Biol. Cybern. 81, 381–402 (1999)

Biological Cybernetics

Exact digital simulation of time-invariant linear systems with applications to neuronal modeling

Stefan Rotter, Markus Diesmann

Neurobiologie und Biophysik, Institut für Biologie III, Universität Freiburg, Freiburg, Germany

Received: 3 October 1998 / Accepted in revised form: 19 March 1999

Abstract. An efficient new method for the exact digital simulation of time-invariant linear systems is presented. Such systems are frequently encountered as models for neuronal systems, or as submodules of such systems. The matrix exponential is used to construct a matrix iteration, which propagates the dynamic state of the system step by step on a regular time grid. A large and general class of dynamic inputs to the system, including trains of δ-pulses, can be incorporated into the exact simulation scheme. An extension of the proposed scheme presents an attractive alternative for the approximate simulation of networks of integrate-and-fire neurons with linear sub-threshold integration and non-linear spike generation. The performance of the proposed method is analyzed in comparison with a number of multi-purpose solvers. In simulations of integrate-and-fire neurons, Exact Integration systematically generates the smallest error with respect to both sub-threshold dynamics and spike timing. For the simulation of systems where precise spike timing is important, this results in a practical advantage in particular at moderate integration step sizes.

1 Introduction

Computational neuroscience, like the study of any complex dynamic system, depends much on the use of reliable and effective numerical methods. Approximate digital simulation of the system behavior under various conditions is exceedingly helpful for exploration and as a first step of analysis, as well as for demonstration purposes. Detailed knowledge of the numerical properties of the simulation method, however, is critical for the success of its application and, eventually, for the credibility of the results.

Correspondence to: S. Rotter
Institut für Biologie III, Schänzlestrasse 1, D-79104 Freiburg, Germany (e-mail: rotter@biologie.uni-freiburg.de
Tel.: +49-761-2032862, Fax: +49-761-2032860)

Approximate digital simulation of a function $y(t)$ involves the computation of a sequence of samples of the function on a discrete temporal grid. This is typically achieved by iteration, where $y(t+\Delta)$ for some step of size $\Delta > 0$ is determined from $y(t)$ and from knowledge about the local properties of the function $y(t)$. The success of this procedure essentially relies on Taylor's formula, which explicitly states what is meant by local information:

$$y(t+\Delta) = y(t) + \dot{y}(t)\Delta + \tfrac{1}{2}\ddot{y}(t)\Delta^2 + \tfrac{1}{6}\dddot{y}(t)\Delta^3 + \cdots \qquad (1)$$

If $y(t)$ is the trajectory of a dynamic system, the various temporal derivatives are related by the differential equations which govern the system dynamics. Different methods for the approximate numerical integration of these equations are distinguished by their ability to faithfully approximate the above power series for sufficiently small steps Δ. State-of-the-art approximate digital simulation of dynamic systems is based on multi-purpose adaptive solvers. Beginning with the initial conditions, the solver steps through time, computing a solution at each time step. If the solution satisfies the prescribed error tolerance criteria, it is a successful step. Otherwise, the results are discarded, the solver shrinks the step size and tries again.

For linear time-invariant systems, the approximate nature of digital simulation can be completely overcome. It is possible to compute the exact trajectories, sampled on a regular grid of arbitrary step size, with a precision depending only on the floating-point arithmetic used. This is achieved by iteration of a linear map on a suitable state space, without reference to a precomputed explicit solution, by using the full Taylor series. To avoid signal aliasing, one only needs to make sure that the grid is dense enough, compared to the time scale of change in the signal.

A fair number of dynamic systems which play a role in neuronal modeling are both linear and time-invariant. The list includes stochastic models for multi-state ion channel kinetics (Colquhoun and Hawkes 1995a,b), compartmental models for the passive spread of current in dendritic cables (Rall 1964; Hines and Carnevale

1997), and the leaky integrator model for spatio-temporal summation of synaptic currents (Tuckwell 1988).

Spatio-temporal integration of synaptic inputs coming as transient changes of ionic conductances (Rall 1964; Hines and Carnevale 1997) are described in terms of linear equations, which are not time-invariant. The dynamics of voltage-sensitive conductances under current-clamp conditions and the generation of action potentials (Hodgkin and Huxley 1952) involves essentially non-linear equations. Both types of systems must be treated approximately with general approximate methods.

Some relevant non-linear systems, however, like the integrate-and-fire neuron, can be viewed as a cascade of a dynamic linear part (sub-threshold leaky integration of inputs) and a static non-linearity (threshold operation for spiking). The proposed integration method can be extended in a natural way to account for this type of non-linearity. In addition, for many problems in neuronal systems modeling, it is mandatory for computational reasons to fix a step size for the iteration. Such is the case for high-dimensional systems with many threshold operations involved (large neural networks), and for systems with stochastic inputs, which are defined by their spectral properties (shot noise). Our exposition in Sects. 3 and 5 gives a more detailed account of the scope of the proposed method.

In the sequel, we first provide the mathematical background and give a short derivation of the exact simulation method (Sect. 2, Appendices A and B). Its application is then explained in detail for a collection of frequently encountered time-invariant linear systems, with and without input (Sect. 3). Finally, we discuss the advantage of using the proposed new method in comparison with classical integration methods (Sect. 4, Appendices C and D).

2 Mathematical background

2.1 Linear differential equations

We consider a time-invariant linear system, the behavior of which is specified by a first-order linear differential equation in n dimensions

$$\dot{y} = Ay + x \ . \tag{2}$$

Here, $y(t)$ is the time-dependent state of the system, and $x(t)$ is the time-dependent input to the system. Both x and y are n-dimensional column vectors with real or complex components. The system is characterized by a fixed square matrix A of numerical constants. By appropriate substitution of variables, any higher-order linear differential equation can be written as a first-order system.

A fundamental system of solutions to the homogeneous (zero-input) equation $\dot{y} = Ay$ is given by the columns of the matrix exponential e^{At}. This can be checked by substituting all derivatives $y^{(k)} = A^k y$ in Taylor's formula (1). In Appendix A, a more precise definition of the matrix exponential and a discussion of its properties is provided. The unique solution of the full equation (2) with initial value $y(s)$ amounts to (Arnol'd 1992; Hirsch and Smale 1974; Walter 1996)

$$y(t) = e^{A(t-s)}y(s) + \int_{s+}^{t} e^{A(t-\tau)}x(\tau)d\tau \tag{3}$$

where the convolution integral extends over the half-open interval $(s, t]$. The first part of the sum is the result of passive propagation of the initial state, whereas the second part represents the input-driven response of the system. Correspondingly, for a system with no input, the matrix e^{At} is termed "time-evolution operator" or "propagator". In contrast, for a system with input but zero initial conditions, the same matrix is called the "impulse response" of the system.

As can be directly seen from the explicit solution (3), the dynamic history of the system prior to time s is completely subsumed in its state $y(s)$ at that time, and is otherwise "forgotten". This fact can be used to set up a simple method for the exact digital simulation of linear system behavior.

2.2 Exact digital simulation in discrete time

Digital simulation means to compute the response $y(t)$ of the system to a prescribed input $x(t)$ on an evenly sampled grid $t_k = k\Delta$, where Δ is a fixed step size and k takes only integer values. We write $y_k \equiv y(t_k)$ for brevity. The function $y(t)$ then corresponds to the sequence y_k of its samples on the grid. For a special type of input functions, the simulation can be performed in an exact way, avoiding potentially inaccurate and unstable integration methods. To this end, we consider (generalized) functions $x(t)$ of the form

$$x(t) = \sum_k x_k \delta(t - t_k) \ , \tag{4}$$

where x_k is an n-dimensional vector for each k, and $\delta(t)$ is the scalar Dirac delta-function; see Appendix B for a short discussion of its properties. Such "pulse trains" are completely defined by the sequence of coefficients x_k. The somewhat different nature of pulse amplitudes x_k and function samples y_k should be kept in mind.

For pulse train inputs which are restricted to the grid, the temporal evolution of the continuous system (2) collapses to a discrete matrix equation. Namely, if we let $s = t_k$ and $t = t_{k+1}$ be two successive points on the grid, it is readily verified that the general solution (3) turns into

$$y_{k+1} = e^{A\Delta}y_k + x_{k+1} \tag{5}$$

which can be interpreted as an iteration. Starting with some initial state y_0 and assuming non-zero input only at t_k for $k = 1, 2, \ldots$, it propagates the exact solution on the grid, step by step. The diagram

$$\begin{array}{ccccccccc} & & x_1 & & x_2 & & x_3 & & \cdots \\ & & \downarrow & & \downarrow & & \downarrow & & \\ y_0 & \rightarrow & y_1 & \rightarrow & y_2 & \rightarrow & y_3 & \rightarrow & \cdots \end{array}$$

depicts the dependency of the current output of the system on its previous output and the current input. We refer to the iteration (5) as the method of "Exact Integration". For a time-invariant system, it is based on the fixed numerical matrix $e^{A\Delta}$, which has to be computed only once by using appropriate standard numerical algorithms (see Appendix A for a more detailed discussion).

2.3 Exact simulation with general inputs

In the context of Exact Integration of a time-invariant linear system, pulse trains of the form (4) can be used to synthesize a rather large class of more general input functions. This is achieved by simply adding equations to the system and then applying the Exact Integration method to the extended system. This qualifies, for instance, piecewise polynomials of bounded degree, sums of sine waves, or sums of exponentials as input functions to a linear system subject to exact digital simulation.

The most general input function $x(t)$ compatible with Exact Integration is itself a linear image of the output $v(t)$ of another linear system with pulse train input $u(t)$

$$\dot{v} = Bv + u$$
$$x = Cv$$

for suitable time-invariant coefficient matrices B and C. An exact digital simulation of the system (2) with this particular input is then accomplished by applying the above methods to the combined system as a whole

$$\begin{bmatrix} \dot{v} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} B & 0 \\ C & A \end{bmatrix} \begin{bmatrix} v \\ y \end{bmatrix} + \begin{bmatrix} u \\ 0 \end{bmatrix} \ .$$

Several examples for this strategy are given in Sect. 3.2.

3 Examples

3.1 Initial value problems

We will now elaborate on the general method of exact digital simulation by discussing a number of simple cases. The examples chosen cover all fundamental types of stable system behavior (Hirsch and Smale 1974): "exponentially damped" (real eigenvalues of the coefficient matrix), "oscillatory" (complex eigenvalues of the coefficient matrix) and "polynomial" (nilpotent coefficient matrix). We consider first linear constant-coefficient initial value problems with no input. In such a case, the iteration (5) reduces to a repeated matrix multiplication, which propagates the initial vector y(0) on a regular time grid in steps of size Δ.

3.1.1 Exponential decay

The most elementary example, which nevertheless illustrates almost all important aspects of the method, is that of a simple exponential decay

$$\dot{\eta} + a\eta = 0, \qquad \eta(0) = \eta_0$$

for a scalar variable η and no external input, whatsoever. The solution of this initial value problem is given by

$$\eta(t) = \eta_0 e^{-at} \ .$$

If we bring the equation to the normal form (2), we identify

$$x = 0, \quad y = \eta, \quad y(0) = \eta_0, \quad A = -a \ .$$

The iteration (5) yields the sequence

$$y_{k+1} = e^{A\Delta} y_k = \left(e^{A\Delta}\right)^k y_0 = \eta_0 e^{-ak\Delta}$$

which is clearly an exact sample of the solution η on the grid. Figure 1 gives a numerical example.

3.1.2 Alpha- and beta-functions

The solution of the second-order equation

$$\ddot{\eta} + (a+b)\dot{\eta} + (ab)\eta = 0, \quad \eta(0) = 0, \ \dot{\eta}(0) = \dot{\eta}_0$$

is called beta-function for $a \neq b$ and alpha-function for $a = b$. Both functions are in use for modeling post-synaptic effects in neurons (Bernard et al. 1994; Jack et al. 1983). The explicit form of alpha- and beta-functions are

$$\eta(t) = \dot{\eta}_0 t e^{-at} \quad \text{and} \quad \eta(t) = \frac{\dot{\eta}_0}{b-a}\left(e^{-at} - e^{-bt}\right) \ ,$$

respectively. The first step to obtain a sample of these functions on the grid without reference to the analytical expressions is to rephrase the differential equation as a two-dimensional first-order system. This can be achieved by many different variable substitutions. Numerical arguments in view of the matrix iteration, however, may favor particular choices. A convenient arrangement in this particular example is a cascade of one-dimensional systems, the first with decay constant a feeding the second with decay constant b. The normal form of the equations then is

$$x = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad y = \begin{bmatrix} b\eta + \dot{\eta} \\ \eta \end{bmatrix},$$
$$y(0) = \begin{bmatrix} \dot{\eta}_0 \\ 0 \end{bmatrix}, \quad A = \begin{bmatrix} -a & 0 \\ 1 & -b \end{bmatrix} \ .$$

An analytical expression for the matrix exponential is

$$e^{A\Delta} = \begin{bmatrix} e^{-a\Delta} & 0 \\ \Delta e^{-a\Delta} & e^{-a\Delta} \end{bmatrix}$$

for the alpha-function, and

$$e^{A\Delta} = \begin{bmatrix} e^{-a\Delta} & 0 \\ \frac{1}{b-a}\left(e^{-a\Delta} - e^{-b\Delta}\right) & e^{-b\Delta} \end{bmatrix}$$

for the beta-function, respectively. For the purpose of exact digital simulation, the matrix exponential can also

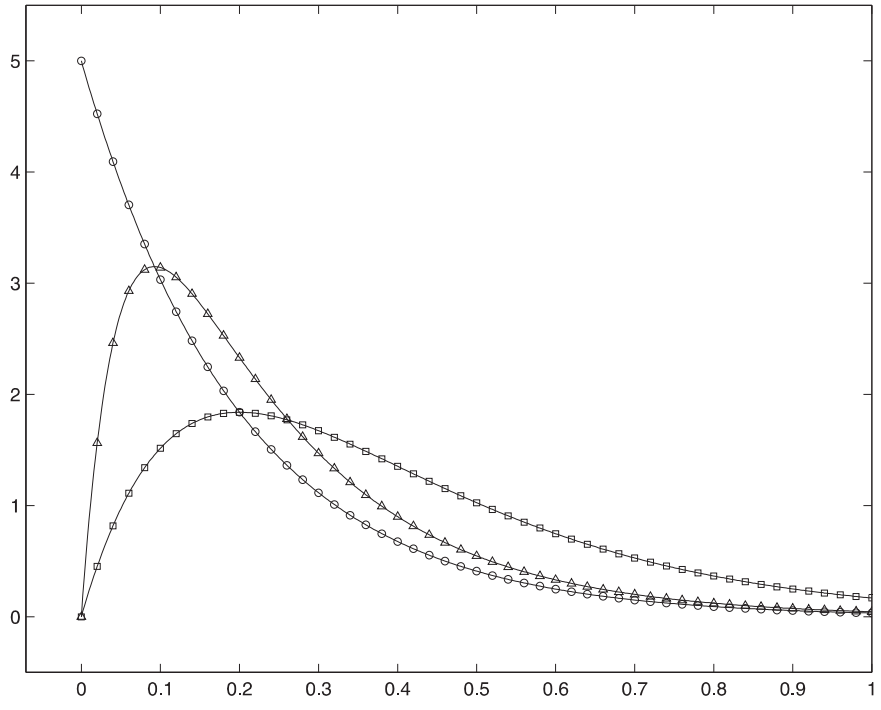

Fig. 1. Exponential decay, alpha- and beta-functions. Exact digital simulation of the initial value problems $\dot{\eta} + a\eta = 0, \eta(0) = a$ with parameter $a = 5$ (exponential decay, *circles*) and $\ddot{\eta} + (a+b)\dot{\eta} + (ab)\eta = 0$, $\eta(0) = 0$, $\dot{\eta}(0) = ab$ for $a = b = 5$ (alpha- function, *squares*) and $a = 5$, $b = 20$ (beta- function, *triangles*). The step size for the iteration was $\Delta = 0.02$. The *solid lines* are plots of the exact solutions $\eta(t) = ae^{-at}$, $\eta(t) = a^2te^{-at}$ and $\eta(t) = \frac{ab}{b-a}\left(e^{-at} - e^{-bt}\right)$, respectively. The initial conditions were chosen such that $\int_0^\infty \eta(t)dt = 1$ in all three cases

be obtained with the help of appropriate numerical routines (see Appendix A), which are as easy to apply as the routines computing the exponential of a real number. Figure 1 gives an illustration of the method for alpha- and beta-functions.

