Mathematical Finance

Solution sheet 10

Solution 10.1

(a) A deterministic function that is not of finite variation.

(b) A fractional Brownian motion with the Hurst index $H \neq \frac{1}{2}$. Indeed, take $(K^n) \subset \mathbb{S}$ defined as

$$K_t^n := \sum_{k=1}^{k_n} \alpha_k^n \mathbb{1}_{]t_{k-1}, t_k]}(t),$$

where $0 = t_0 < \cdots < t_k = T$, $\alpha_1^n = 0$ and $\alpha_k^n = n^{2H-1}(W_{t_{k-1}}^H - W_{t_{k-2}}^H)$. By the Hölder continuity of W^H, we have

$$\sup_{0 \leqslant t \leqslant T} |K_t^n| \xrightarrow{P} 0,$$

but, by the self-similarity of W^H, for the law of the stochastic integral, we have

$$\begin{aligned}(K^n \bullet W^H)_T &= n^{2H-1} \sum_{k=2}^{n} (W_{t_{k-1}}^H - W_{t_{k-2}}^H)(W_{t_k}^H - W_{t_{k-1}}^H) \\ &= T^{2H} \frac{1}{n} \sum_{k=2}^{n} (W_{k-1}^H - W_{k-2}^H)(W_k^H - W_{k-1}^H).\end{aligned}$$

The increments are stationary and the (weak) law of large numbers yields

$$(K^n \bullet W^H)_T \xrightarrow{P} T^{2H}(2^{2H-1} - 1) \neq 0 \text{ for } H \neq \frac{1}{2},$$

i.e., $X := W^H$ is not a good integrator for $H \neq \frac{1}{2}$.

Solution 10.2 Let h be a bounded measurable and consider the value process v for h

$$v(t, x) := E[h(S_T) \mid S_t = x].$$

For $t < T$, we have

$$E[h(S_T)] = E[E[h(S_T) \mid S_t = x]] = \int_0^\infty v(t, x) f(t, x) dx.$$

Differentiating both sides w.r.t. t yields

$$0 = \int_0^\infty [\frac{\partial v}{\partial t} f - v \frac{\partial f}{\partial t}] dx.$$

On the other hand, we know that the value process v satisfies

$$\frac{\partial v}{\partial t}(t, x) + \frac{1}{2}\sigma^2(t, x) x^2 \frac{\partial^2 v}{\partial x^2}(t, x) = 0, \; v(T, x) = h(x),$$

so, we get

$$0 = \int_0^\infty [-\frac{1}{2}\sigma^2 x^2 \frac{\partial^2 v}{\partial x^2} f + v \frac{\partial f}{\partial x}] dx.$$

Integrating by parts twice the first term, we obtain

$$0 = \int_0^\infty \left(-\frac{1}{2}\frac{\partial^2}{\partial x^2}[\sigma^2 x^2 f] + \frac{\partial f}{\partial x} \right) v dx.$$

From the arbitrariness of h (and hence of v), we obtain

$$\frac{\partial f}{\partial x} = \frac{1}{2}\frac{\partial^2}{\partial x^2}[\sigma^2 x^2 f]. \tag{1}$$

Finally,

$$\begin{aligned}
\frac{\partial C}{\partial T}(T,K) &= \int_K^\infty (x-K)\frac{\partial f}{\partial K}(T,x)dx \\
&\overset{(1)}{=} \frac{1}{2}\int_K^\infty (x-K)\frac{\partial^2}{\partial x^2}[\sigma^2 x^2 f](T,x)dx \\
&= -\frac{1}{2}\int_K^\infty \frac{\partial}{\partial x}[\sigma^2 x^2 f](T,x)dx \\
&= \frac{1}{2}\sigma^2(T,K)K^2 f(T,K) \\
&= \frac{1}{2}\sigma^2(T,K)K^2\frac{\partial^2 C}{\partial K^2}(T,K)
\end{aligned}$$

from which the formula follows.

Solution 10.3 We have

$$E_Q[Y_t] = E_Q\left[\int_0^t \kappa(\theta - Y_s)ds + \int_0^t \beta\sqrt{Y_t}dW_s\right] = \kappa\theta t - \kappa E_Q\left[\int_0^t Y_s ds\right].$$

By Tonelli's theorem, we obtain

$$E_Q[Y_t] = \kappa\theta t - \kappa\int_0^t E_Q[Y_s]ds,$$

i.e., $\xi_t := E_Q[Y_t]$ satisfies the ODE

$$\frac{d\xi_t}{dt} = \kappa\theta t - \kappa\xi_t$$

whose solution with the initial condition $\xi_0 = E_Q[Y_0] = Y_0$ is

$$E_Q[Y_t] = \xi_t = Y_0 e^{-\kappa t} + \theta(1 - e^{-\kappa t}).$$

By Tonelli's theorem, we obtain

$$\begin{aligned}
E_Q\left[\frac{1}{T}\int_0^T Y_t dt\right] &= \frac{1}{T}\int_0^T E_Q[Y_t]dt \\
&= \frac{1}{T}\int_0^T \left(Y_0 e^{-\kappa t} + \theta(1 - e^{-\kappa t})\right)dt \\
&= \frac{1 - e^{-\kappa T}}{\kappa T}Y_0 + \left(1 - \frac{1 - e^{-\kappa T}}{\kappa T}\right)\theta.
\end{aligned}$$

Solution 10.4

```
import numpy
import matplotlib.pyplot as plt
from math import sqrt, fabs, exp
from scipy.stats import norm


# Total time.
T = 100.0
# Number of steps.
N = 100
# Time step size
dt = T/N
# Number of realizations to generate.
m = 5000
# The the long-run Y
theta = 0.04
# The mean reversion of Y
kappa = 2.0
# The volatility of Y
beta = -0.05
# The initial value
y_0 = 0.01
# Create an empty array to store the realizations.
y = numpy.empty((m,N+1))
# Initial values of Y
y[:, 0] = y_0
# Generate a sample from the standard normal distribution
z = norm.rvs(size=y[:, 0].shape + (N,),loc=0,scale=1)

# Compute Y (Balanced Implict Scheme)
for i in range(m):
  for j in range(N):
    y[i,j+1]=y[i,j]+(kappa*(theta-y[i,j])*dt+beta*sqrt(y[i,j])*sqrt(dt)*z[i,j])
    /(1.0+kappa*dt+(beta*sqrt(dt)*fabs(z[i,j]))/(sqrt(y[i,j])))

# Print the theoretical value and the Monte-Carlo value
print theta + (1-exp(-kappa*T))/(kappa*T)*(y_0-theta),numpy.cumsum(y)[-1]/(m*N)
```