TMA150/MSG400 Assigment 6

Martin Hansson

2020-10-18

1 Introduction

This report covers *Assignment 6: Stochastic data processing and simulation* in the course TMA150/MSG400. The exercises are solved with use of *Python*

2 Problem description

We have a stochastic process $X : \Omega$ x $[0, 1] \to \mathbb{R}$ given by

$$X(t) = \exp\left(\left(\mu - \frac{\sigma^2}{2}\right)t - \sigma W(t)\right) \tag{1}$$

and its approximation

$$X_h(t_n) = (1 - h\mu)X_h(t_{n-1}) + \sigma X_h(t_{n-1})(W(t_n) - W(t_{n-1})) \tag{2}$$

with $X_h(0) = 1$. For all questions, a sequence of time descretization $h_i = 2^{-i}$ for i=1,...,10 and set $\mu = \sigma = 1$ are used.

2.1 Question 1

Compute and plot the *Brownian motion, W(t),* at all resolutions based on the same noise for all resolutions.

2.2 Question 2

Compute and plot the sample paths of X and X_h at all resolutions and same noise.

2.3 Question 3

Estimate the *strong error* of X_h based on $M = 1000$ Monte Carlo simulations. Plot the error w.r.t. the resolution together with the reference slope $h^{1/2}$ in a loglog-plot.

2.4 Question 4

Estimate the *weak error* of X_h based on $M = 1000$ Monte Carlo simulations. Plot the error w.r.t. the resolution together with the reference slope $h^{1/2}$ in a loglog-plot.

2.5 Question 5

Repeat step 4, but with another, non-linear, test function ϕ. Plot and describe your results.

3 Theory and implementation

A stochastic process is a mathematical description of a time-oriented random process. An example of a stochastic process is a *Brownian motion*, which is subject to study in this report. A *Brownian motion* can physically described as the random motion of free particles position inside a sub-domain of a fluid. As a mathematical object, a Brownian motion $W : \Omega \text{ x } \mathbb{T} \rightarrow \mathbb{R}$ is known as *Wiener process* and satisfies

- $W(0) = 0$,
- W has independent increments,
- $W(t) - W(s)$ $\mathcal{N}(0, t - s)$ for all $0 \leq s \leq t$

Basically you could say (for the 1-dimensional case) that for a standardized Brownian motion, at any time point T, the particle will be given a random velocity that is normally distributed with a mean value $\mu = 0$, a variance h and are independent of all previous velocities. By breaking out the variance, h, the motion between 2 time points can be rewritten as

$$W(t_{n+1}) - W(t_n) = \sqrt{h}\mathcal{N}(0, 1) \tag{3}$$

The stochastic process defined by equation 1, where the *W(t)* is found in the exponent, can be recognized as *Geometric Brownian Motion*. It is process that is extensively used in mathematical finance to model stock prices in the *Black-Scholes model*. The process is log-normal distributed and have an expected value of

$$E[X(T)] = X(0)\exp(\mu T) \tag{4}$$

Since simulation of exponential function are expensive when it comes to computation time, it could be beneficial to simplify the process with an approximation (see equation 2). With such an approximation, we want to know how well it approximates the true process. In this assignment, we are interested in the *weak error* and the *strong error* at the end of the process, i.e. $t = 1$.

If we compute the strong error with *Monte Carlo* based on M simulations it can be written as

$$err_s = E[(X(T) - X_h(T))^2]^{1/2} \approx \left(\frac{1}{M} \sum_{m=1}^{M} ((X(T)^{(m)} - X_h(T)^{(m)})^2 \right)^{1/2} \tag{5}$$

The weak error can be written, for an arbitrary test function ϕ, as

$$err_w = |E[\phi(X(T))] - E[\phi(X_h(T)]| \approx \left| E[\phi(X(T))] - \frac{1}{M} \sum_{m=1}^{M} \phi(X_h(T))^{(m)} \right| \tag{6}$$

It should be noted that the error depends both on the error of the approximation itself and the error of the Monte Carlo simulation. The error is then proportional to

$$err_{tot} \propto \frac{1}{\sqrt{M}} + h^{\gamma} \tag{7}$$

where γ is the order of convergence for the approximation. This means that for small h we also need large M to capture the error of the approximation.