3.1.3 Harmonic oscillations
The solution of

$$\ddot{\eta} + \omega^2\eta = 0, \quad \eta(0) = p, \quad \dot{\eta}(0) = q\omega$$

is a harmonic oscillation with frequency ω, amplitude $c = \sqrt{p^2 + q^2}$ and phase $\phi = \arctan(p/q)$

$$\eta(t) = p\cos(\omega t) + q\sin(\omega t) = c\sin(\omega t + \phi) \ .$$

For the digital simulation of a harmonic oscillation without reference to its analytic description, we first rewrite the differential equation in a form compatible with (2). We choose variables such that the coefficient matrix attains a symmetric shape

$$x = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad y = \begin{bmatrix} \frac{1}{\omega}\dot{\eta} \\ \eta \end{bmatrix}, \quad y(0) = \begin{bmatrix} q \\ p \end{bmatrix},$$
$$A = \begin{bmatrix} 0 & -\omega \\ \omega & 0 \end{bmatrix} \ .$$

The matrix exponential for this particular system is

$$e^{A\Delta} = \begin{bmatrix} \cos(\omega\Delta) & -\sin(\omega\Delta) \\ \sin(\omega\Delta) & \cos(\omega\Delta) \end{bmatrix}$$

corresponding to a rotation in y-space by an angle of $\omega\Delta$. Figure 2 shows a numerical illustration.

3.1.4 Damped harmonic oscillations
We now consider the damped system

$$\ddot{\eta} + 2\mu\dot{\eta} + \left(\mu^2 + \nu^2\right)\eta = 0, \quad \eta(0) = p, \quad \dot{\eta}(0) = q\nu - p\mu \ .$$

It describes a damped oscillation with frequency ν, amplitude decay rate μ, amplitude factor $c = \sqrt{p^2 + q^2}$ and phase $\phi = \arctan(p/q)$

$$\eta(t) = p\cos(\nu t)e^{-\mu t} + q\sin(\nu t)e^{-\mu t} = c\sin(\nu t + \phi)e^{-\mu t} \ .$$

For a digital simulation of this function, we again rewrite the differential equation in a form compatible with (2), putting $\rho = \sqrt{\mu^2 + \nu^2}$

$$x = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad y = \begin{bmatrix} \frac{1}{\rho}\dot{\eta} \\ \eta \end{bmatrix}, \quad y(0) = \begin{bmatrix} \frac{1}{\rho}(q\nu - p\mu) \\ p \end{bmatrix},$$
$$A = \begin{bmatrix} -2\mu & -\rho \\ \rho & 0 \end{bmatrix} \ .$$

In this case now, the matrix exponential is

$$e^{A\Delta} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\cos(\nu\Delta)e^{-\mu\Delta} + \begin{bmatrix} -\frac{\mu}{\nu} & -\frac{\rho}{\nu} \\ \frac{\rho}{\nu} & \frac{\mu}{\nu} \end{bmatrix}\sin(\nu\Delta)e^{-\mu\Delta} \ .$$

Figure 2 shows the numerical simulation of a damped oscillation using Exact Integration.

3.1.5 Polynomial functions
Newton's equations for the ballistic movement of an ideal canon ball imply a parabolic trajectory. More general, for any non-negative integer n, the differential equation

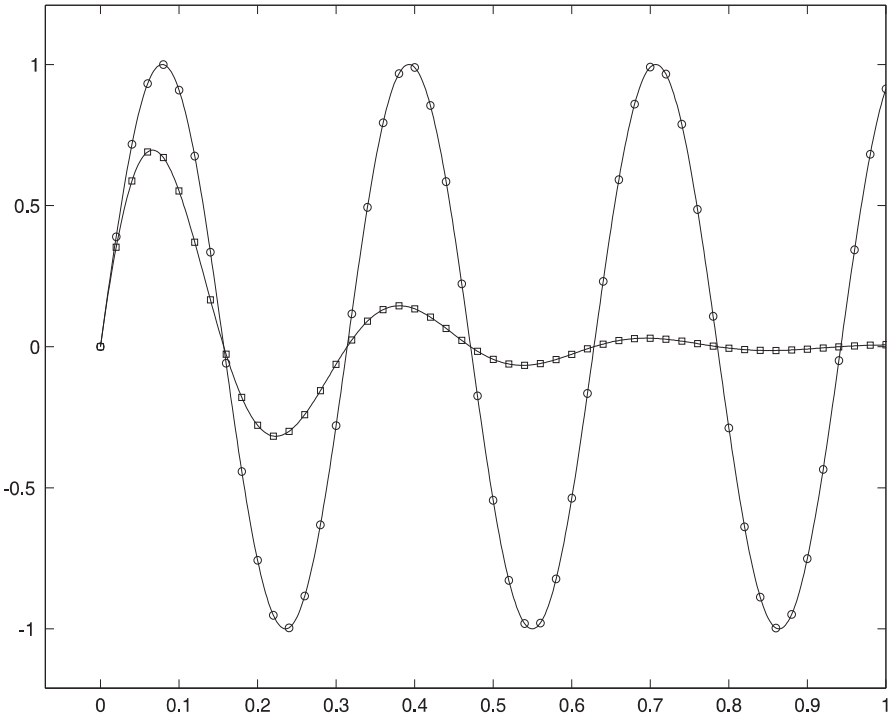

Fig. 2. Oscillations. Exact digital simulation of harmonic oscillations with and without damping. An iterative solution of the initial value problem $\ddot{\eta} + \omega^2 \eta = 0$, $\eta(0) = 0, \dot{\eta}(0) = \omega$ with $\omega = 20$ is indicated by *circles*. The *squares* correspond to the initial value problem $\ddot{\eta} + 2\mu\dot{\eta} + (\mu^2 + \nu^2)\eta = 0$, $\eta(0) = 0$, $\dot{\eta}(0) = \nu$ with $\nu = 20$ and $\mu = 5$. The step size of both iterations was $\Delta = 0.02$. The *solid lines* are plots of the exact analytical solutions $\eta(t) = \sin(\omega t)$ and $\eta(t) = \sin(\nu t)\mathrm{e}^{-\mu t}$, respectively

$$\eta^{(n)} = 0$$

with initial conditions

$$\eta^{(k)}(0) = k! a_k \quad (k = 0, 1, \ldots, n-1)$$

is uniquely solved by the polynomial function

$$\eta(t) = a_0 + a_1 t + \cdots + a_{n-1} t^{n-1} \ .$$

The exact digital simulation of any polynomial function can be obtained by means of an iteration of the form (5). In the case of a cubic polynomial function ($n = 4$), for example, one puts

$$x = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad y = \begin{bmatrix} \dddot{\eta} \\ \ddot{\eta} \\ \dot{\eta} \\ \eta \end{bmatrix}, \quad y(0) = \begin{bmatrix} 6a_3 \\ 2a_2 \\ a_1 \\ a_0 \end{bmatrix},$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} .$$

The matrix exponential for this particular system is

$$\mathrm{e}^{A\Delta} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \Delta & 1 & 0 & 0 \\ \frac{\Delta^2}{2} & \Delta & 1 & 0 \\ \frac{\Delta^3}{6} & \frac{\Delta^2}{2} & \Delta & 1 \end{bmatrix} .$$

Samples for the exact simulation of polynomial functions based on this time evolution operator are given in Fig. 3.

3.2 Linear systems with input

We now seek a digital simulation of the response of specific linear time-invariant systems to prescribed inputs, restricted to a regular grid with a fixed step size Δ. This can be accomplished by using the general form of the iteration (5). We assume zero initial conditions throughout this section.

3.2.1 Low-pass filtered impulse sequence

As a first example, we consider a one-dimensional first-order system at rest with a non-zero input ξ switched on at time 0

$$\dot{\eta} + a\eta = \xi, \qquad \eta(0) = 0 \ .$$

As input, we consider a one-dimensional pulse train on the grid, as discussed in Sect. 2.2. The response η then is a low-pass filtered version of the input

$$\eta(t) = \int_0^t \mathrm{e}^{-a(t-\tau)} \xi(\tau) d\tau \ .$$

As above, one identifies

$$x = \xi, \quad y = \eta, \quad y(0) = 0, \quad A = -a \ .$$

The result of a discrete iteration according to (5) is illustrated in Fig. 4 and describes a scalar system which relaxes from its previous state according to its autonomous dynamics, and which then updates its initial conditions to satisfy the input. This interpretation is valid also for higher-dimensional examples.

3.2.2 Shot noise

The response of a linear system is called "shot noise", if the pulse train input is a realization of a Poisson process

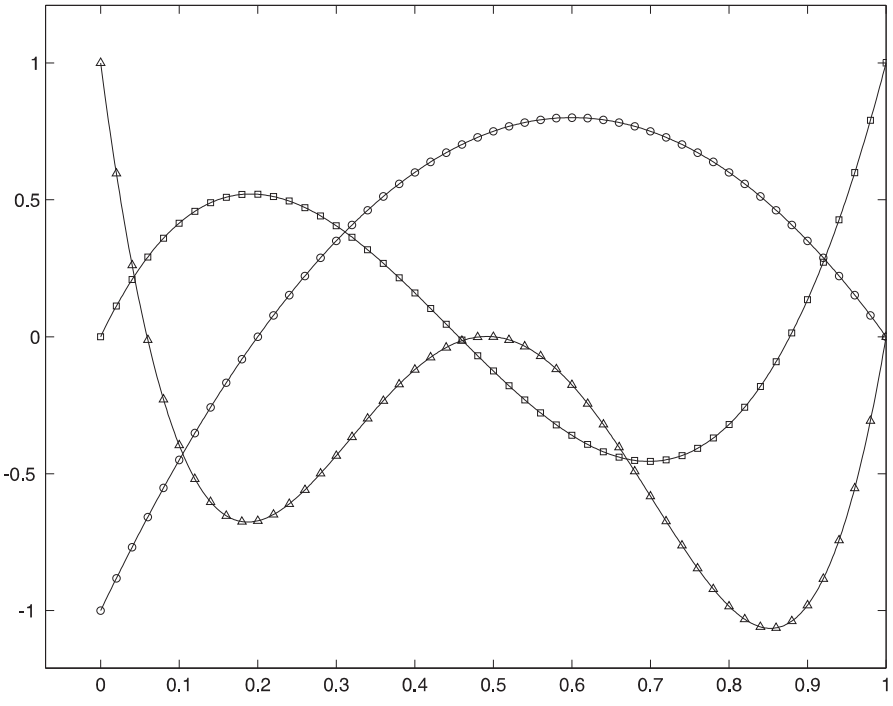

Fig. 3. Polynomial functions. Exact digital simulation of the polynomial functions $\eta(t) = -5t^2 + 6t - 1$, $\eta(t) = 15t^3 - 20t^2 + 6t$, and $\eta(t) = 70t^4 - 143t^3 + 94t^2 - 22t + 1$. Shown are the iterative solutions to the linear differential equation $\eta^{(n)} = 0$ for $n = 3$ (*circles*), $n = 4$ (*squares*), and $n = 5$ (*triangles*), respectively, with appropriate initial conditions. The iteration had a step size of $\Delta = 0.02$. The *solid lines* are plots of the exact analytical functions

(Papoulis 1991). We consider an example from neuronal modeling. The ionic current, which is induced at a synapse in response to a pre-synaptic action potential, is approximated by an alpha-function (see Sect. 3.1.2). Provided that dendritic integration is linear and that all synapses have equal weights, the total current induced by a barrage of pre-synaptic action potentials ξ is proportional to a variable ψ such that

$$\ddot{\psi} + 2a\dot{\psi} + a^2\psi = \xi, \qquad \psi(0) = 0, \quad \dot{\psi}(0) = 0 \ ,$$

where a is the decay constant of the current. The post-synaptic potential η relative to the resting level is a low-pass filtered version of the current

$$\dot{\eta} + b\eta = \psi \ .$$

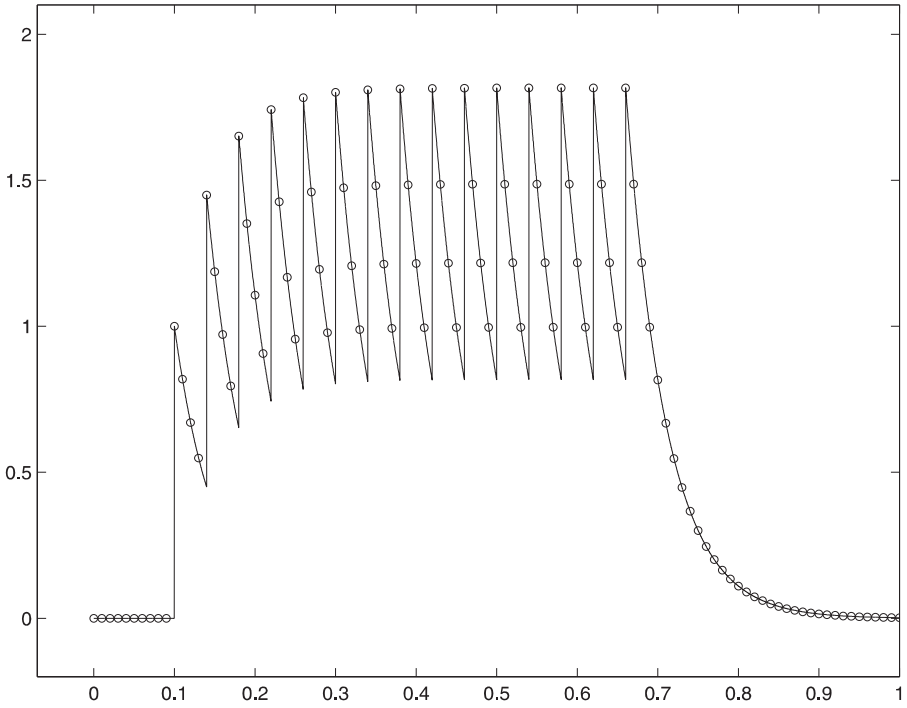

Fig. 4. Low-pass filtered pulse train. Exact digital simulation of the response of the linear system $\dot{\eta} + a\eta = \xi, \eta(0) = 0$ with parameter $a = 20$ to a pulse train ξ of 15 equally spaced impulses with unit amplitude. The step size of the iteration is $\Delta = 0.01$. The *solid line* is a plot of the exact solution $\eta(t) = \int_0^t \mathrm{e}^{-a(t-\tau)}\xi(\tau)d\tau$. Note that the simulated time series cannot fully convey the abrupt changes in the signal, but since all inputs come on the grid, the simulation is nevertheless exact

