Sensitivity approximation by the Peano-Baker series

Numerische Mathematik (2026) 158:303–352
https://doi.org/10.1007/s00211-025-01514-2

Numerische Mathematik

Sensitivity approximation by the Peano-Baker series

Olivia Eriksson[1,2] · **Andrei Kramer-Miehe**[2,3] · **Federica Milinanni**[4] · **Pierre Nyquist**[4,5]

Received: 22 October 2021 / Revised: 10 July 2024 / Accepted: 30 November 2025 /
Published online: 17 December 2025

Abstract
In this paper we develop a new method for numerically approximating sensitivities in parameter-dependent ordinary differential equations (ODEs). Our approach, intended for situations where the standard forward and adjoint sensitivity analyses become too computationally costly for practical purposes, is based on the Peano-Baker series from control theory. Using this series, we construct a representation of the sensitivity matrix $\mathbf{S}$ and, from this representation, a numerical method for approximating $\mathbf{S}$. We prove that, under standard regularity assumptions, the error of our method scales as $\mathcal{O}(\Delta t_{\max}^2)$, where $\Delta t_{\max}$ is the largest time step used when numerically solving the ODE. We illustrate the performance of the method in several numerical experiments, taken from both the systems biology setting and more classical dynamical systems. The experiments show the sought-after improvement in running time of our method compared to the forward sensitivity approach. In experiments involving a random linear system, the forward approach requires roughly $\sqrt{n}$ longer computational time, where n is the dimension of the parameter space, than our proposed method.

Mathematics Subject Classification 65L05 · 65L20

1 Introduction

Mathematical models are used in all areas of science and engineering to model more and more complex real-world phenomena. An important aspect of such modeling is to

✉ Federica Milinanni
fedmil@kth.se

1 School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden

2 Science for Life Laboratory, Solna, Sweden

3 Department of Neuroscience, Karolinska Institute, Solna, Sweden

4 Department of Mathematics, KTH Royal Institute of Technology, Stockholm, Sweden

5 Department of Mathematical Sciences, Chalmers University of Technology and University of Gothenburg, Gothenburg, Sweden

understand how changes and uncertainty in model parameters translate to the output of a model. The first question is the topic of *sensitivity analysis*, an active research area at the intersection of several branches of mathematics and its applications; see e.g. [1–5] and references therein. Motivated by problems arising in systems biology, specifically concerning statistical inference and uncertainty quantification of models based on non-linear dynamical systems, in this paper we consider the rather classical question of local sensitivity analysis in ODE models. More precisely, we are interested in the design of efficient numerical methods for approximating the sensitivity matrix for such models. Although our motivation originally comes from wanting sensitivity approximation methods that are suitable for Markov chain Monte Carlo (MCMC) methods in the systems biology setting–see Section 1.1 for more details and references–we note that the general problem of computing sensitivities in an ODE model is of great interest in a number of fields.

The general starting point is a system of ordinary differential equations (ODEs), which we take to be of the form

$$\begin{cases} \dot{\mathbf{x}}(t) = f(t, \mathbf{x}, \mathbf{p}) \,, \\ \mathbf{x}(t_0) = \mathbf{x}_0 \,, \end{cases} \tag{1.1}$$

where $\mathbf{x}(t)$ represents the state of the system at time t, and $\mathbf{p}$ denotes a set of model parameters[1]; a more precise definition, including the state spaces of the different quantities involved, is given in Sect. 2.

For such a model, it is important to understand how changes in the parameter-vector $\mathbf{p}$ affect the output $\mathbf{x}$: we are interested in computing the derivatives $S_i{}^j(t) = \partial x_i(t)/\partial p_j$, the *sensitivities* of the model, for all component-combinations (i, j)[2]. The sensitivities provide local information about the parameter space that is important for a variety of tasks within modeling where this space is explored, e.g., quantifying uncertainty, finding optimal parameters, and experimental design [2, 3, 5, 6].

For all but very simple systems the sensitivities are not available in an (explicit) analytical form. Instead, we have to turn to different types of approximations [3, 5]. The two standard approaches for obtaining numerical approximations to the sensitivities of an ODE or PDE model are (i) the *forward*, or *variational*, approach [2, 3], and (ii) the *adjoint* approach [7–9]. There is a vast literature on both methods and here we only give a brief review; for a comparison of the two approaches, see [6].

Between the forward and adjoint methods, forward sensitivity analysis is the more straightforward of the two. It is based on finding an ODE system satisfied by the sensitivities $S_i{}^j(t)$, along with appropriate initial values. The solution to this system can then be approximated together with that of the original ODE for $\mathbf{x}$ [3, 6]. It is well-known that for high-dimensional problems–$n_x \times n_p \gg 1$, where n_x and n_p are

[1] In a typical systems biology setting there is also often an input function $\mathbf{u}(t)$ explicit in the arguments of f. Because such inputs $\mathbf{u}(t)$ do not play any role in developing the numerical methods that are the focus of this paper, and for simplicity of notation, we here consider $\mathbf{u}(t)$ as embedded into f.

[2] In some cases we are interested in computing the sensitivities of an output function h applied to the state $\mathbf{x}$, i.e. $\bar{S}_i{}^j(t) = \partial h(\mathbf{x})_i(t)/\partial p_j$. By the chain rule, we can reduce this problem to the computation of the sensitivities $S_i{}^j$ of the state $\mathbf{x}$; in fact, $\bar{S}_i{}^j(t) = \sum_k \frac{\partial h(\mathbf{x})_i}{\partial x_k} S_k{}^j(t)$.

the dimension of the state space and parameter space, respectively–computation of the sensitivities using forward sensitivity analysis becomes slow. As a consequence, this method can be too computationally costly for certain applications, and more efficient methods are needed. Indeed, the manuals of solvers such as CVODES [10] warn against this approach for large systems [11], favouring adjoint methods where possible.

Adjoint sensitivity analysis is designed to ease the computation of sensitivities with respect to $\mathbf{p}$ of an objective function

$$G(\mathbf{x}, \mathbf{p}) = \int_0^T g(\mathbf{x}, t, \mathbf{p}) dt,$$

for some function g (satisfying certain smoothness assumptions). Alternatively, the method can be used to compute sensitivities of $g(\mathbf{x}(T), T, \mathbf{p})$, that is a function only defined at the final time T. The method amounts to introducing an augmented objective function from G by introducing Lagrange multipliers associated with the underlying ODE, and computing the derivatives dG/dp_j of this objective function; see e.g. [7–9] for the details.

In contrast to the forward sensitivity analysis, the adjoint method does not rely on the actual state sensitivities, $\partial x_i / \partial p_j$, but rather on the adjoint state process. It is possible, albeit impractical, to formulate the problem so that the adjoint sensitivity analysis computes $\partial x_i / \partial p_j$, the quantities of interest in this paper: define $g(\mathbf{x}, t, \mathbf{p}) = x_i(t)$ and use the adjoint sensitivity method to compute the sensitivities of g at the specific times $T = t_k$. However, this must be repeated for each component x_i of the state $\mathbf{x}$ and for each time t_k at which an approximation of the sensitivity of the ODE model is sought. As a result, the use of adjoint sensitivity is not recommended for this purpose, especially when (i) the dimension of the state space is large, and (ii) there are many time points t_k at which the sensitivities are to be computed.

The forward sensitivity and the adjoint methods are both too computationally expensive for the type of applications we have in mind, in particular when there is a need to approximate sensitivities in larger models (i.e., high-dimensional state and/or parameter space). As a first step towards resolving this issue, we introduce a new method, based on the Peano-Baker series [12], referred to as the *Peano-Baker Series (PBS) method*. It turns out that this method can be unstable when applied to stiff problems[3], which are common in systems biology applications. We therefore modify the PBS method by adding a stiffness detection mechanism and refining the time steps accordingly: this leads to the *PBS algorithm with refinement (PBSR)*. Additionally, we introduce an algorithm that we call the *Exponential (Exp) algorithm*, which is cheaper, though in general less accurate, and has the advantage of being unconditionally stable. Because we will mainly work with stiff problems, the PBSR and Exp algorithm are the methods that we will implement in practice.

The methods we propose utilize two different approximation techniques, that differ in their stability properties, accuracy and computational time. By the use of these two techniques, we can handle various dynamical properties that arise in different parts

[3] *Stiffness* has no rigorous definition, here we treat the term as meaning that we need an algorithm of *high order* and small *step sizes*; see, e.g., [13].

of the state and parameter space associated with the ODE system, e.g. stiffness and transient versus equilibrium behaviour. We show that by combining these techniques we construct numerical methods to approximate the sensitivity matrix with a computational time and an accuracy that are suitable for use within MCMC methods, which is our main motivation for the sensitivity approximation (see Sect. 1.1). Using the proposed methods in the context of MCMC sampling is, however, ongoing work and in this paper we consider the methods solely from a numerical analysis perspective.

The remainder of the paper is organised as follows. In Section 1.1 we expand briefly on our particular motivation for studying the problem of computing sensitivities in an ODE system. This is followed by an outline of the notation used throughout the paper (Sect. 1.2). In Sect. 2 we give a precise formulation of the ODE model and the associated sensitivities of interest. Next, in Sect. 3 we derive analytical representations of the sensitivities in two different cases: when the solution is at equilibrium (Theorem 3.1), and in the general case (Theorem 3.3). In Sect. 4 the analytical representations are used to propose a new method, the PBS method, outlined in Algorithm 1, for approximating sensitivities in the ODE setting considered in this paper. Before analysing the method, we discuss related stability issues in Sect. 5, where we introduce the PBSR and the Exp algorithms (Algorithms 2 and 3, respectively), and analyze their properties. The main theoretical result of the paper is the error analysis of the PBS and PBSR methods, presented in Sect. 6 in Theorem 6.1 and in Corollary 6.2, respectively; the corresponding proofs are given in Sect. 7. In Sect. 8 we present some numerical experiments that showcase the performance of the proposed methods. In particular, we compare our methods with the forward sensitivity approach and show superior performance. We end the paper with a discussion in Sect. 9.

1.1 Motivation

Our interest in the problem originally stems from uncertainty quantification for models of intracellular pathways, such as those studied in [14]. Forward sensitivity analysis via CVODES is prohibitively slow for such systems; for example, when sensitivity analysis is added for the model used in [14] for modelling the CaMKII system (see also Sect. 8 for additional details on this model), the simulation time increases roughly 100-fold, compared to simulations without sensitivity analysis.

In the systems biology setting, the state process $\mathbf{x}(t)$ corresponds to the concentration of different compounds internal to the cell (e.g. proteins, protein complexes, or ions), and the parameters $\mathbf{p} = (p_1, \ldots, p_{n_p})$ encode e.g. reaction rate coefficients of the chemical reactions involved in the system. In this setting, we are interested in uncertainty quantification for the unknown parameters within a Bayesian framework. One of the standard methods for uncertainty quantification involves sampling from posterior distributions, which can be carried out via MCMC methods [15]. We implement some methods from the MCMC class in the R package UQSA[4] [16].

In general, MCMC sampling often requires one to compute different quantities that will involve the sensitivity matrix $\mathbf{S}$, such as the parameter derivatives of the log-likelihood function. As a motivating example, recent developments in the MCMC

[4] **U**ncertainty **Q**uantification and **S**ensitivity **A**nalysis.

community propose to take a differential-geometric perspective on sampling, exploiting potential underlying concepts related to statistical inference [17–19]. This in turn relies on an appropriate choice of the metric tensor on the parameter space. A good choice of metric tensor [17, 20] is the expected Fisher information. For a generic random variable Y (e.g. a model for the observation of data) and associated parameter θ, with conditional probability density function $p_{Y|\theta}$, the expected Fisher information is defined as:

$$F(\theta) = -E_{Y|\theta}\left[\frac{\partial^2}{\partial\theta^2}\log\left(p_{Y|\theta}(Y \mid \theta)\right)\right]. \tag{1.2}$$

In the case of a Gaussian measurement error model, the distribution of each (independent) observable Y_{ij} has two scalar parameters: μ_{ij} and σ_{ij}, where i is the state index and j the measurement time-point index. When combined with an ODE model, the entries of the Fisher information F then contain the sensitivities of the model alongside the parameters of the noise model [17, 21]:

$$F(\theta)_i^{\ j} = \sum_{m,l} S_m^{\ i}(t_l)\sigma_{ml}^{-2}S_m^{\ j}(t_l)\,, \tag{1.3}$$

where l is the index of measurement times within a *time series*.

The quality of the sensitivity estimates in (1.3), and possibly the log-likelihood and its derivatives, affects the efficiency of MCMC in terms of the convergence speed. Even an efficiently implemented MCMC method, such as the simplified manifold MALA (SMMALA, see [17]), will require repeatedly approximating the sensitivities for all time points l, possibly millions of times for different values of θ. Because the sensitivities $S_i^{\ j}(t)$ are required in (1.3), and also to compute the gradient of the log-likelihood function, the adjoint sensitivity analysis method is not appropriate for this type of setting.

Our main goal is to construct a method that can approximate the sensitivities[5] *after* the initial value problem for $\mathbf{x}(t)$ has been solved independently (not simultaneously)[6]. This decoupling of the two problems allows us to use any numerical solver, in almost any programming language, rather than being tied to a solver that is built to include forward sensitivity analysis (such as CVODES). Furthermore, the problems we encounter in systems biology are often stiff, and so are the associated sensitivity equations. To avoid slowing down numerical solvers, we would like the stiffness of the sensitivity equations (i.e., the ODE system for $\mathbf{S}$) to not affect the solver's step size for the ODE system for $\mathbf{x}$.

In MCMC sampling, when there is an underlying ODE system, the sensitivities are used to inform the MCMC update step. A more precise approximation of the sensitivity

[5] The sensitivities must be approximated for all time points $\{t_l\}$ that appear in the likelihood function, i.e., any phase of the time span under consideration: initial, transient and equilibrium (possibly a limit cycle).

[6] To evaluate the likelihood function $p(Y \mid \theta)$, we need to solve the initial value problem (1.1) for $\mathbf{x}$. This solution has to be as accurate as possible, as it directly affects the inferred shape of the posterior distribution. For this reason, we can safely assume that in all the error terms in this manuscript, the relative error of $\mathbf{x}$ will be comparatively small. In practice, we will use the best available solvers and set tight tolerances when solving for $\mathbf{x}(t)$.

matrix will lead to better–as in more aligned with the true transition kernel, based on the true sensitivity matrix $\mathbf{S}$–proposals for the next state of the underlying Markov chain. This in turn should lead to faster convergence to equilibrium of the chain in the case of MCMC algorithms such as such as SMMALA and RMHMC, meaning fewer MCMC iterations are needed to reach convergence of the Markov chain. However, precise approximations of the sensitivity matrix lead to a higher cost per iteration, increasing the cost of the update step, and thus the entire MCMC scheme. In this paper we propose new methods for approximating the sensitivity matrix that are more efficient for our purposes. The trade-off between accuracy and efficiency of the sensitivity approximation should lead to an increased overall performance of the MCMC method[7]. We are planning to implement the sensitivity approximation methods proposed in this paper into the MCMC methods available in the R package UQSA, and investigate the impact on performance.

1.2 Notation

The following notation is used. Elements in $\mathbb{R}^n$ are denoted by bold font, e.g. $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$, whereas x, y, z denote elements of $\mathbb{R}$; note that the relevant dimension n will change between different vectors. With some abuse of notation, 0 denotes the zero element in $\mathbb{R}^n$ for any $n \geq 1$. The identity matrix is denoted by $\mathbb{I}$, or $\mathbb{I}_n$ whenever we want to emphasise the size $n \times n$. For a matrix $\mathbf{A}$, the matrix exponential $e^{\mathbf{A}}$ is defined in the usual way:

$$e^{\mathbf{A}} = \sum_{h \geq 0} \frac{1}{h!} \mathbf{A}^h.$$

Objects with a hat, such as $\hat{f}$ or $\hat{\mathbf{x}}$, refer to approximations. For $k \in \mathbb{N}$ and an open set A in $\mathbb{R}^n$, $\mathcal{C}^k(A; \mathbb{R}^m)$ is the space of continuous functions on A, taking values in $\mathbb{R}^m$, with continuous partial derivatives up to the kth order; for $m \geq 2$, the latter is defined in the usual way through natural projections. For a compact set K, $\mathcal{C}^k(K; \mathbb{R}^m)$ refers to functions that are $\mathcal{C}^k(A; \mathbb{R}^m)$ on some open neighbourhood A of K.

For a function $f : \mathbb{R}^n \to \mathbb{R}^m$, f_l denotes the l-th entry of the value of f, with $l = 1, \ldots, m$, and ∇f denotes the Jacobian matrix of f.

For a normed space $(\mathcal{X}, \|\cdot\|)$, $\mathcal{C}^0([0, T]; \mathcal{X})$ denotes the space of continuous functions from $[0, T]$ to $\mathcal{X}$, and $\|\cdot\|_{[0,T]}$ denotes the supremum norm over this space: for $g \in \mathcal{C}^0([0, T]; \mathcal{X})$,

$$\|g\|_{[0,T]} = \sup_{t \in [0,T]} \|g(t)\|.$$

Unless otherwise stated, we will let $\|\cdot\|$ denote an arbitrary matrix norm; because we are working on a finite-dimensional space, the specific choice of matrix norm is not important for the results of this paper (discussed more in Sect. 6).

[7] Such a comparison is outside the scope of this manuscript.

When listing the arguments of a function, we omit nested repetitions of arguments: when t, $\mathbf{x}$, and $\mathbf{p}$ appear together in an argument list, they always belong to the same time-slice and parametrization. For example, in (1.1) $f(t, \mathbf{x}, \mathbf{p})$ means the same as $f(t, \mathbf{x}(t), \mathbf{p})$, or indeed $f(t, \mathbf{x}(t, \mathbf{p}), \mathbf{p})$.

Additional notation that is particular to this work is defined as needed, particularly in the error analysis in Sects. 6-7.

2 Problem formulation

The general problem of interest is to compute sensitivities for the solution of the parametrised initial value problem (IVP)

$$\begin{cases} \dot{\mathbf{x}}(t) = f(t, \mathbf{x}, \mathbf{p}) \,, \\ \mathbf{x}(t_0) = \mathbf{x}_0 \,, \end{cases} \tag{2.1}$$

with $\mathbf{x}(t) \in \mathbb{R}^{n_x}$, and $\mathbf{p} \in \mathbb{R}^{n_p}$. As noted in Sect. 1, although our interest in this problem comes from a desire to conduct uncertainty quantification for models in systems biology, the problem is much more general and arises in a wide variety of areas within applied mathematics. We therefore work in a general framework for the remainder of this paper.

We assume the existence and uniqueness of solutions:

Assumption 1 The function $f : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$ is globally Lipschitz continuous in the state variable coordinate, for all values of t and $\mathbf{p}$, and uniformly $C^1(\mathbb{R}^{n_p}; \mathbb{R}^{n_x})$ in the parameter coordinate for all values of t and $\mathbf{x}$.

This is in accordance with the Picard-LindelÃ¶f theorem for existence and uniqueness of the solution to the initial value problem with f and x_0.

Definition 1 Given a solution $\mathbf{x}$ of the ODE system (2.1), the *sensitivity* at time t of the l-th component of $\mathbf{x}$ with respect to parameter p_i is defined as

$$S_l{}^i(t) = \frac{\partial x_l(t)}{\partial p_i}, \quad l = 1, \ldots, n_x, \; i = 1, \ldots, n_p. \tag{2.2}$$

By differentiating (2.1) with respect to $\mathbf{p}$, we obtain that the time derivative of the sensitivities can be expressed in matrix form as

$$\dot{\mathbf{S}}(t) = \nabla_x f(t, \mathbf{x}, \mathbf{p}) \cdot \mathbf{S}(t) + \nabla_p f(t, \mathbf{x}, \mathbf{p}), \tag{2.3}$$

where $\dot{\mathbf{S}}(t), \mathbf{S}(t), \nabla_p f(t, \mathbf{x}, \mathbf{p}) \in \mathbb{R}^{n_x \times n_p}$ are matrices with elements $(l, i) \in \{1, \ldots, n_x\} \times \{1, \ldots, n_p\}$ given, respectively, by

$$\dot{S}_l{}^i(t) \,, \qquad S_l{}^i(t) \qquad \text{and} \qquad \frac{\partial f_l(t, \mathbf{x}, \mathbf{p})}{\partial p_i} \,,$$

and $\nabla_x f(t, \mathbf{x}, \mathbf{p}) \in \mathbb{R}^{n_x \times n_x}$ with entries

$$\nabla_x f(t, \mathbf{x}, \mathbf{p})_l{}^j = \frac{\partial f_l(t, \mathbf{x}, \mathbf{p})}{\partial x_j}, \quad (l, j) \in \{1, \ldots, n_x\} \times \{1, \ldots, n_x\}.$$

Because the initial condition $\mathbf{x}_0$ does not depend on $\mathbf{p}$, we have $S_l{}^i(t_0) = \frac{\partial x_{0,l}}{\partial p_i} = 0$ and equation (2.3) is a linear ODE system with initial condition $\mathbf{S}_{|t=t_0} = \mathbf{S}_0 = 0 \in \mathbb{R}^{n_x \times n_p}$.

Let $\{t_k\}_{k=1}^K$, with $0 = t_0 < t_1 < \cdots < t_K = T$, be the time instants at which we want to compute the sensitivity matrix $\mathbf{S}$, and let $\mathbf{S}_k \in \mathbb{R}^{n_x \times n_p}$ denote the exact sensitivity matrices at times t_k, with $k = 0, \ldots, K$. We can then formulate K ODE problems in an iterative fashion,

$$\begin{cases} \dot{\mathbf{S}}(t) = \nabla_x f(t, \mathbf{x}, \mathbf{p}) \cdot \mathbf{S}(t) + \nabla_p f(t, \mathbf{x}, \mathbf{p}) \\ \mathbf{S}(t_k) = \mathbf{S}_k \end{cases}, \quad k = 0, \ldots, K-1, \tag{2.4}$$

where the $(k+1)$th problem in the sequence is used to determine the solution at time step t_{k+1}, given the sensitivity matrix at the previous time step t_k as initial condition. We will adopt a slight abuse of notation and refer to $\mathbf{S}$ as the sensitivity matrix, and similar for the corresponding approximations, although to be precise it is a matrix-valued function.

The matrix of coefficients $\nabla_x f$ and the forcing term $\nabla_p f$ in (2.4) both depend on t, $\mathbf{x}(t)$, and $\mathbf{p}$, and computing the solution $\mathbf{S}(t)$ therefore becomes a difficult task in general. We address this problem by developing a new, efficient algorithm for approximating the sensitivity matrix $\mathbf{S}$ at times $\{t_k\}_{k=1}^K$.

