Python Solver for Stochastic Differential Equations

BY CHU-CHING HUANG

Abstract
Alternate to proprietary CAS, Matlab and Maple, open source computer scripting language, Python and its applications have potential for doing professional computational work. This article describes a Python module for solving stochastic differential equations (SDEs): PyS³DE avails a) symbolic solvers for SDEs; b) stochastic numerical schemes, Euler and Milstein methods; c) and visualization tools for calculated data and Feynman-Kac simulation etc..

Although PyS³DE can run almost operation system in which Python environment is well installed, an intelligent application, TEXMACS, introduced here to present more interactive communication between Python and graphical user interface (GUI). PyS³DE can also run on the Python-based Sage which supports it doing computation through Internet by Sage’s “notebook()”.

Keywords: Stochastic Differential Equations (SDEs), PyS³DE, Euler and Milstein Schemes, TEXMACS, Sage

1 Introduction

Differential equations are used to illustrate the evolution of a system; stochastic differential equations (SDEs) arise when a random noise (or white noise) is introduced into differentiation equations. White noise process, \( X_t \), is formally defines as the derivatives of the Brownian motion, \( B(t) \):

\[
X_t = \frac{d}{dt}B(t)
\]

Based on I\(\text{ATEX}\) and Guile.

It stands for phenomenon of the violent changes in a system or un-predictable observed data. The theory of SDEs is popular in both theoretical and applied science.

However, white noise does not exist as a function of \( t \) in classical analysis since the trajectory of a Brownian motion which is nowhere differentiable.

During middle of last century, Japanese mathematician K. Itô and Russian mathematician Stratanovich established the theory of stochastic calculus which could compute calculus over such process and its generalization. Too theoretical approach about stochastic calculus arises the learning curve of this field. There are also some researchers who tried to develop some computer package able to do stochastic calculus. However, there are still problems here: either unpopular of the development language (Axiom), or proprietary software (Maple). Under such consideration, we decide to find another clear and reliable approach to develop computer package able to do stochastic calculus and solve SDEs etc.

SDE

Stochastic differential equations (SDE’s) play an important role in stochastic modeling. For example: in economics, solutions of the SDE’s are used to model share prices; in biology, solutions of SDE’s describe sizes of populations.

As one of computer laguages, Python owns exclusive features: it is an open-source object-oriented interpreter language with a large number of libraries, or extensions, for instance: interactive computing, numerical analysis, visualization etc. Except its native Python library, Python can also be accessible in library made from Fortran, C/C++ program.

These days, all quantitative work in scientific research involves calculations on a computer for the work:

a) Solving models (optimization, etc.);
b) Estimation, data manipulation;
c) Simulation;
d) Visualization of results.

minimum

Some simple calculations can be done with desktop application; but computation with enormous manipulation requires more advanced programming environment. As one of computer scripting languages, Python with other packages avails several features we need:

- clean syntax easy to work with;
- Expressions, conditions, loops, etc.
- Data types (numbers, lists, etc.)
- Program structure (functions, objects, modules, etc.)
- Numerical methods (Scipy/Numpy):
  - Working with arrays;
  - Linear algebra;
  - Random number generation;
  - Numerical integration and numerical optimization etc;
- Symbolic calculation (Sympy, Sage): aand user friendly authoring environment
  - Basic Calculus operation;
  - Solvers for Ordinary differential equations (ODEs) and Algebraic equation minimums;
  - Mathematics Expression
- Visualization (Matplotlib, Sage):
  - 2D/3D pictures;
  - animation and movie etc

where quoted texts in emphasized form above are the additional extended Python modules we need in our project needed.

Python is a general purpose, high level language but all the advantages above make it popular in scientific community and academic environment. Python code is written in plain text format and Python distributions contain a simple text editor, idle. We can program and run Python code by its default editor. Once we want to reuse the programming result and reproduce them into other technique format, pdf or \LaTeX for example, such kind of application can not afford it. Furthermore, additioanal documentation is required if we want to put resulted data together with documentation.