Here, b is the decay constant of the neuronal membrane. The cascaded set of equations describing the complete integration process has the normal form

$$x = \begin{bmatrix} \xi \\ 0 \\ 0 \end{bmatrix}, \quad y = \begin{bmatrix} a\psi + \dot{\psi} \\ \psi \\ \eta \end{bmatrix}, \quad y(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$$
$$A = \begin{bmatrix} -a & 0 & 0 \\ 1 & -a & 0 \\ 0 & 1 & -b \end{bmatrix}.$$

Assuming that all pre-synaptic spikes arrive on the grid, the shot noise realization can be exactly simulated by the iteration method, an illustration is given in Fig. 5. The analytic form of the matrix exponential is already somewhat complex

$$e^{A\Delta} = \begin{bmatrix} e^{-a\Delta} & 0 & 0 \\ \Delta e^{-a\Delta} & e^{-a\Delta} & 0 \\ \frac{e^{-b\Delta}-e^{-a\Delta}}{(a-b)^2} - \frac{\Delta e^{-a\Delta}}{a-b} & \frac{e^{-b\Delta}-e^{-a\Delta}}{a-b} & e^{-b\Delta} \end{bmatrix}.$$

It is nevertheless given here to point out the occurrence of the "correction term" in the lower-left corner of the matrix. The corresponding entry in the coefficient matrix A is zero. In the propagator matrix $e^{A\Delta}$, however, it must be nonzero to ensure the exactness of the simulation. This issue will be discussed in Sect. 4 in more detail.

3.2.3 Piecewise-constant input

The requirement that the input to the system must be an impulse sequence can be overcome to some extent by adding equations to the system (see Sect. 2.3). A piecewise constant signal ψ, for instance, is obtained as the solution of

$$\dot{\psi} = \xi, \qquad \psi(0) = \psi_0 \ ,$$

where ξ is again a pulse train. The function ψ solving the equation is constant between any two successive pulses. The heights of the jumps are specified by the input function ξ. We feed ψ as input to a first-order low-pass system, which responds with a function η

$$\dot{\eta} + a\eta = \psi, \qquad \eta(0) = \eta_0 \ .$$

Altogether, one obtains the cascade

$$x = \begin{bmatrix} \xi \\ 0 \end{bmatrix}, \quad y = \begin{bmatrix} \psi \\ \eta \end{bmatrix}, \quad y(0) = \begin{bmatrix} \psi_0 \\ \eta_0 \end{bmatrix}, \quad A = \begin{bmatrix} 0 & 0 \\ 1 & -a \end{bmatrix}.$$

Provided that the jumps occur only on the grid, one can simulate this two-dimensional system in an exact manner. To this end, one needs to compute the appropriate matrix exponential. In practical applications, this would be done numerically. The analytical expression is

$$e^{A\Delta} = \begin{bmatrix} 1 & 0 \\ \frac{1}{a}\left(1 - e^{-a\Delta}\right) & e^{-a\Delta} \end{bmatrix}.$$

The iteration of this system according to (5) has been termed "exponential integration" (MacGregor 1987) and is widely used in the context of neuronal model simulations. Some aspects of its use in the literature are discussed in Sect. 4 and Appendix C.6. An illustration is given in Fig. 6.

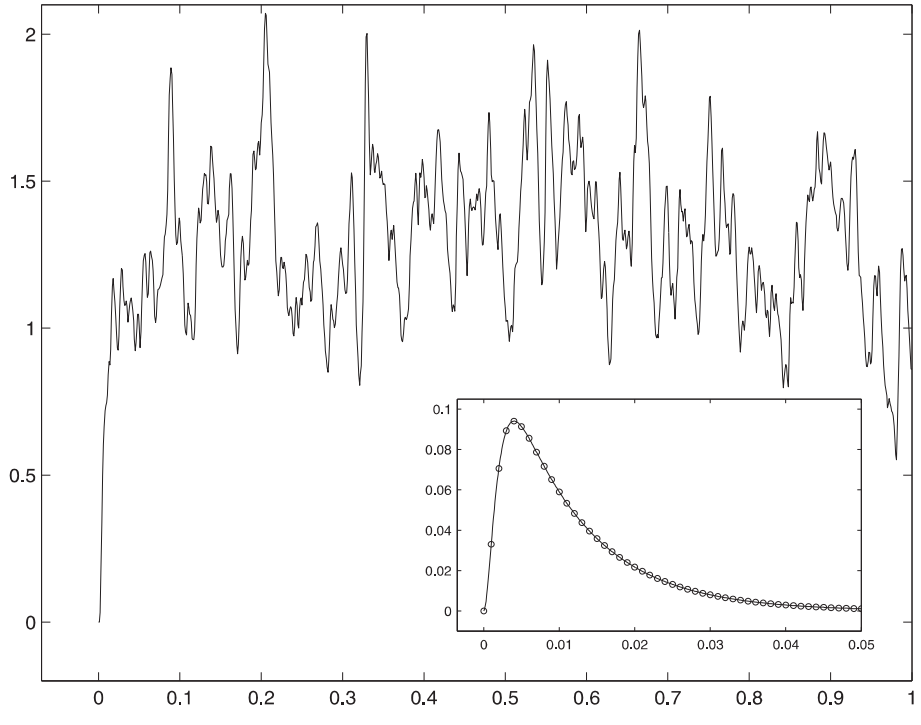

Fig. 5. Shot noise. Exact digital simulation of the response η of a passive linear dendrite to a barrage of action potentials arriving in a Poisson-like manner. The presynaptic process ξ had an expected rate of 1000 spikes for the simulation interval shown. The occurrence of spikes was constrained to the grid with a time step $\Delta = 0.001$. We chose $a = 1000$ for the decay constant of the alpha-function describing unitary post-synaptic currents; the decay constant of the membrane was $b = 100$. After a short transient while the membrane is charged from rest, one observes fluctuations around an equilibrium level. The *inset* shows a unitary postsynaptic potential; the *circles* indicate the time-steps of the simulation. The same (arbitrary) units are used for both figures

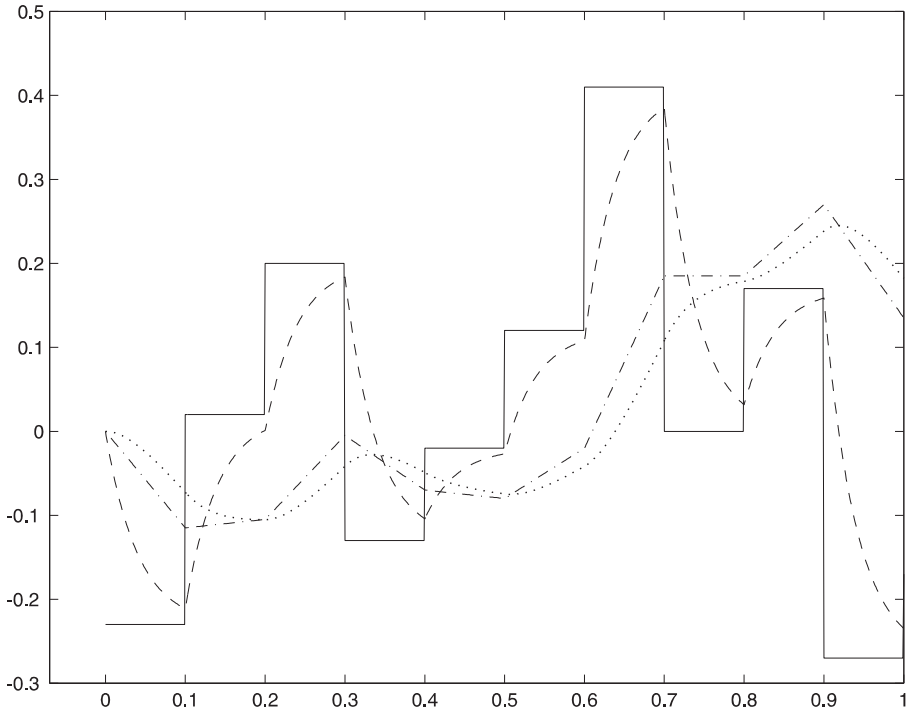

Fig. 6. Piecewise constant and continuous piecewise-linear input. Exact digital simulation of a low-pass system with piecewise constant (ψ: —, η: - - -) and continuous piecewise-linear (ψ: $-\cdot-$, η: $\cdots$) input, respectively. The low-pass system is given by the differential equation $\dot{\eta} + a\eta = a\psi$ with $a = 25$. The step size of both iterations was $\Delta = 0.001$, the graphs shown are linear interpolations of the actual data points

3.2.4 Continuous piecewise-linear input

To go one step further, we want the input ψ to be continuous and piecewise-linear. Such a function is obtained as a solution of

$$\ddot{\psi} = \xi, \qquad \psi(0) = \psi_0, \quad \dot{\psi}(0) = \dot{\psi}_0 \ ,$$

where now ξ is a pulse train describing the changes in slope of the input function. As above, we feed ψ into a first-order low-pass. The equations describing the whole system are then

$$x = \begin{bmatrix} \xi \\ 0 \\ 0 \end{bmatrix}, \quad y = \begin{bmatrix} \dot{\psi} \\ \psi \\ \eta \end{bmatrix}, \quad y(0) = \begin{bmatrix} \dot{\psi}_0 \\ \psi_0 \\ \eta_0 \end{bmatrix},$$

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & -a \end{bmatrix} .$$

The matrix exponential is

$$\mathrm{e}^{A\Delta} = \begin{bmatrix} 1 & 0 & 0 \\ \Delta & 1 & 0 \\ \frac{1}{a^2}\left(\mathrm{e}^{-a\Delta} + a\Delta - 1\right) & \frac{1}{a}\left(1 - \mathrm{e}^{-a\Delta}\right) & \mathrm{e}^{-a\Delta} \end{bmatrix} .$$

A numerical example, where again all changes in slope occur on the grid, is shown in Fig. 6. Input functions with any given degree of regularity, like cubic splines, can be synthesized and incorporated into the description of a linear system by further generalizing the principle indicated by the preceding two examples.

3.2.5 Damped driven oscillations

We now construct an iteration to simulate the response of a damped oscillator system, which is driven by a periodic force $\psi(t) = \cos(\omega t)$. The equations are

$$\ddot{\eta} + 2\mu\dot{\eta} + \left(\mu^2 + v^2\right)\eta = \psi, \qquad \eta(0) = 0, \quad \dot{\eta}(0) = 0$$

damped oscillator, and

$$\ddot{\psi} + \omega^2\psi = 0, \qquad \psi(0) = 1, \quad \dot{\psi}(0) = 0$$

for the periodic driving force. The response of the damped system has two components

$$\eta(t) = p\sin(vt + \phi)\mathrm{e}^{-\mu t} + q\sin(\omega t + \theta) \ ,$$

for suitable parameters p, q, ϕ and θ. The first is a transient oscillation with the eigenfrequency v of the system. The second is an undamped oscillation with the frequency ω of the driving force which eventually dominates the response of the system. For a digital simulation of this behavior, we transform the combined system of differential equations, putting $\rho = \sqrt{\mu^2 + v^2}$

$$x = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad y = \begin{bmatrix} \frac{1}{\omega}\dot{\psi} \\ \psi \\ \frac{1}{\rho}\dot{\eta} \\ \eta \end{bmatrix}, \quad y(0) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix},$$

$$A = \begin{bmatrix} 0 & -\omega & 0 & 0 \\ \omega & 0 & 0 & 0 \\ 0 & 1 & -2\mu & -\rho \\ 0 & 0 & \rho & 0 \end{bmatrix} .$$

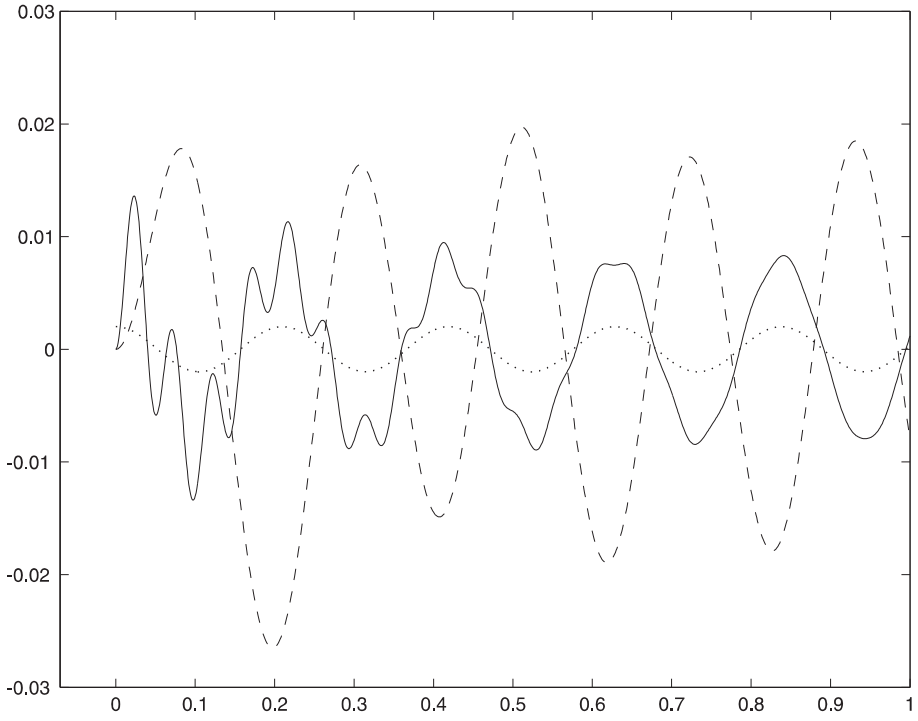

Fig. 7. Damped driven oscillations. Exact digital simulation of two damped oscillators responding to harmonic input. In one case, the eigenfrequency was higher ($\nu = 130$, —); in the other case it was lower ($\nu = 13$, ---) than the frequency of the driving input ($\omega = 30$, $\cdots$). The damping of both oscillators was $\mu = 4$. The displayed amplitude of the driving cosine is reduced by a factor of 500. The step size of both iterations was $\Delta = 0.001$; the graphs shown are linear interpolations of the actual data points

Due to their rather complex dependence on the parameters of the system, neither the explicit form of the parameters p, q, ϕ, θ nor the analytic form of the matrix exponential are given here. The exact digital simulation of the system on the basis of the numerical time-evolution operator, however, is as straightforward as for the previous examples. Figure 7 gives an illustration.

