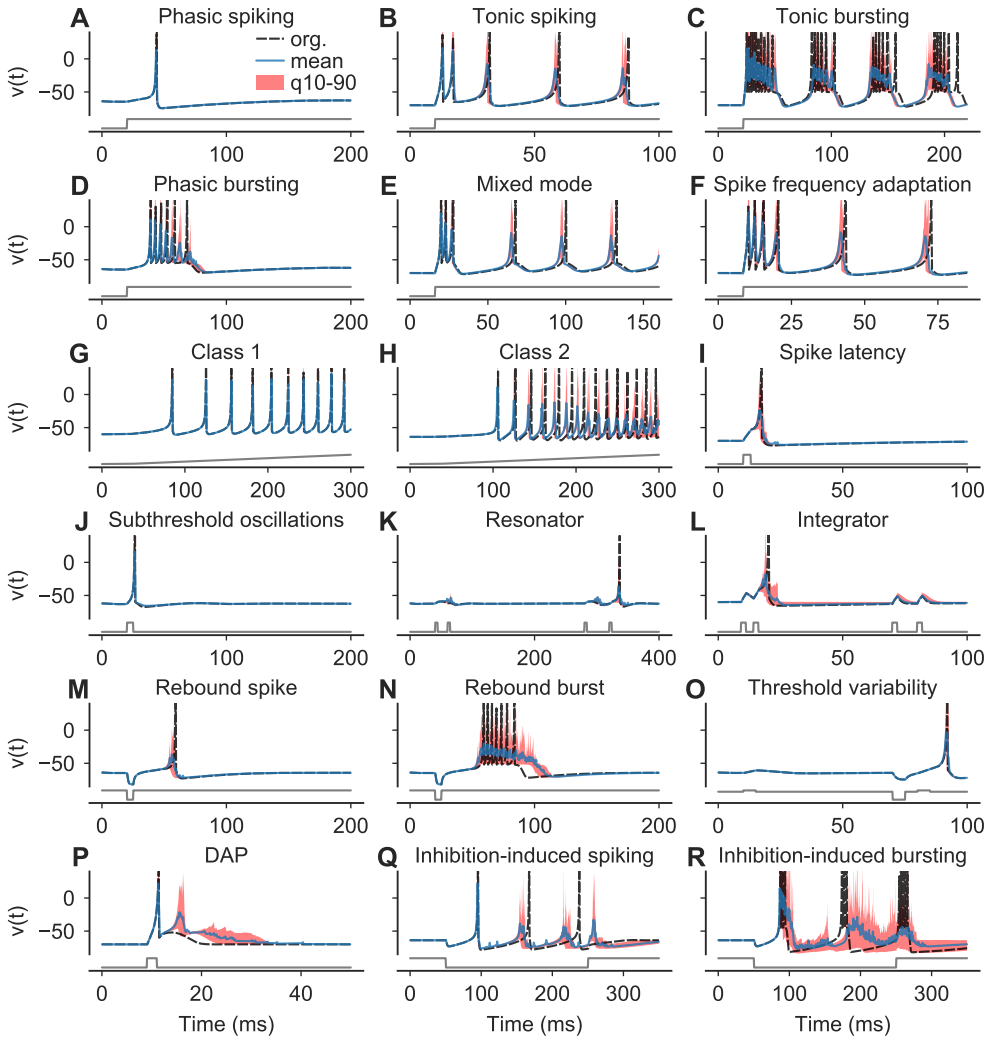

Fig 2. Neuron simulations can be subject to substantial numerical uncertainty. (A-R) Simulations of the IN for different parametrizations θ_i and I_{Stim}. Solutions were computed using the original solver (black dashed) and by drawing 40 samples from a probabilistic $\text{FE}_{\text{f}}^{\text{x}}$ method, summarized as a mean (blue) and 10th-90th percentiles (red). The original solver was similar to a FE solver, expect that $v(t)$ and $u(t)$ where updated sequentially, i.e. first $v(t+\Delta t)$ was computed, and then $u(t+\Delta t)$ was computed using the already updated value $v(t+\Delta t)$. For both solvers, the step-size Δt, which differs between parametrizations, was taken from the original implementation. Normalized stimuli I_{Stim} are shown in the bottom of every panel (gray lines). Solutions for $v(t)$ are clipped at 40. Solutions for $u(t)$ are not shown.

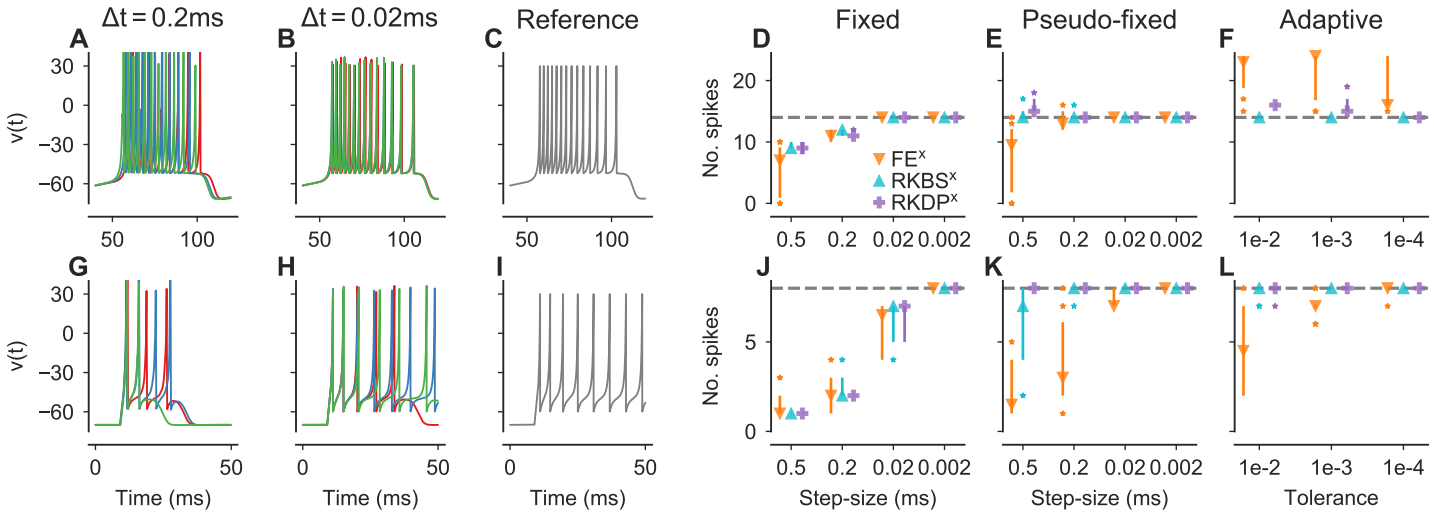

Fig 3. Numerical uncertainty in the spike number of Izhikevich neurons. (A,B) Three sample solutions computed with $\mathrm{FE}_{\mathrm{f}}^{\mathrm{x}}$ for $v(t)$ (clipped at 40) of the "rebound burst" IN (see also Fig. 2N) for a step-size of $\Delta t = 0.2\,\mathrm{ms}$ and $\Delta t = 0.02\,\mathrm{ms}$, respectively. **(C)** A reference solution. **(D-F)** Number of spikes dependent on the step-size / tolerance (x-axis) for different solver methods (legend) for fixed, pseudo-fixed and adaptive step-sizes, respectively. Number of spikes are shown for 40 samples each as means, 10th-90th percentiles (vertical lines) and outliers (stars). The dashed horizontal line refers to the reference solution. **(G-L)** As in (A-F), but for the "DAP" neuron.

$\mathrm{RKDP}^{\mathrm{x}}$ (5th order) (see Table 1) showed some—yet much weaker—variation in the numbers of spikes for larger step-sizes or tolerances (Figs. 3D–3F).

For the "DAP" IN, the number of spikes varied even more for different step-sizes (Figs. 3G–3L). While the original solver produced a single spike with a short depolarization afterwards, this behavior is at least partially an artifact of the solver, with 8 spikes likely corresponding to the true solution (Figs. 3I–3L). In most cases, the probabilistic solvers revealed this uncertainty indicated by a variation in simulated spike numbers (Figs. 3H–3J). However, for relatively large fixed step-sizes neither $\mathrm{RKBS}^{\mathrm{x}}$ nor $\mathrm{RKDP}^{\mathrm{x}}$ indicated any numerical uncertainty (Fig. 3J) even though the number of spikes was much smaller than in the reference solution. Since this effect was not observed for pseudo-fixed step-sizes (Fig. 3K), it is very likely that the discrepancy in the number of spikes for fixed step-sizes is an artifact of the resets—which are restricted to take place during multiples of the step-size—rather than the numerical integration itself.

Numerical uncertainty in a network of Izhikevich neurons

Next we turned to an IN neural network with an excitatory and an inhibitory population, which has been solved with a FE-like solver in the original publication with step-sizes of $\Delta t = 0.5\,\mathrm{ms}$ and $\Delta t = 1\,\mathrm{ms}$ for $v(t)$ and $u(t)$, respectively [20]. Simulating the network with a probabilistic FE_{f} solver with a step-size of $\Delta t = 0.5\,\mathrm{ms}$ revealed large numerical uncertainty in the number and timing of spikes of single neurons (Figs. 4A–4C) and in the firing rate of the network as a whole (Fig. 4D).

