High precision stochastic solvers for large autonomous systems of differential equations

Flavius Guiaş

Abstract—In this paper we propose an improvement of the stochastic Picard–Runge–Kutta solvers for large autonomous systems of ordinary differential equations. These solvers deliver best results for systems with a sparse incidence matrix, for example in the case of spatially discretized partial differential equations. Their basic principle is the fact that the paths of Markov jump processes can be used as an approximation for solutions of systems of differential equations. The step function $\tilde{X}$ computed by the simulation of the jump processes or the Picard–approximation $\bar{X}$ can serve as a predictor which is further improved by suitable correction steps. Given the improved approximation $X^*(t)$ at time t, we compute the corresponding approximation at time $t+h$ by an integral scheme of the form $X^*(t+h) = X^*(t) + \int_t^{t+h} Q(s)\,ds$. For computing the improved approximations $X^*(\cdot)$ we take for the integrand Q a polynomial which interpolates some equidistant intermediate values of $F(\tilde{X}(\cdot))$ or $F(\bar{X}(\cdot))$ between t and $t+h$. By using an exact quadrature formula in order to compute the integral above, we can employ the principle of the Runge–Kutta method in order to compute a better approximation. The goal of this paper is to improve the precision of the intermediate values used by the above described stochastic Runge–Kutta method. The improved values are computed by a stochastic counterpart of a second-order Runge–Kutta method. The result is a high precision scheme with several layers, which starts from the crude approximation delivered by the standard jump process, and based on this data it computes several steps in which the approximations are successively refined.

I. Introduction

In this paper we will show how solutions of large autonomous systems of ordinary differential equations can be efficiently approximated by using a suitable stochastic method. The scheme delivers the best results for systems with a sparse incidence matrix, for example in the case of spatially discretized partial differential equations. Its basic principle is the fact that the dynamics of Markov jump processes can be used as an approximation for solutions of systems of differential equations, as shown in [1].

The motivation comes mainly from the field of chemical kinetics, see [2], [3]. In these papers a particular form of jump process was introduced which, instead of describing the reactions at the molecular level, mimics rather the macroscopic dynamics given by the limit equations $\dot{X} = F(X)$, $F = (F_i)_{i=1}^n$. The approach is called *direct simulation method* and represents a sort of "digitization" of the corresponding differential equations. A path computed by this approach is denoted by $\tilde{X}(\cdot)$, being a vector-valued step function. In contrast to the standard deterministic solvers, here in a computational step only one component $\tilde{X}_i$ is changed. The selection occurs at random, with probabilities proportional to the absolute value of the right hand side of the equations $F_i(\tilde{X}(s))$, whose sign indicates the direction of the change. The chosen component is changed by a value of $\pm 1/N$, i.e. with the weight of a virtual numerical particle, which can be also interpreted as the "resolution" of the approximation. The time is advanced by the usual exponentially distributed waiting time related to the underlying Markov jump process. Its length is adapted, since the mean depends on the current state of the process, being inverse proportional to its total rate. Considerations regarding applications in other fields and implementations aspects are presented in the book [4].

The paper [7] applies the direct simulation method to discretized partial differential equations driven mainly by diffusion and/or convection. The experiments showed that this scheme approximates the macroscopic dynamics given by the differential equation better than alternative modeling approaches, which simulate the flux or even the Brownian motion at the molecular level. Nevertheless, the performance of the stochastic schemes based on jump processes turns out to be far from that of standard deterministic solvers.

One possible approach for improving the direct simulation method is the so called *tau leap* scheme, see [5], [6], where one computes basically the number of jumps in a given (larger) time- interval and then performs all transitions at once.

Improvements based on a different principle are reported in [8] and [9]. The step function $\tilde{X}(\cdot)$ computed by the direct simulation approach can serve as a predictor which is further improved by suitable correction steps. Given the improved approximation $X^*(t)$ at time t, we compute the corresponding approximation at time $t+h$ by an integral scheme of the form

$$X^*(t+h) = X^*(t) + \int_t^{t+h} Q(s)\,ds. \tag{1}$$

One possibility is to take $Q(s) = F(\tilde{X}(s))$, i.e. to perform a Picard iteration based on the predictor $\tilde{X}(\cdot)$ in order to obtain a better approximation denoted in particular by $\bar{X}(\cdot)$. The integral can be computed exactly during the simulation of the jump process, since the integrand is a step function.