4 Approximate numerical integration

In this section, we compare Exact Integration with different approximate methods commonly used in the context of neuronal network modeling. For time-invariant linear systems and for a fixed step size of integration, all approximate methods considered here reduce to matrix iterations which approximate the exact propagator and which have comparable computational costs. Definitions of the alternative integration methods which are considered here, including a short discussion of their properties with respect to accuracy and stability, are given in Appendix C. In particular for moderate-sized integration steps, before signal aliasing due to undersampling starts to become a major cause of malfunction, Exact Integration proves to be more reliable than all other methods tested. This holds for a continuous test system (subthreshold integration in neurons) and an extended non-linear system involving abrupt resets in one variable (integrate-and-fire neuron).

4.1 The test system

As a continuous test system, we chose the passive response of a leaky integrator neuron model to alpha-shaped post-synaptic currents. In order to operate in a realistic parameter regime, we use the following physical constants throughout this section, unless otherwise stated

$$\tau_\alpha = 0.3 \text{ ms}, \quad \tau_\mathrm{m} = 10 \text{ ms},$$
$$\alpha_\mathrm{max} = 50 \text{ pA}, \quad C = 250 \text{ pF} \ .$$

The equations have been introduced in Sect. 3.2.2

$$x = \begin{bmatrix} \xi \\ 0 \\ 0 \end{bmatrix}, \quad y = \begin{bmatrix} \frac{1}{\tau_\alpha}\psi + \dot{\psi} \\ \psi \\ \eta \end{bmatrix}, \quad y(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$$
$$A = \begin{bmatrix} -\frac{1}{\tau_\alpha} & 0 & 0 \\ 1 & -\frac{1}{\tau_\alpha} & 0 \\ 0 & 1 & -\frac{1}{\tau_\mathrm{m}} \end{bmatrix} . \tag{6}$$

At time 0 we present a delta-pulse input $\xi(t) = \beta\delta(t)$, where $\beta = \alpha_\mathrm{max} \mathrm{e} (\tau_\alpha C)^{-1}$ provides the scaling to physical units. Equivalently, we could have set the initial conditions $y(0)$ for the iteration to the value, which is now enforced by the input. The function $\psi(t)$ describes the post-synaptic current, which reaches its peak value α_max at time τ_α. The function

$$\eta(t) = \beta\left(\frac{\mathrm{e}^{-t/\tau_\mathrm{m}} - \mathrm{e}^{-t/\tau_\alpha}}{(1/\tau_\alpha - 1/\tau_\mathrm{m})^2} - \frac{t\mathrm{e}^{-t/\tau_\alpha}}{1/\tau_\alpha - 1/\tau_\mathrm{m}}\right)$$

represents the membrane response, the post-synaptic potential.

In most models of spiking neurons the membrane potential is the critical variable for spike generation. In addition, it is the variable where all time scales of the

system combine their effect. The system (6) with the parameters chosen is, in fact, a moderately stiff system (Mascagni 1989; Press et al. 1992), where the range of characteristic time constants spans more than one order of magnitude. This poses a problem to some methods of approximate integration, which are limited in step size by the smallest time constant, even if one is interested in the system behavior on a larger time scale.

For this particular system, we quantitatively assessed the advantage of using exact integration and compared the applicability and accuracy of different approximate integration methods. To this end, we analyzed the deviation of the approximate solutions from the exact solution, in several respects. The quantitative measures employed for that purpose are defined in Appendix D. As a rule, the errors in the tail of the test function are small as compared to the errors in the peak region. To exclude a systematic bias in our results due to this fact, we also evaluated our test system under shot noise conditions, similar to what is described in Sect. 3.2.2. The system was supplied with input, which was close to a balance of excitation and inhibition (van Vreeswijk and Sompolinsky 1996). The pulse train input was restricted to a 2 ms grid to ensure comparable results for different step sizes. We verified that the results obtained for a single post-synaptic potential also hold for shot noise input.

4.2 Local accuracy and stability

As a measure of the local accuracy of an integration method, we consider the root-mean-square (RMS) of the point-wise deviation of the approximate solution from the analytical solution on the grid, within the first 120 ms. This corresponds to the d_2 measure as described in Appendix D.1; the d_1 and d_∞ measures, however, give essentially the same results. Figure 8 shows the results for a number of selected integration methods. As expected, the total point-wise error of the various approximations increases with the integration step size.

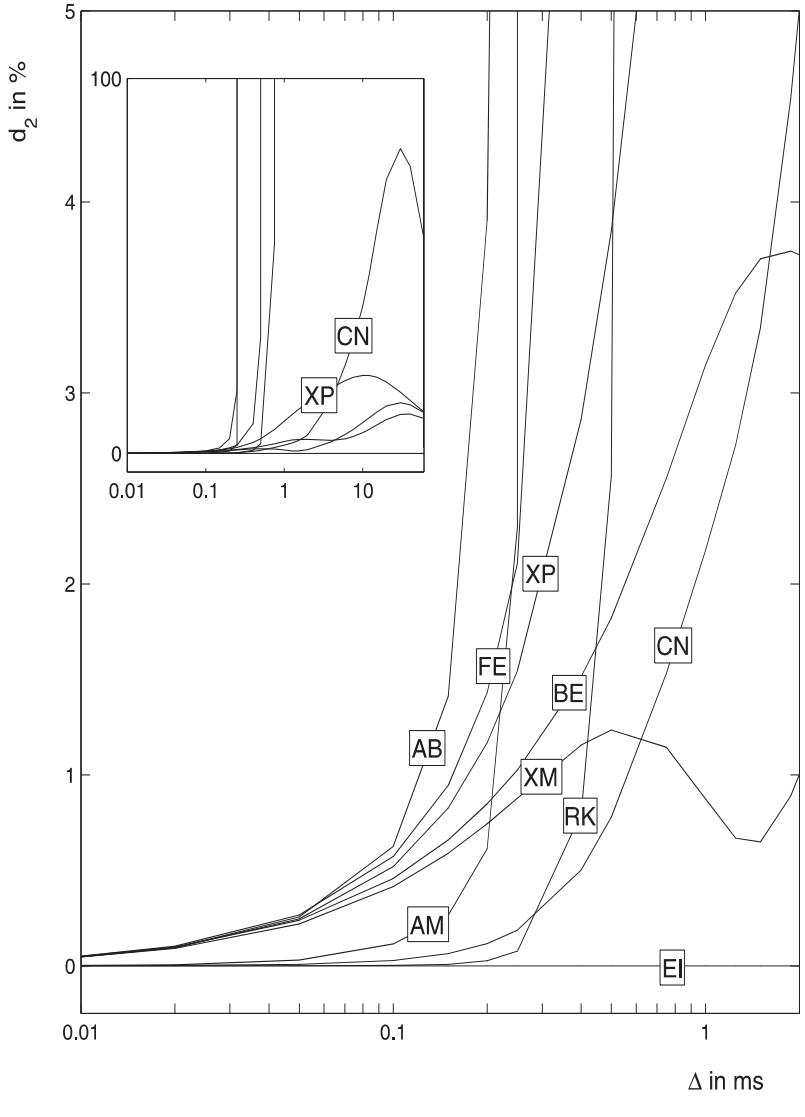

Fig. 8. Local error of various integration methods as a function of the step size (log-scaled *abscissa*). The error is defined as the RMS of the point-wise deviation of the approximation from the analytical solution on the grid, expressed in percentage of the amplitude of the test function (see Appendix. D.1). The test function describing a post-synaptic potential is generated by a system of three linear differential equations with constant coefficients. The step size Δ of the integration is varied from 0.01 ms to 2 ms; the error is evaluated over the first 120 ms. EI denotes the (zero) error of the Exact Integration scheme. Explicit approximate integration methods are Forward-Euler (FE), second-order Adams-Bashforth (AB) with start values $y(-\Delta) = 0$ and $y(0)$, fourth-order Runge-Kutta (RK), and Exponential Integration (XP). Implicit methods are Backward-Euler (BE) and Crank-Nicholson (CN). AM is the result of the second-order Adams-Bashforth integrator, where in contrast to AB start values $y(0)$ and the exact value for $y(\Delta)$ are used. XM is the result for the Exponential Integration where the approximate solution is shifted by $-\Delta$ relative to the exact solution. The *inset* shows the same data with a larger *ordinate* range and an *abscissa* range of 0.01–60 ms. With increasing step size AB ($\Delta = \tau_\alpha$), FE ($\Delta = 2\tau_\alpha$), and RK ($\Delta \approx \frac{8}{3}\tau_\alpha$) become unstable and yield exceedingly large errors

The explicit integration methods of Adams-Bashforth, Forward-Euler and Runge-Kutta lose stability when the step size exceeds the smallest time constant by a method-dependent factor (Wilson and Bower 1989; Hines and Carnevale 1995). The critical value for Δ can be calculated with the methods presented in Appendix C. It turns out that stability depends only on the product of the eigenvalues of the coefficient matrix A and the step size Δ. In our case, where the eigenvalues are all real and negative ($-\frac{1}{\tau_\alpha}$, $-\frac{1}{\tau_\alpha}$, $-\frac{1}{\tau_m}$), the bound for stability scales with the smallest time constant τ_α (see Fig. 16).

In contrast, the implicit integration methods of Backward-Euler and Crank-Nicholson, as well as Exponential Integration and Exact Integration, are absolutely stable (see Appendix C). The average local errors of the implicit methods and Exponential Integration remain bounded and become small again for large step sizes, because then the peak region no longer contributes. Below $\Delta = 0.6\,\text{ms}$, the Crank-Nicholson method has the smallest error of the stable approximate integration methods. At $\Delta = 2\tau_\alpha$, however, the first eigenvalue of its propagator becomes negative and the iteration starts to oscillate. For larger step sizes, the Backward-Euler and the modified Exponential Integration method have smaller errors.

The multi-step Adams-Bashforth method shows the largest error and is the first to become unstable, if the specification of initial values is not done with great care. The method has a fatal tendency to accumulate errors (Press et al. 1992; Bower and Beeman 1997). The error remains small, however, if in addition to the initial condition the exact value for the second step is used. In its stable regime the modified Adams-Bashforth method is the second-best explicit method after that of Runge-Kutta. In the context of neuronal modeling, two-step initialization can be implemented with some additional bookkeeping.

For the test system, Exponential Integration, which is an explicit method, is also stable. Its propagator has, in fact, the same eigenvalues as the exact matrix exponential (see Appendix C.6). For large step sizes, however, it produces considerable errors. This is essentially due to a time delay of one time step relative to the exact solution. If the approximate solution is artificially aligned by a shift backwards in time, the error is reduced (compare Fig. 10). For large step sizes, it becomes even smaller than the error of the implicit methods.

Originally introduced by MacGregor (1987), exponential integration is now widely used in neuronal modeling (Wilson and Bower 1989; Bower and Beeman 1997). This is due to its stability and relative accuracy in typical applications. If, however, Exponential Integration is applied to a stiff system like the one discussed here, care must be taken to appropriately set the initial condition (see Appendix C.6). The non-monotonic behavior of the errors produced by the modified Exponential Integration around $\Delta = 1\,\text{ms}$ is due to a match of shapes (see Fig. 8). In the first iteration step after the peak of the test function, the approximate solution comes to lie close to the exact solution such that all successive steps also generate a small error.

Figure 9 shows the first time steps in the approximate simulation of the test system for different integration methods. The onset delay observed for Exponential Integration as well as for the Forward-Euler and Adams-Bashforth methods (see Fig. 9) is caused by the particular structure of the corresponding propagator matrices for a cascaded system of n differential equations. The ith component of the state vector can influence the jth component only after $j - i$ iterations, and not already after a single step, which is the theoretical minimum for a general matrix. If, for instance, one is only interested in the long-term behavior of a single neuron, this effect could be ignored. In a network of neurons interacting by spikes, the effect can be compensated for, if the synaptic delays are at least $j - i$ time steps long.

4.3 Global accuracy and signal aliasing

In an application, the step size for a numerical simulation may be limited by stability and accuracy of the iteration on the grid. Such problems are clearly reflected by point-wise measures. However, there may also be external criteria such as the detectability of threshold crossings, the accuracy of spectral properties, or the visual appearance of the graph. In the context of threshold models, for instance, we are particularly interested in a faithful approximation of peaks in transient signals. Figure 10 shows the peak region of the test function and its approximations. In contrast, Fig. 11 illustrates the problem of signal aliasing due to under-sampling, which affects any simulation method, including Exact Integration. Although the values obtained are exact on the grid for arbitrary Δ, the shape of the test function is completely lost for step sizes that are too large.

Figure 12 shows the results from the application of a measure which tries to capture the error in global shape, relative to the exact analytical function (see Appendix D.2 for a definition). Now, the error produced by Exact Integration essentially represents a lower bound, because for approximate methods the errors on the grid also contribute to the deviation in global shape.

4.4 Accuracy of threshold crossings

The simplest model for a spiking neuron involves a threshold on the membrane potential. A crossing of the threshold elicits an action potential and a reset of the membrane potential to its resting value. Depolarizations toward threshold are due to passive integration of inputs from other neurons, as discussed above. Therefore, in the sub-threshold regime, the integrate-and-fire model is linear and time-invariant. Only when spikes are generated is the membrane potential trajectory affected by an additional non-linear mechanism.

