MATIMYÁS MATEMATIKA
ISSN 0115-6926

Journal of the Mathematical Society of the Philippines
Vol. 34 Nos. 1-2 (2011) pp. 87-97

Python Solver for Stochastic Differential Equations

Chu-Ching Huang
Center for General Education
Chang-Gung University
Taiwan
email: cchuang@mail.cgu.edu.tw

Abstract

As an alternative to proprietary computer algebra systems (CAS), such as Matlab and Maple, open source computer scripting language Python and its applications carry a variety of comprehensive capacities which have potential for doing professional mathematical work. PyS^3 DE is a Python module with the aims to establish full-featured CAS for studying stochastic differential equations (SDEs). PyS^3 DE features: a) symbolic solvers for SDEs, linear, reduced nonlinear SDE and Kolmogorov backward equation; b) stochastic numerical schemes, Euler and Milstein methods; and c) visualization tools for calculated data and Feynman-Kac simulation etc.

Although PyS^3 DE can run on almost all operation systems in which the Python environment has been well installed, TEXMACS, an intelligent scientific publishing system, is introduced to provide more friendly graphical user interface (GUI) over the LaTeX system and is capable of directly communicating with the Python environment. Incorporated with Sage's notebook, PyS^3 DE's is also accessible remotely through any web browser and is able to do web computation. Creating a viable web server that is able to do Python computation PyS^3 DE is also discussed in this paper.

Keywords: Stochastic Differential Equations (SDEs), PyS^3 DE, Kolmogorov backward equation, 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 defined as the derivative of the Brownian motion, $W(t) = W_t$:

$$X_t = \frac{d}{dt} W_t$$

It stands for the phenomenon of the violent changes in a system or unpredictable data. The theory of SDEs is popular in both theoretical and applied science since it could describe actual evolution of system. However, white noise does not exist as a function of t in classical analysis since the trajectory of a Brownian motion is nowhere differentiable.

During the middle of last century, Japanese mathematician K. Itô and Russian mathematician Stratanovich established the fundamental theory of stochastic calculus which could be used to study SDEs. But a very theoretical approach to stochastic computation also increases the learning curve steeply and makes it very difficult to use. Although there are some computer packages developed to do stochastic calculation, new problems arise – either the development language (Axiom, [Kendall]) is not popular or proprietary software (Maple, [Cyganowski etc]) is used. To solve these problems, we decided to develop a computer package able to do stochastic calculus and solve SDEs based on another reliable approach.

As a computer language, Python owns plenty of cutting-edge features: open-source, object-oriented interpreter language and tons of libraries or extensions (for instance, interactive environment, numerical analysis, visualization etc). Though Python is a general purpose, high level language with compact core library, incorporating its modules/extensions with the following advanced features makes it popular in the scientific community and academe:

- Clean syntax that is easy to work with;
- Expressions, conditions, loops, etc;
- Data types (numbers, lists, array and matrix etc.);
- Program structure (functions, objects, modules, etc.);
- Numerical methods (**Scipy/Numpy**): Working with arrays, linear algebra, random number generation and numerical integration and numerical optimization, etc;
- Symbolic calculation (**Sympy, Sage**): basic calculus operations, solvers for ordinary differential equations (ODEs) and algebraic equation and mathematics expression, etc;
- Visualization (**Matplotlib, Sage**): 2D/3D pictures and animation etc;
- Accessible in library with foreign languages, Fortran and C/C++, etc.

The quoted texts in boldface type above are the additional Python modules already introduced into our project. The superior principles in setting-up our Python computing environment are *compactness* and *efficiency*.

Sage is another comprehensive software that is a complete environment for symbolic and numerical computing, using an extension of Python as programming language. Sage integrates more than 100 Python packages/extensions and is well-tuned for running Python or other mathematical software systems with unified user interface. The symbolic subsystem of Sage provides an environment similar to popular Maple or Mathematica. Sage uses a hybrid C++ and Cython enhanced library, Pynac, on the top of GiNaC (a very efficient C++ library for symbolic computing), to work with symbolic expressions. High level symbolic facilities of Sage use Maxima and naive Sage-cloned module behind the scenes.

With respect to pure Python symbolic environment, we use only Sympy module which is written in pure Python and trivial to install. Both Sympy and Sage have seemingly borrowed naming conventions from GiNaC and Swiginac, a SWIG-based Python interface to GiNaC library, so the syntax differences between two packages are small.