To establish handy environment of routine documentation work after the computing, \TeXMACS would be introduced here. Unlike Python and Sage, \TeXMACS does not provide any computing functionality, the scientific publishing system what-you-see-is-what-you-get (WYSIWYG) publishing system \TeXMACS offers an intelligent what-you-see-is-what-you-get (WYSIWYG) system based on \LaTeX and Guile. \TeXMACS is optimized for efficient typesetting system for mathematical documents: friendly \LaTeX front-end interface and intelligent plug-ins interface for running CAS within \TeXMACS, for example, Maxima, Sage and Python etc. This exclusive plug-ins feature actually shortens the development time in our project. This paper is prepared and completed by \TeXMACS.
It is frequently the case that economic or financial considerations will suggest that a stock price, exchange rate, interest rate, or other economic variable evolves in time according to a stochastic differential equation of the form:

$$dX_t = a(t, X_t)dt + b(t, X_t) dW_t$$

2 Stochastic Differential Equations

2.1 Itô and Stratonovich SDEs

According to the different formulation about differentiation over white noise, there are different mathematical meaning. Itô version uses left rule to present the differential term of Brownian motion, $dB_t$, with respect to the middle rule for Stratonovich version. We use different expressions for two different kinds of one dimensional SDEs:

a) Itô version:

$$dX_t = a(t, X_t)dt + b(t, X_t) dW_t$$

b) Stratonovich version:

$$dX_t = \tilde{a}(t, X_t)dt + b(t, X_t) \circ dW_t$$

Both the solutions, $X_t$, are the same with the relation:

$$\tilde{a}(t, x) = a(t, x) - \frac{1}{2} b(t, x) \frac{\partial b}{\partial x}(t, x)$$

Stratonovich SDE uses midpoint rule to stochastic integration and has the result just like a deterministic calculus, for instance

$$\int_0^T W_t \circ dW_t = \frac{1}{2} W_T^2$$

Itô SDE uses left sum rule to define stochastic integral and has the result:

$$\int_0^T W_t dW_t = \frac{1}{2} W_T^2 - \frac{1}{2} T$$

To solve SDEs, linear type can be solved directly by Itô rule. More complicated case, SDE can be transformed from Itô SDE into Stratonovich SDE such that can be solved by the rule as in deterministic calculus.

2.2 Solvability of SDEs

The theory about existence and uniqueness of solution of linear SDEs can be found in the studies, (Øksendal, Klebaner). First type of SDE is the following linear case:

$$dX_t = (a_1(t)X_t + a_2(t))dt + (b_1(t)X_t + b_2(t)) dW_t$$

By method of integration factor, the solution of this linear SDE is:

$$X_t = \Phi_{t_0, t} \left( X_{t_0} + \int_{t_0}^t (a_2(s) - b_1(s)b_2(s))\Phi_{t_0, s}^{-1} ds + \int_{t_0}^t (b_2(s))\Phi_{t_0, s}^{-1} dW_s \right)$$

where the integration factor is:

$$\Phi_{t_0, t} = \exp \left( \int_{t_0}^t \left( a_2(s) - \frac{1}{2} b_1^2(s) \right) ds + \int_{t_0}^t b_1(s) dW_s \right)$$
This is one the fundamental implement of solver for SDE in our Python package, PySDE.

Other solvable SDEs we consider are discussed here. By method of integration factor, the solution of this linear SDE is:

Consider the case:

\[ dX_t = X_t \, dt + X_t \, dW_t, \]

with initial \( X_0 = x \in \mathbb{R}. \)

The solution is \( \exp (W_t + t/2). \) From the Stratonovich approach, above SDE becomes:

\[ dX_t = \frac{1}{2} X_t \, dt + X_t \circ dW_t \]

Since solving stratonovich SDE is like the way in classical ordinary differential equation, solve last SDE by general method of separating variables:

\[ \frac{1}{X_t} dX_t = \frac{1}{2} dt + 1 \circ dW_t = \frac{1}{2} dt + dW_t \]

\[ \Rightarrow X_t = x + \exp \left( \frac{1}{2} t + W_t \right) \]

Certain types of SDEs can also be solved by such Stratonovich stochastic integration. Consider the following Stratonovich SDE:

\[ dX_t = b(X_t) \circ dW_t \]

Directly integrating SDE obtains the solution as follows:

\[ X_t = \psi^{-1}(W_t + \psi(x)) \]

where

\[ \psi(x) = \int_x^z \frac{ds}{b(s)} \] and \( X_0 = x \)

And this Stratonovich SDE is equivalent to the following Itô SDE:

\[ dX_t = \frac{1}{2} b(X_t) b' X_t \, dt + b(X_t) \, dW_t \]

This means this class of Itô SDE is solvable.