To better understand the reasons of the large uncertainty within the network, we generated solutions with different solvers and step-sizes (Fig. 5). We simulated the network with two stimuli that represent input to the network from a different brain area [20]: a step stimulus that sharply changes its amplitude every millisecond for every neuron and therefore introduces frequent discontinuities in the ODE, and a smooth version of the same stimulus computed by fitting a cubic spline. Comparing both

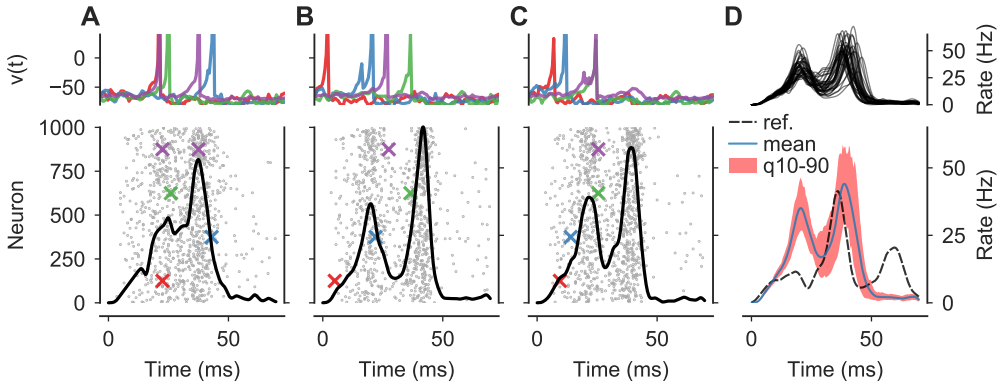

Fig 4. IN network model. (A-C) Simulation with a probabilistic $\mathrm{FE}_{\mathrm{f}}^{\mathrm{x}}$ with $\Delta t = 0.5\,\mathrm{ms}$, which is comparable to the solver originally used. Top: $v(t)$ of four example neurons (clipped at 40). Bottom: Spike raster plots of all 1000 neurons (left y-axis) and the estimated (see Methods) total firing rates (black-lines, right y-axis). Colored crosses highlight the spikes of the example neurons. **(D)** Top: The total firing rate estimates for 40 samples of the network. Bottom: Mean (blue) and 10th-90th percentiles (red) of the 40 samples and a reference solution (black dashed).

stimuli allowed us to analyze the influence of the discontinuities on the numerical uncertainty. To quantify the numerical uncertainty, we generated samples using probabilistic solvers and compared them to a reference solution. For every solution, we estimated the total firing rate (using a kernel density estimate) and computed the Mean Absolute Error (MAE) between the firing rates of samples and the reference to obtain an empirical distribution of the sample-reference distance $\mathrm{MAE}_{\mathrm{SR}}$.

For the step stimulus and fixed step-sizes, using smaller step-sizes decreased the uncertainty, but the step-size had to be drastically decreased to consistently obtain firing rates matching the reference solution (examples in Figs. 5A–5C and summary in Fig. 5G). Interestingly, the benefit from higher order methods was relatively small in this case, likely because of the errors during resets which are of order $O(\Delta t)$. Also pseudo-fixed step-sizes, i.e. allowing intermediate steps during spike resets, yielded only small improvements in the numerical uncertainty (examples in Figs. 5D–5F and summary in Fig. 5G).

Simulating the network with the smooth stimulus changed very little in the numerical uncertainty for fixed-step sizes (Fig. 5H). In contrast, for this stimulus higher-order solvers such as $\mathrm{RKBS}^{\mathrm{x}}$ and $\mathrm{RKDP}^{\mathrm{x}}$ in combination with small pseudo-fixed step-sizes were able to reduce the numerical uncertainty in the network to almost zero (Fig. 5H).

While higher order methods in combination with pseudo-fixed step-sizes can thus decrease numerical uncertainty, it is not clear a priori from the point of computational complexity, whether one should resort to these approaches or rather use a method like FE with small computational costs per step in combination with small step-sizes to reduce numerical uncertainty. We therefore also measured the computational costs of individual samples and compared them across solver setting for both the step (Fig. 6A) and smooth stimulus (Fig. 6B) with respect to the numerical uncertainty. We estimated the numerical uncertainty with probabilistic solvers as the sample-reference distances $\mathrm{MAE}_{\mathrm{SR}}$ and quantified the computational costs as the number of ODE evaluations corresponding deterministic solvers would require per millisecond of simulated time $N_{\mathrm{det.}}(\mathrm{ODE})$ (see Methods).

For the step stimulus and fixed step-sizes (Fig. 6A), we found only relatively small differences across the tested Runge-Kutta-type methods regarding the trade-off between numerical uncertainty and computational costs, with the higher order methods RKBS

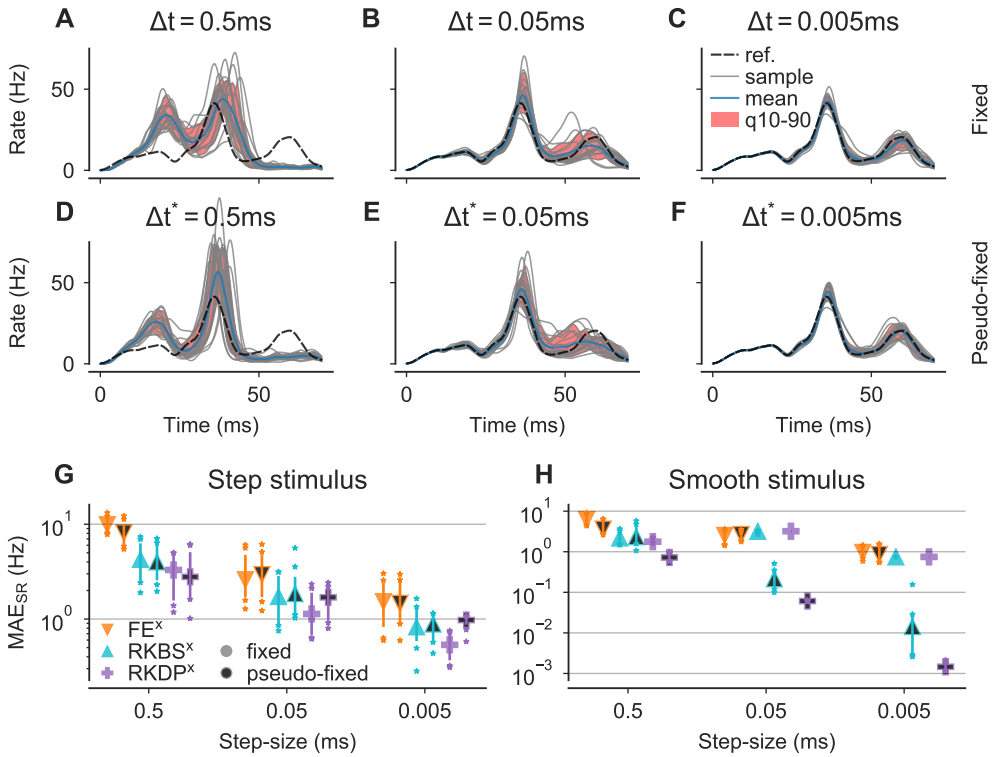

Fig 5. Numerical uncertainty in the Izhikevich neuron network with fixed and pseudo-fixed step-sizes. (A-C) Simulations of the IN network model computed by drawing 20 samples (gray) from a $\mathrm{RKBS}_{\mathrm{f}}^{\mathrm{x}}$ solver for different step-sizes (titles) and with a reference solver (black dashed). Traces represent kernel density estimates of the total firing rate. Samples are summarized as means (blue) and 10th-90th percentiles (red). **(D-F)** As in (A-C), but for pseudo-fixed step-sizes that allow for intermediate steps when spike resets occur. **(G,H)** Sample-reference distances $\mathrm{MAE}_{\mathrm{SR}}$ between firing rates of the reference solution and the samples of different solver schemes (left legend) as a function of the step-size Δt for both fixed and pseudo-fixed step-sizes (right legend), for the step and smooth stimulus, respectively.

and RKDP, being slightly more efficient. Using pseudo-fixed step-sizes instead of fixed step-sizes, increased the computational costs substantially for larger step-sizes without reducing the numerical uncertainty (Fig. 6A). This increase in computational costs for pseudo-fixed step-sizes resulted from an increased number of steps in the solutions. Every spike reset required an additional integration step, which is substantial for larger step-size. For example, for $\Delta t = 0.5\,\mathrm{ms}$ and the total simulated time $T = 70\,\mathrm{ms}$, more than 800 evaluations caused by spikes are added to only 140 evaluations of a corresponding fixed step-size method. Adaptive step-sizes showed the same large offset in computational costs for small tolerances—due to the same reason—and were only efficient in reducing the numerical uncertainty for relatively small tolerances and therefore computationally expensive solutions.

For the smooth stimulus (Fig. 6B), we again found fixed-step sizes to be most efficient if relatively large numerical uncertainties were allowed. Interestingly, for $\mathrm{RKBS}_{\mathrm{f}}^{\mathrm{x}}$ and $\mathrm{RKDP}_{\mathrm{f}}^{\mathrm{x}}$ the accuracy of the solutions decreased when the step-size was decreased from $\Delta t = 0.5\,\mathrm{ms}$ to $\Delta t = 0.05\,\mathrm{ms}$. Similar to the case of the single "DAP" IN (see Fig. 3J) this was likely an artifact of the fixed step-sizes, constraining the spikes to take place on a fixed time grid rather than the inaccurate integration between spikes,

because for pseudo-fixed step-sizes the solutions with $\Delta t 0.05$ ms showed substantially lower numerical uncertainty. To obtain small numerical uncertainties, both RKBS and RKDP in combination with either adaptive or especially pseudo-fixed were much more efficient than fixed step-sizes (Fig. 6B).