In particular, Sympy owns a powerful submodule, **dsolve()**, which can solve ordinary differential equations (ODEs). Sage uses Maxima as backend of ODEs' solver. To keep the consistency capable of running on both desktop and internet environment, we decided to develop SDEs solver independent on neither Sympy's nor Sage's ODEs solver.

Besides available computing abilities for a broad range of mathematical functionality, a full-featured scientific computing environment should support high-quality publishing system for continuing documentation work after computation. The scientific publishing system, TEXMACS, was developed with the aim of reducing cost of making high-quality documentation and producing high-quality user interfaces. Unlike Python or Sage, TEXMACS itself does not provide any computing facilities but offers an intelligent what-you-see-is-what-you-get (WYSIWYG) system based on LATEX and Guile. The purpose of integrating TEXMACS into computational environment is to support optimized typesetting system with friendly LATEX front-end interface and intelligent plug-ins interface for running external applications. The

former make professional mathematical document publishing easily; the latter allows us to run external program (Python, Sage etc) directly within document and print out result. This cutting-edge **plug-ins** feature actually shortens the development time in our project.

Generally, professional computing software requires high-standard running environment, including software and hardware. Live system technology brings a new perspective view of software engineering: nothing is necessarily pre-installed on your hard drive, and a live system provides a wide collection of customized software ready to use after booting from its host media, CD or USB flash driver.

Porteus is the host system we chose. Porteus is a modern, portable, small and fast Linux operating system with a modular approach and outstanding design. Despite its small size (64bit system under 300Mb and 32bit system under 200Mb), other packages and modules are made into individual modules and can be activated to work after booting up. The modular approach allows us to design different platform that allow practical use and make system maintenance more efficienct.

2 Stochastic Differential Equations

The Python computing environment is built up based on Python, Numpy, Scipy, scitools and Sage. The implement for SDEs solver, PyS^3 DE, is named from the first letters of the main software we use:

PyS$^{\mathbf{3}}$ DE =**Py**thon, **S**age, **S**ciPy for **S**tochastic **D**ifferential **E**quations

In brief, this implement includes symbolic solver for SDEs, numerical schemes and visualization function. Here, only symbolic facilities will be introduced in detail.

2.1 Itô and Stratonovich SDEs

The differentiation terms over white noise have different formulations depending on the approach used. In general, there are two kinds of approaches: Itô and Stratonovich approach. Itô version uses left value rule to present the differential term of Brownian motion [Øksendal], dW_t, with respect to the middle value rule for Stratonovich version. In general, different kinds of one dimensional SDEs are represented by different expressions:

a) Itô version:

$$dX_t = a\left(t, X_t\right) dt + b\left(t, X_t\right) dW_t \tag{2.1}$$

b) Stratonovich version:

$$dX_t = \tilde{a}\left(t, X_t\right) dt + b\left(t, X_t\right) \circ dW_t \tag{2.2}$$

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

$$\tilde{a}\left(t, x\right) = a\left(t, x\right) - \frac{1}{2} b\left(t, x\right) \frac{\partial b}{\partial x}\left(t, x\right) \tag{2.3}$$

or reversely:

$$a\left(t, x\right) = \tilde{a}\left(t, x\right) + \frac{1}{2} b\left(t, x\right) \frac{\partial b}{\partial x}\left(t, x\right) \tag{2.4}$$

The stochastic integral (Stratonovich version) when using the midpoint value rule has a result just like in deterministic calculus:

$$\int_0^T W_t \circ dW_t = \frac{1}{2} W_T^2 \tag{2.5}$$

and

$$\int_0^T W_t dW_t = \frac{1}{2}W_T^2 - \frac{1}{2}T \tag{2.6}$$

by using left value rule (Itô version), respectively.

All the linear SDEs can be solved theoretically. Some kinds of Itô SDEs other than the linear case can be converted to Stratonovich SDEs and become solvable by the rules or tools used in deterministic calculus.

