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 professional computational ability.This article describes the use of Python and its packages for symbolic, numerical computations and visualization in stochastic calculus and stochastic differential equations (SDEs). Currently, the SDE python solver contains basic tools for stochastic calulus, Itô formula, Stratonovich formula, SDE solver, Euler , Milstein numerical schemes, SDEplot and Feynman-Kac simulation etc.

Although the package can run almost system, we introduce exclusive running environment, TEXMACS , on which avails more flexible interactive function than running on shell or GUI environment. Python SDE solver can also run as a web notebook server which is based on the Sage notebook. Python SDE solver with fully operation system are made into bootable image which can run on almost PC hardwares. This image can be downloaded from the WWW.

Python

SDE:

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 are the same with the relation:

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

or reversely:

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

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 SDE, 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.

Consider the case:

$$dX_t = X_t dt + X_t dW_t$$

The solution is (see the following python snippet) $\exp(W_t + t/2)$. This Stratonovich SDE becomes:

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

Solve last SDE by general method of seperating variables:

$$\begin{aligned} \frac{1}{X_t} dX_t &= \frac{1}{2} dt + 1 \circ dW_t = \frac{1}{2} dt + dW_t \\ &\Downarrow \\ X_t &= x + \exp\left(\frac{1}{2} t + W_t\right) \end{aligned}$$

where $X_0 = x$.

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

$$dX_t = b(X_t) \circ dW_t$$

Directly integrating SDE obtains the solution as follows:

$$X_t = \psi^{-1}(W_t + \psi(X_0))$$

where

$$\psi(x) = \int^x \frac{ds}{b(s)} \text{ and } X(0) = X_0$$

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 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

$$\begin{aligned} \frac{1}{b(X_t)} dX_t &= \alpha(t) dt + 1 \circ dW_t \\ &\Downarrow \\ \psi(X_t) - \psi(X_0) &= \int^t \alpha(s) ds + W_t \\ &\Downarrow \\ X_t &= \psi^{-1}\left(\int^t \alpha(s) ds + W_t + \psi(X_0) \right) \end{aligned}$$

where ψ 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:

$$dX_t = X_t^m \circ dW_t, \quad X_0 = x \tag{1}$$
$$dX_t = \alpha X_t^2 dt + X_t^2 \circ dW_t, \quad X_0 = x \tag{2}$$

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

  X(W) == (C1 + W - W*m)**(1/(1 - m))

Python] dsolve(X_prime-b, X(W),hint='best')

  X(W) == (C1 + W - W*m)**(1/(1 - m))

Python] fac=a*t+W
Python] bb=x*x
Python] Eq=integrate(1/bb,x)+C
Python] print Eq-fac

 C - W - alpha*t - 1/x

Python] sol=solve(Eq-fac,x)
Python] sol[0].subs({x:X(W)})

  1/(C - W - alpha*t)

