Report

Jaime Alberto Londoño
Andrés Mauricio Villegas

1 Introduction

Ordinary differential equations (ODEs) are a common tool for modelling physical systems. Such models represent idealized versions of real systems, as they are purely deterministic. Stochastic differential equations (SDEs) are the instrument for building more realistic models, as they include the random elements. SDEs are used in many areas of applications, including investment finance, economics, insurance, signal processing and filtering, several fields of biology and physics, population dynamics and genetics.

Since there are very few SDEs for which exact analytical solutions are known, numerical techniques have to be used in the solution of SDEs. Unlike in the solution of ODEs, SDEs can be approximated in two senses: strong and weak. We refer to strong approximations when we want to approximate the trajectory of the solution. When we do not require the whole trajectory, but a function from the solution, for instance a moment, we talk about weak approximations. Some introductions to numerical methods for SDEs are given in [3, 15]

A general Stratonovich SDE has the form

$$dX(s) = a(X,s)ds + \sigma(X,s) \circ dW(s) \qquad X(t_0) = X_0 \qquad X_0 \in \mathbb{R}^n \tag{1}$$

where $a(X,t) \in \mathbb{R}^n$, $\sigma(X,t) \in \mathbb{R}^{n \times d}$, and $W(t)$ is a d-dimensional Brownian motion. Commonly, $a(\cdot)$ and $\sigma(\cdot)$ are called the drift and the diffusion term, respectively.

The aim of this project was the development and implementation of some numerical schemes for the strong approximation of the above equation. The implementation of this numerical schemes implies the solution some common computational mathematics problems, such as: linear algebra problems, random number generation and ODE solution. The importance and the approach we have taken to solve this problems are explained in the following sections.

This document also includes a brief explanation of the proposed numerical methods and an example of the numerical experiments carried out during this project.

2 Linear Algebra (BLAS - LAPACK)

The simulation of stochastic differential equations (SDE) can be extremely time consuming [4]. It is our goal to implement, in a vectorized way, some algorithms for the numerical simulation of SDE and hence produce efficient algorithms. Since in a vectorized implementation a significant amount of execution time may be spent computing basic vector and matrix operation, we need a library that provides us with efficient implementations of the most common linear algebra routines. We have chosen to use BLAS and LAPACK in the implementations because they are the standard for linear algebra computations.

The BLAS (Basic Linear Algebra Subprograms) are high quality "building block" routines for performing basic vector and matrix operations [2]. LAPACK is a library of Fortran 77 subroutines for solving the most commonly occurring problems in numerical linear algebra. They have been designed to be portable and efficient on a wide range of modern high performance computers [1]. For further information about BLAS and LAPACK visit `http://www.netlib.org/blas/`

http://www.netlib.org/lapack/ or see [2, 1].

3 Random Number Generators

Random numbers are the source of all the randomness required in stochastic simulation. The most reliable way of generating random numbers is through deterministic algorithms that produce sequences whose properties resemble those of sequences of independent and identically distributed $U(0,1)$ random variables , (i.e., deterministic sequences that seem to behave like random numbers). Such an algorithm is called a random number generator (RNG).

The quality of simulation results is strongly determined by theoretical and statistical properties of the RNG used. Some of the properties that a good RNG should have are: long period, low serial correlation, efficiency (run fast and use only a small amount of memory) and repeatability (reproduce exactly the same sequence as many times as we want). A complete review on random number generators can be seen in [11]

We built a common interface to random number generators, in order to follow the vectorization fashion of our library. This allows us to use use in the library any random number generator, that follows the interface, without having to make substantial changes to the code. The details of this interface can be seen in the library documentation [17].

In our research we use adapted version of the combined multiple recursive generator of L'Ecuyer and the Mersenne twister of Matsumoto and Nushimura. This two generator were adapted to the proposed interface. We have chosen these RNGs because they have fairly solid theoretical support and have been extensively tested [9]. For more information about this two generators see [9, 10] and [13, 14], respectively.