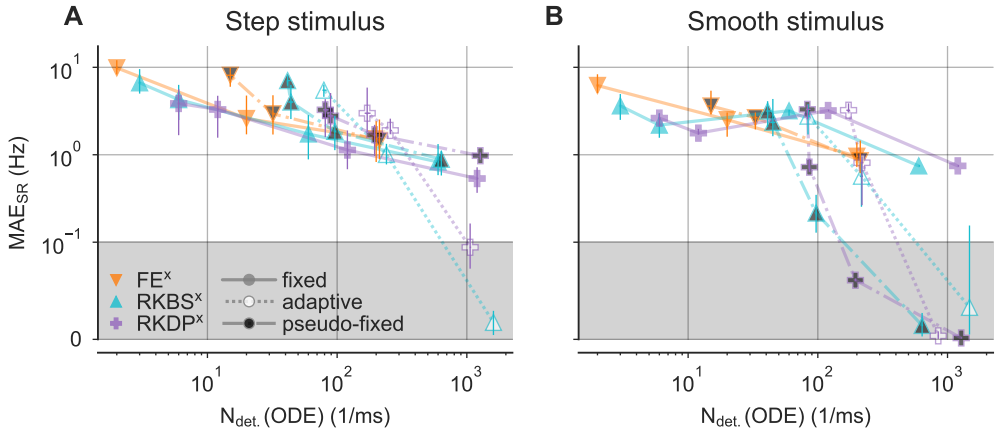

Fig 6. Uncertainty in the IN network model as function of the computational costs. (A) Sample-reference distances $\mathrm{MAE_{SR}}$ between firing rates of a reference solution and samples of probabilistic solvers for the IN network simulated with the step stimulus. $\mathrm{MAE_{SR}}$ distributions are shown for different solver schemes (left legend) with fixed, adaptive and pseudo-fixed step-sizes (right legend) as a function of the computational costs per samples. Computational costs are shown as the number of ODE evaluations a respective deterministic solver would require per millisecond of simulated time $N_{\mathrm{det.}}(\mathrm{ODE})$ (see Methods). $\mathrm{MAE_{SR}}$ and $N_{\mathrm{det.}}(\mathrm{ODE})$ distributions are shown as medians and 10th-90th percentiles. $\mathrm{MAE_{SR}}$ values smaller than 1e−3 (gray background) are shown on a linear scale to highlight the interesting parts. Step-sizes Δt ranged from 0.5 ms for FE and 1 ms for RKBS and RKDP to 0.005 ms; tolerances κ from 1e−2 to 1e−8. **(B)** As in (A), but for the smooth stimulus.

Numerical uncertainty in single Hodgkin-Huxley neurons

We simulated the classical HH neuron with different probabilistic solvers for two stimuli and quantified the numerical uncertainty as a function of the computational costs. We quantified the numerical uncertainty again as $\mathrm{MAE_{SR}}$, but also as the absolute spike-time error of the 7th spike, which was the last spike in the reference solution, for both stimuli. The first stimulus was a current step, with a sharp on- and offset (Fig. 7A), which is a typically used stimulus type. As a second stimulus, we chose a noisy current step with a smooth on- and offset, to emulate noisy input from e.g. another neuron (Fig. 7B).

For both, the step and the noisy stimulus, we found that the exponential integrators $\mathrm{EE_f^x}$ and $\mathrm{EEMP_f^x}$ could terminate with relatively large step-size (up to $\Delta t = 0.5$ ms) and the fewest number of ODE evaluations (as low as $N_{\mathrm{det.}}(\mathrm{ODE}) = 4\,\mathrm{ms}^{-1}$) without becoming numerically unstable (Figs. 7C–7F, respectively). However, when using such large step-sizes the numerical uncertainty was substantial and the discrepancy in the number of spikes between samples and the reference did not allow us to quantify the spike-time error of the 7th spike (Figs. 7E–7F, respectively). For more ODE evaluations, the exponential integrators scaled relatively poorly, with $\mathrm{EEMP_f^x}$ (2nd order) clearly outperforming $\mathrm{EE_f^x}$ (1st order) (Figs. 7C–7F).

For the step stimulus, even the most basic method $\mathrm{FE}_{\mathrm{f}}^{\mathrm{x}}$ outperformed both exponential integrators, if a sufficient number of steps per millisecond was chosen ($N_{\mathrm{det.}}(\mathrm{ODE}) \geq 25\,\mathrm{ms}^{-1}$) to not result in numerical instabilities (Figs. 7C and 7D). For the noisy stimulus, $\mathrm{FE}_{\mathrm{f}}^{\mathrm{x}}$ (1st order) was still much more efficient than $\mathrm{EE}_{\mathrm{f}}^{\mathrm{x}}$ and comparable to $\mathrm{EEMP}_{\mathrm{f}}^{\mathrm{x}}$, despite the method's lower order.

The fixed step-size higher order Runge-Kutta methods $\mathrm{RKBS}_{\mathrm{f}}^{\mathrm{x}}$ and $\mathrm{RKDP}_{\mathrm{f}}^{\mathrm{x}}$ required even more steps per millisecond to terminate ($N_{\mathrm{det.}}(\mathrm{ODE}) \geq 50\,\mathrm{ms}^{-1}$). However, if the step-size was chosen sufficiently small the solutions had low numerical uncertainties for the step stimulus (Figs. 7C and 7E), which were even lower for the noisy stimulus (Figs. 7D and 7F).

When using adaptive step-sizes, the Runge-Kutta methods were able to terminate for substantially fewer ODE evaluations, with $\mathrm{RKBS}_{\mathrm{f}}^{\mathrm{x}}$ needing the fewest ($N_{\mathrm{det.}}(\mathrm{ODE}) \approx 12\,\mathrm{ms}^{-1}$), and $\mathrm{RKCK}_{\mathrm{f}}^{\mathrm{x}}$ being slightly cheaper ($N_{\mathrm{det.}}(\mathrm{ODE}) \approx 17\,\mathrm{ms}^{-1}$) than $\mathrm{RKDP}_{\mathrm{f}}^{\mathrm{x}}$ ($N_{\mathrm{det.}}(\mathrm{ODE}) \approx 22\,\mathrm{ms}^{-1}$). For smaller tolerances and therefore more ODE evaluations, all adaptive Runge-Kutta were highly efficient in reducing the numerical uncertainty. In this regime they were clearly superior to all lower order methods. For example, for a typical number of $N_{\mathrm{det.}}(\mathrm{ODE}) \approx 100\,\mathrm{ms}^{-1}$ (i.e. $\Delta t = 0.01\,\mathrm{ms}$ for a single-stage method like $\mathrm{EE}_{\mathrm{f}}^{\mathrm{x}}$), $\mathrm{RKBS}_{\mathrm{f}}^{\mathrm{x}}$ was more than an order of magnitude more accurate than $\mathrm{EE}_{\mathrm{f}}^{\mathrm{x}}$ ($\overline{\mathrm{MAE}}_{\mathrm{SS}} < 0.02\,\mathrm{mV}$ vs. $\overline{\mathrm{MAE}}_{\mathrm{SS}} > 0.1\,\mathrm{mV}$) for both stimuli (Figs. 7C and 7D). For the step stimulus, where the sharp stimulus onset induced some numerical uncertainty for fixed step-size Runge-Kutta methods, the adaptive stepping also proved beneficial because the step-sizes during the discontinuities were very small (Figs. 7C and 7E). For the smooth noisy stimulus on the other hand, adaptive and fixed step-size Runge-Kutta methods performed approximately equally well.

Next we turned to the STG model and stimulated the response of a single neuron to a current step stimulus (Fig. 8) based on the original publication [25]. Here, we compared the numerical uncertainty in different state variables, namely the voltage $v(t)$ (Fig. 8A), the intracelluar calcium $\mathrm{Ca}(t)$ (Fig. 8B) and a subunit of the sustained calcium channel $h_S(t)$ (Fig. 8C), respectively. We found that the numerical uncertainty differed strongly between these state variables, and was highest for $v(t)$ (Fig. 8D). 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.

Numerical uncertainty in the STG neuron network

To also provide an example of numerical uncertainty quantification for a network consisting of connected Hodgkin-Huxley-like neurons, we simulated the STG network using five different parametrizations (Fig. 9) from the original publication [26]. Overall, we found that the spiking patterns were qualitatively similar between different samples when using a $\mathrm{EE}_{\mathrm{f}}^{\mathrm{x}}$ solver with a step-size of $\Delta t = 0.1$ (Figs. 9A–9E). However, when looking at exact spike-times we found relatively large inconsistencies between samples shifting single spikes and bursts by hundreds of milliseconds (Figs. 9F–9J). Reducing the step-size from $\Delta t = 0.1\,\mathrm{ms}$ to $\Delta t = 0.01\,\mathrm{ms}$ strongly reduced the numerical uncertainty in the firing rate of the neurons (Figs. 9K–9O vs. Figs. 9P–9T) and consistently yielded firing rates close to the reference solution.

Again, we compared the numerical uncertainty for different solvers (Figs. 9U–9Y). Similar to the single HH neuron, the exponential integrators $\mathrm{EE}_{\mathrm{f}}^{\mathrm{x}}$ and $\mathrm{EEMP}_{\mathrm{f}}^{\mathrm{x}}$ were able to solve the ODE with the fewest number of ODE evaluations (as low as $N_{\mathrm{det.}}(\mathrm{ODE}) = 4\,\mathrm{ms}^{-1}$ and consistently for $N_{\mathrm{det.}}(\mathrm{ODE}) \geq 10\,\mathrm{ms}^{-1}$) without becoming numerically unstable. The adaptive Runge-Kutta methods (especially $\mathrm{RKBS}_{\mathrm{a}}^{\mathrm{x}}$), again

