Mathematical Finance

Solution sheet 6

Solution 6.1

(a) Define the stopping time

$$\tau = \inf\{k \geqslant 1 : S_k - S_{k-1} = 1\}.$$

The trading strategy θ is given by

$$\theta_k = 2^{k-1}\mathbb{1}_{\{\tau > k-1\}},\ k \in \mathbb{N}.$$

The self-financing strategy $\varphi = (\eta, \theta)$ associated to $(V_0, \theta) = (0, \theta)$ is determined by the value process

$$\begin{aligned} V_k(\varphi) &= \sum_{j=1}^{k} 2^{j-1}\mathbb{1}_{\{\tau > j-1\}}(S_j - S_{j-1}) \\ &= \mathbb{1}_{\{\tau > k\}} \sum_{j=1}^{k} 2^{j-1}(-1) + \mathbb{1}_{\{\tau \leqslant k\}} \sum_{j=1}^{\tau} 2^{j-1}\mathbb{1}_{\{\tau > j-1\}}(S_j - S_{j-1}) \\ &= \mathbb{1}_{\{\tau > k\}}(1 - 2^k) + \mathbb{1}_{\{\tau \leqslant k\}} \left(\sum_{j=1}^{\tau-1} 2^{j-1}(-1) + 2^{\tau-1}(+1) \right) \\ &= \mathbb{1}_{\{\tau > k\}}(1 - 2^k) + \mathbb{1}_{\{\tau \leqslant k\}},\ k \in \mathbb{N}. \end{aligned}$$

(b) Since $(V_k(\varphi))_{k\in\mathbb{N}}$ is a martingale and $V_0 = 0$, we obtain

$$0 = E[V_k(\varphi)] = (1 - 2^k)P(\tau > k) + P(\tau \leqslant k).$$

Solving for $P(\tau \leqslant k) = \frac{2^k - 1}{2^k}$ and letting $k \to \infty$ yields that $P(\tau < \infty) = 1$. Since $P(\tau < \infty) = 1$, we obtain that

$$V_\infty(\varphi) = \lim_{k\to\infty} V_k(\varphi) = 1.$$

(c) Since $EY_1 = 1 \neq EY_0 = 0$, the process Y is not a martingale. To show that it is a local martingale, choose $\tau_n = \inf\{t \geqslant 0 : |Y_t| \geqslant n\} \wedge n$, $n \geqslant 1$. Let $s < t < 1$ and $A \in \mathcal{F}_s$. We have

$$E[\mathbb{1}_A(Y_t^{\tau_n} - Y_s^{\tau_n})] = E[\mathbb{1}_A(Y_{(\tau^n \wedge t)\vee s} - Y_s)].$$

By the optional stopping theorem, $(Y_t^{\tau_n})_{t\geqslant 0}$ is a uniformly bounded martingale on $t < 1$. Moreover, Y^{τ_n} is continuous at $t = 1$, and constant on $t \geqslant 1$. Therefore, $(Y_t^{\tau_n})_{t\geqslant 0}$ is a martingale, for every $n \geqslant 1$. This shows that Y is a local martingale.

Solution 6.2 Let X be a local martingale. It is enough to show that X is locally a locally integrable process. So, we can suppose that X is a martingale. Define the stopping times

$$\tau_n := \inf\{t \geqslant 0 : |X_t| \geqslant n\} \wedge n$$

for each $n \in \mathbb{N}$. We have $\tau_n \uparrow \infty$ and

$$\sup_{s\leqslant t} |X_s^{\tau_n}| \leqslant n + \mathbb{1}_{\{\tau_n \leqslant t\}}|X_{\tau_n}|,$$

where the right-hand side is integrable by the optional stopping theorem.

Solution 6.3 Let there be two such decompositions $X = M + A = \widetilde{M} + \widetilde{A}$ with $A = \widetilde{A} = 0$. Then $M - \widetilde{M} = \widetilde{A} - A$ is a predictable FV local martingale starting from zero, so, it is identically zero. Indeed, a local martingale is a predictable FV process if and only if it is constant. The equivalence follows from the fact that a local martingale is predictable if and only if it is continuous. To see this take a predictable martingale M and a bounded predictable stopping time τ. By the predictable stopping theorem, we have

$$M_\tau = E[M_\tau \mid \mathcal{F}_{\tau-}] = M_{\tau-}.$$

Solution 6.4

```
import numpy
from math import sqrt
import matplotlib.pyplot as plt

from brownian import brownian

def main():

    # The Wiener process parameter.
    delta = 1
    # Total time.
    T = 2.0
    # Number of steps.
    N = 1000
    # Time step size
    dt = T/N
    # Number of realizations to generate.
    m = 1000
    # Create empty arrays to store the realizations and integrals.
    x = numpy.empty((m,N+1))
    y = numpy.empty((m,N+1))
    z = numpy.empty((m,N+1))
    w = numpy.empty((m,N+1))
    # Initial values of x,y,z,w.
    x[:, 0] = 1
    y[:, 0] = 0
    z[:, 0] = 0

    # Simulate the paths
    brownian(x[:,0], N, dt, delta, out=x[:,1:])
    brownian(y[:,0], N, dt, delta, out=y[:,1:])
    brownian(z[:,0], N, dt, delta, out=z[:,1:])

    # Compute the paths of the inverse Bessel process
    for i in range(m):
      for j in range(N+1):
        w[i,j]=sqrt(x[i,j]**2+y[i,j]**2+z[i,j]**2)**(-1)

    # The mean for (b)
    print w.mean(axis=0)[N]

```

```
    # Stop the paths
    for i in range(m):
      for j in range(N):
        if w[i,j] >= 2:
          w[i,j:] = w[i,j]

    # The mean for (c)
    print w.mean(axis=0)[N]

    # Plot a (stopped) path of the inverse Bessel process
    t = numpy.linspace(0.0, N*dt, N+1)
    plt.step(t, w[1])
    plt.show()

if __name__ == "__main__":
    main()
```