A range of further possibilities for computing improved approximations $X^*(\cdot)$ opens if we take for the integrand Q a polynomial which interpolates some intermediate values of $F(\tilde{X}(\cdot))$ or $F(\bar{X}(\cdot))$. The directly simulated process $\tilde{X}$ or the Picard–approximation $\bar{X}$ are evaluated here at some few equidistant points between t and $t+h$. By using an exact quadrature formula in order to compute the integral in (1), we can employ the principle of the Runge–Kutta method in order

F. Guiaş is with the Department of Mechanical Engineering, Dortmund University of Applied Sciences and Arts, Sonnenstr.96, 44139 Dortmund, Germany, e-mail: flavius.guias@fh-dortmund.de

to further improve our approximation. The length h of the time interval can be taken either fixed, or adapted (controlled by a given number of jumps of the process). Tests on standard benchmark equations show that this improved stochastic method can be comparable or even more efficient than the standard deterministic ODE-solver *ode45* (or Dormand–Prince Runge–Kutta method) implemented under the same conditions as the former one.

In this paper we will discuss the possibility of enhancing the performance of the stochastic solvers presented in [8] and [9] by improving the precision of the intermediate values used within the Runge–Kutta principle.This is done by first applying in this stochastic framework a Runge–Kutta step similar to the corresponding standard deterministic second order method. In section II is given a detailed description of the basic methods and of their improvements, while section III presents the results of numerical experiments. After this, we make some comments and draw some conclusions concerning the results obtained by using the scheme proposed in this paper.

II. Description of the methods

The basic stochastic direct simulation scheme for autonomous systems of ODE $\dot{X} = F(X)$ which is then successively improved delivers paths of a Markov jump process $\tilde{X}(\cdot)$. Its feature is that at every jump only one component of the process is changed with a fixed amount $\pm 1/N$, which can be interpreted as the resolution of the method. The component i which is chosen to be changed in the next step is selected at random with a probability proportional to $|F_i(\tilde{X}(t))|$. The steps of the direct simulation method are the following: While $t \le t_{max}$ do:

1) Given the state vector $\tilde{X}(t)$ of the process at time t:
2) Select a component X_i with probability proportional to $|F_i(\tilde{X}(t))|$.
3) The time step $\Delta t = -\log U/\lambda$ with U uniformly distributed on $(0,1)$ is then exponentially distributed with parameter $\lambda = N\sum_{i=1}^{n} |F_i(\tilde{X}(t))|$.
4) Update the value of the selected component: $\tilde{X}_i \mapsto \tilde{X}_i + \frac{1}{N} sign(F_i(\tilde{X}))$ and set the new time as $t = t + \Delta t$.
5) Update the values of $F_j(\tilde{X}(t))$ for all j (for which $F_j(\tilde{X}(t))$ depends explicitly on the changed component $\tilde{X}_i$ in step 2).
6) GOTO 1.

Writing the ODE system on the time interval $[t, t+h]$ in the integral form yields:

$$X(t+h) = X(t) + \int_t^{t+h} F(X(s))\, ds.$$

Assuming that $\bar{X}(t)$ is an approximation for the exact solution $X(t)$ and that we have simulated a path $\tilde{X}(s),\ t \le s \le t+h$, we can use these data in order to compute an approximation for $X(t+h)$ which improves the crude result $\tilde{X}(t+h)$. This is done by a Picard-iteration:

$$\bar{X}(t+h) = \bar{X}(t) + \int_t^{t+h} F(\tilde{X}(s))\, ds.$$

The integral to be computed is that of a step function and can be computed explicitly, by updating its value after every jump of the Markov process $\tilde{X}(\cdot)$.

A further improvement of the precision of the above schemes is the employment of Runge–Kutta steps of the general form

$$X^*(t+h) = X^*(t) + \int_t^{t+h} Q(s)\, ds, \tag{2}$$

where $X^*(t)$ is a given approximation for the exact solution $X(t)$ and $Q(s)$ is a vector of polynomials which approximates the exact term $F(X(s))$. Note that in the case of Picard iterations we have approximated it by $F(\tilde{X}(s))$, i.e. by using the path of the simulated Markov jump process. The polynomial $Q(s)$ used by the Runge–Kutta steps interpolates some equidistant intermediate values $t+\alpha_i h$ between t and $t+h$ and its integral can be computed by an exact quadrature formula. We basically use three such methods, which are similar to the standard deterministic Runge–Kutta schemes, with the only difference that here the intermediate values are the result of stochastic simulations. If k_i are proper approximations of $F(X(t+\alpha_i h))$, the integral $\int_t^{t+h} Q(s)\, ds$ can be therefore computed by one of the following quadrature schemes:

$$h\left(\frac{1}{2}k_1 + \frac{1}{2}k_2\right) \tag{3}$$

which integrates exactly the linear interpolation for the two nodes and is similar to the deterministic second order Heun method. In combination with (2) we call this the (RK2)-scheme.