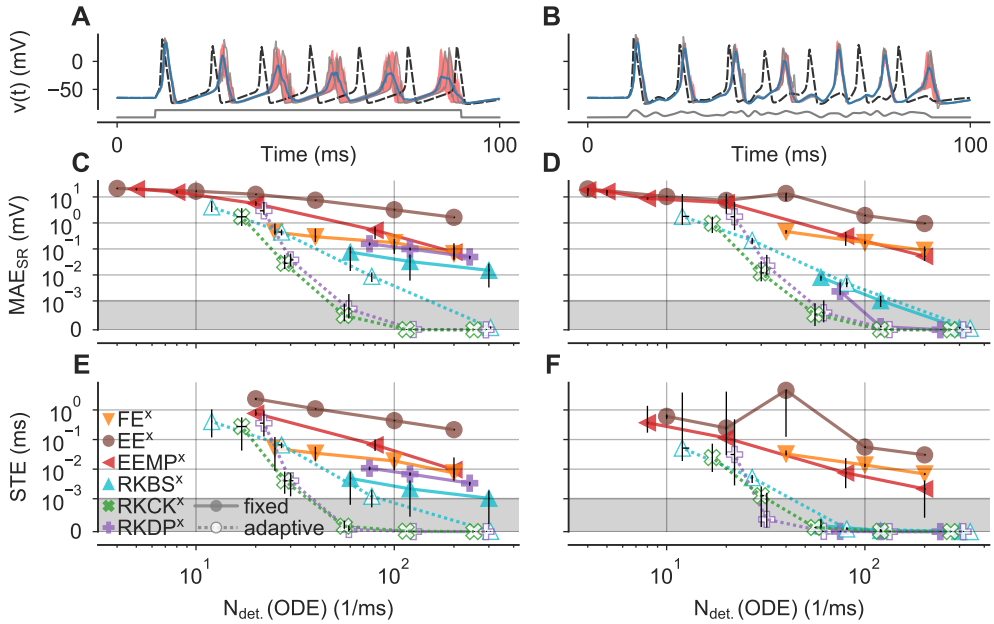

Fig 7. Numerical uncertainty vs. computational costs for the Hodgkin-Huxley neuron. (A,B) Simulations of the classical HH neuron for the step and noisy stimulus, respectively. Solutions were computed using a reference solver (black dashed) and by drawing 20 samples from a probabilistic $\mathrm{EE}_{\mathrm{f}}^{\mathrm{x}}$ method with $\Delta t = 0.25\,\mathrm{ms}$, summarized as a mean (blue) and 10th-90th percentiles (red) and two samples (gray). Normalized stimuli I_{Stim} are shown in the bottom of the panels. **(C,D)** $\mathrm{MAE}_{\mathrm{SR}}$ between voltage traces of samples generated with probabilistic solvers (see legend) and a reference solution. $\mathrm{MAE}_{\mathrm{SR}}$ values are shown as a function of the number of ODE evaluations corresponding deterministic solvers would require per millisecond of simulated time. Data is shown as medians (symbols) and 10th-90th percentiles (lines). y-values smaller than 1e−3 (gray background) are shown on a linear scale to highlight the interesting parts. Step-sizes Δt ranged from 0.5 ms to 0.005 ms; tolerances κ from 1e−2 to 1e−10. Step parameters that caused that simulations to become numerically unstable were removed. Combinations of methods and step parameters that would be computationally more expensive than the shown range (e.g. RKDP with $\Delta t = 0.005\,\mathrm{ms}$ or RKBS with $\kappa = 1\mathrm{e}{-10}$) were either not simulated or removed. **(E,F)** Similar as (C,D), but for the absolute errors of the spike-times of the 7th spike relative to the reference solutions (STE). Solvers yielding samples with different number of spikes are not shown.

performed well in achieving very low numerical uncertainty with respect to their computational costs. This might be surprising considering that in the network there was always at least one neuron (close to) spiking and the periods between spikes were all relatively brief, making the step-size adaptation less efficient compared to a single neuron with longer periods of silence.

Calibration of probabilistic solvers

To meaningfully apply the probabilistic solvers used in this study in practice, the perturbation parameter of these methods has to be sufficiently calibrated. If the perturbation is too small, the numerical uncertainty is underestimated. If it is too large, the solution accuracy suffers. Only if the probabilistic solver is calibrated does the

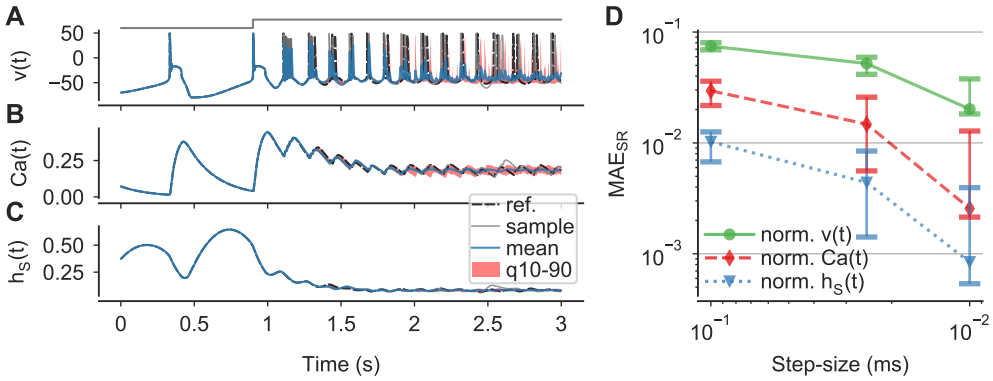

Fig 8. Numerical uncertainty of different state variables for a Hodgkin-Huxley-like STG neuron as a function of the step-size. (A-C) Simulations of the single STG neuron computed using a reference solver (black dashed) and by drawing 40 samples from a probabilistic $\mathrm{EE}_{\mathrm{f}}^{\mathrm{x}}$ method with a step-size of 0.1 ms. Samples are summarized as means (blue) and 10th-90th percentiles (red) for the membrane voltage $v(t)$, the intracellular calcium $\mathrm{Ca}(t)$ and a subunit of the sustained calcium channel $h_S(t)$. The normalized step stimulus I_{Stim} is shown at the top. **(D)** Sample-reference distances $\mathrm{MAE}_{\mathrm{SR}}$ between normalized reference and sample traces for the state variables (legend) in (A-C) as a function of the step-size, shown as medians (symbols) and 10th-90th percentiles (vertical lines). $\mathrm{MAE}_{\mathrm{SR}}$ were computed on normalized traces to facilitate the comparison. For this, first all reference traces were linearly transformed to be between zero and one and then the same linear transformations were applied to the respective samples traces.

sample distribution provide useful information about the numerical uncertainty of the solution. In this case, the inter-sample distance can be used as an approximate measure for the distance between samples and the true solution.

To analyze how the magnitude of the perturbation affects the distributions of inter-sample distances $\mathrm{MAE}_{\mathrm{SS}}$ and sample-reference distances $\mathrm{MAE}_{\mathrm{SR}}$, we simulated the classical HH neuron with a probabilistic $\mathrm{EEMP}_{\mathrm{f}}$ solver for the three perturbation strategies and different perturbation parameters σ (Fig. 10). For the state perturbation, we found that for $\sigma = 0.1$ the inter-sample distance was on average much smaller than the sample-reference distance with all sample-reference distances being tightly distributed around the deterministic-reference distance (Fig. 10A). In this case, the perturbation was too small and all samples were approximately equal to the deterministic solution.

For the default perturbation parameter $\sigma = 1$ on the other hand, the two distributions $\mathrm{MAE}_{\mathrm{SS}}$ and $\mathrm{MAE}_{\mathrm{SR}}$ were largely identical and more broadly distributed around the deterministic-reference distance $\mathrm{MAE}_{\mathrm{DR}}$ (Fig. 10B). This shows, that for $\sigma = 1$ the perturbation was well calibrated with no significant loss of solver accuracy due to the perturbation.

For a further increase of the perturbation parameter to $\sigma = 10$, we observed that, while the two distributions $\mathrm{MAE}_{\mathrm{SS}}$ and $\mathrm{MAE}_{\mathrm{SR}}$ were still largely overlapping, the sample-reference distances were on average much larger than the deterministic-reference distance, showing a loss of accuracy caused by the too large perturbation.

We summarized these findings and additional results for other perturbation parameters as R_N, R_D and their product $R_N R_D$ (see Figs. 10D–10F, respectively). R_N was defined as the normalized ratio between the mean inter-sample distance and the mean sample-reference distance, which should be $R_N \gg 0$ for a calibrated solution. R_D was defined as the ratio between the deterministic-reference distance and the mean