To test the performance of numerical simulation methods we chose an integrate-and-fire neuron with the same sub-threshold integration as discussed above, and

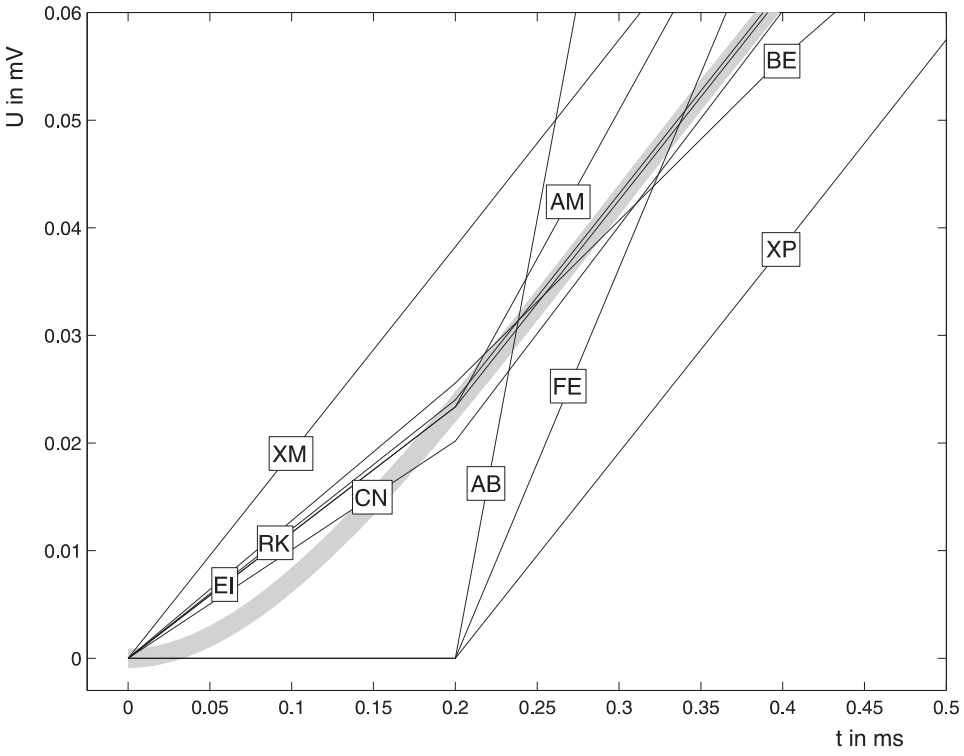

Fig. 9. Onset of the approximate simulation of the test system, a post-synaptic potential (*gray curve*) with parameters given in the text. The different integration methods are labeled as in Fig. 8, we used a step size $\Delta = 0.2$ ms for all methods. The trigger event is at time 0. The initial value for the variable shown (membrane potential) is 0. Therefore, a deviation from 0 can only occur after one computation time step. Forward-Euler, the Exponential Integration and the Adams-Bashforth method show values different from 0 only after one additional time step. This is due to the cascaded structure of the coefficient matrix for the corresponding propagators. In the case of Exponential Integration, the error can be reduced if the approximate solution is shifted to the left by one time step (XM). The second-order Adams-Bashforth method performs an iteration on the basis of the two preceding states. In order to reduce the error, it is therefore natural to start the iteration with the initial value and the exact value after the first step (AM)

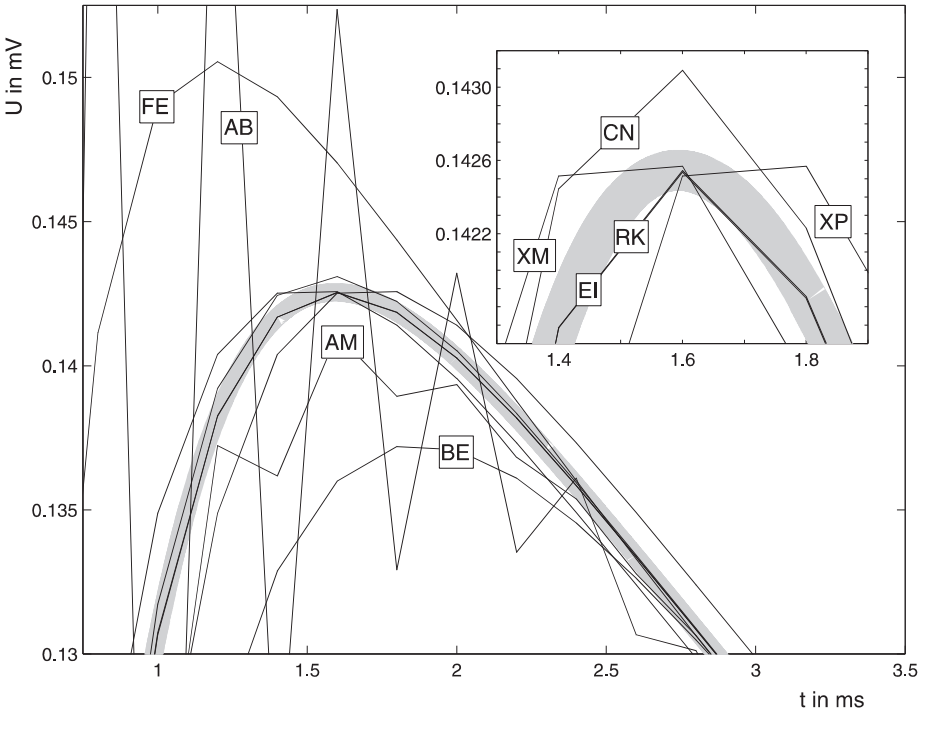

Fig. 10. Peak region of the test function (*gray curve*) simulated at a step size $\Delta = 0.2$ ms. Parameters and labeling are as in Fig. 8. The *inset* shows an enlarged view of the peak region; the curves already labeled in the main figure are omitted. At the chosen step size, Adams-Bashforth integration shows clear oscillatory behavior. This is due to the fact that half of the eigenvalues of its propagator are negative in this regime, and their relative magnitudes increase with Δ (see Appendix C.4). Nevertheless, the variant of the method which is supplied with exact values for the first two steps still captures the peak region. Forward-Euler and Backward-Euler integration both have an error of about 5% peak height. The Crank-Nicholson method has an error of less than 1% peak height, all other methods less than 0.1%. The curve of the Δ-shifted Exponential Integration is closer to the exact solution than the original one everywhere. This reduces both the average local error (Fig. 8) and the global error (Fig. 12) of the method

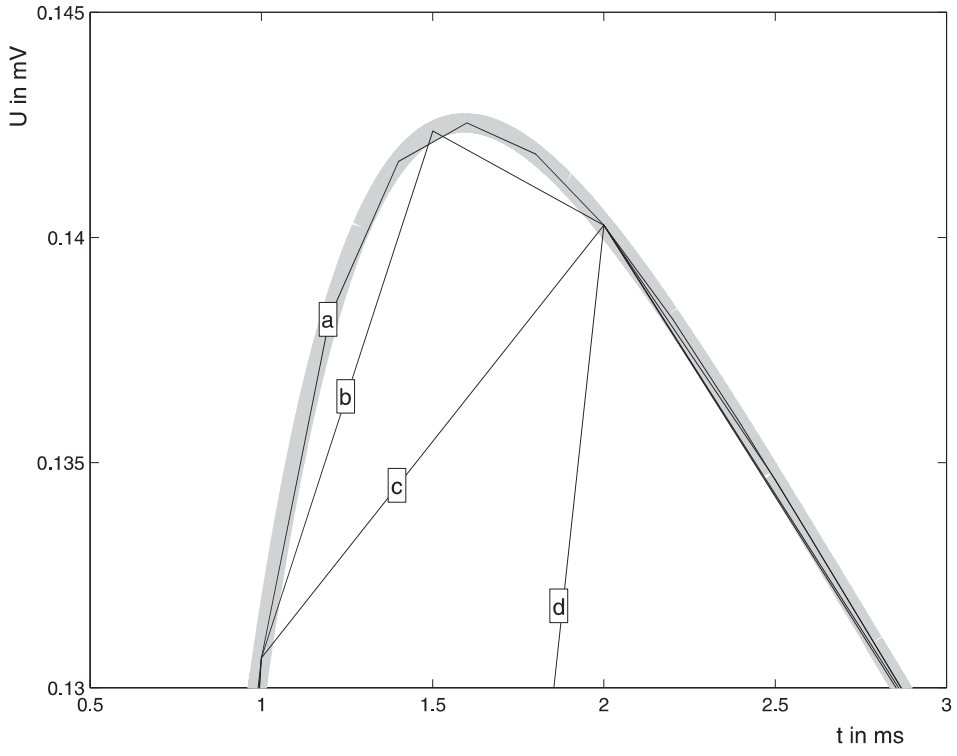

Fig. 11. Under-sampling and signal aliasing in digital simulation. The *solid curves* show the linearly interpolated results of the exact integration method compared to the analytical test function (*gray curve*), for different step sizes (*solid curves*, $\Delta = 0.2$ ms a, $\Delta = 0.5$ ms b, $\Delta = 1$ ms c, $\Delta = 2$ ms d). Parameters are as in Fig. 8. The samples are exact on the grid for arbitrarily large step sizes Δ. Although the solution is exact on the grid, the shape of the test function is completely lost if the step size is too large

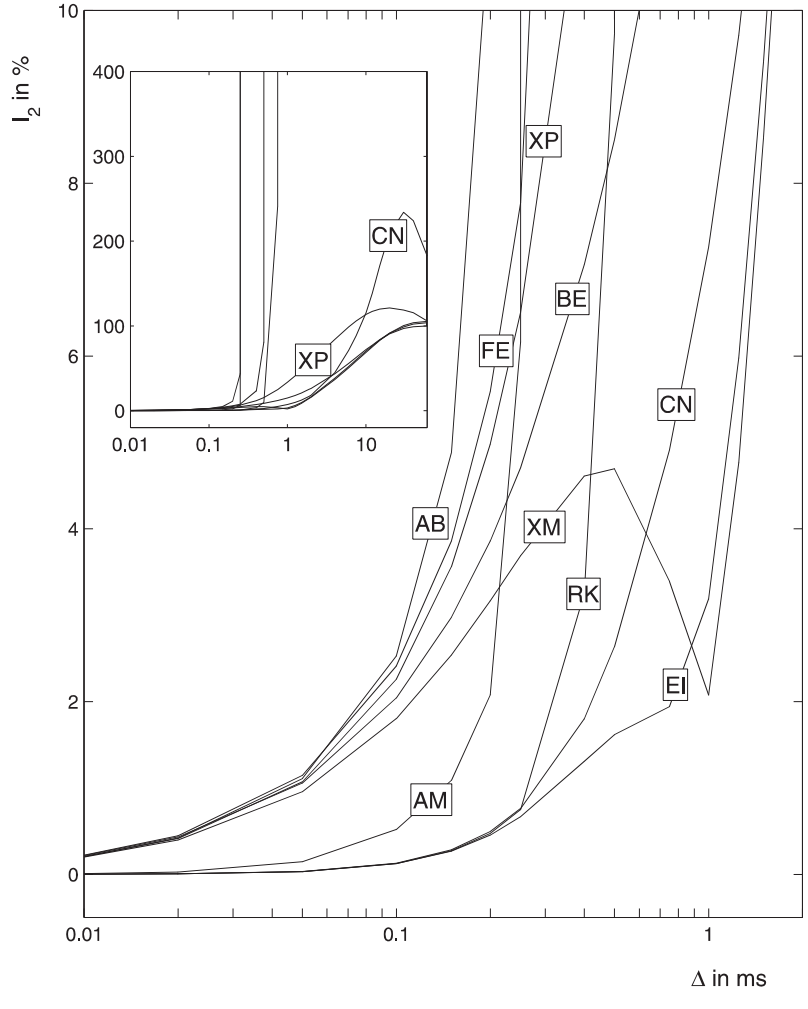

Fig. 12. Global error in the digital simulation of a test function, depending on the integration step size (log-scaled *abscissa*). The error is defined as the relative integrated square error in shape with respect to the analytical solution (see Appendix D.2). Parameters and integration methods are the same as in Fig. 8. For the implicit methods and Exponential Integration, the error converges to 100% for large step sizes because the approximate solutions are then close to the null function and the full shape of the test function contributes to the error. For larger step sizes, under-sampling causes an increasing global error also for Exact Integration (see Fig. 11). For step sizes above 1 ms, Backward-Euler and Exponential Integration come close to Exact Integration. However, for step sizes below 1 ms, Crank-Nicholson has a smaller error than Backward-Euler and the Exponential Integration techniques. The step-size-dependent hierarchy of the integration methods with respect to the global error is essentially the same as for the point-wise error measures (compare Fig. 8). The l_1 and l_∞ measures give essentially the same results

a voltage threshold for spike generation at $U_\theta = 15\,\mathrm{mV}$ above its resting potential. Each time the neuron generated a spike its membrane potential was reset to rest without affecting the input current. The input comprised an almost balanced combination of excitatory and inhibitory events, which had a different sign but otherwise equal amplitudes and time constants. The average spiking frequency of the simulated neuron was 10–15 spikes/s. For comparison, we examined an even more stiff system with all parameters equal, except that the synaptic currents were three times faster.

The simulation of spike trains generated by an integrate-and-fire neuron may be affected by three types of errors (Fig. 13). Firstly, provided there is a fixed step size for the simulation, even Exact Integration cannot be exact for such a system with respect to the precise time of threshold crossing; the occurrence of spikes is constrained to the simulation grid. This gives a clear lower bound for the accuracy of spike trains obtained by this simulation method. This type of error, however, is completely predictable and under control. Secondly, a more serious failure of numerical simulation is the possibility that a threshold crossing is completely overlooked. This can happen for any integration method including Exact Integration, if only the very tip of a peak is super-threshold, but the tip falls in-between two sample points. Clearly, this type of error occurs more frequently for larger step sizes in systems with fast components. Thirdly, approximate integration methods can produce false excess spikes due to integration errors of the continuous threshold variable. Since exact integration does not have this kind of problem, it will systematically indicate the exact number or fewer spikes than the exact system.

We attempted to systematically account for all three types of failures, and to evaluate their combined effect on the quality of numerical simulations. To this end, we first computed cross-correlations of the approximate spike trains with a reference spike train simulated with Exact Integration on a super-fine grid and evaluated the defect, the width and the shift of the respective center bin (Fig. 14). The combined effect of the various types of failures and inaccuracies, however, can be displayed more compactly by employing a distance measure for pulse trains, similar to that employed for continuous signals. A detailed description of the method is given in Appendix D.3; the results of this analysis are displayed in Fig. 15.

