Available online at www.sciencedirect.com

ScienceDirect

SoftwareX 5 (2016) 12–15

SoftwareX

www.elsevier.com/locate/softx

xSPDE: Extensible software for stochastic equations

Simon Kiesewetter, Rodney Polkinghorne, Bogdan Opanchuk, Peter D. Drummond*

Swinburne University of Technology, Melbourne, Victoria 3122, Australia

Received 19 December 2015; accepted 10 February 2016

Abstract

We introduce an extensible software toolbox, xSPDE, for solving ordinary and partial stochastic differential equations. The toolbox makes extensive use of vector and parallel methods. Inputs are exceptionally simple, to reduce the learning curve, with default options for all of the many input parameters. The code calculates functional means, correlations and spectra, checks for errors in both time-step and sampling, and provides several choices of algorithm. Most aspects of the code, including the numerical algorithm, have a modular functional design to allow user modifications.

Keywords: Stochastic software code-generator

Code metadata

Current code version	v1.04
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-15-00087
Legal Code License	MIT license
Code versioning system used	git
Software code languages, tools, and services used	Matlab
Compilation requirements, operating environments & dependencies	Parallel toolbox is preferred, but optional
If available Link to developer documentation/manual	http://xspde-matlab.readthedocs.org
Support email for questions	peterddrummond@gmail.com

1. Motivation and significance

Stochastic differential equations or SDE's are equations with random noise terms [1–3]. They are widely used in many fields, including biology, chemistry, physics, engineering, economics, meteorology and other disciplines. There are very few publicly available, general purpose software packages available to solve them, especially when generalized to stochastic partial differential equations. xSPDE is a Matlab based software toolbox that numerically solves ordinary and partial differential cases of stochastic equations, and graphs the results of correlations and averages. It uses both vectorization and core parallelism for speed. Its object-oriented modular design makes it easy to extend and modify.

The stochastic partial differential equations [4] or SPDEs solved by the program are defined in one time dimension and up to three space dimensions. The xSPDE toolbox is mainly designed to treat Stratonovich equations [2], which are the broad-band limit of a finite band-width random noise equation, but can be used for Ito equations as well. An ordinary or partial stochastic differential equation [5,6] for a real or complex vector field $\mathbf{a}$ is:

$$\frac{\partial \mathbf{a}}{\partial t} = \mathbf{A}\left[\mathbf{a}\right] + \underline{\mathbf{B}}\left[\mathbf{a}\right] \cdot \boldsymbol{\zeta}(t) + \underline{\mathbf{L}}\left(\nabla\right) \cdot \mathbf{a}. \tag{1}$$

Here, $\mathbf{A}$ is a vector or vector field, $\underline{\mathbf{B}}$ a matrix, and $\boldsymbol{\zeta}$ is a real noise vector, delta-correlated in time. In the ordinary

* Corresponding author.
E-mail address: peterddrummond@gmail.com (P.D. Drummond).

http://dx.doi.org/10.1016/j.softx.2016.02.001

differential equation case,

$$\left\langle \zeta_i\left(t\right)\zeta_j\left(t'\right)\right\rangle = \delta\left(t-t'\right)\delta_{ij}. \tag{2}$$

For stochastic partial differential equations, $\underline{\mathbf{L}}\left[\nabla\right]$ is an arbitrary matrix of linear terms and derivatives, diagonal in the vector field component indices, and $\boldsymbol{\zeta} = \left[\boldsymbol{\zeta}^x, \boldsymbol{\zeta}^k\right]$ are real delta-correlated noise fields:

$$\begin{aligned}\left\langle \zeta_i^x\left(t,\boldsymbol{x}\right)\zeta_j^x\left(t',\boldsymbol{x}'\right)\right\rangle &= \delta\left(\boldsymbol{x}-\boldsymbol{x}'\right)\delta\left(t-t'\right)\delta_{ij}\\ \left\langle \zeta_i^k\left(t,\boldsymbol{k}\right)\zeta_j^k\left(t',\boldsymbol{k}'\right)\right\rangle &= f(\mathbf{k})\delta\left(\boldsymbol{k}-\boldsymbol{k}'\right)\delta\left(t-t'\right)\delta_{ij}.\end{aligned} \tag{3}$$

Here $f(\mathbf{k})$ is an arbitrary momentum filter, allowing the treatment of noise with a finite correlation distance. Transverse boundary conditions are currently assumed periodic, although this can be changed due to the modular design.

Typically such equations are extensively used to treat environmental and thermal noise, quantum noise in quantum simulations, and also random effects due to the statistics of discrete particle numbers in birth–death processes, which are found in such disciplines as biology and chemistry.

The significance of xSPDE is its ease of use and adaptability. It has a modular functional design, with an architecture that can easily be extended to other languages in future. The use of extensive, customizable default settings greatly reduces the learning curve. It has interactive and batch modes with compact inputs, and calculates both time-domain and space-domain spectra. It has user-definable features including the algorithm itself. A widely used C++ based toolbox called XMDS [7,8] for solving problems like these is also available. This covers different user requirements, and has less end-user customization.

2. Software description

The main code is a short function called **xspde**. This calls the simulation function **xsim**, then the graphics function **xgraph**. These are also available separately, which is useful for batch mode operation. The simulation function is modular, with 20 different functional components that are easily replaced or modified. On completion, timing and errors are printed.

The xSPDE toolbox permits any definition of the differential and linear functions in the original equations, as well as choices of integration algorithms. Simulations can be carried out in a sequence, to simulate the various stages in an experiment, with general transfer functions and noise inputs.

It calculates averages of any function of complex or real field vectors, with Fourier transforms in both time and space if required. Error estimates due to the finite time-step and sample size are both provided as standard features.

2.1. Data inputs and outputs

To explain the data flow, simulation inputs are stored in the `input` cell array, which describes a *sequence* of simulations, so that `input` = {`in1, in2, ...`}. Each structure `in` has an output which is the input of the next one. Averages over stochastic trajectories are recorded sequentially in the `data` cell array, while raw trajectory data is stored in the `raw` cell array if required. If there is one simulation in the sequence, just one structure is needed. Outputs can be stored and graphed later, by giving a filename. All averages and raw data results can be stored in either Matlab (.mat) or standardized HDF5 (.h5) formats.

2.2. Error control

The final output graphs and data will have error-bars if error checking is specified, which is also the default parameter setting. This is to make sure the final results are accurate. There is a clear strategy if the errors are too large. Either increase the time points or the number of steps between the time-points. The algorithm and extrapolation order can also be changed to improve errors.

Sampling error estimation in xSPDE uses sub-ensemble averaging. Ensembles are specified in three levels. The first level is calculated efficiently using a parallel vector of trajectories. By the central limit theorem, these sample averages are distributed as a normal distribution at large sample number. This approach is limited by available memory.

Next, the sample averages are averaged over two higher level ensembles, whose variance is used to estimate a standard deviation in the mean, since each computed quantity is now a normally distributed result. The next higher level ensemble is carried out in series to minimize memory use, while the highest uses parallel simulations over separate cores or CPUs in a cluster.

3. Illustrative examples

3.1. Example of an SDE

The simplest type of stochastic equation is a random walk, so that $\partial a/\partial t = \zeta(t)$, which has an xSPDE implementation in one line of code:

```
in.da = @(a,z,r) z; xspde(in);
```

Here the notation `in.`[*label*] = [*parameter*] adds a parameter value with a particular label to the structure `in.`, while the notation `@(a,z,r)` is the Matlab notation for an inline or anonymous function, in this case the derivative da/dt, labeled `in.da`. All parameters have default values, and most do not need to be entered. To give a standardized notation, `a` is the stochastic field, `z` the random noise, and `r` is a structure which holds the simulation time, `r.t`, the coordinates of the spatial grid points, `r.x`, `r.y`, `r.z`, and other parameters.

Much greater complexity than this is certainly possible. As another example, frequency spectra have many uses, especially for understanding the steady-state fluctuations of any stochastic physical system. To achieve this, one simply includes the parameter, `in.transforms=1,` to Fourier transform the field over time, giving the spectrum. Ensemble averaging is achieved by including a value for the parameter `in.ensembles`. A random initial distribution is available on defining the function `in.initial`, whose first argument is a random field.

3.2. Example of an SPDE

A typical SPDE is the stochastic Ginzburg–Landau (SGL) equation. In this example, it has one time dimension and two space dimensions, for a total of $d = 3$. It describes symmetry breaking: the system develops a spontaneous phase which can vary spatially as well. The model is widely used to describe lasers, magnetism, superconductivity, superfluidity and even particle physics:

$$\partial a/\partial t = \left(1 - |a|^2\right) a + b\zeta(x,t) + c\nabla^2 a \tag{4}$$

where

$$\left\langle \zeta(x,t)\zeta^*(x',t')\right\rangle = 2\delta\left(t - t'\right)\delta\left(x - x'\right). \tag{5}$$

Solving the SGL equation for $b = 0.001$ and $c = 0.01i$ requires the short script given below, which specifies 10 steps per plotted point for good accuracy, together with 6 movie images to analyze the dynamics in three dimension. The output intensity, $|a|^2$ as a function of time for a single random trajectory is graphed in Fig. 1.

```
in.noises = 2;
in.dimension = 3;
in.steps = 10;
in.linear = @(D,r) i*0.01*(D.x.^2+D.y.^2);
in.observe = @(a,~) abs(a).^2;
in.images = 6;
in.olabels = '|a|^2';
in.da = @(a,z,r) (1-abs(a(1,:)).^2).*a...
     +0.001*(z(1,:)+i*z(2,:));
xspde(in)
```

Here `D.x` is an x-derivative, evaluated using Fourier transforms, and used by the linear response function, `in.linear`. In Matlab syntax, to multiply vectors element-wise, like $a_i = b_i c_i$, the notation `a=b.*c` is used. Fields in any dimension are matrices in xSPDE. The first index is the field component, while the second index indicates different trajectories and lattice locations that can be treated in parallel. The notation `a(1,:)` means that the operation is repeated over all values of the second index, which is the lattice/ensemble index.

4. Impact

The problems solved by xSPDE include many commonly encountered cases of ordinary and partial differential stochastic equations. These equations have uses in biology, chemistry, physics, economics, engineering and meteorology. They are used to model quantum noise and/or environmental noise, all of which have important applications. For greater speed, the design makes it simple to replace components with more specialized GPU-based code [9].

The main utility of the program is its agility. A small number of inputs are required for each new problem, which speeds up

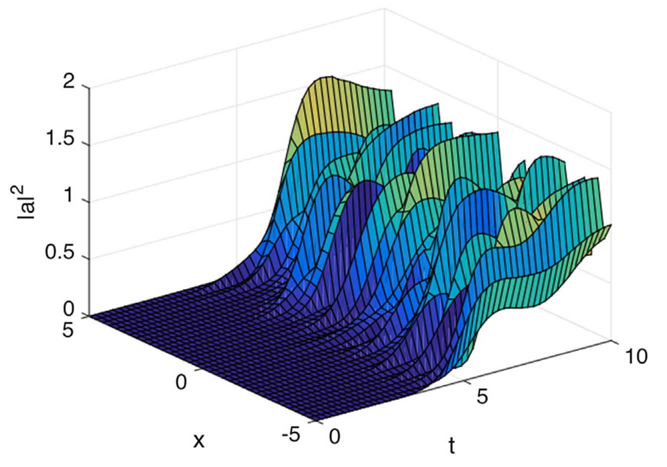

Fig. 1. Simulation of an SGL trajectory in two dimensions, projected onto $y = 0$. Colors correspond to the value of $|a|^2$ and only added for the sake of expressiveness.

development time, and reduces the need for extensive testing in different applications. A previous package, xMDS, has had over 40,000 downloads to date. The improvement provided by xSPDE is its ease of use and greater modularity, allowing simple end-user modification even beyond the extensive functionality already included.

The underlying code body is well-tested. It comes with a set of 19 illustrative benchmark equations, each with analytic solutions. There are default settings for every type of program input, so that the learning curve is minimal. Estimates of time-step and sampling errors give an accuracy bound for any result, apart from lattice discretization error estimates. The toolbox is mainly used by graduate students in computational physics, but is not limited to physics.

4.1. Interactive and batch applications

The interactive use of the xSPDE program can provide a rapid turn-around. This is useful when learning the toolbox, choosing parameter values, or for error control. In the interactive mode, xSPDE inputs and functions are entered directly into the Matlab command window for immediate feedback.

One important advantage of this approach is the speed of prototyping, due to the reduced time required to test and code. The amount of user code required to analyze a new research problem is usually no more than a single page, compared to thousands of lines of code in a hand-coded program with this functionality. In 2015, it was used interactively at the ANZSUP physics summer school, resulting in a 40 student class each coding and solving several unseen, challenging problems in less than four hours, having never used xSPDE previously.

For larger, more time-critical research projects, batch use is routine, and the xSPDE code is unchanged apart from adding a file-name for data storage. Operation over multiple cores and nodes on parallel clusters of any size is obtained simply by adding a single parameter to the `in.ensembles` vector. To adapt the basic functionality to a new user requirement, one can redefine any function or structure name to augment the defaults.

4.2. *Open object-oriented architecture*

The overall data structure is object-oriented, including data and methods. Yet xSPDE has a more open architecture than the usual object-oriented style of programming, which is a deliberate choice to allow extensibility. As an example, in some physics applications it is important to normalize the field after each time-step. In xSPDE, one simply redefines the default time-step function, `xMP`, by adding `in.step = @normstep`, together with a new function definition:

```
function b = normstep(a,z,dt,r)
b = xMP(a,z,dt,r);
norm = sqrt(xint(abs(b).^2,r.dx, r));
b = b/norm;
end
```

5. Conclusion

We have described an open toolbox for integrating stochastic equations with a highly modular and extensible architecture, having applications in many fields.

Acknowledgments

We acknowledge input from R.Y. Teh, K. Ng, L. Rosales-Zarate, R.R. Joseph, J.J. Hope, and students of the 2015 ANZSUP summer school.

References

[1] Arnold L. Stochastic differential equations: theory and applications. reprint ed. Folens Publishers; 1992.
[2] Gardiner CW. Handbook of stochastic methods. 2nd ed. Berlin: Springer-Verlag; 1985.
[3] Drummond PD, Hillery M. The quantum theory of nonlinear optics. Cambridge: Cambridge University Press; 2014.
[4] Werner MJ, Drummond PD. J Comput Phys 1997;132(2):312–26. http://dx.doi.org/10.1006/jcph.1996.5638.
[5] Drummond PD, Mortimer IK. J Comput Phys 1991;93:144–70.
[6] Kloeden PE, Platen E. Numerical solution of stochastic differential equations. Berlin: Springer-Verlag; 1992.
[7] Collecutt GR, Drummond PD. Comput Phys Comm 2001;142:219–23.
[8] Dennis GR, Hope JJ, Johnsson MT. Comput Phys Comm 2013;184: 201–18.
[9] Januszewski M, Kostur M. Accelerating numerical solution of stochastic differential equations with {CUDA}. Comput Phys Comm 2010;181(1): 183–8. http://dx.doi.org/10.1016/j.cpc.2009.09.009. URL http://www.sciencedirect.com/science/article/pii/S0010465509002999.