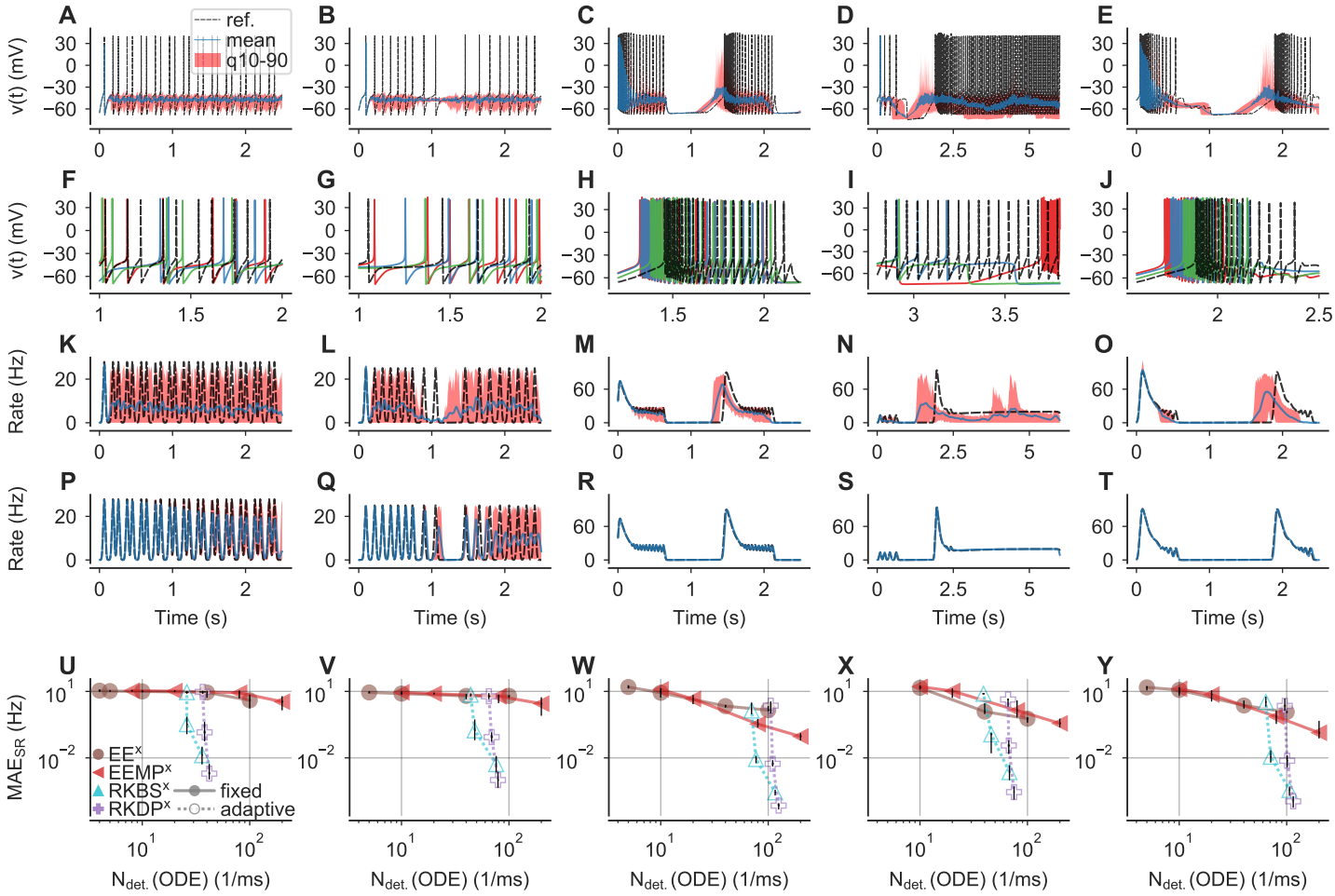

Fig 9. Numerical uncertainty of the PY neuron in the STG network for different synaptic parametrizations. **(A-E)** Membrane voltage traces $v(t)$ of the PY neuron generated by simulating the three neuron STG network for different synaptic parametrizations. Solutions were computed using a reference solver (black dashed) and by drawing 40 samples from a probabilistic $\mathrm{EE}_{\mathrm{f}}^{\mathrm{x}}$ method with a step-size of 0.1 ms. Samples are summarized as means (blue) and 10th-90th percentiles (red). **(F-L)** As in (A,E), but zoomed in and instead of the sample summary three samples are shown (colored solid lines). **(K-O)** As in (A,E), but instead of the voltage $v(t)$, the estimated firing rate is shown. **(P-T)** As in (K-O), but for a step-size of $\Delta t = 0.01$ ms. **(U-Y)** Sample-reference distances $\mathrm{MAE}_{\mathrm{SR}}$ between estimated firing rates of the reference and samples from different solvers (legend). $\mathrm{MAE}_{\mathrm{SR}}$ are shown as a function of the number of ODE evaluations per millisecond of simulated time for a corresponding deterministic solver $N_{\mathrm{det.}}(\mathrm{ODE})$. Data is shown as medians (symbols) and 10th-90th percentiles (black lines). Step-sizes Δt ranged from 0.5 ms to 0.01 ms; tolerances κ from 1e−3 to 1e−7. Step-sizes that caused that simulations to become numerically unstable were removed.

sample-reference distance, which for $R_D \ll 1$ indicates a loss of accuracy caused by the perturbation. Since neither of the two ratios R_N and R_D is sufficient to judge the calibration of the perturbation, we also computed the product $R_N R_D$, which can be used as an approximate measure of calibration that should be maximized.

For the state perturbation and $\sigma = 1$, we found that all conditions for a well calibrated perturbation—$R_N \gg 0$ (Fig. 10D), $R_D \not\ll 1$ (Fig. 10E), and maximized $R_N R_D$ (Fig. 10F)—were full-filled.

We did the same analysis for the step-size perturbation for both the uniform (Figs. 10G–10L) and log-normal distribution (Figs. 10M–10R). For both distributions,

we found that the default parameter $\sigma = 1$ was too small (Figs. 10G and 10N, respectively), whereas $\sigma = 10$ yielded calibrated solutions (Figs. 10I and 10O, respectively). Compared to the state perturbation, the ratios R_N and R_D, and the product $R_N R_D$ showed similar dependencies on $\log(\sigma)$, but were shifted to the right (Figs. 10J–10L and Figs. 10P–10R, respectively).

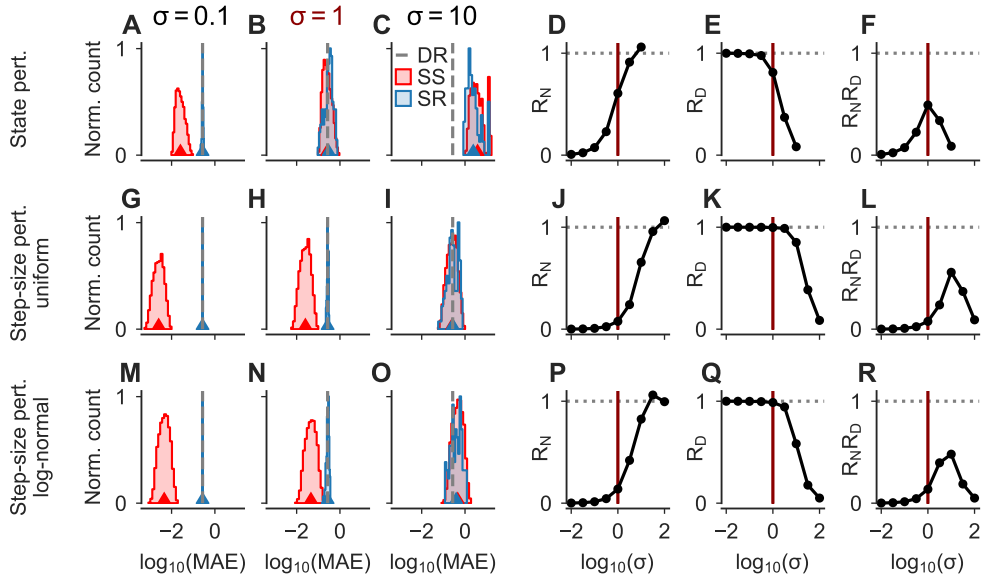

Fig 10. MAE distributions for different perturbation parameters. (A-C) Empirical distributions of $\mathrm{MAE}_{\mathrm{SS}}$ and $\mathrm{MAE}_{\mathrm{SR}}$ for the voltage traces of the classical HH neuron with the noisy stimulus current I_{Stim}. Solutions were computed with a $\mathrm{EEMP}_{\mathrm{f}}^{\mathrm{x}}$ solver ($\Delta t = 0.01\,\mathrm{ms}$, 100 samples), a deterministic version of this solver, and a reference solver. Normalized $\mathrm{MAE}_{\mathrm{SS}}$ (red) and $\mathrm{MAE}_{\mathrm{SR}}$ (blue) distributions are shown with highlighted means (triangles) for different perturbation parameters σ (see titles, shown as $\log_{10}(\sigma)$). For reference, $\mathrm{MAE}_{\mathrm{DR}}$ is shown as well (gray dashed line). **(D)** Normalized MAE ratios $R_N = \overline{\mathrm{MAE}}_{\mathrm{SS}}/\overline{\mathrm{MAE}}_{\mathrm{SR}}/\sqrt{2}$ as a function of the perturbation parameter, where the default perturbation parameter $\sigma = 1$ is highlighted (red vertical line). **(E)** As in (D), but for the sample-reference vs. deterministic-reference distance ratio $R_D = \overline{\mathrm{MAE}}_{\mathrm{SS}}/\mathrm{MAE}_{\mathrm{DR}}$. **(F)** As in (D), but for the ratio product $R_N R_D$. **(G-L)** As in (A-F), but for the state perturbation using a uniform perturbation distribution, i.e. using the solver $\mathrm{EEMP}_{\mathrm{f}}^{\mathrm{t}}$. **(M-R)** As in (A-F), but for the state perturbation using a log-normal perturbation distribution, i.e. using the solver $\mathrm{EEMP}_{\mathrm{f}}^{\mathrm{t}}$.

To quantify the calibration of the three perturbation strategies more generally, we did the same analysis shown in Fig. 10 for different solvers and four different stimuli, and summarized the results as the MAE ratio products $R_N R_D$ (Fig. 11). We chose to simulate a constant stimulus (as a simple baseline), a step stimulus (which are often used in practice), a noisy step stimulus (a quickly changing input without any discontinuities to represent the input from e.g. another neuron) and a subthreshold stimulus (to analyze the uncertainty in the absence of spiking).

For EE_{f}, we found that the state perturbation could not be easily calibrated. For the larger step-size $\Delta t = 0.25\,\mathrm{ms}$ (Fig. 11A, first row) the perturbation was either too small, or the perturbation was so large it resulted in highly distorted solutions and/or numerical instabilities. Only for the subthreshold stimulus, which was less prone to numerical instabilities due to the absence of spiking, the perturbation for $\sigma = 10^{0.5}$ was relatively well calibrated ($R_N R_D = 0.52$). For the smaller step-size $\Delta t = 0.025\,\mathrm{ms}$