Python]
```

There are several solvable SDEs and its solution listed as follows:(Oksendal)

a) $dX_t = \gamma dt + bX_t dW_t$ 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

$$F_t = \exp\left(-bW_t + \frac{1}{2}b^2 t\right)$$

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

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

and $X_t = F_t^{-1}Y_t$.

Example 2. Consider

$$dX_t = \frac{1}{X_t}dt + bX_t dW_t, \quad X_0 = x > 0$$

Let

$$F_t = \exp\left(-bW_t + \frac{1}{2}b^2 t\right)$$

and

$$\begin{aligned}
\frac{dY_t(\omega)}{dt} &= \frac{F_t}{X_t} = \frac{F_t^2}{Y_t} \\
\Rightarrow \quad Y_t^2(\omega) &= x^2 + 2\int_0^t \exp\left(-2bW_s + b^2 s\right)ds \\
\Rightarrow \quad X_t(\omega) &= F_t^{-1}(\omega)Y_t(\omega) \\
&= \exp\left(bW_t - \frac{1}{2}b^2 t\right)\sqrt{x^2 + 2\int_0^t \exp\left(-2bW_s + b^2 s\right)ds}
\end{aligned}$$

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

Define integrating factor, F_t, as follows:

$$F_t = \exp\left\{\left(-rK + \frac{1}{2}b^2\right)t - bW_t\right\}$$

Then the SDE of $Y_t = F_t X_t$ satisfies the following ODE:

$$\begin{aligned}
dY_t &= F_t dX_t + X_t dF_t + [X_t, F_t] \\
&= F_t[rX_t(K - X_t)\,dt + bX_t dW_t] \\
&\quad + \left(\left(-rK + \frac{1}{2}b^2\right)dt - b\,dW_t\right)X_t F_t \\
&\quad + \frac{1}{2}(-bF_t)\cdot(bX_t)dt \\
&= -rY_t^2/F_t
\end{aligned}$$

where $[\bullet,\bullet]$ means the representation of quadratic variation [Klebaner]. Thus the solution of SDE is:

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

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

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

$$\begin{aligned}
dY_t &= \frac{1}{X_t}dX_t - \frac{1}{2}\cdot\frac{1}{X_t^{\,2}}\sigma^2 X_t^2 dt \\
&= \kappa\left(\alpha - Y_t - \frac{\sigma^2}{2\kappa}\right)dt+\sigma dW_t \\
dY_t+\kappa Y_t dt &= \kappa\left(\alpha-\frac{\sigma^2}{2\kappa}\right)dt+\sigma dW_t \\
d\left(e^{\kappa t}Y_t\right) &= e^{\kappa t}\left(\kappa\left(\alpha-\frac{\sigma^2}{2\kappa}\right)dt+\sigma dW_t\right) \\
e^{\kappa t}Y_t - \ln x &= \int_0^t e^{\kappa s}\left(\kappa\left(\alpha-\frac{\sigma^2}{2\kappa}\right)ds+\sigma dW_s\right)ds
\end{aligned}$$

This implies:

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

```
Python] r=Symbol("gamma")
Python] W=Symbol("W")
Python] r=1/X(W)
Python] b=Symbol("b")
Python] F=exp(-integrate(b,W)+integrate(b*b,t)/2)
Python] print F
```

```
exp(-W*b + t*b**2/2)
```

```
Python] Y = Function("Y")(t)
Python] WW=Function("WW")(t)
Python] rr=r.subs({X(W):Y/F})
Python] Y_prime=Derivative(Y, t)
Python] rrr=(rr*F).subs({W:WW})
Python] sol=dsolve(Y_prime-rrr, Y,hint='best')subs({WW:W})
Python] pprint(sol.rhs/F)
```

```
                  ---------------------------------
          2        /          /
       t*b        /           |
 W*b - ----      /            |      2
       2        /             |   t*b   -2*b*W(t)
e          *   /     C1 - 2* | -e    *e           dt
             /                |
            \/               /
 None
```

```
Python]
```

where the constant in the last result is:

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

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.

The solution of SDE,

$$dX_t = \mu(X_t)dt + \sigma(X_t)\, dW_t \tag{3}$$

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

$$\frac{\partial f}{\partial t} = \frac{\partial(\mu(x) f(x,t))}{\partial x} + \frac{\partial(\sigma^2(x) f(x,t))}{\partial x^2} \tag{4}$$

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

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 pdf is given by Wright's formula

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

where ψ is chosen to $\int_\Omega f(x)dx = 1$. The solver avails the KolmogorovFE_Spdf(μ,σ^2[,a,b]) to solve the stationary pdf, $f(x)$. The optional [a,b] is $[-\infty, \infty]$ in default. In the following session, three popular cases are considered:

$$\begin{aligned} \text{Normal case} \quad & dX_t = r(G - X_t)dt + \sqrt{\varepsilon}\, dW_t \\ \text{Gamma case} \quad & dX_t = r(G - X_t)dt + \sqrt{\varepsilon X_t}\, dW_t \\ \text{Beta case} \quad & dX_t = r(G - X_t)dt + \sqrt{\varepsilon X_t(1 - X_t)}\, dW_t \end{aligned}$$

where $r, \varepsilon > 0$.