The next choice is

$$h\left(\frac{1}{6}k_1 + \frac{4}{6}k_2 + \frac{1}{6}k_3\right) \tag{4}$$

which integrates exactly the quadratic and cubic interpolation polynomial for the three nodes, similar to the Simpson's 1/3 (or Kepler's) rule, corresponding to a deterministic Runge–Kutta scheme of third order. In combination with (2) we call this the (RK3)-scheme.

The final possibility which we discuss here is

$$h\left(\frac{1}{8}k_1 + \frac{3}{8}k_2 + \frac{3}{8}k_3 + \frac{1}{8}k_4\right) \tag{5}$$

which integrates exactly the cubic interpolation polynomial for the four nodes and corresponds to Simpson's 3/8 rule, similar to the corresponding deterministic Runge–Kutta method of fourth order. In combination with (2) we call this the (RK4)-scheme.

In all the above schemes, k_i are stochastic approximations of $F(X(t+\alpha_i h))$, i.e. of values at some equidistant time steps $t+\alpha_i h$ between t and $t+h$, which are computed by the direct simulation method possibly followed by a Picard iteration. Additionally, k_1 is a further improved approximation at the end of the previous small time interval of length h by one of the Runge–Kutta steps described above. The length h of these time intervals can be either fixed or automatically adapted, by prescribing a given number of jumps of the Markov process $\tilde{X}(\cdot)$.

The methods based on the previously described steps were introduced in [8] and [9] and showed good results when applied to large ODE systems arising from parabolic PDE's by finite difference spatial discretization.

However, there is still room for improvement. In order to compute the terms k_i used in the Runge–Kutta steps, their precision can be enhanced by using a Picard iteration and/or a Runge–Kutta step of order two, both on the last small (partial) time interval. We illustrate this feature by describing in detail the steps involved in the scheme (4) with a fixed time step h.

1) Start with the state vector $X^*(t)$ at time t.
2) Compute $k_1 = F(X^*(t))$ as an approximation for $F(X(t))$.
3) Compute a path $\tilde{X}(s)$ of the Markov jump process by the above direct simulation method, until the time variable exceeds $h/2$, i.e. the half of the length of the time interval on which the Runge–Kutta scheme (4) is applied.
4) Compute optionally an improved approximation $\bar{X}(t + h/2)$ by a Picard iteration on $[t, t + h/2]$ and then a further improved value $X^*(t + h/2)$ by using a Runge–Kutta step of type (3) on $[t, t + h/2]$ for the integral in (2). With this value compute the approximation $k_2 = F(X^*(t + h/2))$ for $F(X(t + h/2))$.
5) The final time step where we stop the direct simulation of $\tilde{X}(\cdot)$ is when the time variable exceeds $t + h$. By similar steps as above in 4) compute the approximation $k_3 = F(X^*(t + h))$ for $F(X(t + h))$.
6) Compute the approximation $X^*(t+h)$ by using formula (4) in order to compute exactly the integral in (2).

If one uses the method (5) the steps are similar. The length of the time subintervals will be then taken to be $h/3$. We need then three intermediate steps as above, at the moments $t + h/3, t + 2h/3$ and $t + h$ in which we may employ Picard iterations followed by Runge–Kutta steps (3). As in the scheme described above, we compute the values k_1, k_2, k_3, k_4 and use the quadrature formula (5) for the integral in (2), which finally leads to the improved approximation for $X(t + h)$.

At this point a remark is needed. The jump process $\tilde{X}$ stays a random time interval of length Δt in its current state and will change only after this time. Since we need values at intermediate time points, for example $t + h/2$, we take the value of the process at this time, i.e. before the jump. In this case the jump is not performed and the value is that one given after the previous jump, where it has been effectively changed. The same remark also applies if we use Picard iterations. Here we take again the values before the jump, since the terms which are involved are integrals of right continuous step functions, which change only after Δt, but this new value has no effect if integrating over a past time interval.