A comment on the sequence $\{t_k\}_{k=1}^K$ of times at which $\mathbf{S}$ is to be evaluated is in place. In Sect. 1.1 we briefly discussed the specific application of approximating the sensitivity matrix of an ODE in the context of uncertainty quantification in biological models. There, as in many other application areas, it is natural to consider a sequence of time instants t_i, $i = 1, \ldots, n_t$, that represent times of measurements of experimental data. However, in experimental settings it is rather common to have measurements only at a small number n_t of times. In order to obtain a good numerical approximation of (2.4), we therefore need a finer collection of time steps. For this reason, even when there are physical time instants to consider, we introduce a finer collection of time steps $\tilde{t}_0, \ldots, \tilde{t}_K$, such that it contains all the time steps at which measurements are available, i.e. $\{t_1, \ldots, t_{n_t}\} \subset \{\tilde{t}_0, \ldots, \tilde{t}_K\}$; the inclusion is enforced because in this experimental setting the goal is to approximate the sensitivity matrix $\mathbf{S}$ at the n_t measurement times. Moreover, we let $\tilde{t}_0$ be equal to the time of the initial condition: $\tilde{t}_0 = t_0$. To simplify and to conform to the notation used in (2.4), we drop the tilde from the new time steps $\{\tilde{t}_k\}_{k=0}^K$, which in the following are denoted by $t_0, \ldots, t_K$.

3 Analytical solutions for the sensitivity matrix S

In this section we derive analytical representations for the sensitivity matrix $\mathbf{S}(t) \in \mathbb{R}^{n_x \times n_p}$ along the trajectory of the solution $\mathbf{x}(t)$ of the original ODE system (2.1). To simplify notation, we will here denote $\nabla_x f(t, \mathbf{x}, \mathbf{p})$ and $\nabla_p f(t, \mathbf{x}, \mathbf{p})$ as time dependent matrices $\mathbf{A}(t) \in \mathbb{R}^{n_x \times n_x}$ and $\mathbf{B}(t) \in \mathbb{R}^{n_x \times n_p}$, respectively, defined on a closed interval $I \subset \mathbb{R}$ such that $[t_0; t_K] \subset I$. The system (2.4) then takes the form

$$\begin{cases} \dot{\mathbf{S}}(t) = \mathbf{A}(t) \cdot \mathbf{S}(t) + \mathbf{B}(t), \\ \mathbf{S}(t_k) = \mathbf{S}_k, \end{cases} \tag{3.1}$$

which in general represents a first order inhomogeneous linear differential equation with non-constant coefficient matrix $\mathbf{A}(t)$ and forcing term $\mathbf{B}(t)$. Such systems are well-understood from a theoretical point of view [22, 23] and there exist a variety of numerical methods for solving them [24–27]. Although efficient numerical methods are readily available for the n_x-dimensional system in $\mathbf{x}(t)$ (2.1), the potentially high-dimensional nature of the associated sensitivity problem (3.1) renders such methods inefficient in that setting. Indeed, since $\mathbf{S} \in \mathbb{R}^{n_x \times n_p}$, the total dimension of the ODE system for the sensitivity is in general much higher than the dimension n_x of (2.1), in particular for large values of n_x. To remedy this, we approach the problem of approximating $\mathbf{S}(t)$ by exploiting some results from the theory for linear ODEs; some are versions of well-known results and we include them here to make the paper self-contained.

In order to derive solutions of (3.1), we start by considering a solution $\mathbf{x}(t)$ that is in an equilibrium point, $\mathbf{x}(t) \equiv \mathbf{x}_{\text{eq}}$. This leads to a constant matrix of coefficients $\mathbf{A}(t) \equiv \mathbf{A}$ and constant forcing term $\mathbf{B}(t) \equiv \mathbf{B}$, thus the system (3.1) takes the form

$$\begin{cases} \dot{\mathbf{S}}(t) = \mathbf{A} \cdot \mathbf{S}(t) + \mathbf{B}\,, \\ \mathbf{S}(t_k) = \mathbf{S}_k\,. \end{cases} \tag{3.2}$$

The solution of this system is obtained using well-known results from ODE theory; we include a proof for completeness.

Theorem 3.1 *Let* $\mathbf{A} \in \mathbb{R}^{n_x \times n_x}$, $\mathbf{B} \in \mathbb{R}^{n_x \times n_p}$ *and* $\mathbf{S}_k \in \mathbb{R}^{n_x \times n_p}$. *The solution* $\mathbf{S} \in \mathcal{C}^1(I; \mathbb{R}^{n_x \times n_p})$ *at a time* $t \in \mathbb{R}$ *of the first order linear ODE system* (3.2) *is given by*

$$\mathbf{S}(t) = e^{(t-t_k)\cdot\mathbf{A}} \cdot \left(\mathbf{S}_k + \int_{t_k}^{t} e^{(t_k - s)\mathbf{A}} \mathbf{B} ds \right). \tag{3.3}$$

In particular, if $\mathbf{A}$ *is invertible,* (3.3) *is equivalent to*

$$\mathbf{S}(t) = e^{(t-t_k)\cdot\mathbf{A}} \cdot \left(\mathbf{S}_k + \left(\mathbb{I} - e^{-(t-t_k)\cdot\mathbf{A}} \right) \mathbf{A}^{-1} \mathbf{B} \right). \tag{3.4}$$

Proof We show that the matrix-valued function $\mathbf{S}$ defined in (3.3) solves the ODE system (3.2).

The time derivative $\dot{\mathbf{S}}$ of the function $\mathbf{S}$ defined in (3.3) is given by

$$\begin{aligned}\dot{\mathbf{S}}(t) &= \mathbf{A}\cdot e^{(t-t_k)\cdot\mathbf{A}}\cdot\left(\mathbf{S}_k+\int_{t_k}^{t} e^{(t_k-s)\mathbf{A}}\mathbf{B}ds\right)+e^{(t-t_k)\cdot\mathbf{A}}\cdot e^{(t_k-t)\mathbf{A}}\mathbf{B}\\ &= \mathbf{A}\cdot\mathbf{S}(t)+\mathbf{B}.\end{aligned}$$

Moreover, the value of $\mathbf{S}$ at time $t=t_k$ is $\mathbf{S}(t_k)=\mathbf{S}_k$. This proves that $\mathbf{S}$ solves the ODE system (3.2).

Finally, note that when $\mathbf{A}$ is invertible,

$$\begin{aligned}\int_{t_k}^{t} e^{(t_k-s)\mathbf{A}}\mathbf{B}ds &= \int_{t_k}^{t} e^{(t_k-s)\mathbf{A}}\mathbf{A}\mathbf{A}^{-1}\mathbf{B}ds = \int_{t_k}^{t}\frac{d}{ds}\left(-e^{(t_k-s)\mathbf{A}}\right)ds\mathbf{A}^{-1}\mathbf{B}\\ &= \left(\mathbb{I}-e^{-(t-t_k)\mathbf{A}}\right)\mathbf{A}^{-1}\mathbf{B}.\end{aligned}$$

It immediately follows that, when $\mathbf{A}$ is invertible, (3.3) is equivalent to (3.4). □

From a control-theoretic perspective, the matrix exponential $e^{(t-t_k)\cdot\mathbf{A}}$ corresponds to the so-called *state-transition matrix* $\Phi(t;t_k)$ in the specific case of a constant matrix of coefficients $\mathbf{A}$. In general, we consider the first order homogeneous linear ODE system

$$\dot{\mathbf{S}}(t)=\mathbf{A}(t)\cdot\mathbf{S}(t), \tag{3.5}$$

with a time dependent matrix of coefficients $\mathbf{A}(t)$, assumed to be continuous on the closed time interval I. In this more general setting, we define the state-transition matrix $\Phi(t;t_k)$ associated with $\mathbf{A}(t)$ as the matrix-valued function that, given any initial condition $\mathbf{S}(t_k)=\mathbf{S}_k$, allows us to represent the solution of (3.5) at time $t\in I$ as

$$\mathbf{S}(t)=\Phi(t;t_k)\cdot\mathbf{S}_k.$$

The state-transition matrix $\Phi(t;s)$ is a well-studied object; see [23], [22, Chap. 1, Sect. 3] for a detailed description and properties. In particular, in this paper we make repeated use of the the following proposition, which is a combination of well-known results (see e.g. [22, 23]); we omit the proof.

Proposition 3.2 *The matrix-valued function $\Phi: I\times I\to\mathbb{R}^{n_x\times n_x}$ satisfies the following properties for $s,t\in I$:*

1. $\Phi(s;s)=\mathbb{I}$,
2. $\Phi(t;s)^{-1}=\Phi(s;t)$,
3. *for a fixed $s\in I$, the matrix function $\Phi(\cdot;s): I\to\mathbb{R}^{n_x\times n_x}$ satisfies the IVP*

$$\begin{cases}\frac{d}{dt}\Phi(t;s)=\mathbf{A}(t)\cdot\Phi(t;s),\\ \Phi(s;s)=\mathbb{I}.\end{cases}$$

In the case of a constant matrix of coefficients $\mathbf{A}$, the exponential matrix $e^{(t-s)\cdot\mathbf{A}}$ is the state-transition matrix associated with $\mathbf{S}$ [22, Chap. 1, Sect. 5], and as such it satisfies properties 1-3. For time-dependent coefficient and forcing-term matrices $\mathbf{A}(t)$ and $\mathbf{B}(t)$, we can formulate the solution for the first order linear ODE system (3.1) in terms of the matrix-valued function $\Phi(\cdot;t_k) : I \to \mathbb{R}^{n_x \times n_p}$ associated with $\mathbf{A}(t)$.

Theorem 3.3 *Let $\mathbf{A}(\cdot) \in \mathcal{C}^0(I;\mathbb{R}^{n_x\times n_x})$, $\mathbf{B}(\cdot) \in \mathcal{C}^0(I;\mathbb{R}^{n_x\times n_p})$ and $\Phi(\cdot;\cdot) \in \mathcal{C}^1(I \times I;\mathbb{R}^{n_x\times n_x})$ the state-transition matrix associated with $\mathbf{A}(t)$. Let $t_k \in I$ and $\mathbf{S}(t_k) = \mathbf{S}_k \in \mathbb{R}^{n_x\times n_p}$. Then, the solution at time $t \in I$ of the first order inhomogeneous linear ODE system (3.1) is given by*

$$\mathbf{S}(t) = \Phi(t;t_k) \cdot \left(\mathbf{S}_k + \int_{t_k}^{t} \Phi(t_k;\tau)\mathbf{B}(\tau)d\tau\right). \tag{3.6}$$

Proof We show that the matrix-valued function $\mathbf{S}$ defined as (3.6) satisfies the ODE system (3.1).

By differentiating $\mathbf{S}$ in time and using properties 2 and 3 of Proposition 3.2 we obtain

$$\begin{aligned}\dot{\mathbf{S}}(t) &= \left(\frac{d}{dt}\Phi(t;t_k)\right) \cdot \left(\mathbf{S}_k + \int_{t_k}^{t} \Phi(t_k;\tau)\mathbf{B}(\tau)d\tau\right) + \Phi(t;t_k) \cdot \left(\Phi(t_k;t)\mathbf{B}(t)\right)\\ &= \mathbf{A}(t) \cdot \Phi(t;t_k) \cdot \left(\mathbf{S}_k + \int_{t_k}^{t} \Phi(t_k;\tau)\mathbf{B}(\tau)d\tau\right) + \mathbb{I} \cdot \mathbf{B}(t)\\ &= \mathbf{A}(t) \cdot \mathbf{S} + \mathbf{B}(t).\end{aligned}$$

Therefore, the function $\mathbf{S}$ in (3.6) satisfies $\dot{\mathbf{S}}(t) = \mathbf{A}(t) \cdot \mathbf{S} + \mathbf{B}(t)$.

Moreover, using Property 1 of Proposition 3.2,

$$\mathbf{S}(t_k) = \Phi(t_k;t_k) \cdot \left(\mathbf{S}_k + \int_{t_k}^{t_k} \Phi(t_k;\tau)\mathbf{B}(\tau)d\tau\right) = \mathbb{I} \cdot \mathbf{S}_k,$$

thus $\mathbf{S}(t_k) = \mathbf{S}_k$. This completes the proof. □

Theorem 3.3 shows that, if the state-transition matrix $\Phi(t;t_k)$ can be computed (for all relevant values t_k, t), then the solution for the general ODE system (3.1) can be obtained, which in turn solves the problem (2.4) for the sensitivity matrix. An explicit expression for $\Phi(t;s)$ is given by a result from control theory: as proved in [12], the state-transition matrix $\Phi(t;s)$ associated with the continuous coefficient matrix $\mathbf{A}(t)$ can be expressed in terms of the *Peano-Baker series*:

$$\Phi(t;s) = \sum_{n=0}^{\infty} \mathbf{I}_n(t;s), \tag{3.7}$$

where the summands are defined recursively as

$$\mathbf{I}_0(t;s) = \mathbb{I},$$
$$\mathbf{I}_{n+1}(t;s) = \int_s^t \mathbf{A}(\tau)\mathbf{I}_n(\tau;s)d\tau. \tag{3.8}$$

In [12] it is also proved that, assuming $\|\mathbf{A}(\cdot)\|$ locally integrable on the closed time interval I containing s and t, the series (3.7) converges compactly on I. The proof is carried out by showing that the sequence of partial sums $\left\{\sum_{n=0}^{N} \mathbf{I}_n(t;s)\right\}_{N\geq 1}$ is Cauchy on every compact subset $J \subset I$.

Remark 1 We note that if $t = s$, then the integration interval in (3.8) has Lebesgue measure zero, from which it follows that $\mathbf{I}_n(s;s) = 0$ for $n \geq 1$ and $\Phi(s;s) = \mathbf{I}_0(s;s) = \mathbb{I}$.

Remark 2 As previously mentioned, for a constant matrix of coefficients $\mathbf{A}$ the state-transition matrix $\Phi(t;s)$ is given by $e^{(t-s)\cdot\mathbf{A}}$. In this case, it is possible to move the constant matrix $\mathbf{A}$ outside the integral in (3.8). This leads to the recursive definition

$$\mathbf{I}_n(t;s) = \frac{(t-s)^n}{n!} \cdot \mathbf{A}^n,$$

and thus

$$\Phi(t;s) = \sum_{n=0}^{\infty} \frac{(t-s)^n}{n!} \cdot \mathbf{A}^n = e^{(t-s)\cdot\mathbf{A}}.$$

Consider the special case where the solution $\mathbf{x}$ of the underlying ODE (2.1) has reached an equilibrium point $\mathbf{x}_{\text{eq}}$. At equilibrium we have

$$f(t, \mathbf{x}_{\text{eq}}, \mathbf{p}) = 0. \tag{3.9}$$

In this case, the gradients $\nabla_x f(t, \mathbf{x}, \mathbf{p})$ and $\nabla_p f(t, \mathbf{x}, \mathbf{p})$ have constant arguments $(t, \mathbf{x}_{\text{eq}}, \mathbf{p})$, and thus become constant matrices. To simplify notation, we will omit their arguments and denote these matrices as $\nabla_x f := \nabla_x f(t, \mathbf{x}_{\text{eq}}, \mathbf{p})$ and $\nabla_p f := \nabla_p f(t, \mathbf{x}_{\text{eq}}, \mathbf{p})$.

Under this assumption on $\mathbf{x}$, the ODE system (2.4) for $\mathbf{S}$ is a first order linear ODE system with constant matrix of coefficients and constant forcing term. The solution is therefore provided by Theorem 3.1, with $\mathbf{A} = \nabla_x f$ and $\mathbf{B} = \nabla_p f$.

Corollary 3.4 *Suppose that the solution $\{\mathbf{x}(t)\}_{t\in I}$ of (2.1) at time step t_k has reached an equilibrium point $\mathbf{x}_{\text{eq}}$. Assume that $\nabla_x f(t, \mathbf{x}_{\text{eq}}, \mathbf{p})$ is invertible. The solution of (2.4), at time step t_{k+1} is then given by*

$$\mathbf{S}(t_{k+1}) = e^{\Delta t_k \cdot \nabla_x f} \cdot \left(\mathbf{S}_k + (\mathbb{I} - e^{-\Delta t_k \cdot \nabla_x f}) \cdot (\nabla_x f)^{-1} \cdot \nabla_p f\right). \tag{3.10}$$

where we denote $\Delta t_k = t_{k+1} - t_k$.

In the following we will refer to (3.10) as the *exponential formula*, as it makes use of the exponential matrix for the state-transition matrix.

This exact solution can be used as an approximation for $\mathbf{S}(t_{k+1})$ also when $\mathbf{x}(t)$ is not at an equilibrium point, with the approximation improving as $\mathbf{x}(t)$ moves closer to an equilibrium point.

In the following section we will show how we can use Theorem 3.3 to design a formula to approximate the sensitivity matrix $\mathbf{S}(t)$ in the general case (when the solution $\mathbf{x}$ is not at equilibrium).

4 Numerical approximations of S based on the Peano-Baker series

For the problem of finding the sensitivities $\mathbf{S}$ for the solution of the parametrised IVP (2.1), Theorem 3.3 gives a general representation of $\mathbf{S}$. Equipped with this representation, and Corollary 3.4 for the special case when $\mathbf{x}$ has reached an equilibrium point, we are now ready to construct a numerical method for approximating the sensitivity matrix $\mathbf{S}$.

In general we cannot use the assumption of the solution of (2.1) being at equilibrium, as in Sect. 3. That is, in general the matrix of coefficients $\nabla_x f(t, \mathbf{x}, \mathbf{p})$ and the forcing term $\nabla_p f(t, \mathbf{x}, \mathbf{p})$ are not constant matrices. In fact, in the transient of the dynamical system (2.1), there could be drastic variations of these objects. Furthermore, not all systems converge to an equilibrium point (where we can eventually use (3.10) to compute $\mathbf{S}(t_{k+1})$). For example, it is not unusual in systems biology to have the solution $\mathbf{x}(t)$ of (2.1) converge, as $t \to \infty$, to a limit cycle. In such cases, the matrices $\nabla_x f(t, \mathbf{x}, \mathbf{p})$ and $\nabla_p f(t, \mathbf{x}, \mathbf{p})$ are in general never constant. In principle, the trajectory $\mathbf{x}(t)$ could also show a chaotic behaviour, although this is rare in biological systems.

As previously mentioned, the n_x-dimensional ODE problem (2.1) can be solved rather efficiently with standard off-the-shelf ODE solvers. We therefore assume to have available approximations $\hat{\mathbf{x}}_k$ of $\mathbf{x}(t_k)$ at time steps t_k, $k = 1, \ldots, K$. For approximating the sensitivity matrix $\mathbf{S}$, we want to take advantage of the available $\hat{\mathbf{x}}_k$'s and avoid to exploit again an ODE solver for approximating $\mathbf{x}(t)$ at intermediate times $t \notin \{t_0, \ldots, t_K\}$. In addition to a solution $\mathbf{x}$ of (2.1), we assume to have access to the exact expression for the gradients $\nabla_x f(t, \mathbf{x}, \mathbf{p})$ and $\nabla_p f(t, \mathbf{x}, \mathbf{p})$.

For approximating $\mathbf{S}$, we propose a method that is based on approximating the state-transition matrix $\Phi(t;s)$ starting from the Peano-Baker series (3.7). More specifically, the method can be divided into approximating the integrals in (3.6) and (3.8), for which we can apply numerical integration, and approximating the series (3.7) itself, for which we use a truncation that is in a sense optimal (explained in more detail below and in Sect. 7).

For approximating the integrals, because we know approximations of the values $\mathbf{x}_k$ for $k = 1, \ldots, K$, the endpoints of the integration intervals, we apply the trapezoidal rule. In each of the recursive integrals $\mathbf{I}_n$ ($n \geq 1$) in (3.8), the integrands are of the form

$$\nabla_x f(\tau, \mathbf{x}, \mathbf{p}) \cdot \mathbf{I}_{n-1}(\tau;s), \tag{4.1}$$

with $s = t_k$ when we want to compute $\mathbf{S}(t_{k+1})$ given $\mathbf{S}(t_k)$ (based on (3.6)). The trapezoidal rule requires us to evaluate (4.1) for $\tau = t_k$ and $\tau = t_{k+1}$. To this end, consider the approximated solutions $\hat{\mathbf{x}}_k$ and $\hat{\mathbf{x}}_{k+1}$ at time steps t_k and t_{k+1}, respectively, given by the ODE solver. Inserting these into the gradients results in the approximations

$$\begin{aligned} \nabla_x \hat{f}_k &:= \nabla_x f(t_k, \hat{\mathbf{x}}_k, \mathbf{p}) \approx \nabla_x f(t_k, \mathbf{x}, \mathbf{p}), \\ \nabla_x \hat{f}_{k+1} &:= \nabla_x f(t_{k+1}, \hat{\mathbf{x}}_{k+1}, \mathbf{p}) \approx \nabla_x f(t_{k+1}, \mathbf{x}, \mathbf{p}). \end{aligned} \tag{4.2}$$

Moreover, we recall that $\mathbf{I}_0(t_k;t_k) = \mathbb{I}$ (from Property 1 of Proposition 3.2) and $\mathbf{I}_{n\geq 1}(t_k;t_k) = 0$ (from Remark 1). Combining these with the trapezoidal rule gives the approximations,

$$\begin{aligned} \hat{\mathbf{I}}_{0,k} &:= \mathbb{I}, \\ \hat{\mathbf{I}}_{1,k} &:= \frac{\Delta t_k}{2} \cdot \left(\nabla_x \hat{f}_k + \nabla_x \hat{f}_{k+1}\right), \\ \hat{\mathbf{I}}_{n+1,k} &:= \frac{\Delta t_k}{2} \cdot \left(0 + \nabla_x \hat{f}_{k+1} \cdot \hat{\mathbf{I}}_{n,k}\right), \quad n \geq 2, \end{aligned} \tag{4.3}$$

for $\mathbf{I}_0(t_{k+1};t_k)$, $\mathbf{I}_1(t_{k+1};t_k)$ and $\mathbf{I}_{n+1}(t_{k+1};t_k)$, $n \geq 2$, respectively.