```
Welcome to Python
TeXmacs Python interface version 0.8.1. Ero Carrera (c) 2004
http://dkbza.org/tmPython.html
```

```
Python] from sys import path
```

```
Python] path.append(".")
```

```
None
```

```
Python] from pysde import *
Python] x,dx=symbols('x dx')
Python] [a,b,c,d]=[0,1,0,1]
Python] coeff=[a,b,c,d]
Python] t0=0;x0=1
Python] drift=a+b*x
Python] diffusion=c+d*x
Python] sol1=sde.SDE_solver(0,1,0,1)
Python] print sol1
```

```
1 + w
```

```
Python] sol=sde.SDE_solver(drift,diffusion,t0,x0);
```

```
exp(w + 0.5*t) 1 0 0
```

```
Python] print " Solution of dX = (%s) dt + (%s) dw is %s"
        %(drift,diffusion,sol)
```

```
Solution of dX = (x) dt + (x) dw is exp(w + 0.5*t)
```

```
Python]
Python] r,G,e,d=symbols('r G epsilon delta')
Python] print sde.KolmogorovFE_Spdf(r*(G-x),e)
```

```
exp(-r*G**2/(2*epsilon) + G*r*x/epsilon -
r*x**2/(2*epsilon))/(2*pi**(1/2)*epsilon**(1/2)*(1/r)**(1/2))
```

```
Python] print sde.KolmogorovFE_Spdf(r*(G-x),e*x,0,oo)
```

```
x**(-1 + G*r)*(epsilon/r)**(G*r)*exp(-r*x/epsilon)/gamma(G*r)
```

```
Python] l=sde.KolmogorovFE_Spdf(r*(G-x),e*x*(1-x),0,1)
Python]  l.subs({e:r*d})
```

```
 x**(-1 + G/delta)*(1 - x)**(-1 + 1/delta -
G/delta)*gamma(1/delta)/(gamma(G/delta)*gamma(-G*(1 - 1/G)/delta))
```

```
Python]
```

For general SDE,

$$\begin{aligned} dX_s &= \mu(s,X_s)ds+\sigma(s,X_s)\,dW_s \\ X_t &= x \end{aligned}$$

where $t\leqslant s\leqslant T$. Suppose that function of solution is $X_s^{(t,x)}$ Then $u(t,T,x)=E[\varphi(X_T)|\,X_t=x]=E\Big[\,\varphi\Big(X_T^{(t,x)}\Big)\Big]$ satisfies the Kolmogorov backward partial differential equation:

$$\begin{aligned} \frac{\partial u}{\partial t}+\mu(t,x)\frac{\partial u}{\partial x}+\frac{1}{2}\sigma^2(t,x)\frac{\partial u}{\partial x^2} &= 0 \\ u(T,T,x) &= \varphi(x) \\ \frac{\partial u}{\partial t} &= \mu(t,x)\frac{\partial u}{\partial x}+\frac{1}{2}\sigma^2(t,x)\frac{\partial u}{\partial x^2} \\ u(t,t,x) &= \varphi(x) \end{aligned}$$

where $\varphi(x)$ is smooth. For example, suppose that $u(t,T,x)$ satisfies the following pde:

$$\begin{aligned} \frac{\partial u}{\partial t}+\frac{1}{2}\frac{\partial u}{\partial x^2} &= 0 \\ u(T,T,x) &= e^{-x^2/2} \end{aligned}$$

This implies $\mu=0$, $\sigma=1$ and

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

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

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

where the integrand in the integral is because 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
TeXmacs Python interface version 0.8.1. Ero Carrera (c) 2004
http://dkbza.org/tmPython.html
```

```
Python] from sys import path
```

```
Python] path.append(".")
```

```
None
```

```
Python] from sde import *
```

```
Python] T,t,x,y,w = symbols(" T t x y w")
```