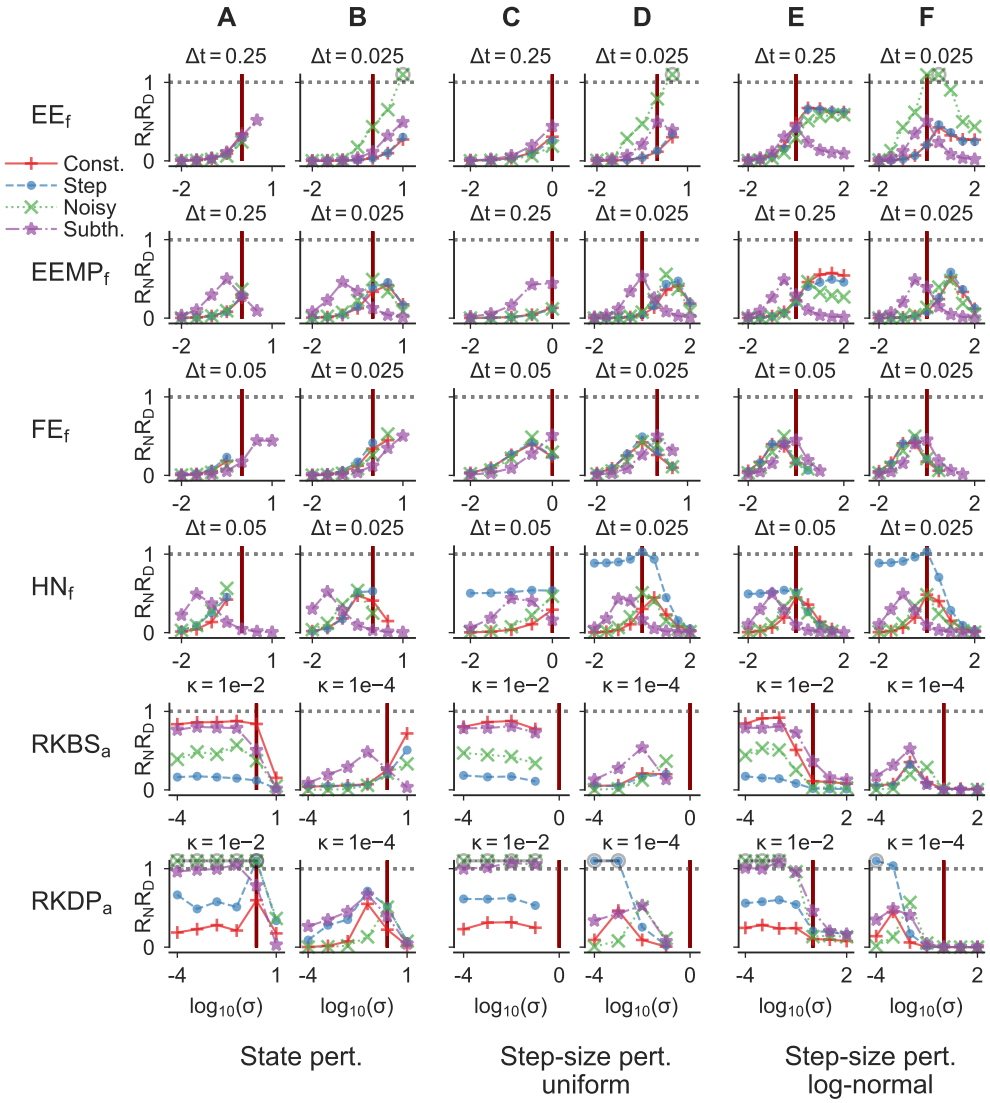

Fig 11. Calibration of the perturbation methods for a Hodgkin-Huxley neuron. (A,B) MAE ratio product $R_N R_D$ as an approximate measure for calibration for 100 samples of different probabilistic solvers (rows, see titles on the left) using the state perturbation method. Samples were computed for the classical HH neuron for four stimuli (legend), different step parameters dependent on the method (panel titles; Δt in milliseconds), and different perturbation parameters σ (x-axis, as $\log_{10}(\sigma)$). The default perturbation parameter $\sigma = 1$ (red vertical line) and $R_N R_D = 1$ (gray horizontal line) are highlighted. The minimum and maximum perturbation parameters tested are shown as x-ticks. Data for parameters resulting in numerical instabilities is not shown. $R_N R_D > 1.1$ are clipped at 1.1 and highlighted (black circles around symbols). **(C,D)** As in (A,B), but for the step-size perturbation method using a uniform perturbation distribution. For all methods, the maximum perturbation parameter σ was limited by the (maximum) step-size and the method order to ensure only positive step-sizes. Therefore, for adaptive step-sizes, where we set the maximum step-size to 1 ms, only perturbation parameters $\sigma < 1$ were simulated. **(E,F)** As in (A,B), but for the step-size perturbation method using a log-normal perturbation distribution.

(Fig. 11B, first row), the solutions looked generally less distorted and a perturbation parameter of $\sigma = 1$ resulted in acceptable calibration for all stimuli. For the noisy stimulus, the perturbation even increased the solution accuracy on average substantially ($\overline{\mathrm{MAE}}_{\mathrm{SR}} = 7.7$ vs. $\mathrm{MAE}_{\mathrm{DR}} = 13.6$). We found quantitative similar calibration results for the uniform step-size perturbation for both the larger (Fig. 11C, first row) and smaller step-size (Fig. 11D, first row) with generally less distorted solutions compared to the state-perturbation. For the log-normal step-size perturbation, which in contrast to the uniform allows for larger perturbation parameters while ensuring only positive step sizes, the method was best calibrated for $\sigma \approx 1$, with $\sigma = 10^{0.5}$ being slightly better for the spiking inducing stimuli. For much larger perturbation parameters, the solutions looked highly distorted again.

For $\mathrm{EEMP_f}$, the most striking difference was observed between the spike inducing stimuli and the subthreshold stimulus (Fig. 11, second row), with only small differences between the spike inducing stimuli. For all perturbation strategies and both tested step-sizes, the best calibration for the subthreshold stimulus was achieved with perturbation parameters approximately an order of magnitude smaller, ranging from 0.1 to 1, compared to the best parameter for the other stimuli, which ranged between 1 for the state perturbation (Figs. 11A and 11B, second row) and 10 for the step-size perturbation (Figs. 11C and 11F, second row).

For $\mathrm{FE_f}$, the state perturbation could not be calibrated for the spiking inducing stimuli when using a larger step-size ($\Delta t = 0.05\,\mathrm{ms}$) because of arising numerical instabilities (Fig. 11A, third row). For the smaller step-size ($\Delta t = 0.025\,\mathrm{ms}$) the calibration was acceptable for the default $\sigma = 1$ for spike inducing stimuli, but could be improved with slightly larger ($\sigma = 10^{0.5}$) values (Fig. 11B, third row). For the subthreshold stimulus, an even larger value could be used to further improve the calibration ($\sigma = 10$). For the step-size perturbation, we found that for both the uniform and log-normal distribution the calibration was ideal for perturbation parameters close to the default (ranging from $\sigma = 10^{-0.5}$ to $\sigma = 1$).

For $\mathrm{HN_f}$, the calibration was strongly dependent on the stimulus, but expect for the step stimulus the differences between the different perturbation strategies were small (Fig. 11, fourth row). For the subthreshold stimulus, the best perturbation parameter (ranging from $\sigma = 10^{-1.5}$ to $\sigma = 0.1$) was—similar to $\mathrm{EEMP_f}$—again much lower than for the spike-inducing stimuli. For the constant and noisy step stimulus, the best perturbation parameter were all close to the default (ranging from $\sigma = 10^{-0.5}$ to $\sigma = 10^{0.5}$), which was also the case for the step stimulus when using state perturbation. However, for the step stimulus in combination with the step-size perturbation the perturbation was well calibrated for all tested perturbation parameters larger than zero and smaller or equal to one (Figs. 11C–11F, fourth row). This was an effect of the sharp stimulus onset at $t_{\mathrm{onset}} = 10\,\mathrm{ms}$. The last stage of the HN method evaluates the ODE at the endpoint ($t_{i+1} = t_i + \Delta t$) of the current step. Therefore, even tiny perturbations in the step-size during the step from $t_i = t_{\mathrm{onset}} - \Delta t$ to $t_{i+1} = t_{\mathrm{onset}}$ result in a $\approx 50\%$ chance of the stimulus being or not being active when the ODE was evaluated at the last stage of this step. In this special case, this was sufficient to improve the average sample accuracy while also having a non-zero mean inter-sample distance.

For the adaptive step-size Runge-Kutta methods $\mathrm{RKBS_a}$ (Fig. 11, next-to-last row) and $\mathrm{RKDP_a}$ (Fig. 11, last row), the relationship between σ and the MAE ratios R_N and R_D was not clearly structured.

For the state perturbation and the larger tolerance ($\kappa = 1\mathrm{e}{-2}$), the solutions were well calibrated for a wide range of perturbation parameters, including the default parameter $\sigma = 1$ (Fig. 11A, last two rows). This wide range was likely a result of the influence of the perturbation—even when very small—on the step-size adaptation. Different step-sizes and therefore different points of ODE evaluation resulted in an

appropriate inter-sample distances without reducing the average sample accuracy, which is—as we define it—a well calibrated probabilistic solution. For the smaller tolerance ($\kappa = 1\text{e}{-4}$), the range of good perturbation parameters narrowed, with the best parameters being dependent on both the stimulus and the Runge-Kutta method. For $\text{RKBS}_\text{a}^\text{x}$, the best calibration was acceptable for the default ($\sigma = 1$), but could be improved with either smaller (for the subthreshold stimulus) or larger perturbation parameters (for the spike inducing stimuli) (Fig. 11B, next-to-last row). For $\text{RKDP}_\text{a}^\text{x}$, the default perturbation also resulted in an at least acceptable calibration, with smaller perturbation parameters ($\sigma = 0.1$) improving the calibration for some stimuli (Fig. 11B, last row).

For the uniform step-size perturbation, the calibration of the adaptive Runge-Kutta methods showed almost no dependence on the perturbation parameters within the tested range for the larger tolerance ($\kappa = 1\text{e}{-2}$) (Fig. 11C, last two rows). For the smaller tolerance ($\kappa = 1\text{e}{-4}$), we observed that only relatively small perturbation parameters resulted in a good calibration (Fig. 11D, last two rows). Here we could not simulate the default perturbation parameter, since it did not fulfill the requirement of positive step-sizes only, as we used a maximum step-size of $\Delta t = 1\,\text{ms}$.

For the log-normal step-size perturbation, the results were similar to the uniform perturbation, except that here larger perturbation parameters ($\sigma \geq 1$) could be used. However, these larger values did not result in a good calibration, and the best parameters were much smaller for both $\text{RKBS}_\text{a}^\text{t}$ (ranging from $\sigma = 1\text{e}{-2}$ to $\sigma = 1\text{e}{-1}$; Fig. 11E, next-to-last row) and $\text{RKDP}_\text{a}^\text{t}$ (ranging from $\sigma = 1\text{e}{-4}$ to $\sigma = 1\text{e}{-2}$; Fig. 11E, last row).

Interestingly, we also observed large differences in calibration for the stimuli in dependence on the Runge-Kutta method. For example, for the state-perturbation and $\kappa = 1\text{e}{-2}$ $\text{RKBS}_\text{a}^\text{x}$ achieved the best calibration for the constant stimulus ($R_N R_D \approx 0.9$), while for $\text{RKDP}_\text{a}^\text{x}$ this was the stimulus resulting in the poorest calibration ($R_N R_D \approx 0.6$).

Computational overhead

The probabilistic methods used in this study are sampling based and therefore require multiple evaluations of the same ODE system to yield an uncertainty estimate. While this process can be parallelized, it nevertheless comes 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]. To empirically determine the number of samples necessary to obtain a reliable measure of numerical uncertainty, we generated probabilistic solutions for the classical HH neuron for the step and noisy stimuli. To this end, we sampled mean inter-sample distances $\overline{\text{MAE}}_\text{SS}^n$ for small numbers of samples n, and divided them by the mean inter-sample distances for a much larger number of samples (300) to obtain normalized inter-sample distances R_SS (Fig. 12). For the step stimulus, we found that in most cases already two samples were sufficient to get a good estimate of the inter-sample distance (for $n = 2$, $> 70\%$ of R_SS were in $[0.5, 2.0]$), with little difference between the methods (Figs. 12C and 12F). For the noisy stimulus, the distributions of inter-sample distances were wider, but still only three samples were sufficient to get a good inter-sample distance estimate in most cases (for $n = 3$, $> 70\%$ of R_SS were in $[0.5, 2.0]$). Here, the distributions for EEMP_f had relatively long tails at the lower end of R_SS (Figs. 12A and 12B) while RKBS_a was more evenly distributed (Figs. 12D and 12E).

The probabilistic methods used in this study do not only require to simulate multiple solutions, but also come with a computational overhead per solution, which we already discussed theoretically above (see Methods). To quantify this overhead empirically, we simulated the HH neuron with different probabilistic solvers and their

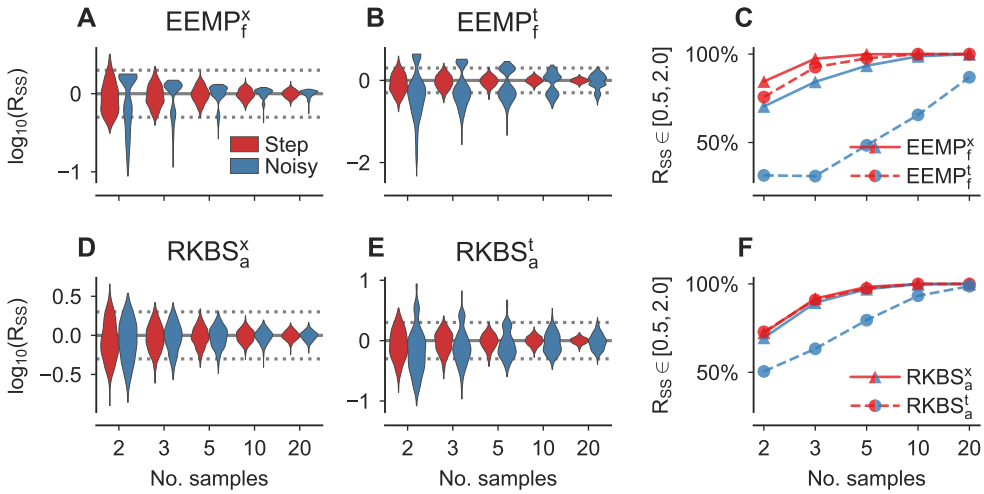

Fig 12. Number of samples necessary to estimate the inter-sample distance. **(A)** Bootstrapped distributions of normalized inter-sample distances R_{SS} for an $\mathrm{EEMP}_{\mathrm{f}}^{\mathrm{x}}$ solver with $\Delta t = 0.1\,\mathrm{ms}$ and $\sigma = 1$ for the classical HH neuron as a function of the numbers of samples n. Distributions were bootstrapped based on 300 samples. For every n, we drew 1000 times n samples without replacement from the 300 samples; and for every draw, we computed the MAE ratio $R_{\mathrm{SS}} = \overline{\mathrm{MAE}}_{\mathrm{SS}}^{n}/\overline{\mathrm{MAE}}_{\mathrm{SS}}^{300}$, where $\overline{\mathrm{MAE}}_{\mathrm{SS}}^{n}$ is the mean inter-sample distance of the drawn samples and $\overline{\mathrm{MAE}}_{\mathrm{SS}}^{300}$ is the mean inter-sample distance of all 300 samples. R_{SS} are shown for the step and the noisy stimulus (legend). Vertical lines highlight values of 0.5, 1 and 2. **(B)** As (A), but for the uniform step-size perturbation and $\sigma = 0.1$ instead of state perturbation. **(C)** Percentages of the distributions of R_{SS} that were between 0.5 and 2 as a function of n for the data shown in (A,B). R_{SS} is shown for the step and noisy stimuli (color as in (A)) and the state and step-size perturbation (legend). **(D-F)** As in (A-C), but for $\mathrm{RKBS}_{\mathrm{a}}$ with $\kappa = 1\mathrm{e}{-3}$.

deterministic counterparts and compared the run times relative to each other. As expected, for the state perturbation, the computational overhead was largest for the lower order methods (Fig. 13). For $\mathrm{FE}_{\mathrm{f}}^{\mathrm{x}}$ and $\mathrm{EE}_{\mathrm{f}}^{\mathrm{x}}$ run times were approximately doubled, for $\mathrm{EEMP}_{\mathrm{f}}^{\mathrm{x}}$ and $\mathrm{RKBS}_{\mathrm{f}}^{\mathrm{x}}$ they increased by $\approx 50\% - 60\%$, and for $\mathrm{RKCK}_{\mathrm{f}}^{\mathrm{x}}$ and $\mathrm{RKDP}_{\mathrm{f}}^{\mathrm{x}}$ by $\approx 40\%$. 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 (25% on average across all adaptive methods and stimuli), with $\mathrm{RKCK}_{\mathrm{a}}^{\mathrm{x}}$, not using the First Same As Last property, being the cheapest with only a $\approx 13\%$ increase. For the step-size perturbation, the increase in run times was on average smaller (11% and 17% on average across all methods and stimuli for the uniform and log-normal state perturbation, respectively) and without large differences between the solver schemes and the usage of adaptive or fixed step-sizes. The step-size perturbation using a log-normal distribution was on average slightly more expensive than uniform step-size perturbation, likely due to the more expensive sampling distribution.

Discussion

Here we studied the effect of numerical uncertainty on the simulation of neurons or neural populations as revealed by probabilistic solvers. 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

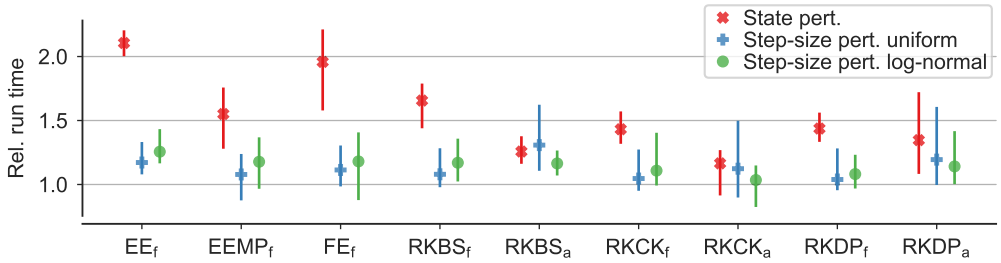

Fig 13. Computational overhead of perturbation methods. Relative run times for different solver schemes (x-axis) and perturbation methods (legend) measured for the classical HH neuron with the step stimulus. For every solver, 100 samples were simulated for both a probabilistic and a respective deterministic solver version. Relative run times were computed by dividing the run times of the probabilistic samples by the run times of the respective deterministic samples. The step-size of fixed step-size methods was $\Delta t = 0.01\,\text{ms}$, and the tolerance of adaptive methods was $\kappa = 1\text{e}{-4}$. The perturbation parameter was $\sigma = 1$ for the state perturbation, and $\sigma = 0.1$ for the step-size perturbation (for both distributions).

some models and parametrizations are more susceptible to numerical uncertainty than others, and that solvers commonly employed in the neuroscience literature typically 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. In particular, our results from the neural network simulations show that it is crucial to systematically assess the effect of the numerical uncertainty on the outcome of neuroscience simulation studies.

The idea to quantify the accuracy/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 [10] 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 [9] 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 put into the proper form and therefore requires specific knowledge about the model and solver. In a more recent study, Chen et al. [11] recommended to use splitting methods, such as second-order Strang splitting, instead of exponential integrators. The diversity in attempts to achieve the best accuracy vs. computational cost trade-off shows that the scientific debate regarding this topic is far from settled and the ideal solver choice is problem specific.

However, usually it is not necessary to pick the "best" solver and it can be sufficient to pick any among the potentially many "good" solvers for a given neuron model. For this purpose, probabilistic solvers may prove very useful as they produce an easy-to-interpret uncertainty measure that can be analyzed without specific knowledge about the solver or solved neuron model. This can facilitate both the choice of the solver and choice of solver settings such as the step-size.

In this study, we used two simple but powerful probabilistic solvers, that perturb the step-wise integration to quantify the numerical uncertainty inherent in simulating the mechanistic neuron models. For both, the state [14] and the step-size perturbation [18],

we showed that this perturbation can be calibrated for mechanistic neuroscience models, and if it is, the numerical uncertainty can be quantified independent of a reference solution. This is crucial, because in many applications it would not make sense to compute both a reference and an uncertainty estimate for a less accurate solution. Moreover, if the reference solution is computed numerically, it may be inaccurate itself (for an example with an analytical derived reference solution see [40]).

Both, the state and step-size perturbation have their advantages and disadvantages. We found that the state perturbation was typically well—even though often not perfectly—calibrated when using the default perturbation parameter, with the first order methods exponential Euler and forward Euler being exceptions that required larger perturbations which then often resulted in highly perturbed solution and numerical instabilities. But since the computational overhead of the state perturbation for these first order methods is relatively large, we recommend to use only higher order methods in combination with the state-perturbation anyway.

For the step-size perturbation, we found the calibration to be slightly more challenging for two reasons. First, the perturbation is influenced by linear scaling of the simulated time, which happens for example if the time unit of the model is changed. And second, for adaptive Runge-Kutta methods the calibration was dependent on the tolerance with the default resulting in relatively poorly calibrated results for the models we analyzed. For the step-size perturbation using a uniform distribution, the perturbation parameter was additionally limited by the (maximum) step-size of the solver, to ensure only positive step-sizes. Despite these constraints on the calibration, the step-size perturbation can be of great use. First, because it comes with comparably little computational overhead especially for first order methods where the default calibration was also relatively good. Second, it can be combined with any explicit solver without the need for an error estimator. And third, it preserves desirable properties of the underlying solver schemes [18]. For example, when Hodgkin-Huxley-like models are solved with exponential integrators like exponential Euler, 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.