In conclusion, the only approximate integration methods which are competitive with Exact Integration

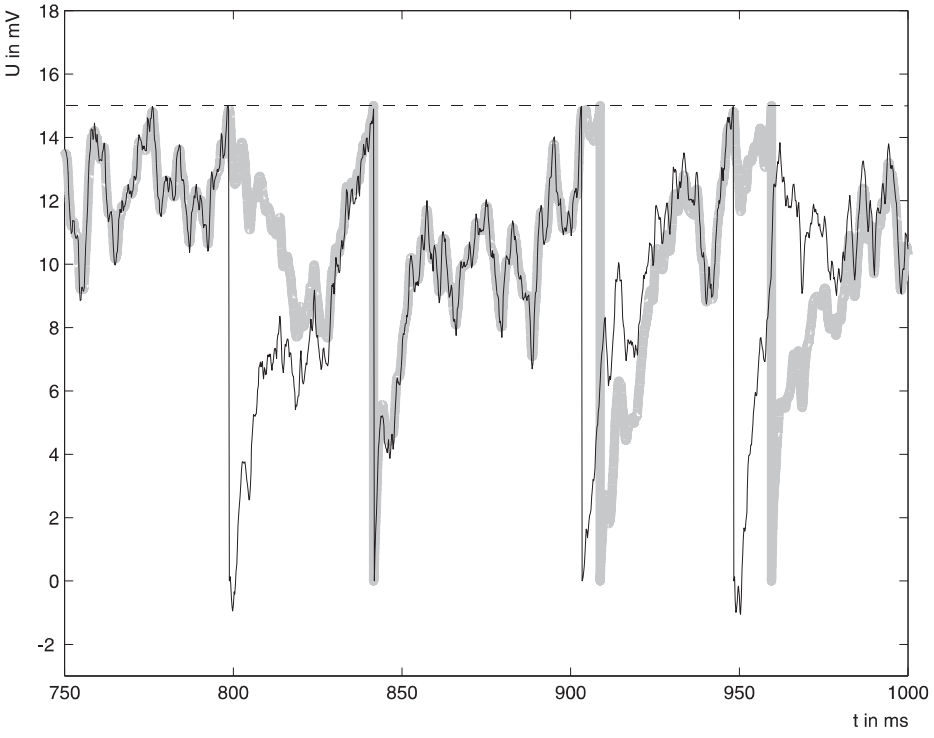

Fig. 13. Simulation of an integrate-and-fire neuron with shot noise input. The reference signal (*gray*) is simulated with Exact Integration on a super-fine grid of step size 0.001 ms. Superimposed is a spike train obtained with a particularly bad approximate integration method (Forward-Euler) at a step size of 0.2 ms (black), a combination not uncommon in applications. Identical input, which was constrained to the coarser grid, was used in both simulations. The first spike in the approximate response signal is falsely generated due to an integration error of the threshold variable. The second spike is accurate except that it is forced to the much coarser grid of the simulation (not visible on the time scale of the figure). The following two spikes come way too early, again due to integration errors. Beyond integration errors, under-sampling may lead to a systematic skipping of threshold crossings. In principle, this problem also affects Exact Integration, in particular at larger integration step sizes. In all cases of failure, however, a complete recovery toward the correct trajectory is the rule for the model under consideration, since the sub-threshold system (like any stable time-invariant linear system) has the property to forget its own previous history

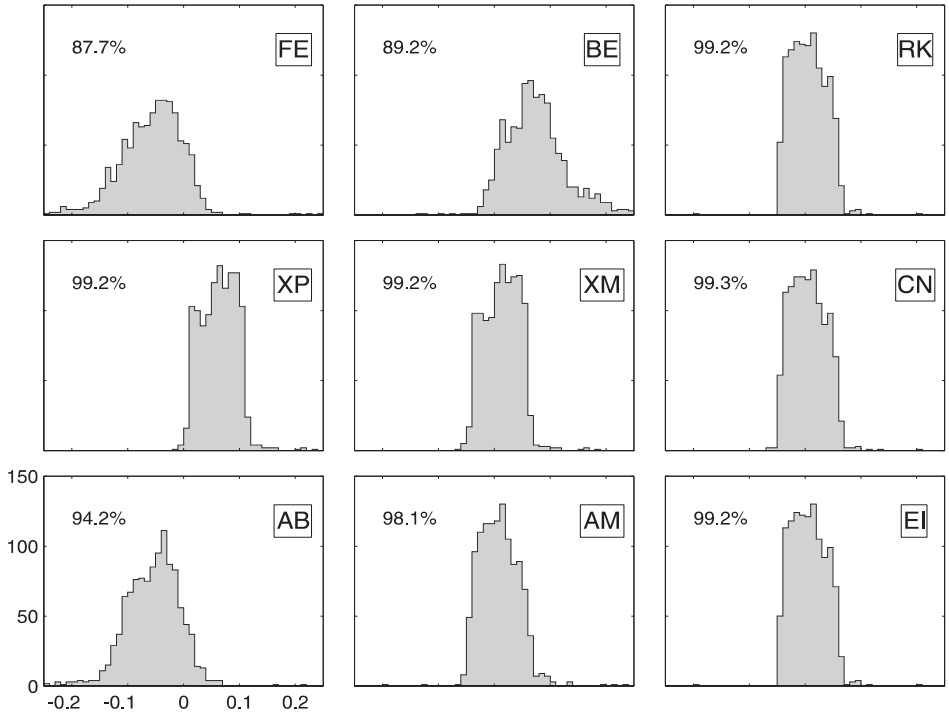

Fig. 14. Accuracy of approximate spike trains. Cross-correlation of a reference spike train (1186 spikes) obtained with Exact Integration on a super-fine grid of step size 0.001 ms and spike trains obtained with various approximate methods at a step size of 0.1 ms. Labels are as in the previous figure, unit of time is 1 ms, histogram shows raw counts, bin width is 0.01 ms. Note that for all integration methods (except for the modified Exponential Integration, which is already corrected) the spikes are put half a time step earlier than the detection of the super-threshold potential. This is done because the precise time of threshold crossing must take place somewhere in the preceding interval, provided there are no integration errors. Most histograms have a rectangular shape, which indicates that all points in the interval qualify for a threshold crossing with equal probability. Therefore, on average, the point of threshold crossing is in the center of the preceding interval. Information about a tendency to indicate a threshold crossing too early (Forward-Euler) or too late (Backward-Euler) can be taken from the location of the center peak relative to the origin. Finally, a systematic skipping or false detection of extra spikes expresses itself in the weight of the center peak, which is indicated here as a percentage of matches within the time window shown

are Crank-Nicholson and, for small enough step sizes, Runge-Kutta. All other methods must be used with care; some of them, most notably Exponential Integration, can be modified for better performance. The first-order methods Forward-Euler and Backward-Euler both suffer from bad accuracy. The method of Adams-Bashforth can be good; its correct implementation, however, is quite difficult and computationally expensive. Especially for moderately sized steps, there is a clear advantage in using Exact Integration with respect to the precision and reliability of the resulting spike train. This advantage may be critical for computer simulations of effects regarding precise synchronization of action potentials.

5 Discussion

5.1 Why is Exact Integration generally useful?

The greatest appeal of the proposed method for digital simulation of linear time-invariant systems lies in the fact that the result is always exact on the grid. In theory, deviations from the analytical solution do not occur, independently of the step size of the iteration. This enables the use of large steps, where high temporal resolution is not of interest, and where the dimensionality of the system would otherwise cause high costs of computation.

An estimate of the computational efficacies of the various integration methods at fixed step size shows that, in general, the exact method is computationally no more expensive than any of the approximate methods. In fact, a large portion of the computational load is concentrated on the determination of the matrix exponential. This, however, has to be done only once, effective and stable algorithms for this purpose are available and have been published. For time-critical simulations, one may want to exploit the zeros of the propagator matrix and generate optimized codes for the actual iteration. It should be noted, however, that a sparsely coupled system does not necessarily lead to a sparse propagator. In contrast, triangular systems always have a triangular propagator. If such is the case, updating of the state vector can be done "in place", which may be advantageous for large systems.

Stiff systems, in general, do not pose a problem to the Exact Integration method, provided that an accurate matrix exponential is available. Systems with eigenvalues of very different magnitude can be exactly integrated with the proposed method, at arbitrarily large step sizes. The post-synaptic potential discussed in Sect. 4 is an

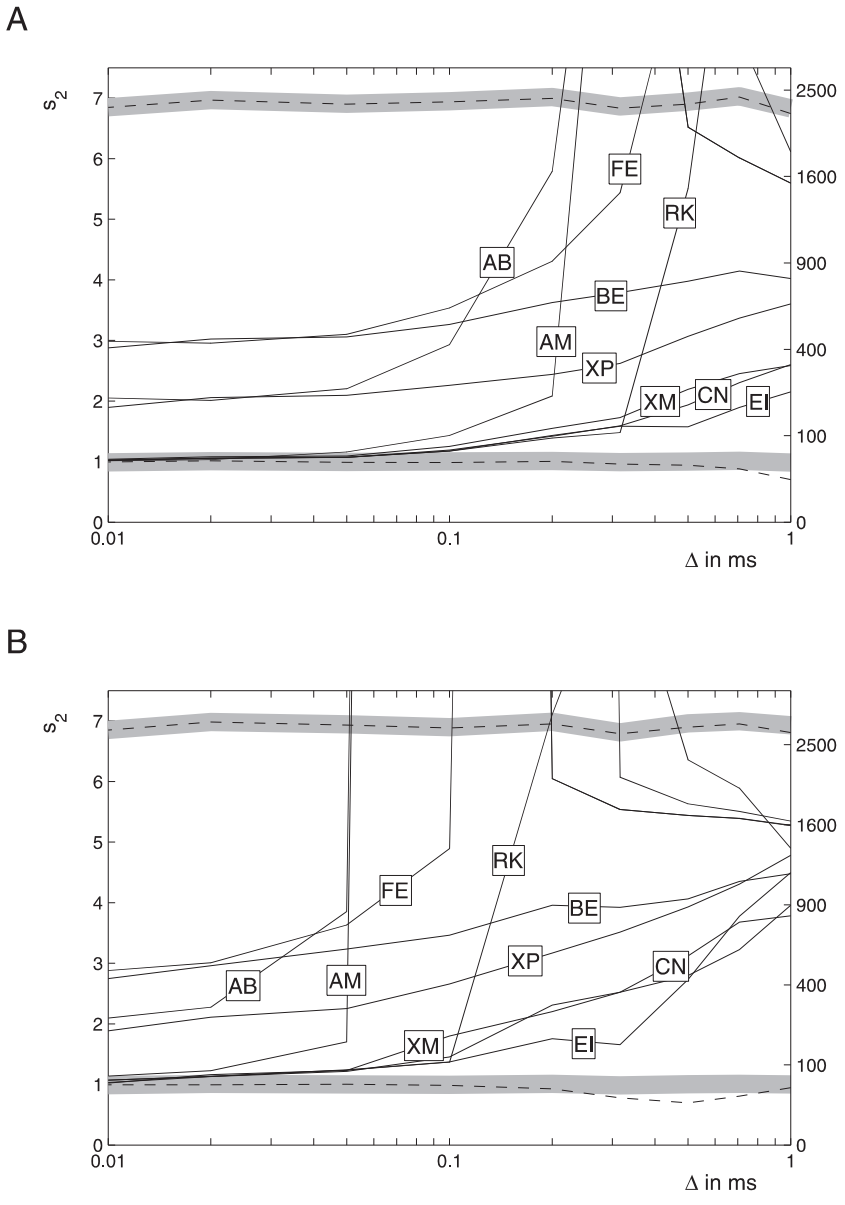

Fig. 15A,B. Error in temporal precision of threshold crossings in simulations of an integrate-and-fire neuron with shot noise input, as a function of the integration step size (log-scaled *abscissa*). Labeling of integration methods is as in the previous figures. We computed the distances between the approximate spike trains and a reference spike train obtained by simulation on a super-fine grid of step size 0.001 ms. Gaussian convolution kernels of standard width equal to the step size of the approximate simulation were used; see Appendix D.3 for a more detailed description of the method. **A** Errors in simulations of a neuron with synaptic currents as in Sect. 4 ($\tau_\alpha = 0.3$ ms), **B** A neuron with faster synaptic currents ($\tau_\alpha = 0.1$ ms). The simulations extended over 100 s, each yielding 1000–1500 spikes. The *bottom dashed line* indicates the error of the exact spike train with all its n spikes forced to the grid; the *bottom gray line* is the value $\sqrt{n/24}$ predicted by theory for that situation. The *left scale bar* expresses all errors in terms of this discretization error, assuming the average number of spikes found in the different simulations. Another interesting limit of this error measure is the case where all spikes are dislocated so far that the spike train is orthogonal to the reference spike train. In this case, if the number of spikes in both spike trains is the same, the theoretical value for the error is $\sqrt{2n}$, as indicated by the *top gray line*. The *top dashed line*, in fact, gives the distance of the reference spike train with a spike train having the same number of spikes at randomly chosen times in the simulation interval. The *right scale* indicates the total number of complete misses or excess spikes in either spike train under the assumption that all other spikes match perfectly. Note that by adapting the metric to the step size of simulation, we normalize errors to the minimal error enforced by the grid. For each fixed metric, errors increase with increasing step size, as is the case for the l_2 error measure (see Fig. 12). The overall performance of the various integration methods for digital simulation of a spiking neuron is comparable to their performance for the previously discussed continuous test system. Especially for moderately sized steps, exact integration has a clear advantage with respect to the precision and reliability of the result

example of a moderately stiff system, which frequently occurs in neuronal modeling.

Possible deviations from the exact solution on the grid, if they occur at all, could have their cause in an accumulation of errors due to finite precision computer arithmetic. Integration of the equations in a coordinate system, where the propagator is diagonal or has Jordan normal form, may then help to reduce the number of multiplications and, therefore, the chance of error accumulation. Another potential source of numerical problems is the computation of the correct matrix exponential itself. Much depends on the choice of a robust algorithm (Moler and van Loan 1978). Relying on the implementations of Matlab and Mathematica, however, we never observed such problems in our simulations (see Appendix A for details). In any case, however, signal aliasing due to under-sampling must be considered a potential source of misinterpretation even of correctly sampled signals.

5.2 How good are approximate methods?

As a rule of thumb, approximate methods dramatically lose accuracy (see Figs. 8 and 12) or even become unstable (see Fig. 16), if they are operated at time steps significantly exceeding the smallest time constant of the system. The use of elaborate higher-order methods can push the borders, but does not present a general cure. Therefore, depending on the system and on the scientific question one has in mind, great care is needed to assess the accuracy of the results of numerical simulation. This is particularly true for compound non-linear systems with thresholds on linear variables (integrate-and-fire neurons) and for large distributed systems (large recurrent neuronal networks), if one is interested in relative timing and synchronization effects.

For linear time-invariant initial value problems and input-output systems, Exact Integration is clearly the method of choice; it is always exact on the grid, and it is computationally almost as effective as any other method. No integration method, however, is immune against the surprising effects of under-sampling, and considerable reconstruction errors must always be taken into account. Therefore, if small step sizes are acceptable, the advantage of exact integration over approximate methods can be small. One may then even come to the conclusion that a carefully chosen multi-purpose solver, which would also be usable for non-linear systems, is good enough.

Depending on the system, adaptive control of the step size may be a desirable feature of any solver. Exact Integration can be operated at different step sizes in sequence, without losing its exactness. A naive criterion for switching to a smaller step size might be the occurrence of large slopes in the trajectories. The price of doing so is that the corresponding matrix exponentials must be made available, for example in the form of a lookup table of precomputed matrices for a fixed set of step sizes. If an analytic expression of manageable complexity for the exact propagator exists, as is the case for most of the examples presented in this paper, more sophisticated step-size control mechanisms can be implemented.

5.3 Exact Integration for neuronal modeling?

The standard kinetic model for the transitions between 'open' and 'closed' states of ion channels employs continuous-time finite-state-space Markov processes. The transient dynamics of the distribution of states for a (large) ensemble of channels under voltage-clamp conditions is governed by a linear time-invariant differential equation of the type discussed in this paper. The model can, therefore, be simulated by the proposed method. Some properties of the matrix exponential have been employed for the general analysis of the system (Colquhoun and Hawkes 1977; 1981; 1982; 1995a,b). Exact digital simulation of multi-state channel kinetics and the corresponding voltage-clamp experiments may add a useful analysis tool and present a worthwhile application of the proposed method.

The simplest model for the sub-threshold behavior of a neuron is that of a leaky integrator. It can be considered as a time-invariant linear system which turns current input into a voltage response. Similarly, multi-compartment models for neurons with spatially extended dendrites can also be viewed as time-invariant linear systems as long as inputs are given as currents and not as conductance changes. The method of Exact Integration can, therefore, be applied. If, however, the input to the dendrite is given as a time-dependent change of synaptic conductances, one ends up with a more general type of linear system, which has to be treated with more general numerical methods.