The adaptive version of these methods use variable lengths h of time intervals which are determined implicitly by prescribing a given number M of jumps (typically of a magnitude order equal to the number of equations). In the adaptive RK2-scheme the value h is given by adding the corresponding random waiting times, for the RK3 (RK4) scheme we take this value as $h/2$ ($h/3$) and perform the intermediate steps on intervals of this length, which can be computed after the prescribed number of jumps has occurred. That is, after automatically determining the length of the first half (third) of the time interval of size h, we can compute the other time moments of the current step where the intermediate values k_i are evaluated.

We note that the structure of the scheme used for computing the approximation $X^*(t + h)$ for the exact solution vector $X(t+h)$ (with fixed or variable time step h) consists in several layers of successive improvements.

At the first level we have a path $\tilde{X}$ of a Markov jump process, which in general is a very crude approximation. However, we can take advantage of the fact that we have computed a full path of it, which also reflects the exact dynamics which it approximates.

Optionally we can compute improved approximations at the intermediate time steps by Picard iterations.

Starting either from these values, or directly from the path of the jump process, we compute a next level of improved approximations of these intermediate values by the (RK2) scheme. Finally, we compute the best approximation for $X(t + h)$ by using these values within the schemes (RK3) or (RK4), calling them respectively (RK23) and (RK24) methods, additionally specifying by '-pic' if also the Picard iteration has been used or '-adap' if the times steps h are not fixed, but adaptively computed.

III. Numerical simulations

We illustrate the application of the described methods at the standard test equation

$$\frac{\partial u}{\partial t} = \Delta u + \frac{5e^{\delta}}{\delta}(2 - u)\exp\left(-\frac{\delta}{u}\right) \tag{6}$$

with initial condition $u_0 \equiv 1$, on $(0, 1)$ with boundary conditions $\partial_\nu u(0) = 0$ and $u(1) = 1$.

The variable u denotes a temperature which increases up to a critical value when ignition occurs (for example for $\delta = 30$ at time $t = 0.240$) resulting in a fast propagation of a reaction front towards the right end of the interval (in the mentioned case $t = 0.244$). Due to the very high speed and the steepness of the front, a very precise time resolution is very important in any numerical approximation of this problem.

The ODE system considered for our numerical tests is a finite difference discretization over $n = 500$ gridpoints of the PDE (6).

We first consider a fixed time discretization step $h = 10^{-7}$ and N denotes the factor which is multiplied with the number n of equations, that is after a jump of the Markov process, the corresponding component is changed by $\pm 1/(Nn)$.

The CPU-times (in seconds) and the corresponding errors (in the max-norm), based on a reference solution computed with the maximal possible precision of the MATLAB solver *ode113*, are plotted in the following tables.

Results for the RK3-type methods:

	$N = 10^2$	$N = 10^3$	$N = 10^4$
RK3	285, 8.36e-6	371, 1.03e-5	1216, 1.69e-5
RK3-pic	678, 6.01e-7	789, 1.31e-7	1823, 9.47e-8
RK23	440, 1.80e-7	523, 1.94e-7	1375, 1.41e-7
RK23-pic	808, 8.18e-8	921, 7.94e-8	2062, 3.79e-8

Results for the RK4-type methods:

	$N = 10^2$	$N = 10^3$	$N = 10^4$
RK4	359, 3.43e-5	446, 5.50 e-6	1295, 2.13e-5
RK4-pic	917, 2.87e-7	1030, 7.30e-7	2184, 2.35e-7
RK24	560, 2.20e-7	652, 4.48e-8	1515, 9.84e-8
RK24-pic	1139, 7.74e-8	1257, 7.35e-8	2331, 6.49e-8

First some general remarks. We note that all solvers tested here deliver high precision computations, which are limited only by the value taken for the time discretization step h and by the structure of quadrature scheme which was used. In the direct simulation method, by increasing N, we may note a steady (however very slow) improvement of the convergence behaviour. Within this family of schemes, this is possible only if we at the same time decrease the time resolution step h. For our purposes it was sufficient to take a proper value, for example $h = 10^{-7}$, which leads to high precision results, but which cannot be improved only by increasing N, as the above results show us. Basically in practice we can use the smallest value of $N = 10^2$. By further increasing N, the precision does not improve anymore (the differences which arise here are mostly due to stochastic fluctuations). By taking a smaller N, for example $N = 10$, the CPU time is only slightly below that corresponding to $N = 10^2$, while the error is slightly larger. This phenomenon of non-linear behaviour of the CPU-time in dependence of N is explained in section 3.3 of [8]. Within this family of schemes a fast computation with low precision is not possible. Due to their inherent structure, even if we take coarse values of N and h, while the precision can be of course decreased, not so the computing time, which cannot be pushed below a certain level.