We also tried to vectorize the algorithm of L'Ecuyer's combined multiple recursive generator but the final vectorize implementation was less efficient, in terms of running time, that the original implementation, given in [9]

4 Ordinary differential equations solver - CVODE

The solution of systems of ordinary differential equation (ODE) plays an important role in the methods of solution of stochastic differential equations that were studied during the research. Particularly, we are interest in the solution of ODE initial value problems. An ODE initial value problem can be written as

$$\frac{dy}{dt} = f(y,t), \qquad y(t_0) = y_0, \qquad y \in \mathbf{R}^N \tag{2}$$

When solving ODE problems almost always numerical techniques must be used since the available analytical techniques are not powerful enough to solve any ODE problem except the simplest. ODE problems are generally divided into two categories, stiff and nonstiff problems. Stiffness is not easy to define, but roughly speaking, if a problem is stiff it will be harder to solve than a nonstiff one. That is why, the numerical methods for solving stiff equations are different from those for solving nonstiff equations. The methods used for solving nonstiff ODEs are based on the Runge-Kutta, Adams, or extrapolation methods. This methods are usually not suitable for solving stiff problems. The interested reader may refer to [6] for a survey on numerical methods for ODEs.

As well as with RNGs a common interface to ODE solvers was written. This interface permits the integration of any ODE solver to our library. The only restriction on the ODE solver is that it is able to solve both stiff and nonstiff problems. The details of this interface can be seen in the library documentation [17].

The ODE solver that we used in our research is called CVODE. CVODE is a solver, written in C, for stiff and nonstiff initial value problems for systems of ordinary differential equations [5]. The underlying integration methods used in CVODE are variable-coefficient forms of the Adams and BDF (Backward Differentiation formula) methods, for nonstiff and stiff problems, respectively. This solver can be downloaded from *http://www.netlib.org/ode/cvode.tar.gz*.

We adapted CVODE, so that it could work with the generic interface for ODE solvers. Since the solution of ordinary differential equations is a very important building block for the numerical methods studied during the research, CVODE was also modified to use BLAS and LAPACK. Inside CVODE, many vector-matrix operations and the solution of systems of linear equations are involved. In this modified version, all the vector and matrix computations are made with BLAS and the solution of linear systems is made with LAPACK. The objective of these changes is to take advantage of the high performance and portability of BLAS and LAPACK. The run-times of the original and the modified versions of CVODE were compared and the version using BLAS and LAPACK showed to be more efficient than the original version.

Taking into account that most of the run-time of the proposed algorithms is spent solving systems of ODE, a possible area for future research would be the development and implementation of algorithms for the solution of ODE. The methods for ODEs would be implemented focusing on the particular characteristics of the ODE systems arising in the solution of stochastic differential equations.

5 Linear Stochastic Differential Equations Solver

Linear Stochastic Differential Equations are an important particular case of SDE. We implement a set of routines to simulate trajectories of a particular type of linear stochastic differential equations. For details on its usage see the library documentation [17]. The general form of a n-dimensional linear SDE of this type with d sources of randomness is

$$dX(s) = [A(s)X(s) + a(s)]ds + \sigma(t)dW(s) \qquad X(t_0) = X_0 \qquad X_0 \in \mathbf{R}^n \tag{3}$$

where $A(t), a(t)$ and $\sigma(t)$ are continuous $n \times n, n \times 1$, and $n \times d$ valued matrices respectively, and $W(t)$ is a d-dimensional Brownian motion.

A detailed explanation of the implemented method is given below. It is known that the exact solution of (3) equation is

$$X(t) = \Phi(t)\left[x + \int_{t_0}^{t} \Phi^{-1}(s)a(s)ds + \int_{t_0}^{t} \Phi^{-1}(s)\sigma(s)dW(s)\right]$$

where $\Phi(t) = (\Phi_1(t), \Phi_2(t), \ldots, \Phi_n(t))$ are the solution of the system of ordinary differential equations

$$\Phi_i'(t) = A(t)\Phi_i(t)ds \qquad \Phi(t_0) = e_i \qquad \text{for } 1 \leq i \leq n \tag{4}$$

, where e_i is the i-esim vector of the canonical base. A formal proof of the above result is given in [7]. It is also known that

$$\delta(t) = \int_{t_0}^{t} \Phi^{-1}(s)\sigma(s)dW(s)$$

is a Gaussian process with mean 0, and covariance

$$V[\delta(t)] = \int_{t_0}^{t} \Phi^{-1}(s)\sigma(s)(\Phi^{-1}(s)\sigma(s))'ds$$

Then

$$h(t) = \int_{t_0}^{t} \Phi^{-1}(s)a(s)ds + \int_{t_0}^{t} \Phi^{-1}(s)\sigma(s)dW(s)$$

is also Gaussian and has mean

$$E[h(t)] = \int_{t_0}^{t} \Phi^{-1}(s)a(s)ds$$

and covariance

$$V[h(t)] = \int_{t_0}^{t} \Phi^{-1}(s)\sigma(s)(\Phi^{-1}(s)\sigma(s))'ds$$

In order to simulate $X(t)$ we just have to calculate

$$X(t) = \Phi(t)[X_0 + N_h] \tag{5}$$

where N_h is a realization of $h(t)$. Thus the simulation of $X(t)$ is reduced to to estimate $\Phi(t)$, $E[h(t)]$ and $V[h(t)]$.

To calculate the integral $I(t) = \int_{t_0}^{t} f(s)ds$ is equivalent to solve the initial value problem

$$dI(t) = f(t)dt \qquad I(t_0) = 0$$

where $f : [t_0, t] \longrightarrow \mathbb{R}^n$. Hence $E[h(t)]$ is the solution of

$$dE[h(t)] = \Phi^{-1}(t)a(t)dt \qquad E[h(t)] = \mathbf{0} \tag{6}$$

and $V[h(t)] = [V_1[h(t)], V_2[h(t)], \ldots, V_n[h(t)]]$ are the solutions of the system of ordinary differential equations

$$dV_i[h(t)] = c_i(t)dt \qquad V_i[h(t_0)] = \mathbf{0} \qquad \text{for } 1 \leq i \leq n \tag{7}$$

where $c_i(t)$ is the i-esim column of $\Phi^{-1}(t)\sigma(t)(\Phi^{-1}(t)\sigma(t))'$.

$\Phi(t)$, $E[h(t)]$ and $V[h(t)]$ can be calculated simultaneously solving a system of ordinary differential equations that contains the systems defined in (4), (6) and (7). The system to be solved is

$$\begin{bmatrix} d\Phi_1(t) \\ d\Phi_2(t) \\ \vdots \\ d\Phi_n(t) \\ dE[h(t)] \\ dV_1[h(t)] \\ dV_2[h(t)] \\ \vdots \\ dV_n[h(t)] \end{bmatrix} = \begin{bmatrix} A(t)\Phi_1(t) \\ A(t)d\Phi_2(t) \\ \vdots \\ A(t)d\Phi_n(t) \\ \Phi^{-1}(t)a(t) \\ c_1(t) \\ c_2(t) \\ \vdots \\ c_n(t) \end{bmatrix} \qquad \begin{bmatrix} \Phi_1(t_0) \\ \Phi_2(t_0) \\ \vdots \\ \Phi_n(t_0) \\ E[h(t_0)] \\ V_1[h(t_0)] \\ V_2[h(t_0)] \\ \vdots \\ V_n[h(t_0)] \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} \tag{8}$$

Once this system is solved, the estimation of $X(t)$ is straightforward using equation (5).

6 Stochastic Differential Equations with multiplicative Noise

We consider the class of Stratonovich stochastic differential equations where the noise term is a function of the state variables. That is

$$dX(s) = a(X, s)ds + \sigma(X) \circ dW(s) \qquad X(t_0) = X_0 \qquad X_0 \in \mathbb{R}^n \tag{9}$$

For this type of equations a Wong-Zakai type numerical solution method was developed and implemented. Wong-Zakai type approximations were first introduced in [19]. The proposed method will be explained below.

Assume that we wish to approximate the above equations at points $t_0 = 0 < t_1 < t_2 < \cdots < t_k = T$ of the interval $[0, T]$. Let X_j be the numerical approximation of $X(t_j)$. For each subinterval $[t_j, t_{j+1}]$,$j = 0, 1, \ldots, k-1$, the solution of equation (9) will be approximated by the solution of the following ordinary differential equation:

$$\frac{dX^*}{dt} = a(X^*, t) + \frac{1}{\sqrt{t_{j+1} - t_j}}\sigma(X^*)Z \qquad X^*(t_j) = X_j \tag{10}$$

where Z is a realization of a Gaussian distributed variable with mean 0 and variance the identity matrix. From the solution of this equation we get $X_{j+1} = X^*(t_{j+1})$. Following this scheme we get all the X_j, $j = 1, \ldots, k$. A detailed explanation of this numerical method and a formal proof of its correctness are given in [12]. For further information on the usage of the implemented methods see the library documentation [17].

When developing numerical algorithms it is necessary to check its accuracy through numerical experiments. Some experiments where made to get an idea of the behavior of the numerical method derived during the research. These numerical experiments can be seen in [18].

In order to illustrate the type of experiments that were run, an example will be shown. Consider a one-dimensional nonlinear problem whose Stratonovich form is,

$$dy = -\alpha(1 - y^2)dt + \beta(1 - y^2) \circ dW(t), \qquad y(0) = 0.5, \qquad t \in [0, 1] \tag{11}$$

According to [8] the exact solution of this equation is

$$y(t) = \frac{(1 + y_0)\exp(-2\alpha t + 2\beta W(t)) + y_0 - 1}{(1 + y_0)\exp(-2\alpha t + 2\beta W(t)) - y_0 + 1}$$

We compute the mean absolute error

$$M(h) = \frac{1}{K}\sum_{i=0}^{K} |y_N^{(i)} - y^{(i)}(t_N)|$$

for 2000 trajectories, $\alpha = -5$ and different β, ranging from 1 to 5. Table 1 presents the simulation results for the first test equation. The mean errors are very small in contrast to those reported in [16], where they use Runge-Kutta type methods. This results suggest that the method obtains the exact solution.

Table 1: Error and convergence rate of (11) for $\alpha = -5.0$ and differtent β and with $h = 1$

β	$M(h)$
1	6.72943e-12
2	9.45192e-12
3	7.41739e-11
4	8.03285e-11
5	8.57892e-11

The results obtained until now suggest that the proposed method is very promising. Future research include the extension of this method to the general case (where there is no restriction on the noise term) and the development of new methods.

References

[1] F. Anderson, et al. LAPACK User's Guide. Society for Industrial and Applied Mathematic. Second Edition. 1995.

[2] Basic Linear Algebra Subprograms Technical Forum Standard. 2001. `http://www.netlib.org/blas/blast-forum`

[3] K. Burrage, P.M. Burrage and T. Tian. Numerical Methods for Strong Solutions of Stochastic Differential Equations: an Overview. Proceedings of the Royal Society of London. Series A 460, 373-402, 2004.

[4] P.M. Burrage. Vectorised simulations for stochastic differential equations. In Jagoda Crawford and A.J. Roberts, editors, *Proc. of 11th Computational techniques and Applications Conference CTAC-2003*, volume 45, pages C350-C363, jun 2004. http://anziamj.austms.org.au/V45/CTAC2003/Burr.

[5] S. Cohen and A. Hindmarsh. CVODE, a Stiff/Nonstiff ODE Solver in C. Computers in Physics, 10(2), March-April 1996, pp 138–143.

[6] G.K. Davis R. Sacks-Davis and P.E. Tischer. A Review of Recent Developments in Solving ODEs. ACM Computing Surveys,Vol. 17, March 1985, pp. 5–47

[7] I. Karatzas, S.E. Shreve. Brownian Motion and Stochastic Calculus, 2 edition. Graduate Texts in Mathematics. Springer, 1997.

[8] P. Kloeden and E. Platen. Numerical Solution of Stochastic Differential Equations, 2 edition. Springer, 1995.

[9] P. L'Ecuyer, R. Simard, E. J. Chen, and W. D. Kelton. An Objected-Oriented Random-Number Package with Many Long Streams and Substreams. Operations Research, 50, 6 (2002), pp 1073–1075.

[10] P. L'Ecuyer. Good Parameter and implementations for combined multiple recursive random number generators. Operations Research, 47, 1 (1999), pp 159–164.

[11] P. L'Ecuyer. Random Number Generation. Chapter 4 of the Handbook on Simulation, Jerry Banks Ed., Wiley, (1998), pp 93–137.

[12] J. A. Londoño. A Wong-Zakai type approximation for solutions of stochastic differential equations. Working paper.

[13] M. Matsumoto and T. Nishimura. Dynamic Creation of Pseudorandom Number Generators. Monte Carlo and Quasi-Monte Carlo Methods 1998, Springer, 2000, pp 56–69.

[14] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Transaction on Modeling and Computer Simulation, 8 (1998), 3–30.

[15] E. Platen, An introduction to numerical methods for stochastic differential equations, Acta. Numer., 8 (1999), pp. 197246.

[16] T.H. Tian and K. Burrage. Two-stage stochastic Runge-Kutta methods for stochastic differential equations. BIT. 2002, Vol. 42, No. 3, pp 625643.

[17] A.M. Villegas and J.A, Londoño. CTSS - library documentation.

[18] A.M. Villegas and J.A, Londoño. CTSS - simulation examples.

[19] E. Wong and M. Zakai. On the relation between ordinary and stochastic differential equations. Int. J. Engin. Sci. , 3 (1965) pp. 213229.