The alpha-function is a frequently used model for the time course of either the ionic current or of the corresponding conductance during a synaptic event (Wilson and Bower 1989; Bernard et al. 1994). In either case, the proposed methods can be used to generate the time course of a synaptic event "online" avoiding the need for large lookup tables. Moreover, the combined effect of all synapses in a single compartment is most easily obtained by integrating the lumped and weighted pulse train of all pre-synaptic neurons. In the case of currents, the membrane response can be incorporated into exact integration. In the case of conductances, the membrane potential must be obtained by a different method.

Models for biological neuronal networks with spiking neurons almost exclusively use delta-functions to represent action potentials. In many of these models, the generation of a spike is the result of a threshold crossing for some internal state variable, such as the membrane potential, the dynamics of which is governed by a linear differential equation. Since a spike in one such neuron typically causes a change of state in many other neurons of the network, the use of adaptive step-size solvers for the dynamic equations of the internal states would be highly ineffective.

Restricting threshold crossings to a fine temporal grid, an approximate simulation of the network

dynamics is obtained by extending the Exact Integration scheme by a threshold operation for each neuron. Post-spike effects, such as a reset of the membrane potential (Stein 1965; Knight 1972; Troyer and Miller 1997), can be modeled as an alteration of the state of the neuron itself. The onset latency of such effects is constrained to non-negative integer multiples of the step size. A minimal delay of one time step for synaptic interactions enforces causality of the system and is required by a parallel update scheme. In such a model, each neuron receives pulse train input and is itself the generator of a pulse train, and the network contributes to its own input. Since pulse train inputs play a special role in combination with the proposed Exact Integration method, its use is natural for such integrate-and-fire models.

The restriction that spikes can only be reported on the grid may be partially overcome by additional measures (Hansel et al. 1998). What remains is the problem that spikes could be completely overlooked, if the integration step size is too large. The advantage of using Exact Integration for the sub-threshold variables results, nevertheless, in an improved accuracy of spike trains, in particular for moderate step sizes. This improvement is of particular practical importance for the examination of precise synchronization of action potentials across neurons.

In conclusion, most neuronal models comprise submodules which can be considered time-invariant linear systems in their own respect. In many cases, it is advantageous to treat these subsystems with the method of Exact Integration. A slight extension of the method leads to a very effective scheme for the accurate simulation of networks of integrate-and-fire neurons. Compared to conventional approximate integration methods, Exact Integration generally yields reliable simulations and more accurate results even for badly conditioned systems.

Appendix A: The matrix exponential

Let $f(x)$ be a holomorphic function in an open set of the complex plane, then a power series expansion of f can be used to define $f(B)$, for any square matrix B. The eigenvalues of the matrix $f(B)$ are of the form $f(\lambda)$, provided that the series converges for all eigenvalues λ of B (Rudin 1991). In particular, the exponential series

$$\mathrm{e}^{B} = \sum_{i=0}^{\infty} \frac{B^i}{i!} = 1 + B + \frac{B^2}{2} + \frac{B^3}{6} + \cdots$$

converges for any square matrix B (Hirsch and Smale 1974; Arnol'd 1992; Bronstein et al. 1996). A number of properties known from the exponential function of scalars are retained for matrices. The most important among these are

$$\mathrm{e}^{0} = 1 \quad \text{and} \quad \mathrm{e}^{B+C} = \mathrm{e}^{B}\mathrm{e}^{C}, \quad \text{if } BC = CB \ .$$

This means in particular that

$$\mathrm{e}^{A(s+t)} = \mathrm{e}^{As}\mathrm{e}^{At} \ ,$$

which is important for the application to linear time-invariant differential equations.

The analytic calculation of the matrix e^{At} for a symbolic matrix of coefficients A can be performed either by hand (Colquhoun and Hawkes 1995b; Hirsch and Smale 1974; Leonard 1996) or with the help of a software package such as Mathematica (Wolfram 1996) or Maple (Heal et al. 1997). For the numerical computation of e^{B} for a fixed matrix B of real or complex coefficients, one can make use of appropriate routines (Moler and van Loan 1998; Golub and van Loan 1996; Druskin et al. 1998; Kenny and Laub 1998) and their ready-to-use implementations in numerical mathematics packages such as Matlab (MathWorks Inc. 1998).

Appendix B: The delta-function

The Dirac delta-function $\delta(t)$ is not a function in the classical sense. Its most important property, however, can be described by its behavior as a kernel under the integral (Arnol'd 1992; Bronstein et al. 1996)

$$\int_{-\infty}^{+\infty} \phi(t)\delta(s-t)dt = \phi(s) \ ,$$

where $\phi(t)$ is an arbitrary function.

In the context of neuronal models, the concept of an infinitely sharp impulse is often used to describe an action potential. A pulse train, which is the sum of delta-functions appropriately located on the time axis, then describes the general form of the spike response of a neuron. By coincidence, this is also the type of input which is best suited for the Exact Integration technique.

Appendix C: Approximate integration methods

There is a large variety of methods to approximate the solution of general, possibly non-linear initial value problems (Press et al. 1992; Bronstein et al. 1996; Stoer and Bulirsch 1996)

$$\dot{y} = f(t,y) \ . \tag{7}$$

We chose a number of representatives from different classes of integration methods (Wilson and Bower 1989; Hines and Carnevale 1995; Bower and Beeman 1997; Hines and Carnevale 1997), which are frequently used in neuronal modeling, and discuss the degree to which they are useful for the simulation of time-invariant linear systems. We restrict our considerations to stable linear systems, where all eigenvalues of the coefficient matrix have negative real parts. We focus on the accuracy of the approximation and on the stability of the iterative solution, and examine the dependence of both on the step size. For time-invariant linear systems, this can be done in a systematic way.

For all methods under consideration, the approximate integration of a linear system turns out to be equivalent to a linear iteration, similarly to the one given by (5). The propagator matrix, however, deviates from the exact matrix exponential, admitting only approximate results. The stability of an approximate iteration can be assessed in terms of the eigenvalues of its time-evolution operator. The condition for stability is that all eigenvalues of the time-evolution operator have magnitude smaller than unity. This yields explicit conditions for the step size Δ of the iteration, which are indicated in Fig. 16 for various approximate integration methods.

C.1. Forward-Euler

The simplest method suggested by Euler explicitly approximates the derivative by a finite difference quotient. This implies the iteration on the grid

$$y_{k+1} = y_k + \Delta f(t_k, y_k) \ .$$

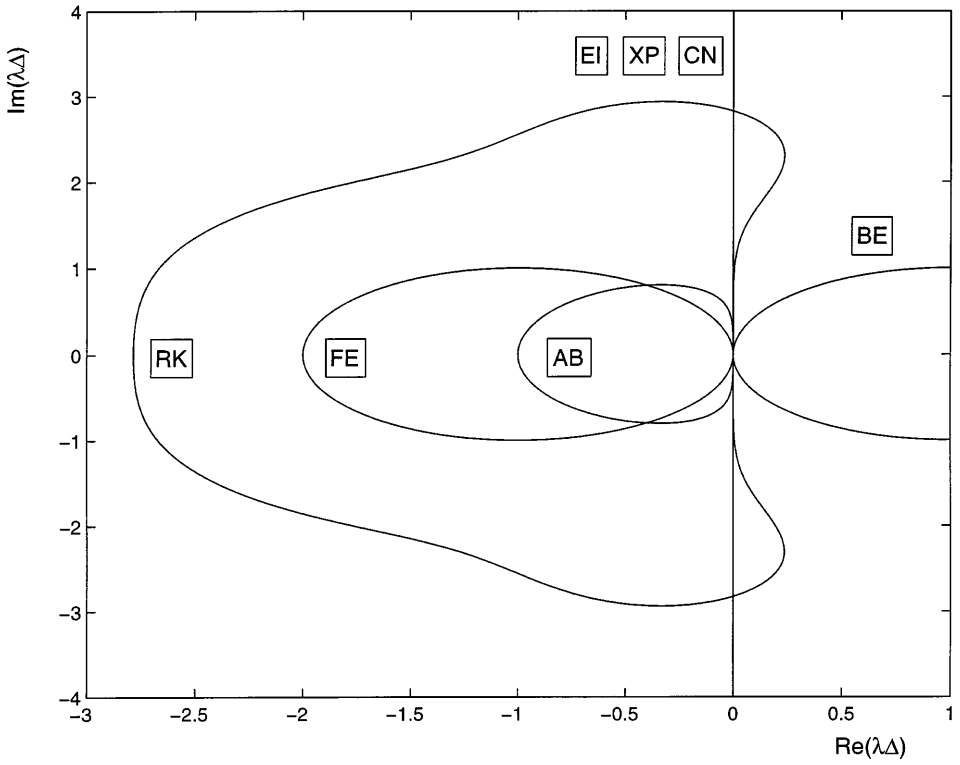

Fig. 16. Stability of various numerical integration methods. The figure shows areas of absolute stability in the complex plane for six approximate integration methods and for the exact method. The condition for stability is that all eigenvalues of the iteration have magnitude smaller than unity. For each method, we show the curve separating stable from unstable values of the product $\lambda\Delta$, where λ is an eigenvalue of the linear system and Δ is the time step of the iteration. The position of the label indicates the area of stability. Only for the Crank-Nicholson method (CN), Exponential Integration (XP), and the exact method (EI) is the stability of the numerical iteration always equivalent to the stability of the linear system itself, which then has all its eigenvalues λ in the left half of the complex plane. The Backward-Euler (BE) and the Runge-Kutta (RK) method may yield a stable iteration even for non-stable systems. For the Adams-Bashforth (AB), the Forward-Euler (FE), and the Runge-Kutta (RK) method, decreasing the step size Δ always has a stabilizing effect, for eigenvalues λ with a negative real part

For a linear time-invariant differential equation. (2), this leads to the linear iteration

$$y_{k+1} = (1 + A\Delta)y_k \ .$$

The time-evolution operator $1 + A\Delta$ of the Forward-Euler method comprises only the first two terms of the power series expansion of the exact matrix exponential $e^{A\Delta}$. The method is indeed known to be neither accurate nor stable. The lack of stability for large step sizes Δ can be understood in terms of the eigenvalues of its propagator. Namely, the corresponding iteration is stable only if all eigenvalues of the matrix $1 + A\Delta$ have magnitude smaller than unity. This yields a condition in terms of the eigenvalues λ of the coefficient matrix A, namely

$$|1 + \lambda\Delta| < 1 \ .$$

C.2 Backward-Euler

The Backward-Euler method evaluates derivatives at the point where the function is to be estimated. This leads to an implicit set of equations

$$y_{k+1} = y_k + \Delta f(t_{k+1}, y_{k+1}) \ ,$$

which, in the case of a linear time-invariant system, yields

$$(1 - A\Delta)y_{k+1} = y_k \ ,$$

which can be solved for y_{k+1} by Gaussian elimination, for example. For stability of the corresponding linear iteration, we must have

$$\left|\frac{1}{1 - \lambda\Delta}\right| < 1$$

for all eigenvalues λ of the coefficient matrix A. If, however, no eigenvalue has a positive real part, this is automatically satisfied for any $\Delta > 0$. This fact makes the Backward-Euler method absolutely stable for any such system. In order to obtain the time-evolution operator for this method, we must solve for y_{k+1}. For small enough Δ, all eigenvalues of the matrix $A\Delta$ have magnitude smaller than unity. In this case, the geometric series expansion for the time-evolution operator converges, and we can write

$$y_{k+1} = \left(\sum_{i=0}^{\infty} (A\Delta)^i\right) y_k \ .$$

Again, this is exact to first order, predicting only moderate accuracy of the integration.

C.3 Crank-Nicholson

A method which is known for its stability under rather general conditions is based on a symmetric evaluation of the right-hand side of (7), yielding the implicit equation

$$y_{k+1} = y_k + \frac{\Delta}{2}(f(t_k, y_k) + f(t_{k+1}, y_{k+1})) \ .$$

It originates from Crank and Nicholson, but has also been termed trapezoidal integration. For linear time-invariant systems we get

$\left(1-\frac{1}{2}A\Delta\right)y_{k+1}=\left(1+\frac{1}{2}A\Delta\right)y_k$.

The condition for stability of the iteration is in this case

$$\left|\frac{2+\lambda\Delta}{2-\lambda\Delta}\right|<1$$

for all eigenvalues λ of the coefficient matrix A. Irrespective of the choice for $\Delta>0$, this condition is satisfied if all eigenvalues have a negative real part. Therefore, the Crank-Nicholson method is absolutely stable for such systems. For small enough Δ, solving for y_{k+1} is possible

$$y_{k+1}=\left(1+\sum_{i=1}^{\infty}\frac{(A\Delta)^i}{2^{i-1}}\right)y_k \ .$$

The power series matches the exponential series up to second order. For small time steps Δ, this predicts a significantly better accuracy of the Crank-Nicholson method compared to the Backward-Euler method.

C.4 Adams-Bashforth

Multi-step methods use information from several previous steps to achieve better accuracy of the solution. As an example, we consider the simple explicit two-step method of Adams and Bashforth

$$y_{k+1}=y_k+\frac{\Delta}{2}(3f(t_k,y_k)-f(t_{k-1},y_{k-1})) \ .$$

For an n-dimensional linear time-invariant system we obtain the two-step iteration

$$y_{k+1}=\left(1+\tfrac{3}{2}A\Delta\right)y_k-\tfrac{1}{2}A\Delta y_{k-1} \ .$$

An equivalent single-step method with dimension $2n$ is given by

$$\begin{bmatrix}y_k\\ y_{k+1}\end{bmatrix}=\begin{bmatrix}0 & 1\\ -\frac{1}{2}A\Delta & 1+\frac{3}{2}A\Delta\end{bmatrix}\begin{bmatrix}y_{k-1}\\ y_k\end{bmatrix} \ .$$

For stability, all the $2n$ eigenvalues of the one-step time-evolution operator must have magnitude smaller than unity. Since the four component matrices commute pairwise, the eigenvalues μ of the compound matrix are the roots of the n characteristic equations (Scheja and Storch 1980)

$$\mu^2-\left(1+\tfrac{3}{2}\lambda\Delta\right)\mu+\tfrac{1}{2}\lambda\Delta=0 \ ,$$

where λ runs through all eigenvalues of A. Provided that the solution is exact at step $k-1$ and at step k, we know that $y_{k-1}=\mathrm{e}^{-A\Delta}y_k$, yielding

$$y_{k+1}=\left(1+\frac{3}{2}A\Delta-\frac{1}{2}A\Delta\sum_{i=0}^{\infty}\frac{(-A\Delta)^i}{i!}\right)y_k \ .$$

This approximates the exponential series correctly up to second order. Deviations from the exact solution, however, may accumulate during the iteration and even corrupt the first-order term.