4 Results

4.1 Question 1

The Brownian motion is shown in figure 1. It can be seen that all graphs are defined based on the same noise created at $h = 2^{-10}$ since they have same value for defined $t\acute{}s$.

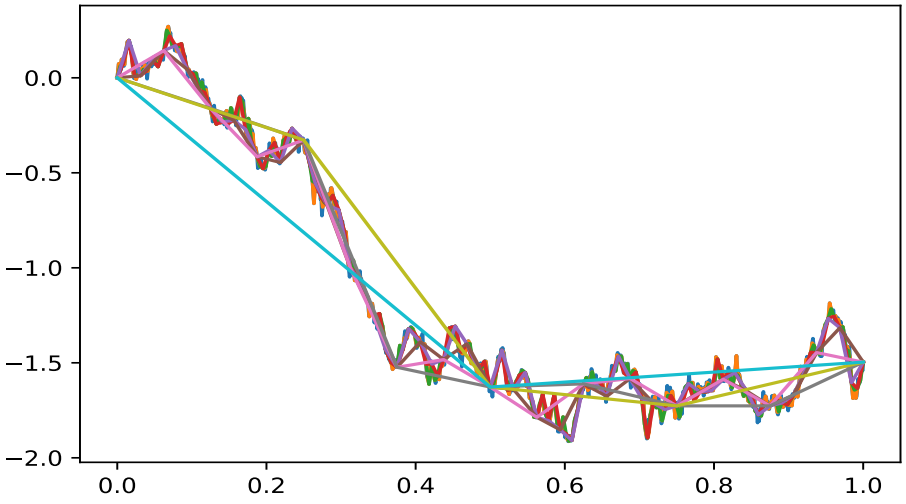

Figure 1: The Brownian Motion with resolution $h_i = 2^{-i}$, $i = 1, ...10$

4.2 Question 2

The sample path of X and X_h with resolution $h_i = 2^{-i}$, $i = 1, ...10$ is shown in figure 2. It can be seen that the error, $X - X_h$ at $t = 1$, increases as the resolution reduces.

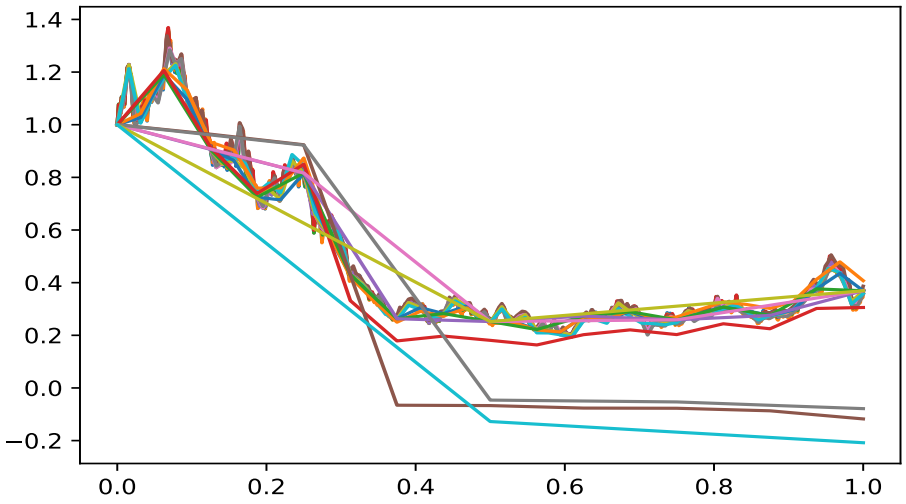

Figure 2: The sample path of X and X_h with resolution $h_i = 2^{-i}$, $i = 1, ...10$

4.3 Question 3

The strong error w.r.t. the time increments are shown in figure 3 for M=1000. It can be seen that the error $X - X_h$ is dominating since it seems to follow the reference slope. This is somewhat expected since

$$\frac{1}{\sqrt{M}} \approx \sqrt{h_{min}}$$

However it would not be surprising if you could see tendencies of the error stopped converging to for the lowest h.