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

```
Python] sol=SDE_solver(drift,diffusion,t,x)
```

```
Python] print sol
```

```
w + x
```

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

```
Python] l=normal_int(func,w)
```

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

```
                                    2
                                   x          2
                           - ------------ + x
                                    1
                                1 - ---------
                                    2*(T - t)
          ---------------
   ___   /   -2*t + 2*T
 \/ 2 *  /  ------------- *e
       \/   1 - 2*T + 2*t
```

```
-----------------------------------------------

                    -------
                  2*\/ T - t
 None
```

```
Python]
```

Here, we also use a special implemented function, normal_int, to evaluate the integral of exp $(Ax^2 + Bx + C)$ since it is necessary in evaluating expectations of Wiener processes but not solvable by Sympy's integrate() function.

Interactivate Web Notebook

Sage has implemented its Notebook as like Mathematica does. Incorporated the Sage's Notebook, we can run Python codes remotely through any web browser. Working within Sage is as same as working within desktop environment:

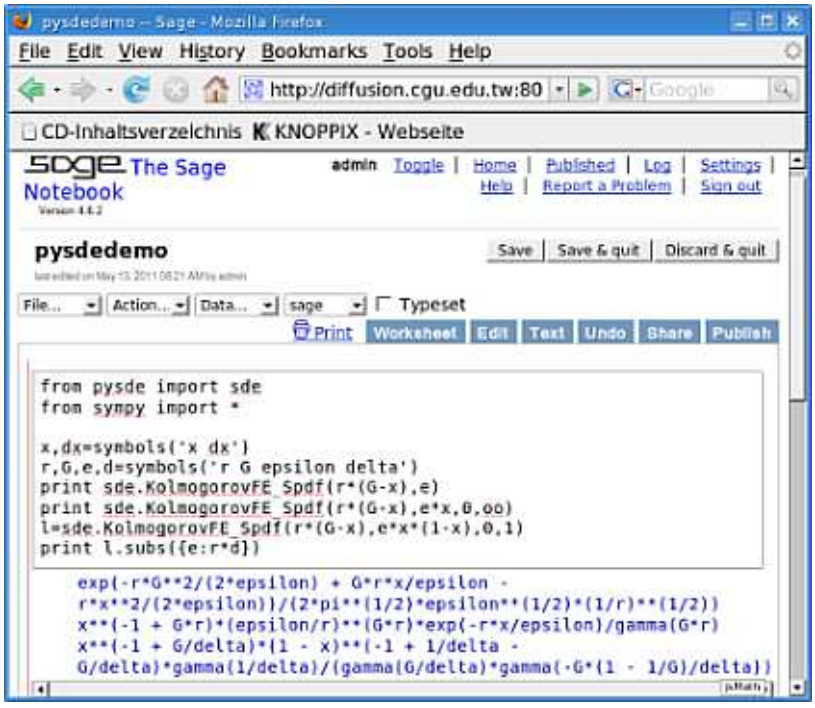

Figure 1. Sage Web interface, Notebook(), on Firefox-4

Although Sage Notebook 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 web interface, Notebook(); instead, visualization output would be auto-displayed on web browser as it was saved. However, Sage has integrated hundred of powerful open-source softwares and capable of working on most operation systems; these advantages make it more viable computing system than proprietary softwares.

Cobb, Loren, Stochastic Differential Equations for the Social Sciences, Mathematical Frontier of the Social and Policy Sciences, Cobb and Thraill eds.,Westview Press, 1981.

Wright, Swell. The distribution of Gene Frequencies under Irreversible Mutation. Proc. Nat'l. Acad. Sci., 24, 1938,253-259.

Oksendal Bernt, Stochastic Differential Equations, An Introduction with Applications, 6th Ed., Springer-Verlag, 2005.

Gard T.C., Introduction to Stochastic Differential Equation, Dekker, 1988.

Klebaner Fima C " Introduction to Stochastic Calculus with Applications, 2nd Ed.", Imperial College Press, 2004.