C.5 Runge-Kutta

Finally, a very robust and reliable explicit solver is the fourth-order Runge-Kutta method. It involves four evaluations of the right-hand side per step

$$k_1=\Delta f(t_k,y_k)$$
$$k_2=\Delta f\left(t_k+\frac{\Delta}{2},y_k+\frac{k_1}{2}\right)$$
$$k_3=\Delta f\left(t_k+\frac{\Delta}{2},y_k+\frac{k_2}{2}\right)$$
$$k_4=\Delta f(t_k+\Delta,y_k+k_3)$$
$$y_{k+1}=y_k+\frac{k_1}{6}+\frac{k_2}{3}+\frac{k_3}{3}+\frac{k_4}{6} \ .$$

For a linear time-invariant system, we again end up with a linear iteration

$$y_{k+1}=\left(1+A\Delta+\frac{(A\Delta)^2}{2}+\frac{(A\Delta)^3}{6}+\frac{(A\Delta)^4}{24}\right)y_k \ .$$

It reproduces the exact power series up to fourth order. The condition for stability of the corresponding iteration is

$$\left|1+\lambda\Delta+\frac{(\lambda\Delta)^2}{2}+\frac{(\lambda\Delta)^3}{6}+\frac{(\lambda\Delta)^4}{24}\right|<1$$

for all eigenvalues λ of the coefficient matrix.

C.6 Exponential Integration

The differential equation

$$\dot{\eta}+a\eta=\xi$$

has the exact solution

$$\eta(t)=\mathrm{e}^{-a(t-s)}\eta(s)+\frac{1}{a}\left(1-\mathrm{e}^{-a(t-s)}\right)\xi(s) \ ,$$

provided a is constant and ξ does not change on the interval $[s,t)$. However, even if a or ξ vary on a time scale which is slow compared to the step size, the corresponding one-dimensional iteration (see Sect. 3.2.3)

$$\eta_{k+1}=\mathrm{e}^{-a\Delta}\eta_k+\frac{1}{a}\left(1-\mathrm{e}^{-a\Delta}\right)\xi_k \tag{8}$$

yields a fairly good approximation to the exact solution. Certain linear systems, which can be viewed as a cascade of several first-order one-dimensional systems, have been treated in this way. For such a system, one can again write the corresponding propagator in matrix form, as we did for the other integration methods. The time-evolution operator for a post-synaptic potential (see Sect. 3.2.2) implemented along these lines is (with $a_1=a_2$)

$$\begin{bmatrix}\mathrm{e}^{-a_1\Delta} & 0 & 0\\ \frac{1}{a_2}\left(1-\mathrm{e}^{-a_2\Delta}\right) & \mathrm{e}^{-a_2\Delta} & 0\\ 0 & \frac{1}{b}\left(1-\mathrm{e}^{-b\Delta}\right) & \mathrm{e}^{-b\Delta}\end{bmatrix} \ .$$

The eigenvalues of the propagator for such a cascade coincide with the correct eigenvalues of the matrix exponential. This guarantees stability irrespective of the parameters of the system. For this reason, Exponential Integration is frequently used in neuronal modeling. However, all non-diagonal elements, most notably the element in the lower-left corner of the matrix, deviate from their correct values. This explains the approximate nature of the iteration.

Input to the system must also be treated in accordance with the assumption leading to (8). Consequently, a delta-pulse $\delta(t-t_k)$ must be represented by a rectangle of height $1/\Delta$ extending over the interval $[k\Delta,(k+1)\Delta)$. Let y_k^i denote the ith component of the state vector at time t_k. The value of y_{k+1}^1 then is

$$\frac{1}{\Delta}\frac{1}{a_1}\left(1-\mathrm{e}^{-a_1\Delta}\right) \ . \tag{9}$$

Since t_k is the arrival time of the pulse, one encounters a delay compared to (5); the input is represented in the state vector only after one time step. This can be compensated for by using (9) as the initial condition for y^1 in the same moment the pulse arrives. The implicit assumption made by using Exponential Integration is that all functions are piecewise constant; the equation for y^2 integrates y^1 using this assumption. If we were to set y_k^1 to 1, the resulting y_{k+1}^2 would be too large, because the decay of y^1 over the time step has not been corrected for. The "correction factor" for the input in (9) is the

average loss in amplitude of y^1 over interval Δ. This factor becomes important if Δ is not small compared to the time constant $1/a$.

Appendix D: Accuracy measures

To assess the accuracy of different integration methods for the chosen test systems we use three families of measures: local measures d_p quantifying the average error of continuous-valued functions on the grid, global measures l_p comparing the overall shape of an approximate continuous-valued solution with the exact solution, and a measure s_p evaluating the accuracy of pulse trains obtained in simulations of spiking neurons.

D.1 Average local error of continuous signals

Let $\eta(t)$ be the exact solution to the initial value problem under consideration. The solution on the grid with step size Δ restricted to the time interval $[0, T]$ is represented by a vector η^Δ with $n^\Delta = \lfloor T/\Delta \rfloor + 1$ components, given by $\eta_k^\Delta = \eta(k\Delta)$. Some approximate solution on the same grid is represented by the vector $\hat{\eta}^\Delta$. Using the vector norm

$$\|f\|_{D_p} = \begin{cases} \left(\sum_{i=0}^n |f_i|^p\right)^{1/p} & 1 \le p < \infty \\ \sup |f_i| & p = \infty \end{cases}$$

for a vector f, we define d_p as the average error per data point expressed in units of the amplitude of the exact solution

$$d_p(\Delta) = \frac{1}{\sup|\eta|}\left(\frac{1}{n^\Delta}\right)^{1/p} \|\hat{\eta}^\Delta - \eta^\Delta\|_{D_p} \ .$$

For $p = 2$, this is the RMS of the point-wise deviations, divided by the amplitude of the exact solution.

D.2 Global error of continuous signals

For a continuous function $f(t)$ evaluated over the interval $[0, T]$ we define the L_p norm in the usual way

$$\|f\|_{L_p} = \begin{cases} \left(\int_0^T |f(t)|^p \, dt\right)^{1/p} & 1 \le p < \infty \\ \sup |f| & p = \infty \end{cases} \ .$$

Let $\eta(t)$ be the exact solution and $\hat{\eta}^\Delta$ a numerical approximation on the grid, as above. We construct a function in continuous time by linear interpolation

$$\hat{\eta}^\Delta(t) = \hat{\eta}^\Delta_{\lfloor t/\Delta \rfloor} + \left(\hat{\eta}^\Delta_{\lfloor t/\Delta \rfloor + 1} - \hat{\eta}^\Delta_{\lfloor t/\Delta \rfloor}\right) \cdot (t/\Delta - \lfloor t/\Delta \rfloor)$$

and define the global error of the approximate solution with respect to the exact solution as

$$l_p(\Delta) = \|\hat{\eta}^\Delta - \eta\|_{L_p} / \|\eta\|_{L_p} \ .$$

For $p = 1$, this is the relative difference in area of the two curves.

D.3. Accuracy of pulse trains

Let $\phi(t)$ be a kernel of standard width w, appropriately normalized Gaussians are used in the context of this paper. By convolution with this kernel, any pulse train $\xi(t) = \sum_i \delta(t - t_i)$ is associated with a continuous-valued function

$$\xi_\phi(t) = (\phi * \xi)(t) = \sum_i \phi(t - t_i) \ .$$

Resorting to norms for continuous-valued functions, this association can be used to define a measure for the distance between any two finite trains of unit pulses

$$s_p(\xi, \xi') = \|\xi_\phi - \xi'_\phi\|_{L_p} \ .$$

We used the Euclidean norm ($p = 2$) for the purpose of this paper. The width w of the kernel specifies the temporal resolution of the distance function. For our application, we consider only kernels that are narrow compared to the typical inter-event interval.

To understand the scaling behavior of the metric s_2, we consider two extreme situations. The first case is that of two pulse trains ξ and ξ' comprising n and n' pulses, respectively. If one can assume that $m \le n, n'$ points perfectly coincide, but all others are out of the range of the kernel, the Euclidean distance of the two pulse trains is

$$s_2(\xi, \xi') \approx \sqrt{n - 2m + n'} \ .$$

This is true provided that the kernel is narrow with respect to the typical inter-event interval. In particular, if one pulse train is simply a subset of the other, we have

$$s_2(\xi, \xi') \approx \sqrt{n - n'} \ .$$

The second case is that of two pulse trains ξ and ξ' with an identical number $n = n'$ of pulses. Corresponding points, however, are randomly shifted with respect to each other. The distribution of shifts has mean 0 and standard deviation σ. If the difference between the two pulse trains is probed by a metric s_2 with a narrow Gaussian kernel, one finds

$$s_2(\xi, \xi') \approx \sqrt{\frac{n}{2}} \cdot \frac{\sigma}{w} \ ,$$

if σ is smaller than or at most equal to w. For larger shifts σ, however, the two pulse trains become more and more orthogonal, with a Euclidean distance approaching

$$s_2(\xi, \xi') = \sqrt{2n} \ .$$

The latter number also represents the worst possible outcome of a digital simulation of spike trains yielding the correct number n of spikes. A lower bound for the error in digital simulations of spike trains on a grid with step size Δ is given by the discretization error, which is introduced by the fact that spikes can only be reported on the grid. The standard deviation of the spike-wise errors is $\sigma = \sqrt{\Delta/12}$ provided that they are uniformly distributed on $[-\Delta/2, \Delta/2]$. Therefore, choosing a kernel of width $w = \Delta$ for convolution yields a measure of the lower bound of the integration error, which is normalized for the step size of integration and depends only on the total number of spikes in the spike train

$$s_2(\xi, \xi') = \sqrt{\frac{n}{24}} \ .$$

Acknowledgements. This work was partially supported by the German-Israeli Foundation (GIF) and the Deutsche Forschungsgemeinschaft (DFG). We gratefully acknowledge the comments of Ad Aertsen, Ellen Baake and Marc-Oliver Gewaltig on earlier versions of the manuscript.

References

Arnol'd VI (1992) Ordinary differential equations, 3rd edn. Springer, Berlin Heidelberg New York

Bernard C, Ge YC, Stockley E, Willis JB, Wheal HV (1994) Synaptic integration of NMDA and non-NMDA receptors in large neuronal network models solved by means of differential equations. Biol Cybern 70:267–273

Bower JM, Beeman D (1997) The book of GENESIS: Exploring realistic neural models with the GEneral NEural SImulation System, 2nd edn. TELOS, Springer, New York Berlin Heidelberg

Bronstein IN, Semendjajew KA, Grosche G, Ziegler V, Ziegler D, Zeidler E (1996) Teubner-Taschenbuch der Mathematik, 25th edn. Teubner, Stuttgart, Leipzig

Colquhoun D, Hawkes AG (1977) Relaxation and fluctuations of membrane currents that flow through drug-operated channels. Proc R Soc Lond B 199:231–262

Colquhoun D, Hawkes AG (1981) On the stochastic properties of ion channels. Proc R Soc Lond B 211:205–235

Colquhoun D, Hawkes AG (1982) On the stochastic properties of bursts of single channel openings and of clusters of bursts. Proc R Soc Lond B 300:1–59

Colquhoun D, Hawkes AG (1995a) The principles of the stochastic interpretation of ion-channel mechanisms. In: Sakmann B, Neher E (eds) Single-channel recording, chap 18, 2nd edn. Plenum Press, New York, pp 397–482

Colquhoun D, Hawkes AG (1995b) A Q-matrix cookbook: how to write only one program to calculate the single-channel and macroscopic predictions for any kinetic mechanism. In: Sakmann B, Neher E (eds) Single-channel recording, chap 20, 2nd edn. Plenum Press, New York, pp 397–482

Druskin V, Greenbaum A, Knizhnerman L (1998) Using nonorthogonal Lanczos vectors in the computation of matrix functions. SIAM J Appl Math 19:38–54

Golub GH, van Loan CF (1996) Matrix computations, 3rd edn. Johns Hopkins University Press, Baltimore

Hansel D, Mato G, Meunier C, Neltner L (1998) On numerical simulations of integrate-and-fire neural networks. Neural Comp 10:467–483

Heal KM, Hansen M, Rickard K (1997) Maple V learning guide for release 5. Springer, Berlin Heidelberg New York

Hines M, Carnival NT (1995) Computer modeling for neurons. In: Arbib MA (ed) The handbook of brain theory and neural networks. MIT Press, Cambridge, Mass, pp 226–230

Hines M, Carnevale NT (1997) The NEURON simulation environment. Neural Comp 9:1179–1209

Hirsch M, Smale S (1974) Differential equations, dynamical systems, and linear algebra. Academic Press, San Diego

Hodgkin AL, Huxley AF (1952) A quantitative description of membrane current and its application to conduction and excitation in nerve. J Physiol (Lond) 117:500–544

Jack JJB, Noble D, Tsien RW (1983) Electric current flow in excitable cells. Oxford University Press, Oxford

Kenny CS, Laub AJ (1998) A Schur-Fréchet a logrithm for computing the logarithm and exponential of a matrix. SIAM J Matrix Anal Appl 19:640– 663

Knight BW (1972) Dynamics of encoding in a population of neurons. J Gen Physiol 59:734–766

Leonard IE (1996) The matrix exponential. SIAM Rev 38:507–512

MacGregor RJ (1987) Neural and brain modeling. Academic Press. San Diego

Mascagni MV (1989) Numerical methods for neuronal modeling. In: Koch C, Segev I (eds) Methods in neuronal modeling: from synapses to networks, chap 13. A Bradford book. MIT Press, Cambridge, Mass, pp 439–440

MathWorks Inc. (1998) MATLAB the language of technical computing: using MATLAB. Natick, Mass

Moler C, von Loan C (1978) Nineteen dubious ways to compute the exponent of a matrix. SIAM Rev 20:801–836

Papoulis A (1991) Probability, random variables, and stochastic processes, 3rd edn. McGraw-Hill, New York

Press WH, Teukolsky SA, Vetterling WT, Flannery BP (1992) Numerical recipes in C, 2nd edn. Cambridge University Press, Cambridge

Rall W (1964) Theoretical significance of dendritic trees for neuronal input-output relations. In: Reiss RF (ed) Neural theory and modeling. Stanford University Press, Palo Alto

Rudin W (1991) Functional analysis, 2nd edn. WBC/McGraw-Hill, New York

Scheja G, Storch U (1980) Lehrbuch der Algebra. Teubner, Stuttgart

Stein RB (1965) A theoretical analysis of neuronal variability. Biophys J 5:173–194

Stoer J, Bulirsch R (1996) Introduction to numerical analysis, 2nd edn. Springer, Berlin Heidelberg New York

Troyer TW, Miller KD (1997) Physiological gain leads to high ISI variability in a simple model of a cortical regular spiking cell. Neural Comp 9:971–983

Tuckwell HC (1988) Introduction to theoretical neurobiology, vol 1. Cambridge University Press, Cambridge

Vreeswijk C van, Sompolinsky H (1996) Chaos in neuronal networks with balanced excitatory and inhibitory activity. Science 274:1724–1726

Walter W (1996) Gewöhnliche Differentialgleichungen: eine Einführung, 6th edn. Springer, Berlin Heidelberg New York

Wilson MA, Bower JM (1989) The simulation of large-scale neural networks. In: Koch C, Segev I (eds) Methods in neuronal modeling: from synapses to networks, chap 9. A Bradford book. MIT Press, Cambridge Mass, pp 291–333

Wolfram S (1996) The Mathematica book, 3rd edn. Wolfram Media/Cambridge University Press, Cambridge