

ECG790C - Problem Set 2 - Spring 2009

1. The following MATLAB function works but exhibits several problems (I can think of at least three problems). Write an improved version of this function and explain the issues your improvements address.

```
% This function simulates stochastic processes
function x=sde(mu,sigma,theta,x0,T,n);
    x=x0; Delta=T/n;
    for i=1:n
        x(:,i+1)=x(:,i)+feval(mu,x(:,i),theta{:})*Delta+ ...
            feval(sigma,x(:,i),theta{:})*sqrt(Delta).*randn(length(x0),1);
    end
```

2. This exercise asks you to write code that will generate Brownian paths backwards. Consider a Brownian path W_t that begins at time 0 with $W_0 = 0$. The path at time T is $N(0, T)$, i.e., is a random normal variable with mean 0 and variance T . Suppose with such a randomly generated point W_T and we want to fill in the path at $n - 1$ additional evenly spaced (in time) points, i.e., points at $(n - 1)\Delta, (n - 2)\Delta, \dots, \Delta$, where $\Delta = T/n$.

The Brownian Bridge formula applied to this problem gives a point $W_{(i-1)\Delta}$ given a point $W_{i\Delta}$ using

$$W_{(i-1)\Delta} = \frac{i-1}{i}W_{i\Delta} + \sqrt{\frac{(i-1)\Delta}{i}}z$$

where $z \sim N(0, 1)$. Notice that for $i = 1$ we get $W_0 = 0$, i.e., we get the starting point for the path.

Implement this algorithm to generate a set of m Brownian paths backwards. Specifically, write a function `BackwardsBrownian` that accepts an m vector W , and integer i and a time step Δ , takes one backwards step and returns the resulting m -vector of values. Also write a function `TerminalBrownian` that accepts m, n and T and returns an m -vector of values of W_T and the value of Δ . Both functions should be well documented!

Use these functions with the following script to generate a set of m Brownian paths (each row of W represents points on a Brownian path).

```

m=5000; n=10; T=1;
[Wt,Delta]=TerminalBrownian(m,n,T);
W=zeros(m,n);
W(:,end)=Wt;
for i=n:-1:2
    Wt=BackwardsBrownian(Wt,i,Delta);
    W(:,i-1)=Wt;
end

```

Validate that your code runs properly by computing the mean and variance of each column of W and comparing it to the true (population) values (which you will need to figure out).

Backwards Brownian generators have proven very valuable in solving asset pricing problems, which generally must be solved backwards from some terminal condition. If the Brownian paths are generated forward and then the asset prices determined backwards, the whole set of time paths need to be stored in memory. This limits (sometimes unacceptably) the number of paths that can be used or the step size that can be taken. By generating paths backwards, only the current values need by maintained, thereby freeing a problem from the memory constraints.

Programming hint: My code consists of two code lines for each function. The functions should not be much larger than this and should contain no loops. Certainly the documentation should be longer than the code.

3. For the following processes and transforms, use Ito's Lemmma to express the stochastic differential equation (SDE) that describes the transformed process. For example, if

$$ds = \mu s dt + \sigma s dW$$

and $y = \ln(s)$, then

$$dy = (\mu - \sigma^2/2)dt + \sigma dW$$

- (a) $ds = \mu s dt + \sigma s dW$
and $y = s/(c + s)$ (for some constant c).
- (b) $ds = \alpha s(\mu - s)dt + \sigma s^{3/2}dW$
and $y = 1/s$. (Hint: the transformed process is a square root process).
- (c) $ds = \alpha s(\mu - s)dt + \sigma s^{3/2}dW$
and $y = 2/\sqrt{s}$. (Hint: the transformed process is a constant volatility process).

4. For the process in the second and third subproblems above and the transforms, write simulators that generate paths for s and y . Transform the y values back to s values and compare the paths of s . Perform your comparison for alternative values of the time step Δ . Draw conclusions about the quality of the simulations. Be sure to use the same Brownian paths in computing the paths for s and y (also note that the transforms change the sign of the random term).

In solving this problem write a generic simulator with the following syntax

```
s=sdegenpath(s0,mu,sigma,dW,Delta,varargin)
```

Here μ and σ are functions and `varargin` represents additional parameters to pass to μ and σ . The easiest way to define these functions is to use what Matlab calls “anonymous functions.” For example, to define the original process use

```
mufunc = @(s,alpha,mu,sigma) alpha*s.*(mu-s);
sigmafunc = @(s,alpha,mu,sigma) sigma*s.^(1.5);
```

(this may not work if you are using version 6, in which case write m-files to define the functions). Call the simulator using

```
s=sdegenpath(s0,mufunc,sigmafunc,dW,Delta,alpha,mu,sigma);
```

Use the parameter values $\alpha = 0.1$, $\mu = 1$ and $\sigma = 0.3$.

5. Suppose that you want to price a call option that is based on the (risk-neutral) process

$$dS = \alpha(\theta - S)dt + \sigma SdW$$

Specifically, the call option pays at time T $\max(0, S_T - K)$. The current value of the option is $E[e^{-rT} \max(0, S_T - K) | S_0 = S]$.

a) Use Monte Carlo simulation to evaluate the current value of the option using the following parameter values: $\alpha = 0.5$, $\theta = 1$, $\sigma = 0.3$, $K = 1$, $T = 1$, $S = 1.1$, $r = 0.05$. Use 250 time step and 10000 time paths and provide an estimate of the quality of the approximation. I obtain a value of approximately 0.126.

b) The Black-Scholes call option pricing model assumes that $dS = rSdt + \sigma SdW$. Use this as a control variate to improve your estimate of the option value. How much improvement is obtained? (Note: the CompEcon Toolbox function `bs` can be used to obtain the Black-Scholes option premium with δ set to 0).