Having said this, let us now discuss the above results. The goal of this paper was to introduce the new methods RK23 and RK24 (without or with Picard iterations) and to compare them with the corresponding method RK3 respectively RK4 (again without or with Picard iterations) which use the usual intermediate values. In all cases we note a decrease of the error by a factor of about 10, while the CPU-time increases only by the factor of at most 1.5. The schemes which deliver the best precision (but also need the longest computation time) are RK23-pic and RK24-pic which involve all possible ingredients, structured in several layers. At the first level we have the values given by the jump process $\tilde{X}$. Better intermediate values for the Runge–Kutta type schemes are obtained by applying first a Picard iteration followed by a RK2-step. Such high precision solvers might be therefore useful for example in computing reference solutions for certain problems.

We will perform next a comparison of several variants of the methods discussed in this paper, either with adaptive or fixed time discretization steps. Figure 1 depicts the efficiency of the schemes. We note that the RK23 and RK24 methods with fixed time step $h = 10^{-7}$ have basically the same efficiency, but the use of adaptive time steps in conjunction with the improvements such as Picard iterations and/or intermediate RK2-steps, the efficiency is significantly increased. At about the same CPU-time, the error is smaller by a factor of 10.

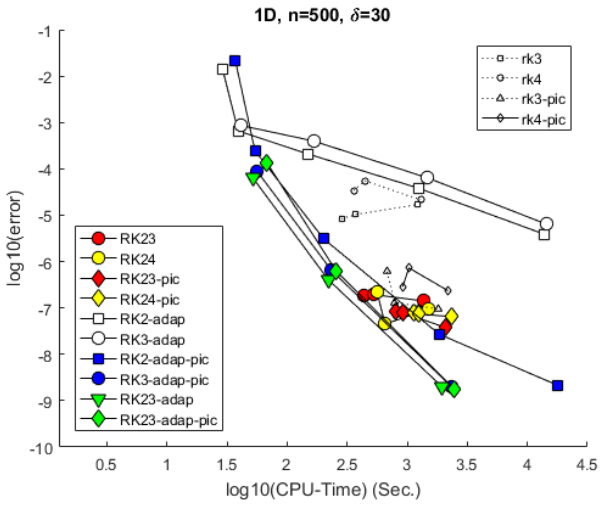

Fig. 1. Efficiency comparison of different methods

IV. Conclusion

In this paper we have introduced a new possibility to improve the stochastic schemes of Runge–Kutta type based on Markov jump processes. The main idea is to enhance the precision of the intermediate values used by these schemes by using a simpler method from the same family, which doesn't involve intermediate time steps. The increase in precision is notable and this fact shows once more that this family of solvers can be successfully applied at large systems of ordinary differential equations, especially if they arise as spatial discretization of certain partial differential equations.

References

[1] S. Ethier and T.G. Kurtz, *Markov Processes: Characterization and Convergence*. Wiley, 1986.

[2] D. Gillespie. General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions. *J. Comp. Phys.*, 22 : 403–434, 1976.

[3] D. Gillespie. Stochastic Simulation of Chemical Kinetics. *Annu. Rev. Phys. Chem.*, 58 : 35–55, 2007.

[4] L. Marchetti, C. Priami, V.H. Thanh. *Simulation Algorithms for Computational Systems Biology.* Texts in Theoretical Computer Science. An EATCS Series. Springer, Cham, 2017.

[5] D.F. Anderson, A. Ganguly, T.G. Kurtz. Error analysis of tau-leap simulation methods. *Ann. Appl. Prob.*, 21 (8) : 2226–2262, 2011.

[6] A. Moraes, R. Tempone, P. Vilanova. Multilevel hybrid Chernoff tau-leap. *Bit Numer. Math.*, 56 (1) : 189–239, 2016.

[7] F. Guiaş. Direct simulation of the infinitesimal dynamics of semi-discrete approximations for convection–diffusion–reaction problems. *Math. Comput. Simulation*, 81 : 820–836, 2010.

[8] F. Guiaş and P. Eremeev, Improving the stochastic direct simulation method with applications to evolution partial differential equations. *Appl. Math. Comput.*, 289 : 353–370, 2016.

[9] F. Guiaş, Stochastic Picard–Runge–Kutta Solvers for Large Systems of Autonomous Ordinary Differential Equations. *2017 Fourth International Conference on Mathematics and Computers in Sciences and in Industry (MCSI)*, DOI 10.1109/MCSI.2017.55 : 298–302, 2017.