Here is an example of how to use Sympy to solve SDEs. Note that variables and functions have to be declared, **symbols("···")** for variable and **Function("···")** for function, before they are used:

Example 1 (Python with TEXMACS Python plugins)

Consider the following SDEs:

$$dX_t = X_t^m \circ dW_t, \quad X_0 = x \tag{2.7}$$

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,a =symbols("W alpha")
Python] b = X(W)**m
Python] X_prime=Derivative(X(W), W)
Python] dsolve(X_prime-b, X(W))
 X(W) == (C1 - W*m + W)**(1/(-m + 1))
```

The steps in the solution are: activating Python plugin, input Python codes and print out result. "**dsolve()**" could solve Stratonovich SDEs directly and Itô SDEs with only little modification.

2.2 Symbolic SDEs Solver, of PyS^3 DE

The theory of existence and uniqueness of solutions of linear SDEs can be found in the references, (Øksendal, Klebaner). The first type we consider is the the following linear SDE:

$$dX_t = (a_1(t)X_t + a_2(t))\,dt + (b_1(t)X_t + b_2(t))\,\mathrm{dW}_t \tag{2.8}$$

Its solution is known as follows:

$$X_t = \Phi_{t_0,t}\left(X_{t_0} + \int_{t_0}^t (a_{21}(s) - b_1(s)b_2(s))\,\Phi_{t_0,s}^{-1}ds + \int_{t_0}^t (b_2(s))\,\Phi_{t_0,s}^{-1}dW_s\right) \tag{2.9}$$

where the integration factor is:

$$\Phi_{t_0,t} = \exp\left(\int_{t_0}^t \left(a_2(s) - \frac{1}{2}b_1^2(s)\right)ds + \int_{t_0}^t b_1(s)\,dW_s\right) \tag{2.10}$$

This result is already implemented in PyS^3 DE.

Certain types of SDEs can also be solved by converting into integrable Stratonovich SDE. Consider the following Itô SDE:

$$dX_t = \left(\alpha(t)\, b(X_t) + \frac{1}{2} b(X_t)\, b' X_t\right) dt + b(X_t)\, dW_t \tag{2.11}$$

which is equivalent to the following Stratonovich SDE:

$$dX_t = \alpha(t)\, b(X_t)\, dt + b(X_t) \circ dW_t$$

And its solution is

$$X_t \quad = \quad \psi^{-1}\left(\int^t \alpha(s)\, ds + W_t + \psi(X_0)\right) \tag{2.12}$$

where

$$\psi(x) = \int^x \frac{ds}{b(s)}$$

This case and linear case forms the main part of SDE solver in PyS^3 DE.

Example 2 Consider the SDE:

$$dX_t = \left(\frac{2X_t}{1+t} - a(1+t)^2\right) dt + a(1+t)^2\, dW_t \tag{2.13}$$

where $t(0) = t_0, X_0 = x_0$ and $a > 0$. By PyS^3 DE, the resulting solution is:

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

The function, **SDE_solver()**, gives the solution of (14) as follows:

$$X_t = \left(\frac{1+t}{1+t_0}\right)^2 \left(x_0 + a\ (t_0 - t)(1+t_0)^2 + a\ (1+t_0)^2 W_t\right) \tag{2.14}$$

Example 3 Consider another SDE:

$$dX_t = -\left(\frac{2X_t}{1-t}\right) dt + \sqrt{t(1-t)} dW_t \tag{2.15}$$

where $0 \leqslant t < 1$ and $X_0 = x_0$. By the PyS^3 DE computing, the solution involves Gaussian processes other than Wiener processes W_t:

```
Python] drift=-2*x/(1+t)
```

```
Python] diffusion=sqrt(t*(1-t))
Python] sol=SDE_solver(drift,diffusion,t0,x0)
Python] print(sol)
(t0 + 1)**2*(x0 + N(0, -t**7/(7*t0**4 + 28*t0**3 + 42*t0**2
+ 28*t0 + 7) - t**6/(2*t0**4 + 8*t0**3 + 12*t0**2 + 8*t0 + 2)
- 2*t**5/(5*t0**4 + 20*t0**3 + 30*t0**2 + 20*t0 + 5) +
t**4/(2*t0**4 + 8*t0**3 + 12*t0**2 + 8*t0 + 2) + t**3/(t0**4 + 4*t0**3 + 6*t0**2
+ 4*t0 + 1) + t**2/(2*t0**4 + 8*t0**3 + 12*t0**2 + 8*t0 + 2)))/(t + 1)**2
```

Simplifying the above computation gives:

$$X_t = \left(\frac{1-t_0}{1-t}\right)^2 \left(x_0 + N\left(0, \frac{\frac{t^2+t^4-t^6}{2} + t^3 - \frac{2}{5}t^5 - \frac{t^7}{7}}{(1+t_0)^4}\right)\right) \tag{2.16}$$

where $N\left(\mu, \sigma^2\right)$ represents the Gaussian process with mean μ and variance σ^2 which is implemented in PyS^3 DE as Python class. It is obvious that X_t is a Gaussian process.

2.3 Stationary Solution for Kolmogorov Forward Equation

The solution of SDE,

$$dX_t = \mu\left(X_t\right)dt + \sigma\left(X_t\right)dW_t \tag{2.17}$$

is a process in which its probability density function, $f\left(x,t\right)$, satisfies Kolmogorov forward equation:

$$\frac{\partial f}{\partial t} = \frac{\partial\left(\mu\left(x\right)f\left(x,t\right)\right)}{\partial x} + \frac{\partial\left(\sigma^2\left(x\right)f\left(x,t\right)\right)}{\partial x^2} \tag{2.18}$$

It is well-known as not all SDEs in (2.18) can be solved, but we can still discuss its probability density function in (2.19) as it has reached its equilibrium. Stationary solution of the probability density function of (2.19) is given by Wright's formula [Wright, Cobb]:

$$f\left(x\right) = \frac{\psi}{\sigma^2}\exp\left[\int^x \frac{\mu\left(s\right)}{\sigma^2\left(s\right)}ds\right] \tag{2.19}$$

where ψ is the factor chosen so that $\int_\Omega f\left(x\right)dx = 1$.

PyS^3 DE provides the function, **KolmogorovFE_Spdf(μ, σ^2[,a,b])**, to solve the stationary probability density function, $f\left(x\right)$. The option, [a,b] is set to be $[-\infty,\infty]$ in default. In the following Python session, three cases are calculated:

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

where $r, \varepsilon > 0$.

```
Python] from pysde import *
Python] x,dx=symbols('x dx')
Python] t0=0;x0=1
Python] r,G,e,d=symbols('r G epsilon delta')
Python] l=sde.KolmogorovFE_Spdf(r*(G-x),e*x*(1-x),0,1)
```

```
Python] pprint(l.subs({e:r*d}))
   G*r*W(W)                G*r*W(W)        r*W(W)
   -------- - 1          - -------- - 1 + ------
    delta                   delta          delta         /r*W(W)\
   x            *(-x + 1)                            *gamma|------|
                                                          \delta /
   ------------------------------------------------------------
                                  /    /     1\     \
                                  |G*r*|-1 + -|*W(W)|
                       /G*r*W(W)\ |    \     G/     |
                  gamma|--------|*gamma|-----------|
                       \ delta  /       \     delta /
```

where gamma (x) represents Gamma function with x parameter.

2.4 Other Python Computation

PyS^3 DE also has basic functions which can be used to a solve Kolmogorov backward partial differential equation. For general SDEs,

$$\begin{aligned} dX_s &= \mu(s, X_s)\, ds + \sigma(s, X_s)\, dW_s \\ X_t &= x \end{aligned} \tag{2.20}$$

where $t \leqslant s \leqslant T$. Suppose that the solution of the system is $X_s^{(t,x)}$. Then

$$u(t, T, x) = E\left[\varphi(X_T) \,|X_t = x\right] = E\left[\varphi\left(X_T^{(t,x)}\right)\right] \tag{2.21}$$

would satisfy the following Kolmogorov backward partial differential equation:

$$\begin{aligned} \frac{\partial u}{\partial t} + \mu(t,x)\frac{\partial u}{\partial x} + \frac{1}{2}\sigma^2(t,x)\frac{\partial u}{\partial x^2} &= 0 \\ u(T,T,x) &= \varphi(x) \\ \frac{\partial u}{\partial t} - \mu(t,x)\frac{\partial u}{\partial x} - \frac{1}{2}\sigma^2(t,x)\frac{\partial u}{\partial x^2} &= 0 \\ u(t,t,x) &= \varphi(x) \end{aligned} \tag{2.22}$$

where $\varphi(x)$ is smooth.

Consider the function which satisfies the following pde:

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

where the diffusion process related to $u(t,T,x)$ is equivalent to the solution of the following SDE:

$$\begin{aligned} dX_s &= dW_s, \quad X_0 = x \\ \Rightarrow X_u &= x + W_u - W_t \end{aligned}$$

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

$$\begin{aligned} u(t,T,x) &= E\left[e^{-X_T^2/2} \,|X_t = x\right] \\ &= \frac{1}{\sqrt{T-t+1}} e^{-\frac{x^2}{2(T-t+1)}} \end{aligned}$$

The following Python codes give the same result:

```
Python] from pysde import *
Python] T,t,x,y,w = symbols(" T t x y w")
Python] drift=simplify(0);diffusion=simplify(1)
Python] sol1=SDE_solver(drift,diffusion,t,x)
Python] func=-(w+x)**2/2-w**2/2/(T-t)
Python] l=normal_int(func,w)/sqrt(2*pi*(T-t))
Python] pprint(simplify(l))
                    2            2
                   x            x
                 - -- + ----------------
                   2      /1       1    \
      ------------        4*|- + ---------|
     /   T - t             \2   2*(T - t)/
    /  -------- *e
  \/   T - t + 1
  ----------------------------------------

                  ------
                \/ T - t
```

Here, we use PyS^3 DE' integration rule, **normal_int()** , for evaluating integral of function, $\exp\left(Ax^2+Bx+C\right)$.

Another reducible nonlinear SDE we consider is in the following form:

Theorem 2.4.1 *The solution of general SDE:*

$$dX_t = f\left(t, X_t\right)dt + b\left(t\right)X_t dW_t, \quad X_0 = x > 0 \tag{2.23}$$

where $f : \mathbb{R}\times\mathbb{R}\to\mathbb{R}$ *and* $b : \mathbb{R}\to\mathbb{R}$ *are continuous, is*

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

where

$$F_t = \exp\left(-\int^t bdW_s + \frac{1}{2}\int^t b^2 ds\right) \tag{2.24}$$

and Y_t *is the solution of the following:*

$$\frac{dY_t}{dt} = F_t f\left(t, F_t^{-1}Y\left(t\right)\right), \quad Y_0 = x > 0 \tag{2.25}$$

Proof: Use the method of integating factor and let $Y_t = F_t X_t$.

Consider another nonlinear SDE solvable by Sympy's dsolve() as follows:

$$dX_t = \frac{1}{X_t}dt + bX_t dW_t, \quad X_0 = x > 0 \tag{2.26}$$

where b, x are constants. Then

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

and let $Y_t = X_t F_t$, we have

$$X_t = \exp\left(bW_t - \frac{1}{2}b^2 t\right)\sqrt{x^2 + 2\int_0^t \exp\left(-2bW_s + b^2 s\right)ds} \tag{2.27}$$

Here the Python codes which solve (2.25) by Sympy's ODE solver, dsolve():

```
Python] W,b,r=symbols("W,b,gamma")
Python] X=Function("W")
Python] F=exp(-integrate(b,W)+integrate(b*b,t)/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')
Python] pprint(sol[0].rhs/F)
```