Next another solvable Stratonovich SDE is as follows:

\[ dX_t = \alpha(t) b(X_t) dt + b(X_t) \circ dW_t \]

And its solution is

\[ \frac{1}{b(X_t)} dX_t = \alpha(t) dt + 1 \circ dW_t \]

\[ \downarrow \]

\[ \psi(X_t) - \psi(X_0) = \int_t^t \alpha(s) ds + W_t \]

\[ \downarrow \]

\[ X_t = \psi^{-1} \left( \int_t^t \alpha(s) ds + W_t + \psi(X_0) \right) \]

where \( \psi \) and \( X_0 \) are the same as defined in previous case. In other words, it is the solution of the following Itô SDE:

\[ dX_t = \left( \alpha(t) b(X_t) + \frac{1}{2} b(X_t) b' X_t \right) dt + b(X_t) \, dW_t \]

Example 1. (SDE Solver on Python Shell)

Consider the following SDEs: where the integration factor is

\[ dX_t = X_t^- \circ dW_t, \quad X_0 = x \quad (4) \]

\[ dX_t = \alpha X_t^2 \, dt + X_t^2 \circ dW_t, \quad X_0 = x \quad (5) \]
where \( m \) is an positive integer. We directly use sympy's ODE solver to solve first case and use it's integration to solve second case:

```python
from sympy import *
from sympy.abc import t,x,k,N,m,C
X = Function("X")
W =Symbol("W")
solve the problem
a=Symbol("alpha")
b = X(W)**m
X_prime=Derivative(X(W), W)
dsolve(X_prime-b, X(W))

X(W) == (C1 + W - W*m)**(1/(1 - m))
dsolve(X_prime-b, X(W),hint='best')

X(W) == (C1 + W - W*m)**(1/(1 - m))
fac=a*t+W
bb=x*x
Eq=integrate(1/bb,x)+C
print Eq-fac

C - W - alpha*t - 1/x
sol=solve(Eq-fac,x)
sol[0].subs({x:X(W)})

1/(C - W - alpha*t)
```

**Example 2.** Consider the SDE:

\[
dX_t = \left(\frac{2X_t}{1+t} - a(1+t)^2\right)dt + a(1+t)^2dW_t
\]

where \( t(0)=t_0, X_0=x_0 \) and \( a > 0 \). By the pysde, the resulting solution is:

```python
from sys import path
path.append(".")

None

from sde import *
x,dx,w,dw,t,dt,a=symbols('x dx w dw t dt a')
x0 =Symbol('x0'); t0 = Symbol('t0')
drift=2*x/(1+t)-a*(1+t)**2; diffusion=a*(1+t)**2
sol=SDE_solver(drift,diffusion,t0,x0)
pprint(sol)

2 / 2 2 2
(1 + t) *\x0 + a*t0*(1 + t0) + a*w*(1 + t0) - a*t*(1 + t0) /
```

Stochastic Differential Equations
None

\begin{verbatim}
2
(1 + t0)
\end{verbatim}

\textbf{Example 3.} Consider another SDE:

\[ dX_t = -\left(\frac{2X_t}{1-t}\right)dt + \sqrt{t(1-t)}dW_t \]

where \(0 \leq t < 1\) and \(X_0 = x_0\). By pysde, the resulting solution is a Gaussian process:

\begin{verbatim}
import pysde

# Define drift and diffusion
drift = -2*x/(1+t)
diffusion = sqrt(t*(1-t))

# Solve the SDE
sol = SDE_solver(drift, diffusion, t0, x0)

# Print the solution
pprint(sol)
\end{verbatim}
\( \mathcal{N}(\cdot) \) is a class which is defined in sde module and represents the Gaussian process.