A study of the error introduced by the various approximations, presented in detail in Sects. 6-7, shows that when the trapezoidal rule is used for the integrals in the Peano-Baker series, it is reasonable to truncate the series at $n = 2$. That is, for $k = 0, 1, \ldots, K-1$, we approximate $\Phi(t_{k+1};t_k)$ by

$$\hat{\Phi}(t_{k+1};t_k) := \hat{\mathbf{I}}_{0,k} + \hat{\mathbf{I}}_{1,k} + \hat{\mathbf{I}}_{2,k}. \tag{4.4}$$

In order to treat the integral in (3.6), where the integrand is

$$\Phi(t_k;\tau) \cdot \nabla_p f(\tau, \mathbf{x}, \mathbf{p}),$$

a similar strategy can be used. Similar to (4.2), we define the approximations of the gradients with respect to $\mathbf{p}$ as

$$\begin{aligned} \nabla_p \hat{f}_k &:= \nabla_p f(t_k, \hat{\mathbf{x}}_k, \mathbf{p}) \approx \nabla_p f(t_k, \mathbf{x}, \mathbf{p}), \\ \nabla_p \hat{f}_{k+1} &:= \nabla_p f(t_{k+1}, \hat{\mathbf{x}}_{k+1}, \mathbf{p}) \approx \nabla_p f(t_{k+1}, \mathbf{x}, \mathbf{p}). \end{aligned} \tag{4.5}$$

By the same reasoning as for (4.4), we can truncate the Peano-Baker series at $n = 2$ and approximate the terms of the truncated series by

$$\begin{aligned} \mathbf{I}_0(t_k;t_{k+1}) &= \mathbb{I} = \hat{\mathbf{I}}_{0,k}, \\ \mathbf{I}_1(t_k;t_{k+1}) &\approx \frac{-\Delta t_k}{2}\left(\nabla_x \hat{f}_k + \nabla_x \hat{f}_{k+1}\right) = -\hat{\mathbf{I}}_{1,k}, \\ \mathbf{I}_2(t_k;t_{k+1}) &\approx \frac{-\Delta t_k}{2}\left(0 + \nabla_x \hat{f}_{k+1} \cdot (-\hat{\mathbf{I}}_{1,k})\right) = \hat{\mathbf{I}}_{2,k}. \end{aligned}$$

The resulting approximation $\hat{\Phi}(t_k;t_{k+1})$ of $\Phi(t_k;t_{k+1})$ is then defined as

$$\hat{\Phi}(t_k;t_{k+1}) := \hat{\mathbf{I}}_{0,k} - \hat{\mathbf{I}}_{1,k} + \hat{\mathbf{I}}_{2,k}.$$

Note that the minus sign in front of the second term is due to the integration interval being reversed compared to (4.4).

Combining the approximation of $\Phi(t_k;t_{k+1})$ with the trapezoidal rule, and the property $\Phi(t_k;t_k) = \mathbb{I}$ (Property 1 of Proposition 3.2), the integral in (3.6) can be approximated as

$$\int_{t_k}^{t_{k+1}} \Phi(t_k;\tau)\nabla_p f(\tau, \mathbf{x}, \mathbf{p})d\tau \approx \frac{\Delta t_k}{2} \cdot \left(\nabla_p \hat{f}_k + \hat{\Phi}(t_k;t_{k+1}) \cdot \nabla_p \hat{f}_{k+1}\right).$$

Next, we combine this approximation with $\hat{\Phi}(t_{k+1};t_k)$ to obtain an approximation of the sensitivity matrix $\mathbf{S}$ at time t_{k+1}, given $\mathbf{S}(t_k)$:

$$\mathbf{S}(t_{k+1}) \approx \hat{\Phi}(t_{k+1};t_k) \cdot \left(\mathbf{S}(t_k) + \frac{\Delta t_k}{2} \cdot \left(\nabla_p \hat{f}_k + \hat{\Phi}(t_k;t_{k+1}) \cdot \nabla_p \hat{f}_{k+1}\right)\right). \tag{4.6}$$

The right-hand side of (4.6) is the approximation, based on the Peano-Baker series, that we use for the solutions $\mathbf{S}$ of the ODE problems (2.4). In the remainder of the paper, we will refer to (4.6) as the *Peano-Baker series (PBS) formula*, as it uses an approximation of the state-transition matrix based on the Peano-Baker series.

We are now ready to introduce the Peano-Baker Series (PBS) algorithm to approximate the sensitivity matrix $\mathbf{S}$ of the solution $\mathbf{x}$ of the ODE problem (2.1). The PBS algorithm, which is described in Algorithm 1, is based on the two formulas to approximate $\mathbf{S}$ that we have presented earlier: the *PBS formula* (4.6) is used in the general case, and the *exponential formula* (3.10) at equilibrium.

In Sect. 8.1 we discuss some criteria for the if-statement in Algorithm 1, i.e. when it is proper to use the *exponential formula* (3.10) to approximate $\mathbf{S}(t_{k+1})$.

Remark 3 Note that the exact solution (3.6) to the ODE system (3.1) can be rewritten as

$$\mathbf{S}(t) = \Phi(t;t_k) \cdot \mathbf{S}_k + \int_{t_k}^{t} \Phi(t;\tau)\mathbf{B}(\tau)d\tau.$$

Based on this formula we can define the following alternative approximation for the sensitivity matrix $\mathbf{S}$:

$$\mathbf{S}(t_{k+1}) \approx \hat{\Phi}(t_{k+1};t_k) \cdot \mathbf{S}(t_k) + \frac{\Delta t_k}{2} \cdot \left(\hat{\Phi}(t_{k+1};t_k)\nabla_p \hat{f}_k + \nabla_p \hat{f}_{k+1}\right). \tag{4.7}$$

A method based on this alternative approximation leads to a global error of the same order as the PBS method (Algorithm 1), which is based on the PBS formula (4.6). We do not investigate further the use of (4.7) for constructing numerical methods in this

Algorithm 1 Peano-Baker Series (PBS) algorithm for the sensitivity matrix $\mathbf{S}$

Require: $\hat{\mathbf{x}}_k \approx \mathbf{x}(t_k), \quad k = 1, \ldots, K$
Ensure: $\hat{\mathbf{S}}_1, \ldots, \hat{\mathbf{S}}_K$
$\nabla_x \hat{f}_k = \nabla_x f(t_k, \hat{\mathbf{x}}_k, \mathbf{p})$
$\nabla_p \hat{f}_k = \nabla_p f(t_k, \hat{\mathbf{x}}_k, \mathbf{p})$
$\hat{\mathbf{S}}_0 = 0 \in \mathbb{R}^{n_x \times n_p}$
for $k \in \{0, \ldots, K-1\}$ **do**
  $\Delta t_k = t_{k+1} - t_k$
  if $\nabla_x f \approx$ const **and** $\nabla_p f \approx$ const **then**
    $\hat{\mathbf{S}}_{k+1} = e^{\Delta t_k \cdot \nabla_x \hat{f}_k} \cdot \left(\hat{\mathbf{S}}_k + \left(\mathbb{I} - e^{-\Delta t_k \cdot \nabla_x \hat{f}_k}\right) \cdot (\nabla_x \hat{f}_k)^{-1} \cdot \nabla_p \hat{f}_k\right)$
  else
    $\hat{\mathbf{I}}_{0,k} = \mathbb{I}_{n_x}$
    $\hat{\mathbf{I}}_{1,k} = \frac{\Delta t_k}{2} \cdot \left(\nabla_x \hat{f}_k + \nabla_x \hat{f}_{k+1}\right)$
    $\hat{\mathbf{I}}_{2,k} = \frac{\Delta t_k^2}{4} \cdot \left(\nabla_x \hat{f}_{k+1} \cdot (\nabla_x \hat{f}_k + \nabla_x \hat{f}_{k+1})\right)$
    $\hat{\Phi}(t_{k+1}; t_k) = \hat{\mathbf{I}}_{0,k} + \hat{\mathbf{I}}_{1,k} + \hat{\mathbf{I}}_{2,k}$
    $\hat{\Phi}(t_k; t_{k+1}) = \hat{\mathbf{I}}_{0,k} - \hat{\mathbf{I}}_{1,k} + \hat{\mathbf{I}}_{2,k}$
    $\hat{\mathbf{S}}_{k+1} = \hat{\Phi}(t_{k+1}; t_k) \cdot \left(\hat{\mathbf{S}}_k + \frac{\Delta t_k}{2} \cdot \left(\nabla_p \hat{f}_k + \hat{\Phi}(t_k; t_{k+1}) \cdot \nabla_p \hat{f}_{k+1}\right)\right)$
  end if
end for

paper. Rather, we will only consider the PBS formula (4.6) and, without confusion, "PBS algorithm" refers to Algorithm 1.

5 Algorithms to approximate S for stiff problems

The PBS algorithm presented in Sect. 4 has the disadvantage that the PBS formula (4.6), which the method relies on, can be unstable in certain regions when applied to stiff problems. Moreover, stiffness of the relevant ODEs is common in the applications that first motivated our interest in methods for computing sensitivities. To overcome this potential stability issue, the PBS approximation should be implemented on small time steps; in general, these time steps should be (much) smaller than the quantities $\Delta t_k = t_{k+1} - t_k$ returned by the ODE solver applied to the system (2.1). With this in mind, we design a stiffness detection mechanism and refine the time steps where the PBS formula (4.6) would otherwise be unstable. To distinguish this new method, with the stiffness detection, from the PBS algorithm (Algorithm 1), we refer to it as the *PBS algorithm with refinement (PBSR)*. An alternative way to overcome the issue of stiffness is to never use the PBS formula (4.6) and, instead, use the exponential formula (3.10), which is unconditionally stable. We refer to the corresponding algorithm as the *Exponential (Exp) algorithm*.

Because of the stability concerns, the PBSR and the Exp algorithms are the methods that are implemented in practice, rather than the PBS algorithm (Algorithm 1). In the following sections we will give precise definitions of the PBSR and Exp algorithms, along with their properties, and discuss the computational cost associated with them.

5.1 The PBSR and exp algorithms

The PBS algorithm relies on the the PBS formula (4.6), which is based on a truncation of the Peano-Baker series (3.7). Because of the truncation, the PBS formula (4.6) can be unstable. Therefore, in the case of stiff problems, it can only be applied on "small enough" time steps.

The approach to approximate the sensitivity matrix $\mathbf{S}$ that we propose in this paper consists in first solving the ODE system (2.1), and then approximate the sensitivities given the numerical ODE solution $\{\hat{\mathbf{x}}_k\}$ and the time steps $\{t_k\}$ returned by the ODE solver. However, the time intervals $[t_k, t_{k+1}]$ provided by the ODE solver can be too large for the PBS formula (4.6). A natural idea is then to detect what time intervals $[t_k, t_{k+1}]$ are too large and refine them into n_{int} uniformly spaced sub-intervals $\{[t_{k,i}, t_{k,i+1}]\}_{i=1}^{n_{\text{int}}}$ of size $\Delta t_k/n_{\text{int}}$, with $t_{k,i} = t_k + (i-1)\Delta t_k/n_{\text{int}}$, $i = 1, \ldots, n_{\text{int}}+1$, where we can apply the PBS formula (4.6). To implement (4.6), we also need an approximation of the state $\mathbf{x}$ on the finer time grid. For this purpose we linearly interpolate the ODE solution $\hat{\mathbf{x}}$ on the refinement; i.e. for $s \in (t_k; t_{k+1})$ we approximate $\mathbf{x}(s) \approx \hat{\mathbf{x}}_k + (s - t_k)\frac{\hat{\mathbf{x}}_{k+1} - \hat{\mathbf{x}}_k}{\Delta t_k}$.

The number n_{int} of sub-intervals should be large enough to guarantee that the PBS formula (4.6) is properly used, i.e., that the size $\Delta t_k/n_{\text{int}}$ of the sub-intervals is sufficiently small. In Sect. 8 we describe how we determine a good choice of n_{int} in our numerical examples in order to overcome stiffness.

Introducing a finer time grid comes with an increase in computational cost. To reduce this additional computational burden, on time intervals $[t_k, t_{k+1}]$ where n_{int} becomes prohibitively large, we avoid the refinement. For such intervals, rather than relying on the PBS formula (4.6), which would come with stability issues, we use the exponential formula (3.10): although in general less accurate than the PBS formula, an advantage of the exponential formula is that it is stable independently of the value of Δt_k. The use of the exponential formula, as opposed of refining and using (4.6), is triggered in two situations, where the second one was already used in Algorithm 1:

(i) if n_{int} is "too large", and therefore the refinement would lead to a prohibitive computational cost;
(ii) if $\nabla_x f$ and $\nabla_p f$ are approximately constant on $[t_k, t_{k+1}]$, and therefore the use of the exponential formula (3.10) is motivated by Corollary 3.4.

The method just described is the PBS algorithm with refinement (PBSR) and its pseudo-code is presented in Algorithm 2. In Sect. 8 we describe the definitions of the conditions (i), that is the stiffness detection mechanism, and (ii) that are implemented in our numerical experiments; see specifically Sect. 8.1. This mechanism allows us to overcome the problem of stiffness, making the method robust. However, the exact definitions and implementation of the two conditions can vary, e.g., depending on what applications one has in mind or how one values speed (i.e., computational cost) vs. accuracy.

While the PBS formula (4.6) must be implemented on small time steps when the problem is stiff, the exponential formula (3.10) is stable regardless of the size of the time intervals $[t_k, t_{k+1}]$. Thus, another way of handling the problem of stiffness consists in using the exponential formula (3.10) at each time step, without refining

Algorithm 2 Peano-Baker Series algorithm with Refinement (PBSR) for the sensitivity matrix $\mathbf{S}$

Require: $\hat{\mathbf{x}}_k \approx \mathbf{x}(t_k), \quad k = 1, \ldots, K$
Ensure: $\hat{\mathbf{S}}_1, \ldots, \hat{\mathbf{S}}_K$
$\nabla_x \hat{f}_k = \nabla_x f(t_k, \hat{\mathbf{x}}_k, \mathbf{p})$
$\nabla_p \hat{f}_k = \nabla_p f(t_k, \hat{\mathbf{x}}_k, \mathbf{p})$
$\hat{\mathbf{S}}_0 = 0 \in \mathbb{R}^{n_x \times n_p}$
for $k \in \{0, \ldots, K-1\}$ **do**
  $\Delta t_k = t_{k+1} - t_k$
  estimate n_{int}
  if n_{int} "too large" **or** $\nabla_x f \approx \text{const}$ **and** $\nabla_p f \approx \text{const}$ **then**
    $\hat{\mathbf{S}}_{k+1} = e^{\Delta t_k \cdot \nabla_x \hat{f}_k} \cdot \left(\hat{\mathbf{S}}_k + \left(I - e^{-\Delta t_k \cdot \nabla_x \hat{f}_k}\right) \cdot \nabla_x \hat{f}_k^{-1} \cdot \nabla_p \hat{f}_k\right)$
  else
    $\hat{\mathbf{S}}_{\text{temp}} = \hat{\mathbf{S}}_k$
    for $h \in \{0, \ldots, n_{\text{int}} - 1\}$ **do**
      $t_a = t_k + \frac{h}{n_{\text{int}}}(t_{k+1} - t_k)$
      $t_b = t_k + \frac{h+1}{n_{\text{int}}}(t_{k+1} - t_k)$
      $\Delta t = t_b - t_a$
      $\hat{\mathbf{x}}_a = \hat{\mathbf{x}}_k + \frac{h}{n_{\text{int}}}(\hat{\mathbf{x}}_{k+1} - \hat{\mathbf{x}}_k)$
      $\hat{\mathbf{x}}_b = \hat{\mathbf{x}}_k + \frac{h+1}{n_{\text{int}}}(\hat{\mathbf{x}}_{k+1} - \hat{\mathbf{x}}_k)$
      $\nabla_x \hat{f}_a = \nabla_x f(t_a, \hat{\mathbf{x}}_a, \mathbf{p})$
      $\nabla_x \hat{f}_b = \nabla_p f(t_b, \hat{\mathbf{x}}_b, \mathbf{p})$
      $\nabla_p \hat{f}_a = \nabla_p f(t_a, \hat{\mathbf{x}}_a, \mathbf{p})$
      $\nabla_p \hat{f}_b = \nabla_p f(t_b, \hat{\mathbf{x}}_b, \mathbf{p})$
      $\hat{\mathbf{I}}_1 = \frac{\Delta t}{2} \cdot \left(\nabla_x \hat{f}_a + \nabla_x \hat{f}_b\right)$
      $\hat{\mathbf{I}}_2 = \frac{\Delta t^2}{4} \cdot \left(\nabla_x \hat{f}_b \cdot (\nabla_x \hat{f}_a + \nabla_x \hat{f}_b)\right)$
      $\hat{\Phi}(t_b; t_a) = I + \hat{\mathbf{I}}_1 + \hat{\mathbf{I}}_2$
      $\hat{\Phi}(t_a; t_b) = I - \hat{\mathbf{I}}_1 + \hat{\mathbf{I}}_2$
      $\hat{\mathbf{S}}_{\text{temp}} = \hat{\Phi}(t_b; t_a) \cdot \left(\hat{\mathbf{S}}_{\text{temp}} + \frac{\Delta t}{2} \cdot \left(\nabla_p \hat{f}_a + \hat{\Phi}(t_a; t_b) \cdot \nabla_p \hat{f}_b\right)\right)$
    end for
    $\hat{\mathbf{S}}_{k+1} = \hat{\mathbf{S}}_{\text{temp}}$
  end if
end for

the time intervals. This corresponds to treating the Jacobians *as if* they were constant on each time interval $[t_k, t_{k+1}]$, and therefore, by Corollary 3.4, we can express the (approximate) solution for the sensitivity matrix using the exponential formula (3.10). The corresponding method is the Exp algorithm [21], whose pseudo-code is displayed in Algorithm 3.

Algorithm 3 Exponential (Exp) algorithm for the sensitivity matrix $\mathbf{S}$

Require: $\hat{\mathbf{x}}_k \approx \mathbf{x}(t_k), \quad k = 1, \ldots, K$
Ensure: $\hat{\mathbf{S}}_1, \ldots, \hat{\mathbf{S}}_K$
$\nabla_x \hat{f}_k = \nabla_x f(t_k, \hat{\mathbf{x}}_k, \mathbf{p}), \quad \nabla_p \hat{f}_k = \nabla_p f(t_k, \hat{\mathbf{x}}_k, \mathbf{p})$
$\hat{\mathbf{S}}_0 = 0 \in \mathbb{R}^{n_x \times n_p}$
for $k \in \{0, \ldots, K-1\}$ **do**
  $\Delta t_k = t_{k+1} - t_k$
  $\hat{\mathbf{S}}_{k+1} = e^{\Delta t_k \cdot \nabla_x \hat{f}_k} \cdot \left(\hat{\mathbf{S}}_k + \left(I - e^{-\Delta t_k \cdot \nabla_x \hat{f}_k}\right) \cdot \nabla_x \hat{f}_k^{-1} \cdot \nabla_p \hat{f}_k\right)$
end for

Note that in the case of severe stiffness, the PBSR algorithm can trigger the use of the exponential formula (3.10) in most of the time intervals; consequently, in such cases the PBSR algorithm is equivalent to the Exp algorithm.

5.2 Computational cost and relative accuracy

The PBSR and Exp algorithms are, along with the FS method, the methods we implement in our numerical experiments (see Sect. 8). To complement this numerical study, it is of interest to analyze the computational cost of the three algorithms, and compare them in terms of accuracy.

The FS method is the most expensive of the three algorithms. In particular, the larger the dimension n_p of the parameter space is, the less efficient the FS method becomes when compared to the other two. The reason is that the FS method is based on solving an ODE system of dimension $n_x \times (n_p + 1)$, because it couples the ODE for the state $\mathbf{x}$ (of dimension n_x) with the ODE for the sensitivity matrix $\mathbf{S}$ (of dimension $n_x \times n_p$). In contrast, the PBSR and Exp methods only require solving the (smaller) n_x-dimensional ODE system for $\mathbf{x}$ (which in general requires a coarser time grid compared to the ODE system in the FS method), as well as performing some additional operations on each of the time intervals $[t_k, t_{k+1}]$ that are returned by the ODE solver. The cost of these additional operations along with solving the n_x-dimensional ODE system does not outweigh the cost of solving the $n_x \times (n_p + 1)$-dimensional ODE system in the FS method; the larger n_p is, the more expensive FS becomes, when compared to the PBSR and Exp methods.

To better understand, and compare, the cost of the PBSR and Exp methods, we now outline the operations used by the two methods over each ODE time interval $[t_k, t_{k+1}]$; we omit cheaper computations, such as scalar multiplication and matrix summation.
Exp algorithm. In this method the exponential formula (3.10) is applied on each time interval. The (expensive) operations associated with (3.10) are:

- 1 matrix-matrix multiplication (dimension $n_x \times n_x$),
- 1 matrix exponentiation (dimension $n_x \times n_x$),
- solving 1 linear system $\nabla_x f \cdot v = \nabla_p f$, where the dimension of $\nabla_x f$ is $n_x \times n_x$, and the dimension of $\nabla_p f$ is $n_x \times n_p$ (1 QR decomposition + n_p forward substitution sequences; or 1 LU decomposition + n_p substitution sequences).

PBSR algorithm. The cost of the PBSR method for each time interval depends on whether or not the current time interval is refined into finer sub-intervals:

(i) if the refinement is applied, the (expensive) operations are $3 \cdot n_{\text{int}}$ matrix-matrix multiplications (dimension $n_x \times n_x$), because the time interval is refined into n_{int} sub-intervals and the PBS formula (4.6) consists of 3 matrix-matrix multiplications;
(ii) if the exponential formula (3.10) is applied (and thus no refinement), the cost is the same as for the Exp algorithm (described above).

The outline of the (expensive) operations used for each method suggests that, in general, because of the refinement, the PBSR algorithm requires a larger number of

operations, and therefore we expect the Exp algorithm to be less computationally expensive. This will also be confirmed in the numerical experiments in Sect. 8.

The advantage of the PBSR over the Exp algorithm is the higher accuracy it provides. The reasons for this higher accuracy are (i) the refinement performed in the PBSR algorithm, and (ii) the higher accuracy of the PBS formula 4.6 (used in the PBSR algorithm) over the exponential formula 3.10: when the system is not at equilibrium and Δt_k is sufficiently small, the PBS formula (4.6) is more accurate than (3.10), as it takes into account the changes in $\nabla_x f$ and $\nabla_p f$ in the interval $[t_k, t_{k+1}]$, whereas only the values of the Jacobians at time t_k enter into the exponential formula (3.10).

Note that when the system reaches a steady state, the Jacobians $\nabla_x f$ and $\nabla_p f$ are (approximately) constant, and the error in the sensitivity matrix approximated with the Exp algorithm is of the same order as the PBSR approximation. This behaviour can also be observed in the numerical example in Sect. 8 where we approximate the sensitivity matrix in the model for the CaMKII signalling pathway, which reaches convergence in the time span considered (see Figure 3). If one is interested in the approximation of the sensitivities *only* at steady state, then the Exp algorithm is to be preferred, as it is cheaper and provides similar approximation to the PBSR algorithm (at convergence). However, in our applications (see Sect. 1.1) we need to compute the Fisher information (1.3), which requires the sensitivities at all time-points $\{t_l\}$ at which the state of the system is measured. These time-points are typically in the transient phase. Thus for our applications, and in general for cases where a good approximation of the sensitivity matrix is required in the transient phase, the PBSR algorithm is more suitable than the Exp algorithm, despite the slight increase in computational time.