```
                 ____________________________
                /        /                              2
               /        |                              b *t
              /         |                2        W*b - ----
    ___      /          |  -2*b*WW(t)  b *t              2
\/ 2 *  /     C1 +  | e           *e     dt *e
      /            |
     \/           /
```

where the constant C_1 in the last result is x^2 by initial condition $X_0 = x$. Sympy's dsolve() gives two solution of ODE and the second one is taken since it is positive. This is also implemented in PyS^3 DE and named as reduced2() function.

3 Web-based Computing Environment

Using Python interactively is not too difficult in desktop application environment by the help of Idle, IPython or TEXMACS. But there are two two potential barriers to run them remotely:

i. Space and time limit: such applications are not available to users outside the environment (campus or laboratory) which the applications are installed;

ii. budget limit: proprietary softwares might avail remote solution for web computation but they are too expensive to afford for everyone.

Modern internet technology and contributions from volunteers make Python remote programming possible and the field is developing rapidly. Sage is a standard example and comes with all-in-one modules it needs. After downloading and installing source by simple "make" procedure, Sage can run right away. And a twisted implement for web service, called **notebook()**, also comes with Sage. Sage/Python codes can run remotely by this interface as well as Mathematica does. Sage does not contain PyS^3 DE library; we have to copy it into Sage/Python third-party libraries direcory before using. Working within Sage is the same as working within a desktop environment. Sage also avails a more refined environment (mathematical-readable syntax, mathematical functions and high-quality LATEXbased publishing system, etc.) than native Python environment does. The picture in the following page shows the use of PyS^3 DE within Sage's notebook() and through internet.

Although the Sage Notebook web interface brings more flexible computing environment than desktop application does, it is necessary for us to adjust traditional user custom working in desktop environment. Directly displayingthe visualization output after computing may not work well through Sage's notebook(); instead, visualization output would be auto-displayed on web browser while it was **saved**. notebook() interface has also been implemented by last IPython release which also provides a light-weight web-based Python computing application than Sage does.

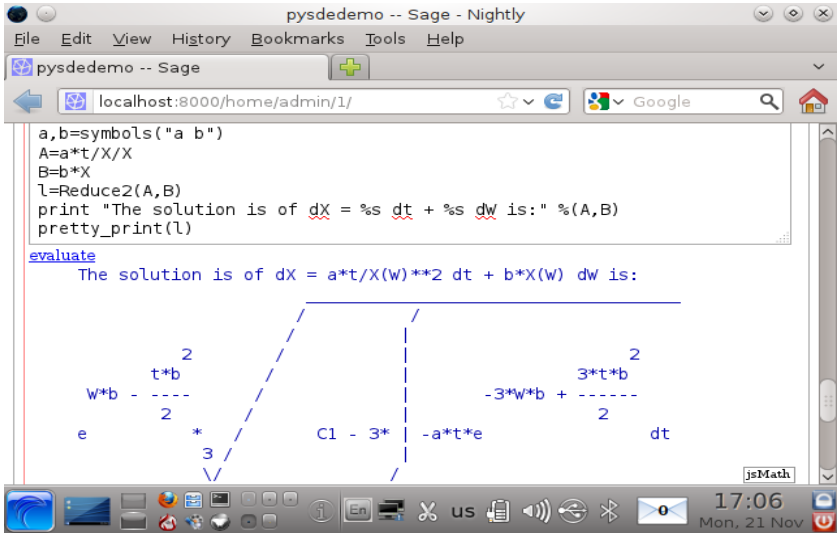

Figure 1: Sage Web interface, notebook(), runs on Firefox

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 stuff and provides high-performance functionality for computation. Now the library for SDE, PyS^3 DE, includes symbolic solvers, numerical schemes and visualization tools. All of them can run on both traditional desktop and web environments.

On the other hand, for the purpose of user interface enhancement, we also introduce a scientific publishing system, TEXMACS, and integrated CAS, Sage, into environment we study. Not only can we work more friendly with TEX/LATEX via TEXMACS as its front-end interface, but its plugin also makes it capable of building flexible Python computing platform with interactivity. Sage is introduced to establish and manage scalable Python computing system through internet which could actually overcome the barriers of budget limit and space-time restriction.

All library and tools we developed may be hosted on mobile device, CDROM or USB flash drive, with pre-installed bootable and self-configuring Linux system. The benefits from such kind of live system include:

- usable on any recent desktop or laptop;
- usable in computer labs as machines rebooted into computing environment and continuing last work;
- support for use to set-up heterogeneous working environment;
- enable to try software on new hardware to reliable test and compatibility;
- extensible: replace buggy package, renew last module or add new package without a long re-installation.

PyS^3 DE and Live system (for 32bit and 64bit Intel-CPU clone set) are released in open source licence and could be downloaded from our official site [diffusion].

Acknowledgments.

This article is supported by 9th TPSOA (MECO-TECO project approved by NSC Taiwan) and Chang-Gung University, Taiwan.

References

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

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

[3] Øksendal Bernt, Stochastic Differential Equations, An Introduction with Applications, 6th Ed., Springer-Verlag, 2007.

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

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

[6] Eröcal, Burçin and Stein, W., The Sage Project: Unified Free Mathematical Software to create a Viable Alternative to Magma, Maple, Mathematica and Matlab, In Komei Fukuda, Joris van der Hoeven, Michael Joswig, and Nobuki Takayama (ed.), ICMS 2010: Proceedings of the Third International Congress on Mathematical Software. Springer, Lecture Notes in Computer Science, volume 6327, p. 12-27, 2010.

[7] Langtangen, Hans Petter, Python Scripting for Computational Science 3rd Ed., Springer, 2009.

[8] Oliphant, Travis E., Python for Scientific Computing, Computing in Science and Engineering, May/June, p. 10-21, 2007.

[9] S. Cyganowski, P.E. Kloeden and J. Ombach, From Elementary Probability to Stochastic DEs with MAPLE, Springer-Verlag, Heidelberg, 2001.

[10] W.S. Kendall, Computer algebra and stochastic calculus, Notices Amer. Math. Soc. 37 (1990), 1254-1256.

[11] William A. Stein et al. Sage Mathematics Software (Version 4.7.2), The Sage Development Team, 2011, http://www.sagemath.org

[12] PyS^3 DE and Live System: http://diffusion.cgu.edu.tw/ftp