There are several solvable SDEs and its solution listed as follows:\(^{(Oksendal)}\)

\( a) \quad dX_t = \gamma dt + b X_t dW_t \) where \( a, b \) are real constants:

\[
X_t = F_t^{-1} X_0 + F_t^{-1} \int_0^t \gamma F_s \, ds
\]

with integrating factor

\[
F_t = \exp \left( -b W_t + \frac{1}{2} b^2 t \right)
\]

\( b) \quad dX_t = \gamma (t, X_t) \, dt + b(t) X_t dW_t \): Define

\[
Y_t(\omega) = F_t(\omega) X_t(\omega)
\]

where integrating factor

\[
F_t = \exp \left( -\int_0^t b(s) \, dW_t + \frac{1}{2} \int_0^t b^2(s) \, ds \right)
\]

Then \( Y_t(\omega) \) satisfies the ordinary differential equation:

\[
dY_t(\omega) = F_t(\omega) \cdot \gamma(t, F_t^{-1}(\omega)Y_t(\omega)), \quad Y_0 = x
\]

and \( X_t = F_t^{-1} Y_t \).

**Example 4.** Consider

\[
dX_t = \frac{1}{X_t} dt + b X_t dW_t, \quad X_0 = x > 0
\]

Let

\[
F_t = \exp \left( -b W_t + \frac{1}{2} b^2 t \right)
\]

and

\[
\frac{dY_t(\omega)}{dt} = \frac{F_t}{X_t} = \frac{F_t^2}{Y_t},
\]

\[
\Rightarrow \quad Y_t^2(\omega) = x^2 + 2 \int_0^t \exp(-2b W_s + b^2 s) \, ds
\]

\[
\Rightarrow \quad X_t(\omega) = F_t^{-1}(\omega) Y_t(\omega)
\]

\[
= \exp \left( b W_t - \frac{1}{2} b^2 t \right) \sqrt{x^2 + 2 \int_0^t \exp(-2b W_s + b^2 s) \, ds}
\]

\( c) \quad dX_t = r X_t (K - X_t) \, dt + b X_t dW_t, \quad X_0 = x > 0, \ K > 0 \) and \( r, b \in \mathbb{R} \) [Gard]:

Define integrating factor, \( F_t \), as follows:

\[
F_t = \exp \left\{ \left( -r K + \frac{1}{2} b^2 \right) t - b W_t \right\}
\]

Then the SDE of \( Y_t = F_t X_t \) satisfies the following ODE:

\[
dY_t = F_t dX_t + X_t dF_t + [X_t, F_t]
\]

\[
= F_t[r X_t (K - X_t) \, dt + b X_t dW_t]
\]

\[
+ \left( \left( -r K + \frac{1}{2} b^2 \right) t - b dW_t \right) X_t F_t
\]

\[
+ \frac{1}{2} (-b F_t^2 + (b X_t)) dt
\]

\[
= -r Y_t^2 / F_t
\]
where $[\bullet, \bullet]$ means the representation of quadratic variation [Klebaner]. Thus the solution of SDE is:

$$X_t = \exp \left\{ \left( rK - \frac{1}{2}b^2 \right)t + bW_t \right\} \frac{1}{x^{-1} + r \int_0^t \exp \left\{ \left( rK - \frac{1}{2}b^2 \right)s + bW_s \right\} ds}$$

d) $dX_t = \kappa X_t(\alpha - \log X_t) dt + \sigma X_t dW_t$, $X_0 = x, \kappa, \alpha, \sigma > 0$ gives the solution:

Change the variable, $Y_t = \ln X_t$. Then

$$dY_t = \frac{1}{X_t} dX_t - \frac{1}{2} \frac{1}{X_t^2} \sigma^2 Y_t^2 dt$$

$$= \kappa \left( \alpha - Y_t - \frac{\sigma^2}{2\kappa} \right) dt + \sigma dW_t$$

$$dY_t + \kappa Y_t dt = \kappa \left( \alpha - \frac{\sigma^2}{2\kappa} \right) dt + \sigma dW_t$$

$$d(e^{\kappa Y_t}) = e^{\kappa Y_t} \left( \kappa \left( \alpha - \frac{\sigma^2}{2\kappa} \right) ds + \sigma dW_s \right)$$