We end this section with a brief discussion of another possible modification of the algorithms presented so far: an algorithm that performs the same refinement as the PBSR and applies the exponential formula (3.10) on each sub-interval. Although a natural idea to consider, we argue that it is not worthwhile to implement such an algorithm. In fact, because the PBS formula (4.6) is more accurate than the exponential formula (3.10), such a method becomes less accurate than the PBSR algorithm. Moreover, the three (expensive) operations in the exponential formula (3.10) (1 matrix-matrix multiplication, 1 matrix exponentiation, solving 1 linear system) are much more expensive than those in the PBS formula (4.6) (3 matrix-matrix multiplications), as can be seen from Table 1, where we compare the cost for the different operations. We do this in two situations corresponding to two of the models used in Sect. 8. The conclusion is that combining refinement of the time intervals with (only) the exponential formula comes with a higher computational cost, but no improvement in accuracy when compared to the PBSR.

6 Error analysis of the approximations of S

In this section we present the result of an error analysis for the approximation of the sensitivity matrix $\mathbf{S}$ obtained by applying the PBS formula (4.6) to the iteratively defined ODE systems (2.4), as $\max_k \Delta t_k \to 0$ (or equivalently, as $K \to \infty$).

Table 1 Relative numerical costs (calculation time ratio) between different operations applied to matrices of sizes 11×11 and 21×21 corresponding to Jacobians $\nabla_x f$ arising from the two models from systems biology described in Sect. 8 (PKA and CaMKII, respectively) listed in the first column. Operations: double precision general matrix-matrix multiplication (`dgemm`) (used as reference for each of the two models), matrix exponentiation via the *scale and square* algorithm (`expm`), solve linear system in place (`svx`), LU decomposition (`LU`). These numerical results are obtained from a C implementation using the GNU Scientific Library. The standard error is indicated in parentheses (concise error notation)

Model	Matrix dimension	`dgemm`	`expm`	`svx`	`LU`
PKA	11×11	1	5.96(31)	4.79(27)	0.74(03)
CaMKII	21×21	1	8.46(99)	3.22(79)	0.48(07)

The errors involved are represented as vectors or matrices that correspond to the difference between an exact term and its approximation. In order to provide error estimates, and conclude on the order of convergence, we will consider the norm $\|\cdot\|$ of the errors. Since we are working in finite-dimensional vector spaces, all norms are equivalent and the exact choice is irrelevant to the analysis. Recall also from Sect. 2 that we have assumed enough regularity of the problem so that existence and uniqueness of a solution of (2.1) is guaranteed.

To simplify the notation, we define $\mathbf{J}_k$ as

$$\mathbf{J}_k := \int_{t_k}^{t_{k+1}} \Phi(t_k;\tau) \cdot \nabla_p f(\tau, \mathbf{x}, \mathbf{p}) d\tau,$$

and the corresponding approximation

$$\hat{\mathbf{J}}_k := \frac{\Delta t_k}{2} \cdot (\nabla_p \hat{f}_k + \hat{\Phi}(t_k;t_{k+1}) \cdot \nabla_p \hat{f}_{k+1}).$$

Having established the notation, we can use Theorem 3.3 to formulate the exact solution of the ODE system (2.4) for the sensitivity matrix at time step t_{k+1} as

$$\mathbf{S}(t_{k+1}) = \Phi(t_{k+1};t_k) \cdot (\mathbf{S}(t_k) + \mathbf{J}_k), \tag{6.1}$$

while the approximate solution is defined as

$$\hat{\mathbf{S}}_{k+1} = \hat{\Phi}(t_{k+1};t_k) \cdot \left(\hat{\mathbf{S}}_k + \hat{\mathbf{J}}_k\right). \tag{6.2}$$

Figure 1 illustrates the terms and the corresponding errors in the approximation (6.2) in a graph. Each node in the graph represents a new source of error and the directed edges show the propagation of error from one node (an approximation) to another, as a result of the approximation being used instead of the exact quantity.

The main result of this section, Theorem 6.1, is a characterisation of the error in the approximation of $\mathbf{S}$, at time step $k+1$, in terms of the largest time step $\Delta t_{\max}$ used by the underlying ODE solver. Before we can state this result, we list the assumptions we

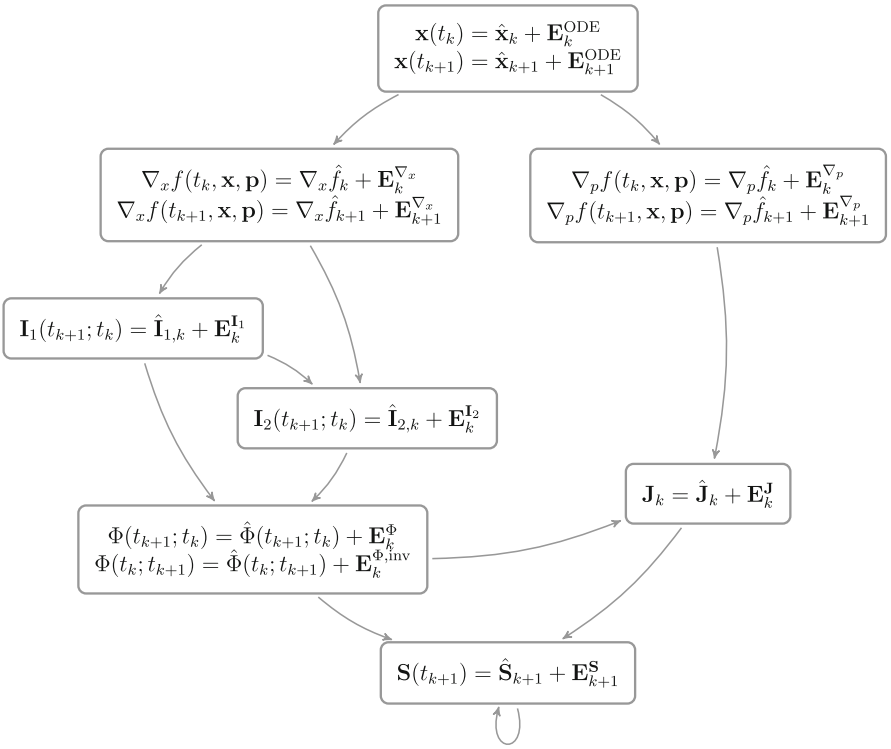

Fig. 1 Error propagation graph

make on the functions involved. The first assumption concerns the numerical solution $\hat{\mathbf{x}}_k$, at each time step t_k, of the ODE (2.1),

$$\mathbf{E}_k^{\text{ODE}} = \mathbf{x}(t_k) - \hat{\mathbf{x}}_k,$$

which is defined as an n_x-dimensional error vector. We make the following assumption on the global error of the underlying numerical method.

Assumption 2 The ODE solution of (2.1) has a global error of at least second order, i.e. for all $k = 0, \ldots, K-1$, $\left\|\mathbf{E}_k^{\text{ODE}}\right\| = \mathcal{O}(\Delta t_{\max}^2)$.

Next, for partial derivatives of f with respect to $\mathbf{x}$ and $\mathbf{p}$ we use the notation,

$$(H_{xx})_l^{jh} := \frac{\partial^2 f_l}{\partial x_h \partial x_j},$$
$$\left(H_{xp}\right)_l^{ih} := \frac{\partial^2 f_l}{\partial x_h \partial p_i}.$$

Assumption 3 The third order tensors $H_{xx}(t, \mathbf{x}, \mathbf{p})$ and $H_{xp}(t, \mathbf{x}, \mathbf{p})$ of second partial derivatives with respect to $\mathbf{x}$ and $\mathbf{p}$ are both bounded with respect to the supremum norm $\|\cdot\|_{[0,T]}$.

The fourth assumption concerns boundedness of certain maps with respect to $\|\cdot\|_{[0,T]}$.

Assumption 4 The following maps are bounded with respect to $\|\cdot\|_{[0,T]}$:

(4.1). $t \mapsto \nabla_x f(t, \mathbf{x}, \mathbf{p})$,
(4.2). $t \mapsto \nabla_p f(t, \mathbf{x}, \mathbf{p})$,
(4.3). $t \mapsto \frac{d^2}{dt^2} \nabla_x f(t, \mathbf{x}, \mathbf{p})$,
(4.4). $t \mapsto \frac{d^2}{dt^2} (\nabla_x f(t, \mathbf{x}, \mathbf{p}) \mathbf{I}_1(t; t_k))$,
(4.5). $t \mapsto \frac{d^2}{dt^2} \left(\Phi(t_k; t) \cdot \nabla_p f(t, \mathbf{x}, \mathbf{p})\right)$.

Assumption (4.1) is used in the study of the remainder term of the Peano-Baker series, after truncation at $n = 2$, and Assumption (4.2) in the proof of the error in the terms $\mathbf{J}_k$. The remaining assumptions (4.3)–(4.5) are used for arguments involving the trapezoidal rule, which explains why the second derivative with respect to time appears.

The final assumption concerns the sensitivity matrix $\mathbf{S}$ and the integrands in the $\mathbf{J}_k$s.

Assumption 5 There exist finite, positive constants $C_{\mathbf{S}}$ and $C_{\mathbf{J}}$, such that $C_{\mathbf{S}} \geq \|\mathbf{S}(t)\|$ for all $t \in [0, T]$, and $C_{\mathbf{J}} := \max_{k=0,\ldots,K-1} \|\Phi(t_k; t) \cdot \nabla_p f(t, \mathbf{x}, \mathbf{p})\|_{[0,T]}$.

The assumptions are stated not with the aim of full generality, but rather to provide enough regularity for the overall results to not get obscured by technical details. It is clear that the assumptions can be made both more explicit and less restrictive if aimed at specific examples. As an example, the assumptions of Lemma 7.2 are satisfied if, for example, $f \in \mathcal{C}^3(\mathbb{R}, \mathbb{R}^{n_x}, \mathbb{R}^{n_p})$. Similarly, if the vector field f in the ODE (2.1) is not time-dependent, more explicit error estimates can be obtained. We leave such refinements of the results for future work considering more specific formulations of the underlying IVP (2.1).

We are now ready to state the main theorem on the error of the proposed approximation; the proof is carried out in Sect. 7.

Theorem 6.1 *Suppose Assumptions 2–5 hold. Given a discretization* $0 = t_0 < t_1 < \cdots < t_{K-1} < t_K = T$ *of the time interval* $[0, T]$, *let* $\Delta t_{\max} := \max_{h=0,\ldots,K-1} \Delta t_h$ *and* $\Delta t_{\min} := \min_{h=0,\ldots,K-1} \Delta t_h$, *and assume that there exists a finite constant* Ξ *such that* $\frac{\Delta t_{\max}}{\Delta t_{\min}} \leq \Xi$, *as* $\Delta t_{\max} \to 0$. *Then, the error* $\mathbf{E}^{\mathbf{S}}_{k+1}$ *in the approximation of the sensitivity matrix* $\mathbf{S}$ *at time step* t_{k+1} *obtained by applying the PBS formula* (4.6) *to the ODE systems* (2.4) *satisfies*

$$\left\| \mathbf{E}^{\mathbf{S}}_{k+1} \right\| = \mathcal{O}(\Delta t^2_{\max}), \tag{6.3}$$

as $\Delta t_{\max} \to 0$.

Theorem 6.1 covers the error associated with the PBS algorithm. It turns out that, if the map $t \mapsto \frac{d}{dt} f(t, \mathbf{x}, \mathbf{p})$ is bounded with respect to $\|\cdot\|_{[0,T]}$, the result holds also when a refinement of the time intervals Δt_k is used, and the PBS formula (4.6) is applied on the sub-intervals, as in the PBSR algorithm (Algorithm 2). This is formalised in the following corollary, the proof of which is also provided in Section 7.

Corollary 6.2 *Suppose Assumptions 2–5 hold and the map* $t \mapsto \frac{d}{dt} f(t, \mathbf{x}, \mathbf{p})$ *is bounded with respect to* $\|\cdot\|_{[0,T]}$. *Given a discretization* $0 = t_0 < t_1 < \cdots < t_{K-1} < t_K = T$ *of the time interval* $[0, T]$, *let* $\Delta t_{\max} := \max_{h=0,\ldots,K-1} \Delta t_h$ *and* $\Delta t_{\min} := \min_{h=0,\ldots,K-1} \Delta t_h$, *and assume that there exists a finite constant* Ξ *such that* $\frac{\Delta t_{\max}}{\Delta t_{\min}} \leq \Xi$, *as* $\Delta t_{\max} \to 0$. *Consider any refinement of the time intervals* $[t_h, t_{h+1}]$ *and linearly interpolate the numerical solution* $\hat{\mathbf{x}}$ *of the ODE system* (2.1) *on each sub-interval. Then, the error* $\mathbf{E}^{\mathbf{S}}_{k+1}$ *in the approximation of the sensitivity matrix* $\mathbf{S}$ *at time step* t_{k+1} *obtained by applying the PBS formula* (4.6) *to the ODE systems* (2.4) *defined on the refined time grid satisfies*

$$\left\| \mathbf{E}^{\mathbf{S}}_{k+1} \right\| = \mathcal{O}(\Delta t^2_{\max}), \tag{6.4}$$

as $\Delta t_{\max} \to 0$.

7 Proof of the error estimates for the approximations of S

In the previous section we stated Theorem 6.1 and Corollary 6.2 about the error estimates for the approximations of $\mathbf{S}$ using the PBS formula (4.6) on the ODE systems (2.4) (without and with refinement of the time intervals, respectively). Here we provide the associated proofs, starting from Theorem 6.1, for which we will use a series of lemmas, corresponding roughly to the directed edges in Figure 1. The first source of error comes from the numerical solution $\hat{\mathbf{x}}_k$, at each time step t_k, of the ODE (2.1), represented by the uppermost node in Figure 1. Under Assumption 2, the error $\mathbf{E}^{\text{ODE}}_k$ associated with $\hat{\mathbf{x}}_k$ is $\mathcal{O}(\Delta t^2_{\max})$.

Next, we consider the approximations of the Jacobians, defined in in (4.2) and (4.5). As shown by the two directed edges going from the uppermost node in Figure 1, the error $\mathbf{E}^{\text{ODE}}_k$ propagates to both these terms: the errors that arise in the Jacobians are due to the evaluation of the exact Jacobians $\nabla_x f$ and $\nabla_p f$ in the approximated solution $\hat{\mathbf{x}}_k$ instead of the exact counterpart $\mathbf{x}(t_k)$:

$$\begin{aligned} \mathbf{E}^{\nabla_x}_k &= \nabla_x f(t_k, \mathbf{x}, \mathbf{p}) - \nabla_x \hat{f}_k, \\ \mathbf{E}^{\nabla_p}_k &= \nabla_p f(t_k, \mathbf{x}, \mathbf{p}) - \nabla_p \hat{f}_k. \end{aligned}$$

The following result is standard and included for completeness.

Lemma 7.1 *Under Assumptions 2 and 3, the errors* $\mathbf{E}^{\nabla_x}_k$ *and* $\mathbf{E}^{\nabla_p}_k$ *are both of second order in* $\Delta t_{\max}$, *i.e.*

$$\left\| \mathbf{E}^{\nabla_x}_k \right\| = \mathcal{O}(\Delta t^2_{\max}) \quad \text{and} \quad \left\| \mathbf{E}^{\nabla_p}_k \right\| = \mathcal{O}(\Delta t^2_{\max}),$$

as $\Delta t_{\max} \to 0$.

Proof The errors $\mathbf{E}_k^{\nabla_x}$ and $\mathbf{E}_k^{\nabla_p}$ are matrices of dimension $n_x \times n_x$ and $n_x \times n_p$, respectively. For an arbitrary $h \in \{1, \ldots, n_x\}$, let $\left(\mathbf{E}_k^{\mathrm{ODE}}\right)_h$ denote the hth component of the n_x-dimensional error vector $\mathbf{E}_k^{\mathrm{ODE}}$. The entries of $\mathbf{E}_k^{\nabla_x}$ and $\mathbf{E}_k^{\nabla_p}$ are then given by

$$\begin{aligned}\left(\mathbf{E}_k^{\nabla_x}\right)_l^{\ j} &= \frac{\partial f_l(t_k, \mathbf{x}, \mathbf{p})}{\partial x_j} - \frac{\partial f_l(\hat{\mathbf{x}}_k)}{\partial x_j} \\ &= \frac{\partial^2 f_l(t_k, \mathbf{x}, \mathbf{p})}{\partial x_h \partial x_j} \cdot \left(\mathbf{x}(t_k) - \hat{\mathbf{x}}_k\right)_h + \mathcal{O}\left(\left\|\mathbf{E}_k^{\mathrm{ODE}}\right\|^2\right) \\ &= \left(H_{xx}(t, \mathbf{x}, \mathbf{p})\right)_l^{\ jh} \left(\mathbf{E}_k^{\mathrm{ODE}}\right)_h + \mathcal{O}\left(\left\|\mathbf{E}_k^{\mathrm{ODE}}\right\|^2\right),\end{aligned}$$

and

$$\begin{aligned}\left(\mathbf{E}_k^{\nabla_p}\right)_l^{\ i} &= \frac{\partial f_l(t_k, \mathbf{x}, \mathbf{p})}{\partial p_i} - \frac{\partial f_l(\hat{\mathbf{x}}_k)}{\partial p_i} \\ &= \frac{\partial^2 f_l(t_k, \mathbf{x}, \mathbf{p})}{\partial x_h \partial p_i} \cdot \left(\mathbf{x}(t_k) - \hat{\mathbf{x}}_k\right)_h + \mathcal{O}\left(\left\|\mathbf{E}_k^{\mathrm{ODE}}\right\|^2\right) \\ &= \left(H_{xp}(t, \mathbf{x}, \mathbf{p})\right)_l^{\ ih} \left(\mathbf{E}_k^{\mathrm{ODE}}\right)_h + \mathcal{O}\left(\left\|\mathbf{E}_k^{\mathrm{ODE}}\right\|^2\right).\end{aligned}$$

Given these forms for the entries, we obtain the following bounds,

$$\begin{aligned}\left\|\mathbf{E}_k^{\nabla_x}\right\| &\le C_1 \left\|H_{xx}(t_k, \mathbf{x}, \mathbf{p})\right\| \left\|\mathbf{E}_k^{\mathrm{ODE}}\right\| + \mathcal{O}\left(\left\|\mathbf{E}_k^{\mathrm{ODE}}\right\|^2\right), \\ \left\|\mathbf{E}_k^{\nabla_p}\right\| &\le C_2 \left\|H_{xp}(t_k, \mathbf{x}, \mathbf{p})\right\| \left\|\mathbf{E}_k^{\mathrm{ODE}}\right\| + \mathcal{O}\left(\left\|\mathbf{E}_k^{\mathrm{ODE}}\right\|^2\right),\end{aligned}$$

where the constants $C_1 > 0$ and $C_2 > 0$ depend on the specific dimensions n_x and n_p considered and on the choice of norms.

Under Assumption 3 on $H_{xx}(t, \mathbf{x}, \mathbf{p})$ and $H_{xp}(t, \mathbf{x}, \mathbf{p})$, there exist finite constants $C_{H_{xx}} := \|H_{xx}\|_{[0,T]}$ and $C_{H_{xp}} := \left\|H_{xp}\right\|_{[0,T]}$. By the definition of the supremum norm, we obtain uniform (with respect to time) bounds for the errors in the Jacobians:

$$\begin{aligned}\left\|\mathbf{E}_k^{\nabla_x}\right\| &\le C_1 C_{H_{xx}} \left\|\mathbf{E}_k^{\mathrm{ODE}}\right\| + \mathcal{O}\left(\left\|\mathbf{E}_k^{\mathrm{ODE}}\right\|^2\right), \\ \left\|\mathbf{E}_k^{\nabla_p}\right\| &\le C_2 C_{H_{xp}} \left\|\mathbf{E}_k^{\mathrm{ODE}}\right\| + \mathcal{O}\left(\left\|\mathbf{E}_k^{\mathrm{ODE}}\right\|^2\right).\end{aligned}$$

This shows that the norm of either error is of the same order as the error in the ODE solution, which by Assumption 2 is $\mathcal{O}(\Delta t_{\max}^2)$ as $\Delta t_{\max} \to 0$. □

Following the error propagation in Figure 1, the approximations of $\nabla_x f(t_k, \mathbf{x}, \mathbf{p})$ as $\nabla_x \hat{f}_k$, and its counterpart at time step $k+1$, are used to approximate the integrals $\mathbf{I}_1(t_{k+1};t_k)$ and $\mathbf{I}_2(t_{k+1};t_k)$, respectively, with $\hat{\mathbf{I}}_{1,k}$ and $\hat{\mathbf{I}}_{2,k}$, as defined in (4.3); note that the approximation $\hat{\mathbf{I}}_{1,k}$ is used in $\hat{\mathbf{I}}_{2,k}$ (see Figure 1). The errors that arise are a result of both the application of the trapezoidal rule for numerical integration and the errors in the Jacobians, as described by Lemma 7.1.

Lemma 7.2 *Suppose Assumptions 2, 3 and (4.3) hold. Then the error*

$$\mathbf{E}_k^{\wedge,\mathbf{I}_1} = \mathbf{I}_1(t_{k+1}; t_k) - \frac{\Delta t_k}{2} \cdot (\nabla_x f(t_k, \mathbf{x}, \mathbf{p}) + \nabla_x f(t_{k+1}, \mathbf{x}, \mathbf{p}))$$

due to the approximation of $\mathbf{I}_1(t_{k+1};t_k)$ *by the trapezoidal rule*[8] *is*

$$\left\| \mathbf{E}_k^{\wedge,\mathbf{I}_1} \right\| = \mathcal{O}(\Delta t_k^3), \tag{7.1}$$

and the error $\mathbf{E}_k^{\mathbf{I}_1}$*, from the approximation of* $\mathbf{I}_1(t_{k+1}; t_k)$ *by* $\hat{\mathbf{I}}_{1,k}$*, satisfies*

$$\left\| \mathbf{E}_k^{\mathbf{I}_1} \right\| = \mathcal{O}(\Delta t_{\max}^3), \tag{7.2}$$

as $\Delta t_{\max} \to 0$.

Proof It can be proved [27, Sec. 5.2.1] that the error of the trapezoidal rule applied to the integral of a function $g \in C^2(A, \mathbb{R})$, with $A \subset \mathbb{R}$, on the interval $[a, b] \subset A$ is

$$\int_a^b g(x)dx - \frac{b-a}{2} \cdot (g(a) + g(b)) = -\frac{(b-a)^3}{12} \cdot g''(\xi), \tag{7.3}$$

for some $\xi \in (a, b)$. Using the latter, the components of $\mathbf{E}_k^{\wedge,\mathbf{I}_1}$ are

$$\left(\mathbf{E}_k^{\wedge,\mathbf{I}_1}\right)_l^j = -\frac{\Delta t_k^3}{12} \cdot \left(\frac{d^2}{dt^2} \nabla_x f(t, \mathbf{x}, \mathbf{p})\right)_l^j \Bigg|_{t=\xi}, \tag{7.4}$$

where in general $\xi \in (t_k, t_{k+1})$ is different for each pair of indices l, j. From the properties of matrix norms, we have that for every $t \in [t_k, t_{k+1}]$ and $l, j = 1, \ldots, n_x$,

$$\left\| \left(\frac{d^2}{dt^2} \nabla_x f(t, \mathbf{x}, \mathbf{p})\right)_l^j \right\| \leq C \left\| \frac{d^2}{dt^2} \nabla_x f(t, \mathbf{x}, \mathbf{p}) \right\|.$$

for a constant $C > 0$ that only depends on the choice of norm $\|\cdot\|$.

[8] Indicated by the wedge symbol $\wedge$.

Using Assumption (4.3) for $\frac{d^2}{dt^2}\nabla_x f(t, \mathbf{x}, \mathbf{p})$, there exist a constant $C' < \infty$ such that $C' = \left\| \frac{d^2}{dt^2}\nabla_x f(t, \mathbf{x}, \mathbf{p}) \right\|_{[0,T]}$. It follows that, for every l, $j = 1, \ldots, n_x$,

$$\left\| \left(\mathbf{E}_k^{\wedge, \mathbf{I}_1} \right)_l^j \right\| \leq \frac{CC'}{12} \Delta t_k^3.$$

Using the fact that all matrix norms are equivalent, there exists a constant $C'' > 0$, which depends on the choice of norm, such that

$$\left\| \mathbf{E}_k^{\wedge, \mathbf{I}_1} \right\| \leq C'' \max_{l, j = 1, \ldots, n_x} \left\| \left(\mathbf{E}_k^{\wedge, \mathbf{I}_1} \right)_l^j \right\|.$$

Combined with the previous inequality this yields the following upper bound on the error from the trapezoidal rule,

$$\left\| \mathbf{E}_k^{\wedge, \mathbf{I}_1} \right\| \leq \frac{CC'C''}{12} \Delta t_k^3.$$

This proves (7.1).

Having established an order of convergence for the error due to the trapezoidal rule, we now expand the error arising from the approximation of $\mathbf{I}_1(t_{k+1}; t_k)$:

$$\begin{aligned}
\mathbf{E}_k^{\mathbf{I}_1} &= \mathbf{I}_1(t_{k+1}; t_k) - \hat{\mathbf{I}}_{1,k} \\
&= \mathbf{I}_1(t_{k+1}; t_k) - \frac{\Delta t_k}{2} \cdot (\nabla_x \hat{f}_k + \nabla_x \hat{f}_{k+1}) \\
&= \mathbf{I}_1(t_{k+1}; t_k) - \frac{\Delta t_k}{2} \cdot (\nabla_x f(t_k, \mathbf{x}, \mathbf{p}) + \nabla_x f(t_{k+1}, \mathbf{x}, \mathbf{p})) \\
&\quad + \frac{\Delta t_k}{2} \cdot \left(\nabla_x f(t_k, \mathbf{x}, \mathbf{p}) - \nabla_x \hat{f}_k + \nabla_x f(t_{k+1}, \mathbf{x}, \mathbf{p}) - \nabla_x \hat{f}_{k+1} \right) \\
&= \mathbf{E}_k^{\wedge, \mathbf{I}_1} + \frac{\Delta t_k}{2} \cdot (\mathbf{E}_k^{\nabla_x} + \mathbf{E}_{k+1}^{\nabla_x}).
\end{aligned}$$

Together with Lemma 7.1, this yields the upper bound

$$\left\| \mathbf{E}_k^{\mathbf{I}_1} \right\| \leq \left\| \mathbf{E}_k^{\wedge, \mathbf{I}_1} \right\| + \mathcal{O}(\Delta t_{\max}^3),$$

which proves the claimed order of convergence (7.2). □

With these estimates for the errors in $\nabla_x \hat{f}_k$ and $\hat{\mathbf{I}}_{1,k}$, we now turn to the approximation of $\mathbf{I}_{2,k}$. The first part of this lemma, concerning the error due to an application of the trapezoidal rule, is analogous to the first part of Lemma 7.2.

Lemma 7.3 *Suppose Assumptions 2, 3, (4.1)-(4.4) hold. Then, the error*

$$\mathbf{E}_k^{\wedge, \mathbf{I}_2} = \mathbf{I}_2(t_{k+1}; t_k) - \frac{\Delta t_k}{2} \cdot \nabla_x f(t_{k+1}, \mathbf{x}, \mathbf{p}) \cdot \mathbf{I}_1(t_{k+1}; t_k)$$

due to the approximation of $\mathbf{I}_2(t_{k+1};\, t_k)$ *by the trapezoidal rule is*

$$\left\| \mathbf{E}_k^{\wedge,\mathbf{I}_2} \right\| = \mathcal{O}(\Delta t_k^3), \tag{7.5}$$

and the error $\mathbf{E}_k^{\mathbf{I}_2}$*, from the approximation of* $\mathbf{I}_2(t_{k+1};t_k)$ *by* $\hat{\mathbf{I}}_{2,k}$*, is*

$$\left\| \mathbf{E}_k^{\mathbf{I}_2} \right\| = \mathcal{O}(\Delta t_{\max}^3), \tag{7.6}$$

as $\Delta t_{\max} \to 0$.

Proof The proof of (7.5) is analogous to the proof of Lemma 7.2: by replacing $\mathbf{E}_k^{\wedge,\mathbf{I}_1}$ with $\mathbf{E}_k^{\wedge,\mathbf{I}_2}$, and $\frac{d^2}{dt^2}\nabla_x f(t,\mathbf{x},\mathbf{p})$ with $\frac{d^2}{dt^2}\left(\nabla_x f(t,\mathbf{x},\mathbf{p})\mathbf{I}_1(t,t_k)\right)$, and defining

$$\tilde{C}' := \left\| \frac{d^2}{dt^2}\left(\nabla_x f(t,\mathbf{x},\mathbf{p})\mathbf{I}_1(t,t_k)\right) \right\|_{[0,T]},$$

we obtain

$$\left\| \mathbf{E}_k^{\wedge,\mathbf{I}_2} \right\| \leq \frac{C\tilde{C}'C''}{12}\Delta t_k^3.$$

This proves the order of convergence (7.5).

To show (7.6), we note that this error can be expressed as

$$\begin{aligned}
\mathbf{E}_k^{\mathbf{I}_2} &= \mathbf{I}_2(t_{k+1};t_k) - \hat{\mathbf{I}}_{2,k} \\
&= \mathbf{I}_2(t_{k+1};t_k) - \frac{\Delta t_k}{2}\cdot\nabla_x f(t_{k+1},\mathbf{x},\mathbf{p})\cdot\mathbf{I}_1(t_{k+1};t_k) \\
&\quad + \frac{\Delta t_k}{2}\cdot\left(\nabla_x f(t_{k+1},\mathbf{x},\mathbf{p})\cdot\mathbf{I}_1(t_{k+1};t_k) - \nabla_x \hat{f}_{k+1}\cdot\hat{\mathbf{I}}_{1,k}\right) \\
&= \mathbf{E}_k^{\wedge,\mathbf{I}_2} + \frac{\Delta t_k}{2}\cdot\Big(\nabla_x f(t_{k+1},\mathbf{x},\mathbf{p})\cdot\mathbf{I}_1(t_{k+1};t_k) - \nabla_x f(t_{k+1},\mathbf{x},\mathbf{p})\cdot\hat{\mathbf{I}}_{1,k} \\
&\qquad\qquad + \nabla_x f(t_{k+1},\mathbf{x},\mathbf{p})\cdot\hat{\mathbf{I}}_{1,k} - \nabla_x \hat{f}_{k+1}\cdot\hat{\mathbf{I}}_{1,k}\Big) \\
&= \mathbf{E}_k^{\wedge,\mathbf{I}_2} + \frac{\Delta t_k}{2}\cdot\left(\nabla_x f(t_{k+1},\mathbf{x},\mathbf{p})\cdot\mathbf{E}_k^{\mathbf{I}_1} + \mathbf{E}_{k+1}^{\nabla_x}\cdot\hat{\mathbf{I}}_{1,k}\right).
\end{aligned} \tag{7.7}$$

Applying the norm operator to (7.7) and using the triangle inequality gives

$$\left\| \mathbf{E}_k^{\mathbf{I}_2} \right\| \leq \left\| \mathbf{E}_k^{\wedge,\mathbf{I}_2} \right\| + \frac{\Delta t_k}{2}\cdot\left(\left\|\nabla_x f(t_{k+1},\mathbf{x},\mathbf{p})\right\|\left\| \mathbf{E}_k^{\mathbf{I}_1} \right\| + \left\| \mathbf{E}_{k+1}^{\nabla_x} \right\|\left\| \hat{\mathbf{I}}_{1,k} \right\|\right).$$

From Assumption (4.1), $\|\nabla_x f(t_{k+1},\mathbf{x},\mathbf{p})\| = \mathcal{O}(1)$ and, from the definition of $\hat{\mathbf{I}}_{1,k}$ (see (4.3)), we have $\left\|\hat{\mathbf{I}}_{1,k}\right\| = \mathcal{O}(\Delta t_k)$. From the upper bound in the last displayed formula, combined with Lemmas 7.1-7.2, we obtain

$$\left\|\mathbf{E}_k^{\mathbf{I}_2}\right\| \leq \left\|\mathbf{E}_k^{\wedge,\mathbf{I}_2}\right\| + \mathcal{O}(\Delta t_{\max}^4),$$

which, together with (7.5), proves the order of convergence (7.6). □

Next, we move to the approximation of $\Phi(t_{k+1};t_k)$. As illustrated in Figure 1, this approximation depends directly on the approximations of $\mathbf{I}_1(t_{k+1};t_k)$ and $\mathbf{I}_2(t_{k+1};t_k)$, and thus Lemmas 7.2-7.3 will be used to obtain the order of convergence of the associated error.

Lemma 7.4 *Suppose that Assumptions 2, 3 and (4.1)-(4.4) hold. Then, the error* $\mathbf{E}_k^{\Phi}$ *in the approximation of* $\Phi(t_{k+1};t_k)$ *by* $\hat{\Phi}(t_{k+1};t_k)$ *is*

$$\left\|\mathbf{E}_k^{\Phi}\right\| = \mathcal{O}(\Delta t_{\max}^3),$$

as $\Delta t_{\max} \to 0$.

Proof Recalling the definition (4.4) of the approximation $\hat{\Phi}(t_{k+1};t_k)$, we can express the associated $\mathbf{E}_k^{\Phi}$ as

$$\begin{aligned}
\mathbf{E}_k^{\Phi} &= \Phi(t_{k+1};t_k) - \mathbb{I}_{n_x} - \hat{\mathbf{I}}_{1,k} - \hat{\mathbf{I}}_{2,k} \\
&= \Phi(t_{k+1};t_k) - \sum_{n=0}^{2} \mathbf{I}_n(t_{k+1};t_k) \\
&\quad + \mathbb{I}_{n_x} + \mathbf{I}_1(t_{k+1};t_k) + \mathbf{I}_2(t_{k+1};t_k) - \mathbb{I}_{n_x} - \hat{\mathbf{I}}_{1,k} - \hat{\mathbf{I}}_{2,k} \\
&= \sum_{n=3}^{\infty} \mathbf{I}_n(t_{k+1};t_k) + \mathbf{E}_k^{\mathbf{I}_1} + \mathbf{E}_k^{\mathbf{I}_2},
\end{aligned} \tag{7.8}$$

Lemmas 7.2 and 7.3 describe the behaviour of $\mathbf{E}_k^{\mathbf{I}_1}$ and $\mathbf{E}_k^{\mathbf{I}_2}$ in terms of Δt_k. To study the term $\sum_{n=3}^{\infty} \mathbf{I}_n(t_{k+1};t_k)$, we recall from the definition (3.8) of the summands $\mathbf{I}_n(t_{k+1};t_k)$,

$$\mathbf{I}_n(t;s) = \int_s^t \nabla_x f(\tau, \mathbf{x}, \mathbf{p}) \mathbf{I}_{n-1}(\tau;s) d\tau, \quad n \geq 1.$$

Since $\mathbf{I}_0(t,s) = \mathbb{I}_{n_x}$, each such term can be expressed as

$$\mathbf{I}_n(t_{k+1};t_k) = \int_{t_k}^{t_{k+1}} \int_{t_k}^{\tau_1} \cdots \int_{t_k}^{\tau_{n-1}} \nabla_x f(\tau_n, \mathbf{x}, \mathbf{p}) \cdots \nabla_x f(\tau_1, \mathbf{x}, \mathbf{p}) d\tau_n \cdots d\tau_1.$$

Using this identity we can obtain an upper bound on the norm of $\sum_{n=3}^{\infty} \mathbf{I}_n(t_{k+1};t_k)$, the part of the sum that is removed in the truncation term:

$$\begin{aligned}
&\left\| \sum_{n=3}^{\infty} \mathbf{I}_n(t_{k+1};t_k) \right\| \leq \sum_{n=3}^{\infty} \|\mathbf{I}_n(t_{k+1};t_k)\| \\
&= \sum_{n=3}^{\infty} \left\| \int_{t_k}^{t_{k+1}} \int_{t_k}^{\tau_1} \cdots \int_{t_k}^{\tau_{n-1}} \nabla_x f(\tau_n, \mathbf{x}, \mathbf{p}) \cdots \nabla_x f(\tau_1, \mathbf{x}, \mathbf{p}) d\tau_n \cdots d\tau_1 \right\| \\
&\leq \sum_{n=3}^{\infty} \int_{t_k}^{t_{k+1}} \int_{t_k}^{\tau_1} \cdots \int_{t_k}^{\tau_{n-1}} \|\nabla_x f(\tau_n, \mathbf{x}, \mathbf{p}) \cdots \nabla_x f(\tau_1, \mathbf{x}, \mathbf{p})\| \, d\tau_n \cdots d\tau_1 \\
&\leq \sum_{n=3}^{\infty} \int_{t_k}^{t_{k+1}} \int_{t_k}^{\tau_1} \cdots \int_{t_k}^{\tau_{n-1}} \|\nabla_x f(\tau_n, \mathbf{x}, \mathbf{p})\| \cdots \|\nabla_x f(\tau_1, \mathbf{x}, \mathbf{p})\| \, d\tau_n \cdots d\tau_1 \\
&= \sum_{n=3}^{\infty} \frac{1}{n!} \left(\int_{t_k}^{t_{k+1}} \|\nabla_x f(\tau, \mathbf{x}, \mathbf{p})\| \, d\tau \right)^n \leq \sum_{n=3}^{\infty} \frac{1}{n!} (C_{\nabla_x})^n (\Delta t_k)^n, \qquad (7.9)
\end{aligned}$$

with $C_{\nabla_x} := \|\nabla_x f(t, \mathbf{x}, \mathbf{p})\|_{[0,T]}$, which is finite by Assumption (4.1). The last equality is due to the fact that the multiple integrals

$$\int_{t_k}^{t_{k+1}} \int_{t_k}^{\tau_1} \cdots \int_{t_k}^{\tau_{n-1}} \|\nabla_x f(\tau_n, \mathbf{x}, \mathbf{p})\| \cdots \|\nabla_x f(\tau_1, \mathbf{x}, \mathbf{p})\| \, d\tau_n \cdots d\tau_1,$$

can be seen as the summands of the Peano-Baker series for the one-dimensional ODE $\dot{z}(t) = \|\nabla_x f(t, \mathbf{x}, \mathbf{p})\| \cdot z(t)$; since the terms $\|\nabla_x f(\tau_i, \mathbf{x}, \mathbf{p})\|$ commute (being scalar functions), such multiple integrals are shown in [12] to be equal to the simpler terms $\frac{1}{n!} \left(\int_{t_k}^{t_{k+1}} \|\nabla_x f(\tau, \mathbf{x}, \mathbf{p})\| \, d\tau \right)^n$.

If we assume $\Delta t_k < 1$ (in fact, we consider $\Delta t_{\max} \to 0$), then $\Delta t_k^3 \geq \Delta t_k^n$ for every $n \geq 3$, and we obtain the upper bound

$$\left\| \sum_{n=3}^{\infty} \mathbf{I}_n(t_{k+1};t_k) \right\| \leq (\Delta t_k)^3 \sum_{n=3}^{\infty} \frac{1}{n!} (C_{\nabla_x})^n \leq \Delta t_k^3 \sum_{n \geq 0} \frac{1}{n!} (C_{\nabla_x})^n = \Delta t_k^3 e^{C_{\nabla_x}}. \qquad (7.10)$$

Taking the norm of (7.8), and using the latter upper bound for $\left\| \sum_{n=3}^{\infty} \mathbf{I}_n(t_{k+1};t_k) \right\|$, we obtain

$$\left\| \mathbf{E}_k^{\Phi} \right\| \leq \Delta t_k^3 e^{C_{\nabla_x}} + \left\| \mathbf{E}_k^{\mathbf{I}_1} \right\| + \left\| \mathbf{E}_k^{\mathbf{I}_2} \right\|,$$

where all three terms at the right-hand side are $\mathcal{O}(\Delta t_{\max}^3)$. This concludes the proof. □

Before we proceed with analysing the approximation of $\mathbf{J}_k$, which is the last term to consider before we move on to the approximation of $\mathbf{S}$ (the final node in Figure 1), we discuss the choice of truncating the Peano-Baker series at $n = 2$. Introducing additional terms in the approximation (i.e., truncating after a larger number of summands) would

lead to a higher power of Δt_k in the upper bound (7.10), which in turn would imply faster convergence. However, we rely on approximations of the summands in the Peano-Baker series rather than on the exact terms $\mathbf{I}_n$, and these approximations retain an error of order $\mathcal{O}(\Delta t_{\max}^3)$ (see Lemmas 7.2 and 7.3). Therefore, although including additional terms in the series would suggest a higher order of convergence, this would be cancelled by the $\mathcal{O}(\Delta t_{\max}^3)$ appearing in $\hat{\mathbf{I}}_{1,k}$ and $\hat{\mathbf{I}}_{2,k}$.

We could also opt to truncate the Peano-Baker series at $n = 1$ instead of $n = 2$, i.e. retaining only $\mathbb{I}_{n_x}$ and $\mathbf{I}_1(t_{k+1};t_k)$. In this case, the opposite situation would arise: we would lower the power of Δt_k in (7.10) to Δt_k^2, and the error $\mathbf{E}_k^{\Phi}$ would be $\mathcal{O}(\Delta t_{\max}^2)$. Thus, we would not benefit from the third order convergence of $\hat{\mathbf{I}}_{1,k}$. The conclusion is that if the integrals in $\mathbf{I}_1(t_{k+1};t_k)$ and $\mathbf{I}_2(t_{k+1};t_k)$ are approximated with the trapezoidal rule (which produces $\mathcal{O}(\Delta t_k^3)$ errors), then it is optimal to truncate the Peano-Baker series at $n = 2$; optimal here means obtaining the highest possible order of convergence with as few summands as possible.

We now move to the analysis of the approximation error associated with $\mathbf{J}_k$, the final error to consider before proving Theorem 6.1. Recalling the definition,

$$\mathbf{J}_k := \int_{t_k}^{t_{k+1}} \Phi(t_k;\tau) \cdot \nabla_p f(\tau, \mathbf{x}, \mathbf{p}) d\tau,$$

we note that an application of the trapezoidal rule will lead to the state-transition matrix $\Phi(t_k;t_{k+1})$–the inverse of $\Phi(t_{k+1};t_k)$ (see Proposition 3.2)– appearing. However, for greater efficiency, we compute $\Phi(t_k;t_{k+1})$ by the Peano-Baker series, instead of computing the inverse $\Phi(t_{k+1};t_k)^{-1}$. In particular, we observe that $\Phi(t_k;t_{k+1})$ can be obtained from $\Phi(t_{k+1};t_k)$ by interchanging the limits of integration in each term $\mathbf{I}_n$ (see (3.7) and (3.8)); interchanging the limits of integration, does not change the norm of an integral. Similarly, the approximations of the $\mathbf{I}_n$s by the trapezoidal rule are the same (up to the sign) for both $\Phi(t_{k+1};t_k)$ and $\Phi(t_k;t_{k+1})$; thus, they have the same norm. By replicating the arguments we applied to $\Phi(t_{k+1};t_k)$ throughout Lemmas 7.2-7.4, also on its inverse $\Phi(t_k;t_{k+1})$, we obtain the same bound for the error term $\mathbf{E}_k^{\Phi,\text{inv}}$: as $\Delta t_{\max} \to 0$,

$$\left\| \mathbf{E}_k^{\Phi,\text{inv}} \right\| = \left\| \Phi(t_k;t_{k+1}) - \hat{\Phi}(t_k;t_{k+1}) \right\| = \mathcal{O}(\Delta t_{\max}^3).$$

Lemma 7.5 *Suppose that Assumptions 2, 3, 4 hold. Then, the error* $\mathbf{E}_k^{\wedge,\mathbf{J}}$ *associated with the approximation of the integral* $\mathbf{J}_k$ *by the trapezoidal rule is*

$$\left\| \mathbf{E}_k^{\wedge,\mathbf{J}} \right\| = \mathcal{O}(\Delta t_k^3), \tag{7.11}$$

and the whole error in the approximation of $\mathbf{J}_k$ *with* $\hat{\mathbf{J}}_k$ *is*

$$\left\| \mathbf{E}_k^{\mathbf{J}} \right\| = \mathcal{O}(\Delta t_{\max}^3), \tag{7.12}$$

as $\Delta t_{\max} \to 0$.

Proof As in the proofs of Lemmas 7.2 and 7.3, we can invoke the error of the trapezoidal rule (7.3) and obtain

$$\left(\mathbf{E}_k^{\wedge,\mathbf{J}}\right)_i^j = -\frac{\Delta t_k^3}{12} \cdot \left(\frac{d^2}{dt^2}\left(\Phi(t_k;t) \cdot \nabla_p f(t,\mathbf{x},\mathbf{p})\right)\right)_i^j \Bigg|_{t=t_\xi},$$

for some $\xi \in (t_k;t_{k+1})$. The proof of (7.11) is now analogous to Lemmas 7.2 and 7.3 and we omit the details.

Moving to the error $\mathbf{E}_k^{\mathbf{J}}$, based on approximating the integral $\mathbf{J}_k$ with the trapezoidal rule, we first expand the error similar to what was done for $\mathbf{E}_k^{\mathbf{I}_1}$ and $\mathbf{E}_k^{\mathbf{I}_2}$:

$$\begin{aligned}
\mathbf{E}_k^{\mathbf{J}} &= \int_{t_k}^{t_{k+1}} \Phi(t_k;\tau) \cdot \nabla_p f(\tau,\mathbf{x},\mathbf{p}) d\tau - \frac{\Delta t_k}{2} \cdot (\nabla_p \hat{f}_k + \hat{\Phi}(t_k;t_{k+1}) \cdot \nabla_p \hat{f}_{k+1}) \\
&= \int_{t_k}^{t_{k+1}} \Phi(t_k;\tau) \cdot \nabla_p f(\tau,\mathbf{x},\mathbf{p}) d\tau \qquad (7.13) \\
&\quad - \frac{\Delta t_k}{2} \cdot (\nabla_p f(t_k,\mathbf{x},\mathbf{p}) + \Phi(t_k;t_{k+1}) \cdot \nabla_p f(t_{k+1},\mathbf{x},\mathbf{p}) \\
&\quad + \frac{\Delta t_k}{2} \cdot (\nabla_p f(t_k,\mathbf{x},\mathbf{p}) + \Phi(t_k;t_{k+1}) \cdot \nabla_p f(t_{k+1},\mathbf{x},\mathbf{p}) \\
&\quad - \frac{\Delta t_k}{2} \cdot (\nabla_p \hat{f}_k + \hat{\Phi}(t_k;t_{k+1}) \cdot \nabla_p \hat{f}_{k+1}) \\
&= \mathbf{E}_k^{\wedge,\mathbf{J}} + \frac{\Delta t_k}{2} \cdot \left(\mathbf{E}_k^{\nabla_p} + \Phi(t_k;t_{k+1}) \cdot \nabla_p f(t_{k+1},\mathbf{x},\mathbf{p}) - \hat{\Phi}(t_k;t_{k+1}) \cdot \nabla_p \hat{f}_{k+1}\right) \\
&= \mathbf{E}_k^{\wedge,\mathbf{J}} + \frac{\Delta t_k}{2} \cdot \big(\mathbf{E}_k^{\nabla_p} + \Phi(t_k;t_{k+1}) \cdot \nabla_p f(t_{k+1},\mathbf{x},\mathbf{p}) - \Phi(t_k;t_{k+1}) \cdot \nabla_p \hat{f}_{k+1} \\
&\quad + \Phi(t_k;t_{k+1}) \cdot \nabla_p \hat{f}_{k+1} - \hat{\Phi}(t_k;t_{k+1}) \cdot \nabla_p \hat{f}_{k+1}\big) \\
&= \mathbf{E}_k^{\wedge,\mathbf{J}} + \frac{\Delta t_k}{2} \cdot \left(\mathbf{E}_k^{\nabla_p} + \Phi(t_k;t_{k+1}) \cdot \mathbf{E}_{k+1}^{\nabla_p} + \mathbf{E}_k^{\Phi,\mathrm{inv}} \cdot \nabla_p \hat{f}_{k+1}\right). \qquad (7.14)
\end{aligned}$$

From the previous results and Assumption (4.2), the terms in the parenthesis in (7.14) are $\mathcal{O}(\Delta t_{\max}^2)$, hence

$$\left\|\mathbf{E}_k^{\mathbf{J}}\right\| \leq \left\|\mathbf{E}_k^{\wedge,\mathbf{J}}\right\| + \mathcal{O}(\Delta t_{\max}^3) = \mathcal{O}(\Delta t_{\max}^3),$$

from which the claim (7.12) follows . □

With Lemma 7.5, we now have estimates for all errors that propagate–as illustrated in Figure 1–into the approximation $\hat{\mathbf{S}}_{k+1}$ of the sensitivity matrix at time step t_{k+1}. We are now ready to prove Theorem 6.1, the characterisation of the error in the approximation of the sensitivity matrix $\hat{\mathbf{S}}_{k+1}$ at time step t_{k+1}.

Proof (Proof of Theorem 6.1) To estimate the error $\mathbf{E}_{k+1}^{\mathbf{S}}$ in the approximation of $\mathbf{S}$, we start from the exact expression for $\mathbf{S}(t_{k+1})$, given in (6.1):

$$\mathbf{S}(t_{k+1}) = \Phi(t_{k+1};t_k) \cdot (\mathbf{S}(t_k) + \mathbf{J}_k).$$

The approximation naturally takes a similar recursive form, using the approximations $\hat{\Phi}(t_{k+1};t_k)$ and $\hat{\mathbf{J}}_k$,

$$\hat{\mathbf{S}}_{k+1} = \hat{\Phi}(t_{k+1};t_k) \cdot \left(\hat{\mathbf{S}}_k + \hat{\mathbf{J}}_k\right).$$

In order to analyse the associated error, first we replace every exact term in the definition of $\mathbf{S}(t_{k+1})$ with the sum of the corresponding approximation and error; for example, we replace $\mathbf{S}(t_{k+1})$ with $\hat{\mathbf{S}}_{k+1} + \mathbf{E}^{\mathbf{S}}_{k+1}$. This leads to the following relation for the approximating terms and errors, implicitly defining the error $\mathbf{E}^{\mathbf{S}}_{k+1}$,

$$\hat{\mathbf{S}}_{k+1} + \mathbf{E}^{\mathbf{S}}_{k+1} = \left(\hat{\Phi}(t_{k+1};t_k) + \mathbf{E}^{\Phi}_k\right) \cdot \left(\left(\hat{\mathbf{S}}_k + \mathbf{E}^{\mathbf{S}}_k\right) + \left(\hat{\mathbf{J}}_k + \mathbf{E}^{\mathbf{J}}_k\right)\right).$$

By expanding the product, we identify the expression $\hat{\Phi}(t_{k+1;t_k}) \cdot \hat{\mathbf{S}}_k + \hat{\mathbf{J}}_k$ on the right-hand side, which cancels $\hat{\mathbf{S}}_{k+1}$ on the left-hand side.

As a result, the error in the sensitivity matrix can be expressed as

$$\begin{aligned}\mathbf{E}^{\mathbf{S}}_{k+1} &= \hat{\Phi}(t_{k+1};t_k) \cdot \left(\mathbf{E}^{\mathbf{S}}_k + \mathbf{E}^{\mathbf{J}}_k\right) + \mathbf{E}^{\Phi}_k \cdot \left(\hat{\mathbf{S}}_k + \mathbf{E}^{\mathbf{S}}_k + \hat{\mathbf{J}}_k + \mathbf{E}^{\mathbf{J}}_k\right) \\ &= \hat{\Phi}(t_{k+1};t_k) \cdot \mathbf{E}^{\mathbf{S}}_k + \hat{\Phi}(t_{k+1};t_k) \cdot \mathbf{E}^{\mathbf{J}}_k + \mathbf{E}^{\Phi}_k \cdot (\mathbf{S}(t_k) + \mathbf{J}_k),\end{aligned}$$

where in the last equality we re-introduced the exact terms for the sensitivity matrix at time step t_k and the exact integral $\mathbf{J}_k$. This recursive expression can be expanded, using that $\mathbf{E}^{\mathbf{S}}_0 = 0$,

$$\mathbf{E}^{\mathbf{S}}_{k+1} = \sum_{h=0}^{k} \left(\prod_{j=h+1}^{k} \hat{\Phi}(t_{j+1};t_j) \cdot \left(\hat{\Phi}(t_{h+1};t_h) \cdot \mathbf{E}^{\mathbf{J}}_h + \mathbf{E}^{\Phi}_h \cdot (\mathbf{S}(t_h) + \mathbf{J}_h)\right)\right).$$

Taking the norm of the error, we obtain the following bound

$$\begin{aligned}&\left\|\mathbf{E}^{\mathbf{S}}_{k+1}\right\| \\ &\leq \sum_{h=0}^{k} \left(\prod_{j=h+1}^{k} \left\|\hat{\Phi}(t_{j+1};t_j)\right\| \cdot \left(\left\|\hat{\Phi}(t_{h+1};t_h)\right\| \cdot \left\|\mathbf{E}^{\mathbf{J}}_h\right\| + \left\|\mathbf{E}^{\Phi}_h\right\| \cdot (\|\mathbf{S}(t_h)\| + \|\mathbf{J}_h\|)\right)\right).\end{aligned} \tag{7.15}$$

To understand the convergence rate of the error $\mathbf{E}^{\mathbf{S}}_{k+1}$, it now suffices to consider the terms on the right-hand side of (7.15).

We start by considering $\hat{\Phi}(t_{j+1};t_j)$. From the definition (4.4), applying the norm operator we obtain

$$\left\|\hat{\Phi}(t_{j+1};t_j)\right\| \leq 1 + \Delta t_j C_{\nabla_x} + \frac{\Delta t_j^2}{2} C^2_{\nabla_x},$$

for every $j = 0, \ldots, k$; here $C_{\nabla_x} = \|\nabla_x f(\mathbf{x}(\cdot))\|_{[0,T]}$, which is finite by Assumption (4.1). This term is $\mathcal{O}(1)$ for $\Delta t_j \to 0$. For arguments used later in the proof, it is convenient to define $C_{\Delta t} := C_{\nabla_x} \cdot \left(1 + \frac{\Delta t_{\max} C_{\nabla_x}}{2}\right)$, which is not a constant, but depends on $\Delta t_{\max}$ and $C_{\Delta t} \to C_{\nabla_x}$ as $\Delta t_{\max} \to 0$. With this definition we can rewrite the upper bound as

$$\left\| \hat{\Phi}(t_{j+1}; t_j) \right\| \leq 1 + C_{\Delta t} \Delta t_j .$$

From Lemmas 7.5 and 7.4 we know that the error terms $\mathbf{E}_h^{\mathbf{J}}$ and $\mathbf{E}_h^{\Phi}$ are $\mathcal{O}(\Delta t_{\max}^3)$.

By Assumption 5, the exact sensitivity matrix at time step t_h can be bounded as $\|\mathbf{S}(t_h)\| \leq C_{\mathbf{S}}$, hence $\|\mathbf{S}(t_h)\| = \mathcal{O}(1)$, and $\left\| \Phi(t_h; t) \cdot \nabla_p f(t, \mathbf{x}, \mathbf{p}) \right\|_{[0,T]} \leq C_{\mathbf{J}}$ for every $h = 0, \ldots, k$. It follows that the integral term $\mathbf{J}_h$ can be bounded as

$$\|\mathbf{J}_h\| = \left\| \int_{t_h}^{t_{h+1}} \Phi(t_h; \tau) \cdot \nabla_p f(\tau, \mathbf{x}, \mathbf{p}) d\tau \right\| \leq C_{\mathbf{J}} \Delta t_h ,$$

and we conclude that $\|\mathbf{J}_h\| = \mathcal{O}(\Delta t_h)$. As a consequence, the term

$$\left\| \hat{\Phi}(t_{h+1}; t_h) \right\| \cdot \left\| \mathbf{E}_h^{\mathbf{J}} \right\| + \left\| \mathbf{E}_h^{\Phi} \right\| \cdot (\|\mathbf{S}(t_h)\| + \|\mathbf{J}_h\|),$$

in (7.15) is $\mathcal{O}(\Delta t_{\max}^3)$ and we can define a new constant $0 < C < \infty$, such that

$$\begin{aligned} \left\| \mathbf{E}_{k+1}^{\mathbf{S}} \right\| &\leq \sum_{h=0}^{k} \left(\prod_{j=h+1}^{k} \left\| \hat{\Phi}(t_{j+1}; t_j) \right\| \cdot C \Delta t_{\max}^3 \right) \\ &\leq \sum_{h=0}^{k} \left(\prod_{j=h+1}^{k} \left(1 + C_{\Delta t} \Delta t_j\right) \cdot C \Delta t_{\max}^3 \right). \end{aligned}$$

Moreover, we can bound Δt_j by $\Delta t_{\max}$, which gives an upper bound for $\mathbf{E}_{k+1}^{\mathbf{S}}$ in terms of $\Delta t_{\max}$,

$$\left\| \mathbf{E}_{k+1}^{\mathbf{S}} \right\| \leq \sum_{h=0}^{k} (1 + C_{\Delta t} \Delta t_{\max})^{k-h} \cdot C \Delta t_{\max}^3 .$$

The sum $\sum_{h=0}^{k} (1 + C_{\Delta t} \Delta t_{\max})^{k-h}$ can be rewritten as $\sum_{h=0}^{k} (1 + C_{\Delta t} \Delta t_{\max})^{h}$, which admits the closed formula

$$\begin{aligned} \sum_{h=0}^{k} (1 + C_{\Delta t} \Delta t_{\max})^{h} &= \frac{1 - (1 + C_{\Delta t} \Delta t_{\max})^{k+1}}{1 - (1 + C_{\Delta t} \Delta t_{\max})} \\ &= \frac{(1 + C_{\Delta t} \Delta t_{\max})^{k+1} - 1}{C_{\Delta t} \Delta t_{\max}} . \end{aligned} \tag{7.16}$$

First, we consider the term $(1 + C_{\Delta t}\Delta t_{\max})^{k+1}$ in the numerator. Since the exponent k runs over $k = 0, \ldots, K-1$, and the term inside the parenthesis is non-negative,

$$(1 + C_{\Delta t}\Delta t_{\max})^{k+1} \leq (1 + C_{\Delta t}\Delta t_{\max})^{K}. \tag{7.17}$$

Second, the total number of time steps K can be bounded from above,

$$K \leq \frac{T}{\Delta t_{\min}} = \frac{T}{\Delta t_{\min}} \frac{\Delta t_{\max}}{\Delta t_{\max}} = \Xi \frac{T}{\Delta t_{\max}}.$$

Inserting this in (7.17) yields

$$(1 + C_{\Delta t}\Delta t_{\max})^{K} \leq (1 + C_{\Delta t}\Delta t_{\max})^{\Xi \frac{T}{\Delta t_{\max}}}.$$

To finish the proof, we use that the function $g : (0, \infty) \to \mathbb{R}$, $g(x) = (1 + x)^{\frac{1}{x}}$ is monotonically decreasing and that $\lim_{x \to 0^+} g(x) = e$. Since $C_{\Delta t}\Delta t_{\max}$ is increasing in $\Delta t_{\max}$, it follows that

$$(1 + C_{\Delta t}\Delta t_{\max})^{\frac{1}{C_{\Delta t}\Delta t_{\max}}} < e,$$

hence

$$(1 + C_{\Delta t}\Delta t_{\max})^{\Xi \frac{T}{\Delta t_{\max}}} < e^{\Xi C_{\Delta t} T}.$$

The previous results show that (7.16) is bounded from above by

$$\frac{e^{\Xi C_{\Delta t} T} - 1}{C_{\Delta t}\Delta t_{\max}}.$$

Lastly, inserting this into the upper bound for $\mathbf{E}^{\mathbf{S}}_{k+1}$ yields

$$\left\| \mathbf{E}^{\mathbf{S}}_{k+1} \right\| \leq \frac{e^{\Xi C_{\Delta t} T} - 1}{C_{\Delta t}\Delta t_{\max}} \cdot C \Delta t^3_{\max} = \frac{C(e^{\Xi C_{\Delta t} T} - 1)}{C_{\Delta t}} \Delta t^2_{\max}.$$

Since $C_{\Delta t} = \mathcal{O}(1)$, this proves the claimed bound (6.4). □

Remark 4 Collecting all the error estimates determined in the proofs of Lemmas 7.1-7.5 and the proof of Theorem 6.1, we obtain that the constant inside the $\mathcal{O}$ in (6.4) in Theorem 6.1 is

$$\frac{\left(e^{\Xi C_{\Delta t} T} - 1\right)}{C_{\Delta t}} \cdot C,$$

with Ξ, $C_{\Delta t}$, T and C defined above.

Finally, Corollary (6.2) is a direct consequence of the following lemma, which shows that the error in the linearly interpolated numerical ODE solution $\hat{\mathbf{x}}$ is $\mathcal{O}(\Delta t^2_{\max})$.

This is the same order that we assume for $\mathbf{E}_k^{\text{ODE}}$ in Assumption 2. As a consequence, Lemmas 7.1-7.5 continue to hold, and therefore the same proof as Theorem 6.1 can be applied also when a refinement of the time intervals is performed and the ODE solution $\hat{\mathbf{x}}$ is linearly interpolated.

Lemma 7.6 *Assume that the map* $t \mapsto \frac{d}{dt} f(t, \mathbf{x}, \mathbf{p})$ *is bounded with respect to* $\|\cdot\|_{[0,T]}$. *For* $i \in \{1, \ldots, n_{int}\}$ *let* $t_{k,i} = t_k + \frac{i-1}{n_{int}} \Delta t_k$ *and*

$$\hat{\mathbf{x}}_{k,i} = \hat{\mathbf{x}}_k + \frac{i-1}{n_{int}} (\hat{\mathbf{x}}_{k+1} - \hat{\mathbf{x}}_k).$$

Let

$$\mathbf{E}_{k,i}^{ODE} = \mathbf{x}(t_{k,i}) - \hat{\mathbf{x}}_{k,i}$$

be the error of approximating the exact ODE solution $\mathbf{x}$ *at time* $t_{k,i}$ *with the interpolation* $\hat{\mathbf{x}}_{k,i}$ *of the ODE numerical solution,* $\{\hat{\mathbf{x}}_k\}_{k=0}^K$. *Under Assumption 2,*

$$\|\mathbf{E}_{k,i}^{ODE}\| = \mathcal{O}(\Delta t_{\max}^2).$$

Proof Let

$$\mathbf{x}_{k,i}^{\text{interp}} = \mathbf{x}(t_k) + \frac{i-1}{n_{\text{int}}} (\mathbf{x}(t_{k+1}) - \mathbf{x}(t_k))$$

be the interpolation of the exact ODE solution $\mathbf{x}$ at time $t_{k,i}$. Because $t \mapsto \frac{d}{dt} f(t, \mathbf{x}, \mathbf{p})$, i.e. $t \mapsto \frac{d^2}{dt^2} \mathbf{x}(t)$, is bounded with respect to $\|\cdot\|_{[0,T]}$, the linear interpolation error satisfies

$$\|\mathbf{x}(t_{k,i}) - \mathbf{x}_{k,i}^{\text{interp}}\| = \mathcal{O}(\Delta t_k^2),$$

as showed in [28, Chap. 5]. Observe that

$$\begin{aligned}
\mathbf{x}_{k,i}^{\text{interp}} - \hat{\mathbf{x}}_{k,i} &= \mathbf{x}(t_k) + \frac{i-1}{n_{\text{int}}} (\mathbf{x}(t_{k+1}) - \mathbf{x}(t_k)) - \hat{\mathbf{x}}_k - \frac{i-1}{n_{\text{int}}} (\hat{\mathbf{x}}_{k+1} - \hat{\mathbf{x}}_k) \\
&= \mathbf{E}_k^{\text{ODE}} + \frac{i-1}{n_{\text{int}}} \left(\mathbf{E}_{k+1}^{\text{ODE}} - \mathbf{E}_k^{\text{ODE}} \right).
\end{aligned}$$

Consequently,

$$\|\mathbf{x}_{k,i}^{\text{interp}} - \hat{\mathbf{x}}_{k,i}\| \leq \|\mathbf{E}_k^{\text{ODE}}\| + \frac{i-1}{n_{\text{int}}} \left(\|\mathbf{E}_{k+1}^{\text{ODE}}\| + \|\mathbf{E}_k^{\text{ODE}}\| \right) = \mathcal{O}(\Delta t_{\max}^2),$$

by Assumption 2. We conclude that

$$\|\mathbf{E}_{k,i}^{\text{ODE}}\| = \|\mathbf{x}(t_{k,i}) - \hat{\mathbf{x}}_{k,i}\| \leq \|\mathbf{x}(t_{k,i}) - \mathbf{x}_{k,i}^{\text{interp}}\| + \|\mathbf{x}_{k,i}^{\text{interp}} - \hat{\mathbf{x}}_{k,i}\|,$$

where both error on the right hand side of the last inequality are of order $\mathcal{O}(\Delta t_{\max}^2)$, hence the result. □

8 Numerical results

In this section we complement the theoretical analysis in Sects. 4-7 with numerical experiments, illustrating the performance of the proposed sensitivity approximation methods in a number of examples. First, in Sect. 8.1 we describe some implementation details of the PBSR and Exp algorithms, first presented in Sect. 5. Next, in Sect. 8.2 we outline how the accuracy of the different methods is evaluated and show numerical results for four examples: two biological models, referred to as PKA and CaMKII, a random linear system, and a model with a limit cycle (the Chua's circuit model). The GitHub repository associated with this paper[9] presents more implementation details; another repository[10] contains an alternative implementation in C of the methods described in this manuscript and sets up more examples.

8.1 Implementation details

To test the accuracy and run time of the sensitivity approximation methods proposed in this paper (described in Sect. 5), we have implemented the methods in the Julia language. For this purpose, we used the Julia packages `DifferentialEquations.jl` and `Sundials.jl` to solve the n_x-dimensional ODE system for the state variable $\mathbf{x}$. In particular, we used the solver `CVODE_BDF` from `Sundials.jl`, for which we set absolute and relative tolerances of 1e-6 and 1e-5, respectively. The ODE solver uses an adaptive time stepping algorithm, and therefore returns the approximate ODE solution $\{\hat{\mathbf{x}}_k\}$ on a non-uniform time grid $\{t_k\}$, where time steps $[t_k, t_{k+1}]$ can be arbitrarily large (within the set time span).

As discussed in Sect. 5, the size Δt_k of the time intervals $[t_k, t_{k+1}]$ does not give rise to stability concerns in the case of the Exp algorithm (Algorithm 3), as this method relies only on the exponential formula (3.10), which is stable for all choices of Δt_k. For the PBSR algorithm (Algorithm 2), in the case of stiff problems, the time intervals $[t_k, t_{k+1}]$ can be too large for the PBS formula (4.6), causing stability issues. Therefore, the time intervals are refined into n_{int} sub-intervals $\{[t_{k,i}, t_{k,i+1}]\}_{i=1}^{n_{\text{int}}}$ of length $\Delta t_k/n_{\text{int}}$; note that we suppress the n_{int}s dependence on k in the notation. To determine a good choice for the number of sub-intervals n_{int}, we consider some of the error terms in the PBS sensitivity approximation, analysed in Sect. 7. In particular, a potentially problematic term is the error due to the truncation of the Peano-Baker series, i.e., the omission of $\sum_{n=3}^{\infty} \mathbf{I}_n(t_{k,i+1}; t_{k,i})$, on the sub-interval $[t_{k,i}, t_{k,i+1}]$ (see (7.8)). With majorations similar to (7.9), we obtain the upper bound

$$\left\| \sum_{n=3}^{\infty} \mathbf{I}(t_{k,i+1}; t_{k,i}) \right\| \leq \sum_{n=3}^{\infty} \frac{1}{n!} \left(\frac{\Delta t_k}{n_{\text{int}}} C \right)^n = e^{\frac{\Delta t_k}{n_{\text{int}}} C} - 1 - \left(\frac{\Delta t_k}{n_{\text{int}}} C \right) - \frac{1}{2} \left(\frac{\Delta t_k}{n_{\text{int}}} C \right)^2,$$

[9] https://github.com/federicamilinanni/JuliaSensitivityApproximation.

[10] https://github.com/a-kramer/CSensApprox.

where $C = \|\nabla_x f\|_{[t_{k,i}, t_{k,i+1}]}$. One way to ensure that this value is sufficiently small is to require that $\frac{\Delta t_k}{n_{\text{int}}} C \lesssim 0.1$. Based on this, the choice for the number of sub-intervals becomes $n_{\text{int}} = \lceil 10 \Delta t_k \|\nabla_x f(\hat{\mathbf{x}}_k)\| \rceil$.

Refining the time grid increases the running time of the algorithm, in particular when the time span of the whole simulation is of the order of hundreds or thousands time units, as in the examples considered in the following subsections. To make the algorithm more efficient, on time intervals $[t_k, t_{k+1}]$ where the proposed n_{int} is deemed too large, we apply the exponential formula (3.10) instead of refining and using the PBS formula (4.6). As outlined in Sect. 5, this means that the exponential formula is used in two situations: for time intervals $[t_k, t_{k+1}]$ for which (i) n_{int} is "too large", or (ii) $\nabla_x f$ and $\nabla_p f$ are approximately constant. In our implementation of the PBSR algorithm we consider n_{int} "too large" when it exceeds the value 100, and we treat the Jacobians $\nabla_x f$ and $\nabla_p f$ as constant when

$$\frac{\|\nabla_x \hat{f}_{k+1} - \nabla_x \hat{f}_k\|}{\|\nabla_x \hat{f}_k\|} < 10^{-4} \quad \text{and} \quad \frac{\|\nabla_p \hat{f}_{k+1} - \nabla_p \hat{f}_k\|}{\|\nabla_p \hat{f}_k\|} < 10^{-4},$$

respectively. Combining the two criteria, the `if`-statement in Algorithm 2 that triggers the use of the exponential formula (3.10) is implemented as:

$$\texttt{if} \quad n_{\text{int}} > 100 \quad \texttt{OR} \quad \frac{\|\nabla_x \hat{f}_{k+1} - \nabla_x \hat{f}_k\|}{\|\nabla_x \hat{f}_k\|} < 10^{-4} \quad \texttt{AND} \quad \frac{\|\nabla_p \hat{f}_{k+1} - \nabla_p \hat{f}_k\|}{\|\nabla_p \hat{f}_k\|} < 10^{-4}. \tag{8.1}$$

The exponential formula (3.10) requires an implementation of matrix exponentiation. Each programming language offers a built in[11] function for this, typically named `expm` (we use `Base.exp()` in Julia). Such methods often rely on the scaling and squaring method; Julia specifically cites [29]. An unstable/bad implementation of `expm` could lead to large numerical inaccuracies, however this is not something we have observed in practice.

Lastly, in the numerical examples we also implement the Forward Sensitivity method to approximate the sensitivity matrix. For this purpose we use the Julia package `DiffEqSensitivity.jl`.

8.2 Simulation experiments

In the numerical experiments that follow, we evaluate the accuracy and speed of the Exp and PBSR algorithms (Algorithms 2 and 3). The accuracy is measured in two different ways: one is to compare the two algorithms with the FS method, which is considered to have a high accuracy, and one is to consider minor perturbations, with respect to the parameter vector, of the ODE solution.

As our first performance measure, after solving the ODE system (2.1) for $\mathbf{x}$, we consider the time points t_k, $k = 0, \ldots, K$, returned by the ODE solver, and for each

[11] Alternatively, a linear algebra package.

time point t_k compute approximations of the sensitivity matrix $\mathbf{S}$ using the PBSR and Exp algorithms: $\hat{\mathbf{S}}_k^{\text{PBSR}}$ (PBSR), and $\hat{\mathbf{S}}_k^{\text{Exp}}$ (Exp). Besides, we apply the FS method to obtain the FS approximation of the sensitivity matrix. The time steps used by the FS method are in general different from those returned by the ODE solver mentioned above; therefore, we linearly interpolate the FS sensitivity matrix approximation in the same time points $\{t_k\}$ used for the PBSR and Exp algorithms, thus obtaining the approximation $\{\hat{\mathbf{S}}_k^{\text{FS}}\}$.

Next, for each time point t_k, we compute the scalar relative error of the PBSR (E_k^{PBSR}) and Exp (E_k^{Exp}) algorithms with respect to the FS method:

$$E_k^{\text{PBSR}} = \frac{\|\hat{\mathbf{S}}_k^{\text{PBSR}} - \hat{\mathbf{S}}_k^{\text{FS}}\|}{\|\hat{\mathbf{S}}_k^{\text{FS}}\|}, \qquad E_k^{\text{Exp}} = \frac{\|\hat{\mathbf{S}}_k^{\text{Exp}} - \hat{\mathbf{S}}_k^{\text{FS}}\|}{\|\hat{\mathbf{S}}_k^{\text{FS}}\|}.$$

The method just outlined treats $\hat{\mathbf{S}}_k^{\text{FS}}$, obtained using the FS method, as the true sensitivity matrix at time t_k. However, the FS method also yields an approximation, and thus $\hat{\mathbf{S}}_k^{\text{FS}}$ comes with an error. We therefore introduce an alternative, in a sense more objective, way to evaluate the accuracy of the proposed methods. The idea is to use the sensitivity matrix to approximate the solution of the underlying ODE model after making a small perturbation $\delta_{\mathbf{p}}$ in the parameter vector $\mathbf{p}$. Consider making such a perturbation by either adding or subtracting (an appropriate) $\delta_{\mathbf{p}}$ from $\mathbf{p}$: from a Taylor expansion, we have,

$$\mathbf{x}(t, \mathbf{p} + \delta_{\mathbf{p}}) = \mathbf{x}(t, \mathbf{p}) + \mathbf{S}(t, \mathbf{x}, \mathbf{p}) \cdot \delta_{\mathbf{p}} + \frac{1}{2}\delta_{\mathbf{p}}^{\mathrm{T}} \cdot \mathbb{S}(t, \mathbf{x}, \mathbf{p}) \cdot \delta_{\mathbf{p}} + \mathcal{O}(\delta_{\mathbf{p}}^3), \tag{8.2}$$

$$\mathbf{x}(t, \mathbf{p} - \delta_{\mathbf{p}}) = \mathbf{x}(t, \mathbf{p}) - \mathbf{S}(t, \mathbf{x}, \mathbf{p}) \cdot \delta_{\mathbf{p}} + \frac{1}{2}\delta_{\mathbf{p}}^{\mathrm{T}} \cdot \mathbb{S}(t, \mathbf{x}, \mathbf{p}) \cdot \delta_{\mathbf{p}} + \mathcal{O}(\delta_{\mathbf{p}}^3), \tag{8.3}$$

where $\mathbb{S}$ is the second order sensitivity tensor (i.e. the Hessian of $\mathbf{x}(t, \mathbf{p})$, with respect to $\mathbf{p}$). Subtracting (8.3) from (8.2) leads to a natural criterion to judge how well the sensitivity works as a linear approximation of the solution with respect to parameter changes,

$$\mathbf{x}(t, \mathbf{p} + \delta_{\mathbf{p}}) - \mathbf{x}(t, \mathbf{p} - \delta_{\mathbf{p}}) = 2\mathbf{S}(t, \mathbf{x}, \mathbf{p}) \cdot \delta_{\mathbf{p}} + \mathcal{O}(\delta_{\mathbf{p}}^3). \tag{8.4}$$

For an accurate approximation $\hat{\mathbf{S}}$ of $\mathbf{S}$, the difference between the left-hand side and the right-hand side of (8.4), with $\mathbf{S}$ replaced by $\hat{\mathbf{S}}$, should be small and this therefore works as a performance measure. In line with this, for an approximation $\hat{\mathbf{S}}$, we define, for a given $\delta_{\mathbf{p}}$, the following (objective) error estimate:

$$\bar{\varkappa}_{\delta_{\mathbf{p}}} = \frac{\|\mathbf{x}(t, \mathbf{p} + \delta_{\mathbf{p}}) - \mathbf{x}(t, \mathbf{p} - \delta_{\mathbf{p}}) - 2\hat{\mathbf{S}}(t, \mathbf{x}, \mathbf{p}) \cdot \delta_{\mathbf{p}}\|}{\|\epsilon + \mathbf{x}(t, \mathbf{p} + \delta_{\mathbf{p}}) - \mathbf{x}(t, \mathbf{p} - \delta_{\mathbf{p}})\|}, \tag{8.5}$$

where ϵ is a small quantity, used to regularize the fraction in the cases where the denominator in (8.5) is vanishingly small[12]. In practice, we pick several random $\delta_{\mathbf{p}}$ values and average the resulting errors $\bar{\varkappa}_{\delta_{\mathbf{p}}}$:

$$\bar{\varkappa} = \frac{1}{N} \sum_{i=1}^{N} \bar{\varkappa}_{\delta_{\mathbf{p}}^i}, \quad \delta_{\mathbf{p}}^i = \mathbf{h}^i \odot \mathbf{p}, \tag{8.6}$$

where $\odot$ denotes point-wise multiplication, or Hadamard product, between the vectors $\mathbf{h}^i$ and $\mathbf{p}$ (both of dimension n_p), and for each $i = 1, \ldots, N$, $\mathbf{h}^i$ is a random vector with components

$$h_j^i \sim \mathcal{U}(10^{-5}, 10^{-4}), \quad j = 1, \ldots, n_p.$$

With this definition, the perturbations $\delta_{\mathbf{p}}^i$ are small compared to $\mathbf{p}$, in the sense that each component $\delta_{\mathbf{p},j}$ satisfies $|\delta_{\mathbf{p},j}|/p_j \ll 1$, $j = 1, \ldots, n_p$.

In each of the examples in the following subsections, the error estimate $\bar{\varkappa}$ is computed for all the three methods (PBSR, Exp and FS) at each time t_k, denoted by $\bar{\varkappa}_k^{\text{PBSR}}$, $\bar{\varkappa}_k^{\text{Exp}}$ and $\bar{\varkappa}_k^{\text{FS}}$. Note that there is some arbitrariness in the chosen distribution for $\delta_{\mathbf{p}}$.

A benefit of this alternative, seemingly more objective, performance measure is that it provides an estimate also of the error in the FS approximation of the sensitivity matrix, and removes the reliance on any other numerical approximation in evaluating the accuracy of the PBSR and the Exp algorithms.

Models for molecular signaling pathways: PKA and CaMKII models

The first numerical experiments highlight the original motivation of considering models from systems biology (see Sect. 1): in this section we apply the PBSR Algorithm 2 and the Exp Algorithm 3 to two models that describe molecular signaling pathways within neurons involved in learning and memory. In particular, these mechanisms are involved in the strengthening or weakening of neuron synapses, referred to as long term potentiation (LTP) and long term depression (LTD), respectively.

A crucial role in signaling pathways is played by protein kinases and phosphatases, thanks to their ability to, respectively, phosphorylate and dephosphorylate substrate proteins. In the two models that we consider, the phosphorylating role is performed by the cAMP-dependent protein kinase A (PKA) and the Ca^{2+}/calmodulin-dependent protein kinase II (CaMKII), which give the two models their names: PKA and CaMKII–for more details see references [14, 30, 31].

In the two ODE models, the state vectors $\mathbf{x}(t)$ represent the concentrations of the different forms of the modelled proteins and ions at time t; the dimension of the state space is $n_x = 11$ in the PKA model, and $n_x = 21$ in the CaMKII model. In both models the parameter vector $\mathbf{p}$ corresponds to the kinetic constants that characterize the reactions in the underlying pathway. The dimension of the parameter space is $n_p = 35$ in the PKA, and $n_p = 59$ in the CaMKII model.

[12] Initially, at t_0, the sensitivity is exactly 0 in all components, thus $\mathbf{x}(t_0, \mathbf{p} + \delta_{\mathbf{p}}) = \mathbf{x}(t_0, \mathbf{p} - \delta_{\mathbf{p}})$.

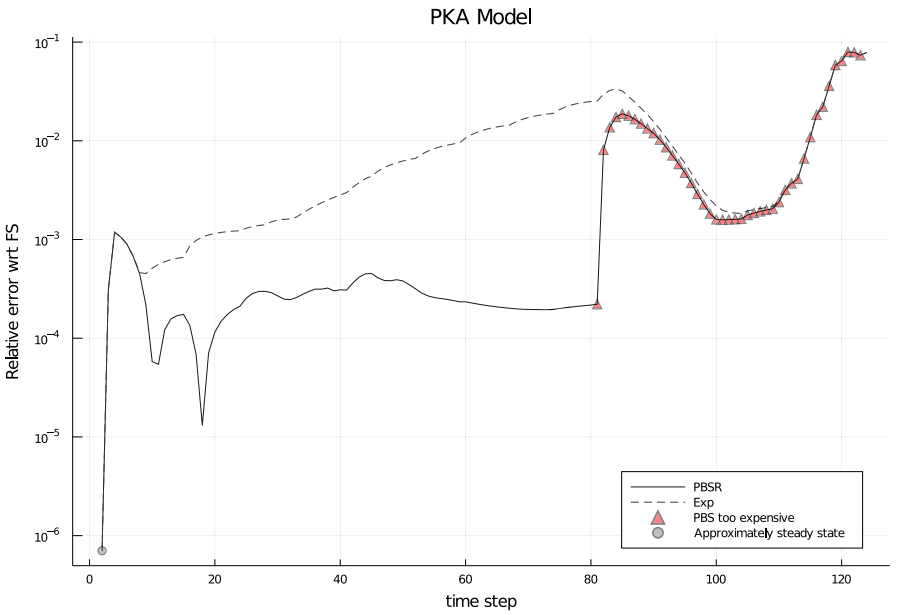

Fig. 2 Relative error of the sensitivity matrix, against time step, for the PBSR (solid line) and the Exp (dashed line) algorithms for the PKA model. The simulations are run over the time span [0, 600]. The gray circles superimposed on the solid line indicate the time steps at which within the PBSR algorithm the exponential formula (3.10) is used instead of the PBS formula (4.6) because the system is approximately at equilibrium; the small red triangles show when the exponential formula (3.10) is used instead of the PBS formula (4.6) because the number of sub-intervals n_{int} in the PBSR refinement is too large

In Figures 2 and 3 we show the relative errors E^{PBSR} (solid line) and E^{Exp} (dashed line) against the time step for the PKA and CaMKII model, respectively. The small gray circles and red triangles superimposed on the solid line (PBSR algorithm) refer to the time steps at which the `if`-statement is satisfied and the exponential formula (3.10) is used: gray circles correspond to time steps where the use of the exponential formula (3.10) is triggered because the system is approximately at equilibrium, while red triangles indicate time steps where (3.10) is used because the estimated value of n_{int} is too large. We observe that the results obtained by the PBSR algorithm are 1 to 2 order of magnitude more accurate than the Exp algorithm at the time steps where the PBSR formula (4.6) is used (when FS is considered as providing the true sensitivity matrix).

Figures 4 and 5 show the objective errors $\bar{\varkappa}$, defined in (8.6), with $N = 100$. The blue and red shaded areas indicate simulation time points where the use of the exponential formula (3.10) is triggered in the PBSR algorithm: blue corresponds to simulation time points where the system is approximately in equilibrium, and red indicates time points where the estimated number of sub-intervals n_{int} for the refinement is too large. Note that in Figure 2, the PKA system is simulated in the time span [0, 600], during which it does not reach convergence (final time points are labelled with red triangles), whereas in Figure 4 we simulate the system over the time span [0, 60000], and the system then reaches equilibrium (final time points are depicted in a shaded blue area).

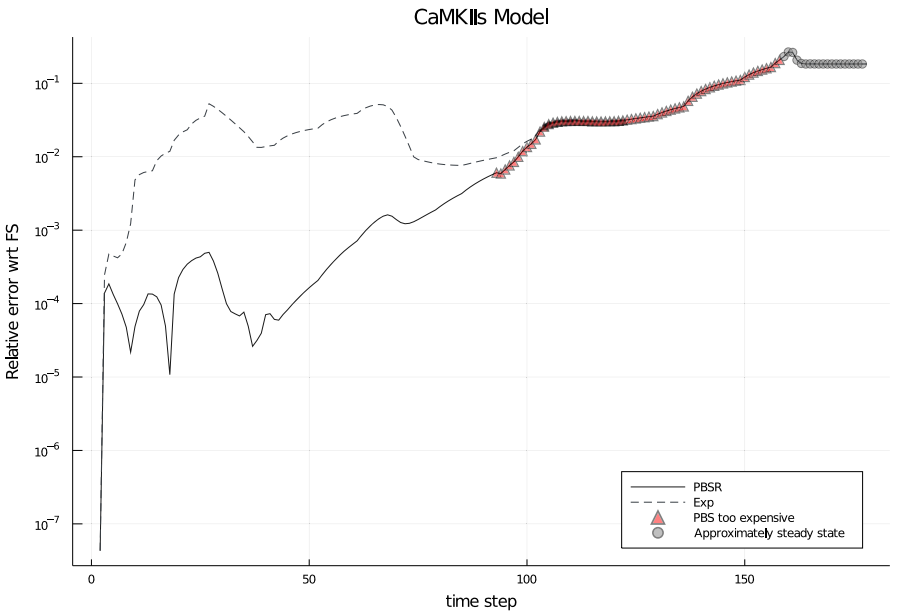

Fig. 3 Relative error of the sensitivity matrix, against time step, for the PBSR (solid line) and the Exp (dashed line) algorithms for the CaMKII model. The simulations are run over the time span [0, 600]. The gray circles superimposed on the solid line indicate the time steps at which within the PBSR algorithm the exponential formula (3.10) is used instead of the PBS formula (4.6) because the system is approximately at equilibrium; the small red triangles show when the exponential formula (3.10) is used instead of the PBS formula (4.6) because the number of sub-intervals n_{int} in the PBSR refinement is too large

We observe that in the first instants of the simulations, the errors $\bar{\varkappa}^{\text{PBSR}}$ and $\bar{\varkappa}^{\text{FS}}$ are about one order of magnitude lower than $\bar{\varkappa}^{\text{Exp}}$; after the initial phase, the errors in the three methods result to be close to each other, with $\bar{\varkappa}^{\text{Exp}}$ being slightly higher in the model of larger dimension (CaMKII), between 10^{-2} and 10^{-1}. At the time points where the PBSR algorithm implements the exponential formula (3.10) (red and blue regions in the Figures) the error $\bar{\varkappa}^{\text{PBSR}}$ is comparable with $\bar{\varkappa}^{\text{Exp}}$. Interestingly, we observe that in the larger model (CaMKII) the errors $\bar{\varkappa}^{\text{PBSR}}$ and $\bar{\varkappa}^{\text{Exp}}$ in the PBSR and Exp approximations decay to 10^{-5} when the system converges to equilibrium, whereas the error $\bar{\varkappa}^{\text{FS}}$ in the FS approximation remains of an order of 10^{-1}. This explains why in Figure 3 the errors E^{PBSR} and E^{Exp} in the PBSR and Exp approximations are large in the final time steps, even though the PBSR and Exp approximation have a sufficiently high accuracy: in fact, E^{PBSR} and E^{Exp} correspond to the relative errors of the PBSR and Exp sensitivity approximations with respect to the FS approximation, which in this case is much less accurate.

As a final test, we compared the computational time for the different algorithms in the PKA and CaMKII models by computing the average runtime (averaged over 100 iterations) for the three methods; the results are presented in Table 2. As expected, the PBSR shows an increased computational cost compared to the Exp algorithm, whereas compared to the FS method, the PBSR algorithm is faster. The latter becomes more

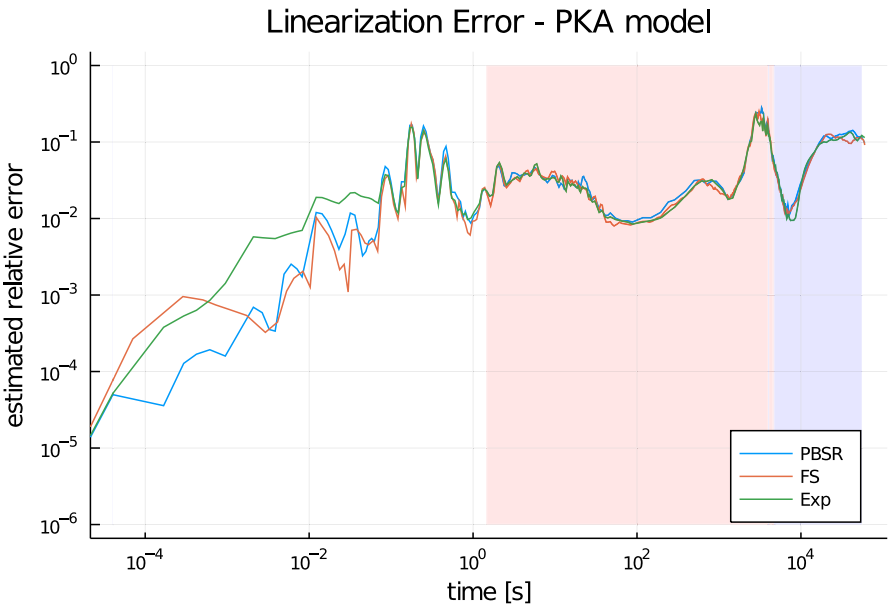

Fig. 4 The estimated relative error $\tilde{\varkappa}$ against time for the PKA model. The simulations are run over the time span [0, 60000]. The blue and red shaded areas both indicate that within the PBSR method the exponential formula (3.10) is used: *blue* because the system is close to a (stable) steady state, *red* because refinement is numerically prohibitive. The error estimate $\tilde{\varkappa}$ uses small parameter perturbations $\delta_{\mathbf{p}}$ to test how well the sensitivity matrix predicts the changes in the solution (averaged over $N = 100$ small, random parameter perturbations)

pronounced when the dimension of the system is rather high (CaMKII model vs. PKA model).

Random linear system

In order to numerically investigate the impact of an increase in the dimensions n_x and n_p of the underlying ODE system on the algorithms' computational cost, we considered a random linear system for which we can choose an arbitrary dimensions n_x; for convenience, we take this to be equal to the parameter space dimension n_p and denote both with n.

The model is obtained by first generating a random matrix $\mathbf{B}$ of dimension $n \times n$ and a random vector $\mathbf{p}$ of length n, both with values uniformly distributed in the interval [0, 1]. Next, we define $\mathbf{A} := -\mathbf{B}^{\mathrm{T}}\mathbf{B}$, and let $\mathbf{p}^2$ be the element-wise square of $\mathbf{p}$, and $\mathbf{u}$ an n-dimensional vector of ones. The associated (random) linear system is defined as

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x} + \mathbf{p}^2 + \mathbf{u}\,,$$

and we use this system to test how the dimension of the system affects the runtime of the three algorithms (PBSR, Exp and FS).

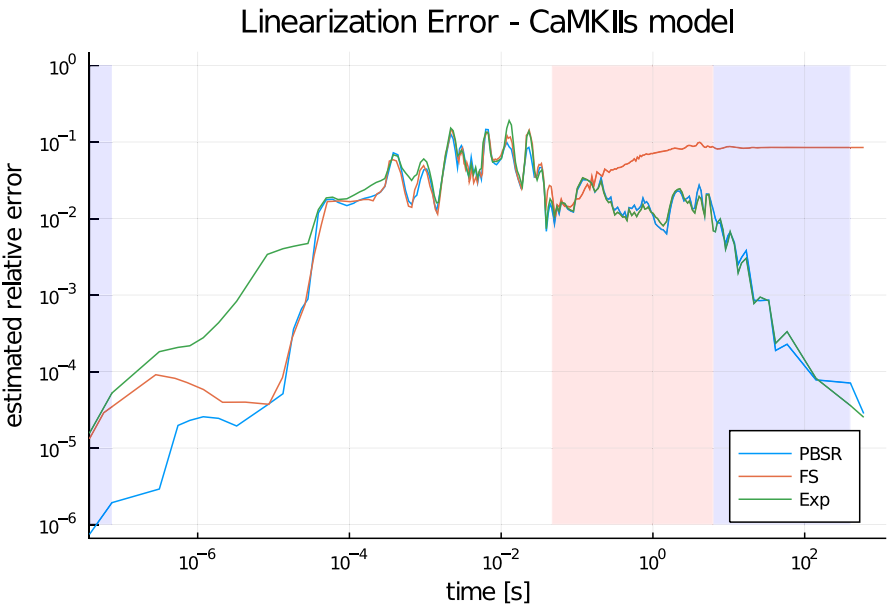

Fig. 5 The estimated relative error $\bar{\varkappa}$ against time for the CaMKMIIs model. The simulations are run over the time span $[0, 600]$. The blue and red shaded areas both indicate that within the PBSR method the exponential formula (3.10) is used: *blue* because the system is close to a (stable) steady state, *red* because refinement is numerically prohibitive. The error estimate $\bar{\varkappa}$ uses small parameter perturbations $\delta_{\mathbf{p}}$ to test how well the sensitivity matrix predicts the changes in the solution (averaged over $N = 100$ small, random parameter perturbations)

Table 2 Average runtime (averaged over 100 iterations) of FS, PBSR and Exp algorithms applied to the PKA and CaMKII models

	PKA	CaMKII
FS	0.056 *s*	1.518 *s*
PBSR	0.047 *s*	0.132 *s*
Exp	0.008 *s*	0.030 *s*

In this example, the Jacobians $\nabla_x f$ and $\nabla_p f$ are, by construction, constant with respect to the state $\mathbf{x}$ and time–$\nabla_x f \equiv \mathbf{A}$ and $\nabla_p f = 2\mathbf{p}$–and the `if`-statement (8.1) in Algorithm 2 is always true. Thus, in the PBSR algorithm, there would never be a refinement of the time intervals $[t_k; t_{k+1}]$ and the exponential formula (3.10) would always be applied, leading to the same output as the Exp algorithm. Therefore, to test the approximation based on the Peano-Baker series, we remove the `if`-statement and always apply the PBS formula (4.6), including the refinement of the time grid, to approximate the state-transition matrix and compute $\mathbf{S}$.

We performed tests of the three algorithms on this type of random linear system of dimensions $n = 5, 10, \ldots, 95, 100$. In Figure 6 we show the average runtime (over 10 iterations) of the FS algorithm (solid line), the PBSR algorithm (dashed line) and Exp (dotted line). To determine how the runtime scales with the dimension n of the system, we perform regressions on the form runtime $= a \cdot \text{dimension}^b$, using runtime

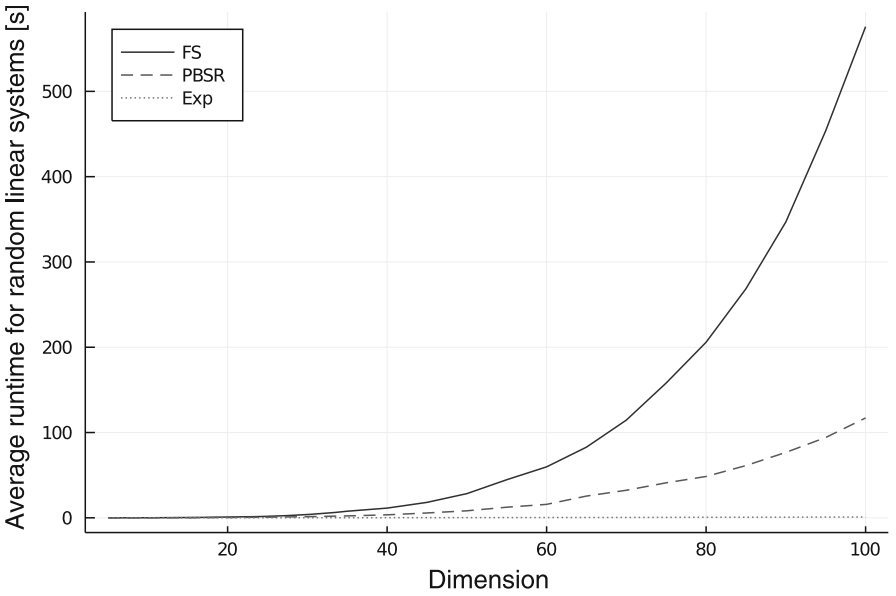

Fig. 6 Average running time (over 10 iterations) of the FS (solid line), PBSR (dashed line) and Exp (dotted line) algorithms applied to random linear systems of dimension 5, 10, . . . , 100

data for each of the three algorithms. The results suggest a runtime of $\mathcal{O}(n^{2.1})$ for the Exp algorithm, $\mathcal{O}(n^{3.7})$ for the PBSR and $\mathcal{O}(n^{4.2})$ for the FS, indicating that the PBSR algorithm gives an improvement of approximately $n^{0.5}$ compared to the FS method for this particular system.

Example with a limit cycle: Chua's circuit model

As our last example, we consider the following three-dimensional dynamical system,

$$\begin{cases} \dot{x}_1 = p_1(x_2 - x_1 - f(x)), \\ \dot{x}_2 = x_1 - x_2 + x_3, \\ \dot{x}_3 = -p_2 x_2, \end{cases}$$

with $f(x) = -8/7x_1 + 4/63x_1^3$, parameters $p_1 = 7$, $p_2 = 15$, and initial condition $\mathbf{x}_0 = (0, 0, -0.1)^{\mathrm{T}}$. This system models Chua's circuit, an electrical circuit consisting of two capacitors and an inductor, and the choice of parameters and initial condition causes the system to converge to a limit cycle.

In this example, the Jacobians $\nabla_x f$ and $\nabla_p f$ exhibit high variability within time steps $[t_k, t_{k+1}]$. Because the PBS formula (4.6) can capture such variations, the PBSR algorithm (Algorithm 2) provides an approximation of the sensitivity matrix that is roughly two orders of magnitude more accurate than the Exp algorithm (Algorithm 3); this is according to both types of error estimates, comparing E_k^{PBSR} and E_k^{Exp} (see Figure 7), and $\bar{\varkappa}_k^{\mathrm{PBSR}}$ and $\bar{\varkappa}_k^{\mathrm{Exp}}$ (see Figure 8), respectively. Moreover, in Figure 8 we

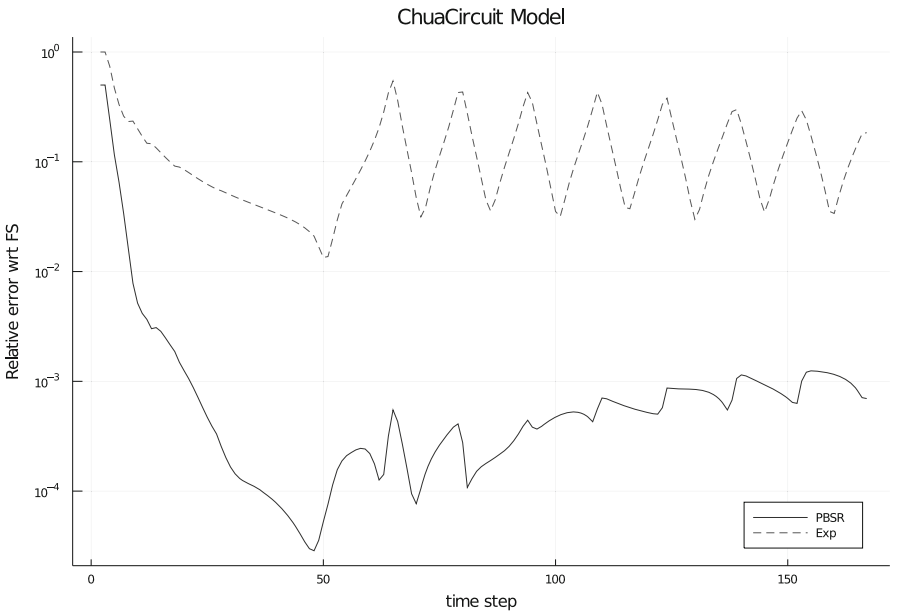

Fig. 7 Relative error of the sensitivity matrix, against time step, for the PBSR (solid line) and the Exp (dashed line) algorithms for the Chua's system. The simulations are run over the time span [0, 10]

note that aside from the initial phase, the FS method has an error that is essentially equivalent to that of the PBSR.

9 Conclusion and future work

In this paper we have addressed the problem of computing the sensitivity matrix of parameter-dependent ODE models in high-dimensional settings, where the forward and adjoint methods become too slow for practical purposes. This situation arises in, e.g., uncertainty quantification using Bayesian methods, where there can be a need to compute the sensitivity matrix at a large number of time steps and parameter values, coupled with a high-dimensional parameter space.

We developed a new numerical method based on the Peano-Baker series from control theory, which appears in the exact solution of $\mathbf{S}$ (Theorem 3.3). By truncating the series and applying the trapezoidal rule for the integrals involved in the exact formula, we constructed an approximation to $\mathbf{S}$ that is suitable for numerical computation, referred to here as the *PBS formula* (4.6). In addition to the general representation of $\mathbf{S}$, given in Theorem 3.3, in Corollary 3.4 we derived a simplified expression for the solution $\mathbf{S}$ in the setting of constant coefficients and forcing term in the ODE for $\mathbf{S}$. This led to a second formula to approximate the sensitivity matrix $\mathbf{S}$–the *exponential formula* (3.10)–which is exact when the system is at equilibrium, and a good approximation when the vector field of the ODE system has almost-constant Jacobians $\nabla_x f$

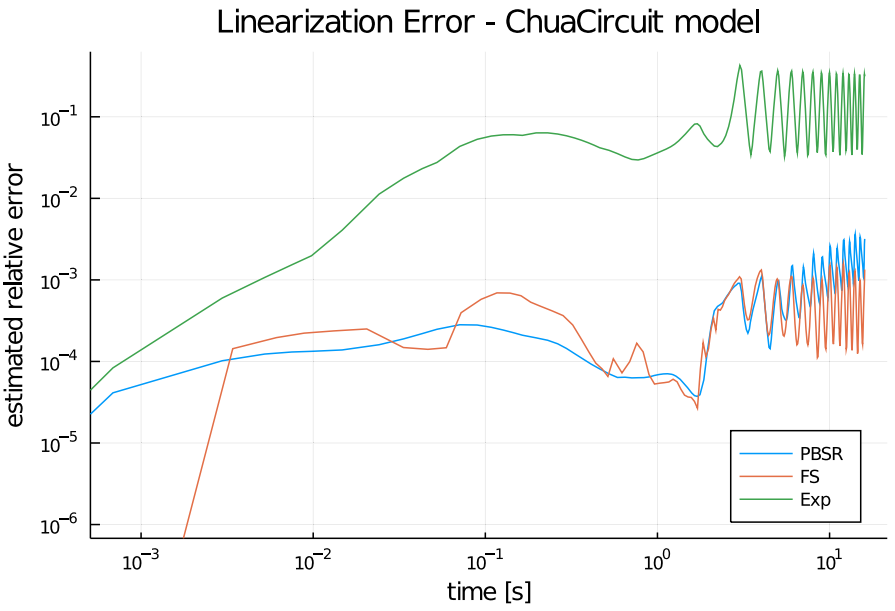

Fig. 8 The estimated relative error $\bar{\varkappa}$ for the Chua's Circuit model. The error estimate $\bar{\varkappa}$ uses small parameter perturbations $\delta_{\mathbf{p}}$ to test how well the sensitivity matrix predicts the changes in the solution (averaged over $N = 100$ small, random parameter perturbations)

and $\nabla_p f$. Combining these two formulas, we defined the *Peano-Baker series (PBS) algorithm* in Sect. 4.

As discussed in Sect. 5, the PBS algorithm may suffer from stability issues, in particular for stiff problems. To overcome this, we define two additional algorithms: the *PBS with refinement (PBSR) algorithm*, where we implement a refinement of the time grid, along with what can be viewed as a stiffness detection mechanism, and the *Exponential (Exp) algorithm*, where only the exponential formula is used (and no refinement is performed). For applications, such as the type of uncertainty quantification and MCMC sampling that originally motivated this work (see Sect. 1.1), these are the two algorithms we propose to use.

A rigorous error analysis, carried out in Sects. 6-7, showed that, under standard regularity assumptions, the proposed algorithm, based on the Peano-Baker series (with or without refinement of the time grid), admits a global error of order $\mathcal{O}(\Delta t_{\max}^2)$. The analysis also showed that the proposed method is optimal in the sense of at what term the Peano-Baker series is truncated.

The theoretical error analysis was complemented by several numerical experiments in Sect. 8. We compared the performance of the different methods to that of the forward sensitivity method for two ODE models from systems biology, a random linear system and a system modelling Chua's circuit. The results showed that both our algorithms produced accurate approximations of the sensitivity matrix with significant speed-up, that seemed to increase rapidly with the dimension of the problem. In the dynamical system modeling Chua's circuit, the limit trajectory of the ODE was a

limit cycle and thus the Jacobians never become (close to) constant. In this example, the PBSR algorithm produced approximations that had an accuracy of two orders of magnitude better than that of the Exp algorithm, motivating the use of the PBSR algorithm for accuracy in general problems. Further motivation for the PBSR comes from the applications in neuroscience that we considered, where ODE models are commonly characterised by time-dependent inputs (e.g. so-called Ca-spike trains). In such systems, the framework is similar to the Chua's circuit example, where the Jacobians are time-variant. Therefore, the PBSR algorithm is expected to be more precise, and thus more useful, than the Exp algorithm.

The methods presented in this paper are expected to be useful in the context of MCMC methods, particularly for problems arising in, e.g., systems biology: the speed-up provided by the PBSR and Exp algorithms, compared to forward sensitivity analysis, has the potential to drastically increase the efficiency of MCMC methods. As a first test of this, we equipped an implementation (in C, using the CVODES solver) of the simplified manifold Metropolis-adjusted Langevin algorithm (SMMALA, see, e.g., [17]) with the Exp algorithm to compute the Fisher Information and posterior gradients, and compared it to SMMALA with conventional forward sensitivity analysis. The (real) data used for MCMC was obtained at or near the steady-state of the system, so the use of the Exp algorithm was suitable. The near-steady-state sensitivity approximation enabled sampling approximately 100 times faster from the posterior distribution, compared to using CVODES' forward sensitivity analysis[13]. Similar tests for the PBSR algorithm will be part of future work on the integration of the proposed methods in the MCMC setting.

In addition to applications to MCMC sampling, future work includes further investigation of the impact of different implementation aspects of the PBSR algorithm–e.g. the criteria (based on the Jacobians $\nabla_x f$ and $\nabla_p f$) to switch from PBSR to exponential formula, and the refinement of the time grid of the ODE solver. Additional comparisons with existing methods and a better understanding of the methods performance, particularly in large-scale systems, is another important direction.

Acknowledgements We sincerely thank the referees for their thorough and insightful feedback regarding the first version of the paper. Their comments and questions have led to significant improvements, both in terms of presentation and content. The list of changes inspired by their reports is too long to include here, however, we highlight the discussion about stability issues throughout the paper, and the error analysis covering the case with refined time intervals (see Corollary 6.2); both additions are direct results of the feedback from the referees.

Funding Open access funding provided by Royal Institute of Technology. The research was supported by Swedish e-Science Research Centre (SeRC) and Science for Life Laboratory. The research of AK and OE was supported by EU/Horizon 2020 no. 945539 (HBP SGA3). The research of AK was also supported by the EU/Horizon 2020 no. 101147319 (EBRAINS 2.0 Project), European Union's Research and Innovation Program Horizon Europe no. 101137289 (the Virtual Brain Twin Project) and the Swedish Research Council (VR-M-2020-01652). The research of PN was supported in part by the Swedish Research Council (VR-2018-07050, VR-2023-03484) and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.
The computations were enabled by resources provided by the Swedish National Infrastructure for Computing (NAISS) at the PDC Center for High Performance Computing, KTH Royal Institute of Technology, partially

[13] With large samples sizes, a speed-up of that magnitude is difficult to test thoroughly, due to the time required by the slower sampler.

funded by the Swedish Research Council through grant agreement no. 2022-06725 and by LUNARC, The Centre for Scientific and Technical Computing at Lund University.

Declarations

Conflicts of Interest The authors declare no conflicts of interest. The funding is disclosed in the Acknowledgment section. This numerical study uses free software, the code and numerical results have been made available through publicly accessible repositories.

References

1. Frank, P.M.: Introduction to System Sensitivity Theory. Academic Press, Harcourt Brace Jovanovich, New York-London (1978)
2. Caracotsios, M., Stewart, W.E.: Sensitivity analysis of initial value problems with mixed ODEs and algebraic equations. Comput. Chem. Eng. **9**(4), 359–365 (1985)
3. Cacuci, D.G.: Sensitivity and Uncertainty Analysis, vol. I. Chapman & Hall/CRC, Boca Raton, FL (2003)
4. Saltelli, A., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Ratto, M.: Global Sensitivity Analysis: The Primer, 1st edn. Wiley-Interscience, New York (2008)
5. Cacuci, D.G., Ionescu-Bujor, M., Navon, I.M.: Sensitivity and Uncertainty Analysis, vol. II. Chapman & Hall/CRC, Boca Raton, FL (2005)
6. Calver, J., Enright, W.: Numerical methods for computing sensitivities for ODEs and DDEs. Numer. Algorithms **74**(4), 1101–1117 (2017)
7. Marchuk, G.I.: Adjoint Equations and Analysis of Complex Systems. Mathematics and its Applications, vol. 295. Kluwer Academic Publishers Group, Dordrecht (1995)
8. Marchuk, G.I., Agoshkov, V.I., Shutyaev, V.P.: Adjoint Equations and Perturbation Algorithms in Nonlinear Problems. CRC Press, Boca Raton, FL (1996)
9. Cao, Y., Li, S., Petzold, L., Serban, R.: Adjoint sensitivity analysis for differential-algebraic equations: The adjoint DAE system and its numerical solution. SIAM J. Sci. Comput. **24**(3), 1076–1089 (2003)
10. Gardner, D.J., Reynolds, D.R., Woodward, C.S., Balos, C.J.: Enabling new flexibility in the SUNDIALS suite of nonlinear and differential/algebraic equation solvers. ACM Transactions on Mathematical Software (TOMS) **48**(3), 1–24 (2022). https://doi.org/10.1145/3539801
11. Hindmarsh, A.C., Serban, R., Balos, C.J., Gardner, D.J., Reynolds, D.R., Woodward, C.S.: User Documentation for CVODES. Chap. 2.7. https://computing.llnl.gov/sites/default/files/cvs_guide-5.7.0.pdf
12. Baake, M., Schlägel, U.: The Peano-Baker series. Proceedings of the Steklov Institute of Mathematics **275** (2010)
13. Higham, D.J., Trefethen, L.N.: Stiffness of odes. BIT Numer. Math. **33**, 285–303 (1993)
14. Eriksson, O., Jauhiainen, A., Maad Sasane, S., Kramer, A., Nair, A.G., Sartorius, C., Hellgren Kotaleski, J.: Uncertainty quantification, propagation and characterization by Bayesian analysis combined with global sensitivity analysis applied to dynamical intracellular pathway models. Bioinformatics **35**, 284–292 (2019)
15. Gelman, A.: Bayesian Data Analysis, 3rd edn. Chapman Hall/CRC, United Kingdom (2013)
16. Kramer, A., Milinanni, F., Nyquist, P., Jauhiainen, A., Eriksson, O.: UQSA–an R-package for uncertainty quantification and sensitivity analysis for biochemical reaction network models. arXiv preprint arXiv:2308.05527 (2023)

17. Girolami, M., Calderhead, B.: Riemann manifold Langevin and Hamiltonian Monte Carlo methods. J. R. Stat. Soc. Ser. B Stat Methodol. **73**(2), 123–214 (2011)
18. Garcia Trillos, N., Sanz-Alonso, D.: The Bayesian update: variational formulations and gradient flows. Bayesian Anal. **15**(1), 29–56 (2020)
19. Barp, A., Briol, F.-X., Kennedy, A.D., Girolami, M.: Geometry and dynamics for Markov chain Monte Carlo. Annu. Rev. Stat. Appl. **5**, 451–474 (2018)
20. Rao, C.R.: Information and the Accuracy Attainable in the Estimation of Statistical Parameters. In: Kotz, S., Johnson, N.L. (eds.) Breakthroughs in Statistics: Foundations and Basic Theory, vol. 1, pp. 235–247. Springer, New York, NY (1992)
21. Kramer, A.: Stochastic Methods for Parameter Estimation and Design of Experiments in Systems Biology, p. 161. Logos Verlag, Berlin (2016)
22. Brockett, R.W.: Finite Dimensional Linear Systems, Siam edn. Classics in applied mathematics ; 74. siam, Cambridge, Massachusetts (2015)
23. Rugh, W.J.: Linear System Theory. Prentice Hall, Englewood Cliffs, N.J. (1993)
24. Hairer, E., Nørsett, S.P., Wanner, G.: Solving Ordinary Differential Equations. I, 2nd edn. Springer Series in Computational Mathematics, vol. 8. Springer, Berlin (1993)
25. Hairer, E., Wanner, G.: Solving Ordinary Differential Equations. II. Springer Series in Computational Mathematics, vol. 14. Springer, Berlin (2010)
26. Griffiths, D.F., Higham, D.J.: Numerical Methods for Ordinary Differential Equations. Springer Undergraduate Mathematics Series. Springer, London (2010)
27. Sauer, T.: Numerical Analysis. Pearson/Addison Wesley, Boston (2006)
28. Scarborough, J.B.: Numerical Mathematical Analysis, 2. ed. edn., Baltimore (1950)
29. Higham, N.J.: The scaling and squaring method for the matrix exponential revisited. SIAM Rev. **51**(4), 747–764 (2009)
30. Church, T.W., Tewatia, P., Hannan, S., Antunes, J., Eriksson, O., Smart, T.G., Hellgren Kotaleski, J., Gold, M.G.: AKAP79 enables calcineurin to directly suppress protein kinase a activity. bioRxiv (2021)
31. Nair, A.G., Gutierrez-Arenas, O., Eriksson, O., Jauhiainen, A., Blackwell, K.T., Kotaleski, J.H.: Modeling intracellular signaling underlying striatal function in health and disease. In: Blackwell, K.T. (ed.) Computational Neuroscience. Progress in Molecular Biology and Translational Science, vol. 123, pp. 277–304. Academic Press, Cambridge, Massachusetts (2014)