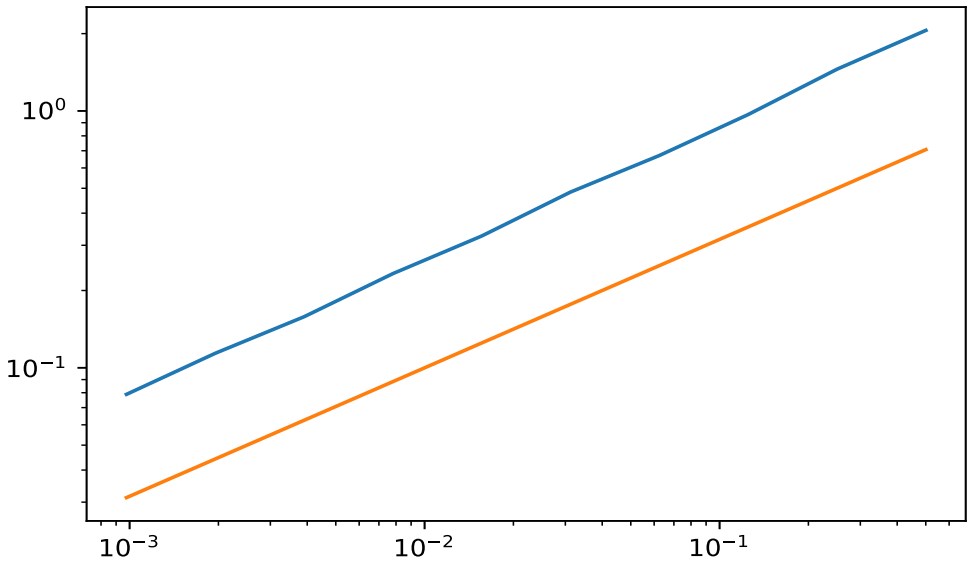

Figure 3: The strong error (blue) (y-axis) w.r.t. h (x-axis) for M=1000 together with the reference slope $\sqrt{h}$ (yellow)

4.4 Question 4

The weak error w.r.t. the time increments are shown in figure 3 for M=1000. It can be seen that the error $X - X_h$ is dominating for for large h since it seems to follow the reference slope. However, for lower h the error in Monte Carlo simulation takes over and the blue curve will stop follow the yellow curve. Compared for Question 3 the convergence rate for the approximation is now faster as is converges linearly, i.e.

$$\frac{1}{\sqrt{M}} > h_{min}$$

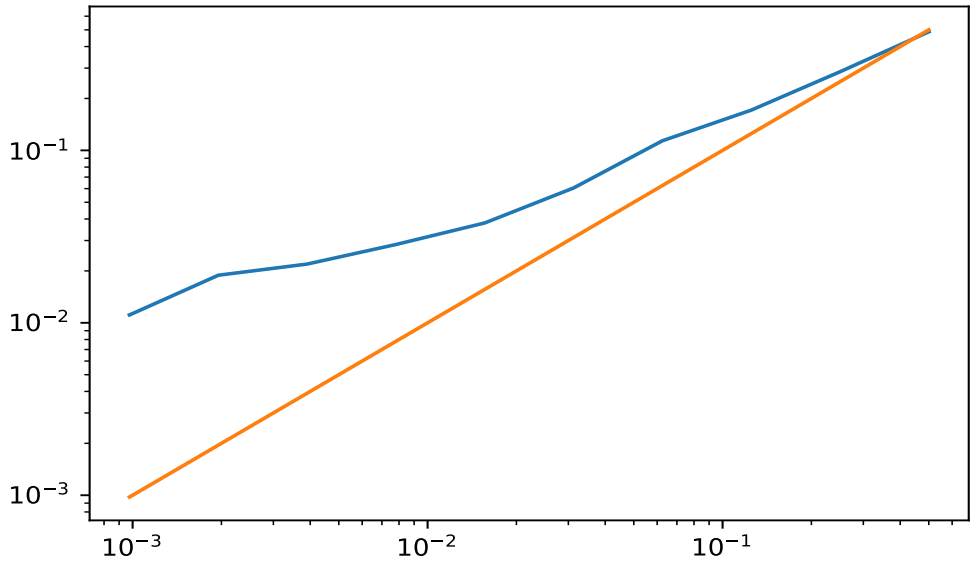

Figure 4: The weak error (blue) (y-axis) w.r.t. h (x-axis) for M=1000 together with the reference slope h (yellow)

4.5 Question 5

Let us define another test function

$$\phi = X^2$$

We know that $E[X(1)] = \exp(\mu)$ if $X(t)$ is defined by equation 1. We can write

$$\phi(X(1)) = \exp\left(2\left(\mu - \frac{\sigma^2}{2}\right)t - 2\sigma W(t)\right) = \exp\left(\left(2\mu - \frac{\sigma_{new}^2}{4}\right)t - \sigma_{new}W(1)\right)$$

$$= \exp\left(\left(2\mu - \frac{\sigma_{new}^2}{4} + \frac{\sigma_{new}^2}{2} - \frac{\sigma_{new}^2}{2}\right)t - \sigma_{new}W(1)\right) = \exp\left(\left(2\mu + \frac{\sigma_{new}^2}{4} - \frac{\sigma_{new}^2}{2}\right)t - \sigma_{new}W(1)\right)$$

Then we can define

$$\mu_{new} = 2\mu + \frac{\sigma_{new}^2}{4} = 2\mu + \sigma^2 \implies E[\phi(X(1))] = \exp(2\mu + \sigma^2)$$

Repeating Question 4 with the new expected value and with an M=100000, the new test function

gives the weak error shown in figure 5. It can be seen that the weak error with this test function comes out to be higher and requires a larger M in the Monte Carlo simulation to follow the reference slope.

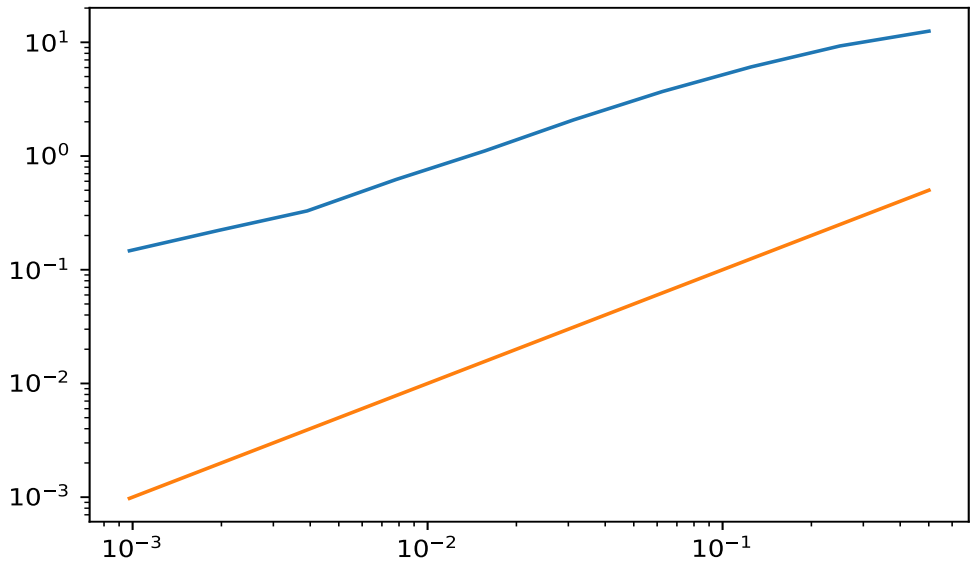

Figure 5: The weak error (blue) (y-axis) w.r.t. h (x-axis) for M=100000 together with the reference slope h (yellow). $\phi = X^2$

Appendix - code

Question 1-2 python-code

```
"""
Created on Fri Oct 16 10:24:05 2020

@author: Martin
"""
import random
import numpy as np
import matplotlib.pyplot as plt
mu=1
s=1

#Question 1 and 2
max_i=10
h=2**-max_i
N=1/h
t=np.arange(0,1+h,h)
W=np.zeros(int(N+1))
xh=np.ones(int(N+1))
x=np.ones(int(N+1))
eta=np.sqrt(h)*np.random.randn(int(N)) #the noise
for j in range(int(N)): #create W,xh and x for ith highest resolution
    W[j+1]=W[j]+eta[j]
    xh[j+1]=(1+h*mu)*xh[j]+s*xh[j]*(W[j+1]-W[j])
    x[j+1]=np.exp(((mu-s**2/2)*t[j+1])+s*W[j+1])
plt.figure(1) #plot for Brownian motion
plt.plot(np.arange(0,1+h,h),W)
plt.figure(2) #plot for xh motion
plt.plot(np.arange(0,1+h,h),x)
plt.plot(np.arange(0,1+h,h),xh)

for i in range(max_i-1,0,-1): #create W,xh,x for all other i's
    h=2**-i
    N=1/h
    W=np.zeros(int(N+1))
    xh=np.ones(int(N+1))
    x=np.ones(int(N+1))
    eta_n=np.zeros(int(N))
    t_n=np.zeros(int(N+1))
    for j in range(int(N)):
        eta_n[j]=eta[2*j]+eta[2*j+1]
        t_n[j+1]=t[2*(j+1)]
        W[j+1]=W[j]+eta_n[j]
        xh[j+1]=(1+h*mu)*xh[j]+s*xh[j]*(W[j+1]-W[j])
        x[j+1]=np.exp((mu-s**2/2)*t_n[j+1]+s*W[j+1])
    plt.figure(1) #plot for Brownian motion
    plt.plot(np.arange(0,1+h,h),W)
    plt.plot(np.arange(0,1+h,h),x)
    plt.figure(2) #plot for xh motion
    plt.plot(np.arange(0,1+h,h),xh)
    eta=eta_n
    t=t_n


plt.figure(1)
```

```
plt.savefig('Q1.pdf')
plt.figure(2)
plt.savefig('Q2.pdf')
```

Question 3-5 python-code

```
"""
Created on Fri Oct 16 12:15:48 2020

@author: Martin
"""
import random
import numpy as np
import matplotlib.pyplot as plt
mu=1
s=1
a=1
#Question 3-5
M=10000
max_i=7
err_s=np.zeros(max_i) #vector for strong error
E1=np.zeros(max_i) #vector for E in when phi=X
E2=np.zeros(max_i) #vector for E in when phi=X^2
for m in range(M):
    h=2**-max_i
    N=1/h
    t=np.arange(0,1+h,h)
    W=np.zeros(int(N+1))
    xh=np.ones(int(N+1))
    x=np.ones(int(N+1))
    eta=np.sqrt(h)*np.random.randn(int(N))
    for j in range(int(N)):
        W[j+1]=W[j]+eta[j]
        xh[j+1]=(1+h*mu)*xh[j]+s*xh[j]*(W[j+1]-W[j])
        x[j+1]=np.exp(((mu-s**2/2)*t[j+1])+s*W[j+1])
    err_s+=(x[j+1]-xh[j+1])**2
    E1[0]+=xh[j+1]
    E2[0]+=xh[j+1]**2
    for i in range(max_i-1,0,-1):
        h=2**-i
        N=1/h
        W=np.zeros(int(N+1))
        xh=np.ones(int(N+1))
        x=np.ones(int(N+1))
        eta_n=np.zeros(int(N))
        t_n=np.zeros(int(N+1))
        for j in range(int(N)):
            eta_n[j]=eta[2*j]+eta[2*j+1]
            t_n[j+1]=t[2*(j+1)]
            W[j+1]=W[j]+eta_n[j]
            xh[j+1]=(1+h*mu)*xh[j]+s*xh[j]*(W[j+1]-W[j])
            x[j+1]=np.exp((mu-s**2/2)*t_n[j+1]+s*W[j+1])
        err_s[max_i-i]+=(x[j+1]-xh[j+1])**2 #sum for strong error
        E1[max_i-i]+=xh[j+1] #sum for weak error phi=X
        E2[max_i-i]+=xh[j+1]**2 #sum for weak error with phi=X^2
        eta=eta_n #renew eta with doubled increment
        t=t_n #renew eta with doubled increment
```

```
err_s=np.sqrt(err_s/M)
err_w1=np.abs(np.exp(mu)-E1/M)
err_w2=np.abs(np.exp(2*mu+a*s)-E2/M)

step=np.zeros(max_i)
for i in range(max_i):
    step[i]=2**-(max_i-i)

plt.figure(3) #plot strong error
plt.loglog(step,err_s)
plt.loglog(step,np.sqrt(step))
plt.savefig('Q3.pdf')

plt.figure(4) #plot weak error phi=X
plt.loglog(step,err_w1)
plt.loglog(step,step)
plt.savefig('Q4.pdf')

plt.figure(5) #sum for weak error with phi=X
plt.loglog(step,err_w2)
plt.loglog(step,step)
plt.savefig('Q5.pdf')
```