This implies:

$$X_t = \exp \left( e^{-\kappa \ln x + \left( \alpha - \frac{\sigma^2}{2\kappa} \right)(1 - e^{\kappa t}) + \sigma e^{-\kappa t} \int_0^t e^{\kappa s} dW_s} \right)$$

```python
r = Symbol("gamma")
W = Symbol("W")
r = 1/X(W)
b = Symbol("b")
F = exp(-integrate(b,W)+integrate(b*b,t)/2)
print F
exp(-W*b + t*b**2/2)
Y = Function("Y")
WW = Function("WW")
rr = r.subs({X(W):Y/F})
Y_prime = Derivative(Y, t)
rrr = (rr*F).subs({W:WW})
sol = dsolve(Y_prime-rrr, Y, hint='best').subs({WW:W})
pprint(sol.rhs/F)
```

```python
2
W*b - ---- / | 2
\ | 2
\ | t*b -2*b*W(t)
e* / C1 - 2* | -e** dt
/ |
/ |
/ |
None
```
where the constant in the last result is:

\[
X_0 = x \\
= \exp\left( bW_0 - \frac{1}{2}b^2 \cdot 0 \right) \sqrt{C1 + 2 \int_0^0 \exp(-2bW_s + b^2s)ds} \\
= \sqrt{C1} \\
\Rightarrow \quad C1 = x^2
\]

One of advantages of Python language is that supports namespace. It is very useful to acclaim variable \( x \), and define function, \( Y \), in Python programming similar to the basic idea we use in mathematics study. Here, Sympy package avails the “Symbol/symbols” for variable acclaiming and “Function” for function defining.

2.3 Stationary Solution for Komolgorov Forward Equation

The solution of SDE,

\[
dX_t = \mu(X_t)dt + \sigma(X_t)\,dW_t
\]

is a process with which its probability density function, \( f(x, t) \), satisfies Komolgorov forward equation:

\[
\frac{\partial f}{\partial t} = \frac{\partial}{\partial x} \left( \mu(x)f(x, t) \right) + \frac{\partial^2}{\partial x^2} \left( \sigma^2(x)f(x, t) \right)
\]

Generally, an explicit solution is not solvable; however, it is quiet easy to discuss the probability density function as it has reached its equilibrium. Stationary solution of probability density function is given by Wright’s formula:

\[
f(x) = \frac{\psi}{\sigma^2} \exp \left[ \int_{\Omega} \frac{\mu(s)}{\sigma^2(s)} \, ds \right]
\]

where \( \psi \) is chosen to \( \int_{\Omega} f(x)\,dx = 1 \).

PySDE also provides the \texttt{KolmogorovFE_Spdf(\mu, \sigma^2, [a,b])} to solve the stationary probability density function, \( f(x) \). The optional \([a,b]\) is \([-\infty, \infty]\) in default. In the following Python session, three popular cases are considered:

Normal case
\[
dX_t = r(G - X_t)dt + \sqrt{\varepsilon} \, dW_t
\]

Gamma case
\[
dX_t = r(G - X_t)dt + \sqrt{\varepsilon X_t} \, dW_t
\]

Beta case
\[
dX_t = r(G - X_t)dt + \sqrt{\varepsilon X_t(1 - X_t)} \, dW_t
\]

where \( r, \varepsilon > 0 \).
2.4 Other Python Computation

For general SDE,

\[ dX_s = \mu(s, X_s)ds + \sigma(s, X_s) dW_s \]
\[ X_t = x \]

where \( t \leq s \leq T \). Suppose that solution of equations is \( X_s^{(t,x)} \) Then \( u(t, T, x) = E[\varphi(X_T) | X_t = x] = E[\varphi(X_T^{(t,x)})] \) would satisfy the following Kolmogorov backward partial differential equation:

\[ \frac{\partial u}{\partial t} + \mu(t, x) \frac{\partial u}{\partial x} + \frac{1}{2} \sigma^2(t, x) \frac{\partial^2 u}{\partial x^2} = 0 \]
\[ u(T, T, x) = \varphi(x) \]
\[ \frac{\partial u}{\partial t} - \mu(t, x) \frac{\partial u}{\partial x} - \frac{1}{2} \sigma^2(t, x) \frac{\partial^2 u}{\partial x^2} = 0 \]
\[ u(t, t, x) = \varphi(x) \]

where \( \varphi(x) \) is smooth.