In a different line of work, probabilistic ODE solvers are constructed using techniques from (nonlinear) Gaussian filtering and smoothing [12, 13, 41–45]. 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 [13]. In particular for large neural network simulations, such efficient methods will be key in quantifying uncertainty.

In addition to playing a diagnostic role to assess the reliability of simulations, probabilistic numerical methods could be useful for identifying a set of parameters such that simulations from the mechanistic neuron model are aligned with a set of measurements. For this task, recently a variety of methods based on likelihood-free inference techniques have been proposed [5, 7]. However, also the uncertainty estimates of probabilistic numerical methods for ODEs have been shown to be useful for parameter inference. For instance, Kersting et al. [46] show how filtering-based ODE solvers give rise to efficient gradient-based sampling and optimization schemes in the parameter space. It will be interesting to investigate how such probabilistic numerical methods can be applied for efficient parameter inference in neuron modeling.

What we have not addressed in this study are implicit solvers. For stiff problems, e.g. multi-compartment neuron models [47], explicit solvers may not be a good choice and implicit solvers like implicit Euler or higher order methods should be used instead.

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. [48] that automatically determines whether an implicit or an explicit solver should be used. In a future study it would be interesting to test implicit probabilistic solvers and compare them to the explicit probabilistic solvers used in this study.

Taken together, in this study we showed that the numerical uncertainty in common mechanistic neuroscience models can be substantial. Further, we demonstrated how probabilistic perturbation methods can be used to reveal and quantify this uncertainty and how it can guide the choice of a solver and its settings.

Acknowledgements

This research was funded by the Deutsche Forschungsgemeinschaft through a Heisenberg Professorship (BE5601/4-1, PB), the Excellence Cluster 2064 "Machine Learning — New Perspectives for Science" (ref number 390727645, PB and PH) and the Priority Program "Computational Connectomics" (BE5601/2-1, PB), the German Ministry of Education and Research through the Bernstein Award (01GQ1601, PB), ADIMEM (01IS18052C and 01IS18052B to PB and PH) and the Tübingen AI Center (FKZ: 01IS18039A, PB and PH) as well as 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.

References

1. Pillow JW, Shlens J, Paninski L, Sher A, Litke AM, Chichilnisky E, et al. Spatio-temporal correlations and visual signalling in a complete neuronal population. Nature. 2008;454(7207):995–999.

2. Gerwinn S, Bethge M, Macke JH, Seeger M. Bayesian inference for spiking neuron models with a sparsity prior. In: Advances in Neural Information Processing Systems; 2008. p. 529–536.

3. Gerstner W, Kistler WM. Spiking neuron models: Single neurons, populations, plasticity. Cambridge University Press; 2002.

4. Izhikevich EM. Dynamical systems in neuroscience. MIT Press; 2007.

5. Gonçalves PJ, Lueckmann JM, Deistler M, Nonnenmacher M, Öcal K, Bassetto G, et al. Training deep neural density estimators to identify mechanistic models of neural dynamics. bioRxiv doi: 101101/838383. 2019;.

6. Oesterle J, Behrens C, Schroeder C, Herrmann T, Euler T, Franke K, et al. Bayesian inference for biophysical neuron models enables stimulus optimization for retinal neuroprosthetics. bioRxiv doi: 101101/20200108898759. 2020;.

7. Papamakarios G, Sterratt DC, Murray I. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. arXiv:180507226. 2018;.

8. Hennig P, Osborne MA, Girolami M. Probabilistic numerics and uncertainty in computations. Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences. 2015;471(2179).

9. Stewart RD, Bair W. Spiking neural network simulation: numerical integration with the Parker-Sochacki method. Journal of Computational Neuroscience. 2009;27(1):115–133.

10. Borgers C, Nectow AR. Exponential Time Differencing for Hodgkin–Huxley-like ODEs. SIAM Journal on Scientific Computing. 2013;35(3):B623–B643.

11. Chen Z, Raman B, Stern A. Structure-Preserving Numerical Integrators for Hodgkin–Huxley-Type Systems. SIAM Journal on Scientific Computing. 2020;42(1):B273–B298.

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

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

14. Conrad PR, Girolami M, Särkkä S, Stuart A, Zygalakis K. Statistical analysis of differential equations: introducing probability measures on numerical solutions. Statistics and Computing. 2017;27:1065–1082.

15. Chkrebtii OA, Campbell DA, Calderhead B, Girolami MA. Bayesian solution uncertainty quantification for differential equations. Bayesian Analysis. 2016;11:1239–1267.

16. Teymur O, Zygalakis K, Calderhead B. Probabilistic Linear Multistep Methods. In: Lee DD, Sugiyama M, Luxburg UV, Guyon I, Garnett R, editors. Advances in Neural Information Processing Systems. Curran Associates, Inc.; 2016. p. 4314–4321.

17. Teymur O, Lie HC, Sullivan T, Calderhead B. Implicit probabilistic integrators for ODEs. In: Advances in Neural Information Processing Systems; 2018. p. 7244–7253.

18. Abdulle A, Garegnani G. Random time step probabilistic methods for uncertainty quantification in chaotic and geometric numerical integration. Statistics and Computing. 2020;.

19. Izhikevich EM, Edelman GM. Large-scale model of mammalian thalamocortical systems. Proceedings of the National Academy of Sciences. 2008;105(9):3593–3598.

20. Izhikevich EM. Simple model of spiking neurons. IEEE Transactions on Neural Networks. 2003;14(6):1569–1572.

21. Domhof JW, Tiesinga PH. Balance between inhibitory cell types is necessary for flexible frequency switching in adult mouse visual cortex. bioRxiv doi: 101101/20200118911271. 2020;.

22. Galán RF, Fourcaud-Trocmé N, Ermentrout GB, Urban NN. Correlation-induced synchronization of oscillations in olfactory bulb neurons. Journal of Neuroscience. 2006;26(14):3646–3655.

23. Izhikevich EM. Which model to use for cortical spiking neurons? IEEE Transactions on Neural Networks. 2004;15(5):1063–1070.

24. Hodgkin AL, Huxley AF. A quantitative description of membrane current and its application to conduction and excitation in nerve. The Journal of Physiology. 1952;117(4):500–544.

25. Prinz AA, Billimoria CP, Marder E. Alternative to hand-tuning conductance-based models: construction and analysis of databases of model neurons. Journal of Neurophysiology. 2003;90(6):3998–4015.

26. Prinz AA, Bucher D, Marder E. Similar network activity from disparate circuit parameters. Nature Neuroscience. 2004;7(12):1345–1352.

27. Ermentrout GB, Terman DH. The hodgkin–huxley equations. In: Mathematical foundations of neuroscience. Springer; 2010. p. 1–28.

28. Hairer E, Nørsett SP, Wanner G. Solving Ordinary Differential Equations I – Nonstiff Problems. Springer; 1993.

29. Dormand JR, Prince PJ. A family of embedded Runge-Kutta formulae. Journal of Computational and Applied Mathematics. 1980;6(1):19–26.

30. Oates CJ, Sullivan TJ. A modern retrospective on probabilistic numerics. Statistics and Computing. 2019;29:1335–1351.

31. Cockayne J, Oates CJ, Sullivan TJ, Girolami M. Bayesian probabilistic numerical methods. SIAM Review. 2019;64:756–789.

32. Lie HC, Stuart AM, Sullivan TJ. Strong convergence rates of probabilistic integrators for ordinary differential equations. Statistics and Computing. 2019;29:1265–1283.

33. Bogacki P, Shampine LF. A 3 (2) pair of Runge-Kutta formulas. Applied Mathematics Letters. 1989;2(4):321–325.

34. Cash JR, Karp AH. A variable order Runge-Kutta method for initial value problems with rapidly varying right-hand sides. ACM Transactions on Mathematical Software (TOMS). 1990;16(3):201–222.

35. Oh J, French DA. Error analysis of a specialized numerical method for mathematical models from neuroscience. Applied mathematics and computation. 2006;172(1):491–507.

36. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods. 2020;17:261–272. doi:https://doi.org/10.1038/s41592-019-0686-2.

37. Cranmer K, Brehmer J, Louppe G. The frontier of simulation-based inference. Proceedings of the National Academy of Sciences. 2020;117(48):30055–30062.

38. Gonçalves PJ, Lueckmann JM, Deistler M, Nonnenmacher M, Öcal K, Bassetto G, et al. Training deep neural density estimators to identify mechanistic models of neural dynamics. Elife. 2020;9:e56261.

39. Butera RJ, McCarthy ML. Analysis of real-time numerical integration methods applied to dynamic clamp experiments. Journal of Neural Engineering. 2004;1(4):187.

40. Henker S, Partzsch J, Schüffny R. Accuracy evaluation of numerical methods used in state-of-the-art simulators for spiking neural networks. Journal of Computational Neuroscience. 2012;32(2):309–326.

41. Kersting H, Hennig P. Active Uncertainty Calibration in Bayesian ODE Solvers. Uncertainty in Artificial Intelligence (UAI). 2016;.

42. Magnani E, Kersting H, Schober M, Hennig P. Bayesian Filtering for ODEs with Bounded Derivatives. arXiv:170908471. 2017;.

43. Kersting H, Sullivan TJ, Hennig P. Convergence Rates of Gaussian ODE Filters. arXiv:180709737v2. 2019;.

44. John D, Heuveline V, Schober M. GOODE: A Gaussian Off-The-Shelf Ordinary Differential Equation Solver. In: International Conference on Machine Learning; 2019. p. 3152–3162.

45. Tronarp F, Särkkä S, Hennig P. Bayesian ODE Solvers: The Maximum A Posteriori Estimate. arXiv:200400623. 2020;.

46. Kersting H, Krämer N, Schiegg M, Daniel C, Tiemann M, Hennig P. Differentiable Likelihoods for Fast Inversion of 'Likelihood-Free' Dynamical Systems. arXiv:200209301. 2020;.

47. Mascagni MV, Sherman AS, et al. Numerical methods for neuronal modeling. Methods in neuronal modeling. 1989;2.

48. Blundell I, Plotnikov D, Eppler JM, Morrison A. Automatically selecting a suitable integration scheme for systems of differential equations in neuron models. Frontiers in neuroinformatics. 2018;12:50.