For example, suppose that \( u(t, T, x) \) satisfies the following pde:

\[ \frac{\partial u}{\partial t} + \frac{1}{2} \frac{\partial^2 u}{\partial x^2} = 0 \]
\[ u(T, T, x) = e^{-x^2/2} \]
This implies $\mu = 0, \sigma = 1$ and

$$
\begin{align*}
    dX_s & = dW_s \\
    \Rightarrow X_u & = x + W_u - W_t
\end{align*}
$$

Therefore, the solution of PDE can be derived as follows:

$$
\begin{align*}
    u(t, T, x) & = E\left[ e^{-X_T^2/2} | X_t = x \right] \\
    & = E[\exp(-(x + W_T - W_t)^2/2)] \\
    & = \int_{-\infty}^{\infty} e^{-(x+y)^2/2} \frac{1}{\sqrt{2\pi(T-t)}} e^{\frac{-y^2}{2(T-t)}} dy \\
    & = \frac{1}{\sqrt{T-t+1}} e^{\frac{x^2}{1-2(T-t)+2t}}
\end{align*}
$$

since the distribution of $W_T - W_t + x$ being $N(x, T - t)$. The full steps of pde solver in pysde could be simply implemented by solving SDE for $X_t$, and evaluating the conditional expectation as follows:

Welcome to Python and user friendly authoring environment

TeXmacs Python interface version 0.8.1. Ero Carrera (c) 2004

http://dkbza.org/tmPython.html

```python
from sys import path

path.append(".")

from pysde import *

T,t,x,y,w = symbols(" T t x y w")

drift=0;diffusion=1;f_init=exp(-x**2/2)

sol=SDE_solver(drift,diffusion,t,x)

print sol

w + x

func=sol**2-w**2/2/(T-t)

l=normal_int(func,w)

pprint(simplify(l/sqrt(2*pi*(T-t))))
```

Here, we also use a special implemented function, normal_int (included in PySDE), to evaluate the integral of function in such form, $\exp(Ax^2 + Bx + C)$, since it is necessary in evaluating probability of function involving Wiener processes but could not solvable by SymPy’s integrate() function.
3 Interactive Environment

Using Python interactively is not too difficult in desktop application environment; Idle, IPython or \TeX\textsc{macs} support such function. Modern internet technology makes Python programming possibly. Sage has implemented its Notebook() as like Mathematica does. Incorporated the Sage’s Notebook(), we can run Python codes remotely by any web browser. Working within Sage is as same as working within desktop environment:

![Figure 1. Sage Web interface, Notebook(), on Firefox](image)

Although Sage Notebook interface brings more flexible environment for computation than ever, some using experiences have to be adjusted for desktop users. Directly display visualization output after computing can not work again through Sage’s Notebook(); instead, visualization output would be auto-displayed on web browser while it was saved. Notebook() interface has also been supported by last IPython which provides a light-weight web-based Python computing application.

4 Conclusion

The focus of our research was to develop a computer package for solving SDEs. To accomplish our objective we investigated the use of computer scripting language, Python and its packages, which allows us to formulate mathematical staff and provides high-performance functionality for computation. Our investigation centered on symbolic SDEs solver and integrated working environment with interactivity and publishing system.

Stochastic differential equations appear to hold considerable promise for sciences. It is more reasonable to consider and study a model based on both stochastic and deterministic components. This feature overcomes the major drawback of ordinary differential equations in scientific researches. The PyS\textsuperscript{3}DE provides the solvers for SDEs, stationary probability density function, numerical schemes and visualization. In other words, it is relatively easy to derive the probability density function of dependent variable as it has reached statistical equilibrium.

Acknowledgments.

This article is supported by NSC project and METO.

References: