Financial Risk Theory

Zhaozhi Liu
s2058599
University of Edinburgh

Section A

Question 1

Solution:
Let $V_t = |X_t - \theta^*|^2$, by Ito's formula, we have

$$\begin{aligned} dV_t = d(|X_t - \theta^*|^2) &= 2(X_t - \theta^*)dX_t + (dX_t)^2 \\ &= 2(X_t - \theta^*)(-h(X_t)dt + \sqrt{\frac{2}{\beta}}dB_t) + \frac{2}{\beta}dt \\ |X_t - \theta^*|^2 - |X_0 - \theta^*|^2 &= -2\int_0^t (X_s - \theta^*)h(X_s)ds + 2\sqrt{\frac{2}{\beta}}\int_0^t (X_s - \theta^*)dB_s + \frac{2}{\beta}t \end{aligned}$$

Since X_t is continuous on $[0, T]$, we have that $\mathbb{E}\left[\int_0^T X_t^2 dt\right] < \infty$. Therefore we have

$$\mathbb{E}\left[\int_0^t (X_s - \theta^*)dB_s\right] = 0$$

Then take expectation on both sides we will get

$$\mathbb{E}\left[|X_t - \theta^*|^2\right] = (X_0 - \theta^*)^2 - 2\mathbb{E}\left[\int_0^t (X_s - \theta^*)h(X_s)ds\right] + \frac{2}{\beta}t$$

By assumption 2

$$-\langle X_s - \theta^*, h(X_s) - h(\theta^*)\rangle \leq -m(X_s - \theta^*)^2 \quad \text{for some } m > 0$$

Therefore,

$$\mathbb{E}\left[|X_t - \theta^*|^2\right] \leq (X_0 - \theta^*)^2 - 2m\left[\int_0^t \mathbb{E}\left[|X_s - \theta^*|^2\right] ds\right] + \frac{2}{\beta}t$$

The RHS of above inequality is continuous with respect to t, therefore, for $t \in [0, T]$, the RHS of above is bounded.
Then we apply Gronwall's lemma here,

$$\mathbb{E}\left[|X_t - \theta^*|^2\right] \leq \left[(X_0 - \theta^*)^2 + \frac{2}{\beta}t\right] e^{-2mt} < \infty$$

Hence,

$$\sup_{t\in[0,T]} \mathbb{E}\left[|X_t - \theta^*|^2\right] < \infty$$

Since the above holds for all $\theta^* \in \mathbb{R}$, without loss of generality, take $\theta^* = 0$, we have

$$\sup_{t\in[0,T]} \mathbb{E}\left[|X_t|^2\right] < \infty$$

Question 2

Solution:
From Question 1 we have

$$|X_t - \theta^*|^2 - |X_0 - \theta^*|^2 = -2\int_0^t (X_s - \theta^*)\, h(X_s)\, ds + 2\sqrt{\frac{2}{\beta}}\int_0^t (X_s - \theta^*)\, dB_s + \frac{2}{\beta}t$$

Then we define the stopping time

$$\tau_R := \inf_t \{t : |X_t| \geq R\}$$

$$|X_{t\wedge\tau_R} - \theta^*|^2 - |X_0 - \theta^*|^2 = -2\int_0^{t\wedge\tau_R} (X_s - \theta^*)\, h(X_s)\, ds + 2\sqrt{\frac{2}{\beta}}\int_0^{t\wedge\tau_R} (X_s - \theta^*)\, dB_s + \frac{2}{\beta}(t\wedge\tau_R)$$

Since X_t is continuous on $[0, t\wedge\tau_R]$, we have that $\mathbb{E}\left[\int_0^{t\wedge\tau_R} X_t^2 dt\right] < \infty$. Therefore we have

$$\mathbb{E}\left[\int_0^{t\wedge\tau_R} (X_s - \theta^*) dB_s\right] = 0$$

Take expectation

$$\begin{aligned}
\mathbb{E}\left[|X_{t\wedge\tau_R} - \theta^*|^2\right] &= |X_0 - \theta^*|^2 + \mathbb{E}\left[\frac{2}{\beta}(t\wedge\tau_R)\right] - 2\mathbb{E}\left[\int_0^{t\wedge\tau_R} (X_s - \theta^*)\, h(X_s)\, ds\right] \\
&= |X_0 - \theta^*|^2 + \frac{2}{\beta}\int_0^t \underbrace{\mathbb{E}\left[\mathbf{1}_{s\leq\tau_R}\right]}_{\leq 1} ds + 2\int_0^t \mathbb{E}\underbrace{\left[-(X_{s\wedge\tau_R} - \theta^*)h(X_{s\wedge\tau_R})\right]}_{\leq -m(X_{s\wedge\tau_R} - \theta^*)^2} ds \\
&\leq |X_0 - \theta^*|^2 + \frac{2}{\beta}t - 2m\int_0^t \mathbb{E}\left[(X_{s\wedge\tau_R} - \theta^*)^2\right] ds < \infty
\end{aligned}$$

Similar argument can be put into above, due to continuity, we have the above is finite, by Gronwall's lemma, we have

$$\mathbb{E}\left[|X_{t\wedge\tau_R} - \theta^*|^2\right] \leq \left[(X_0 - \theta^*)^2 + \frac{2}{\beta}t\right] e^{-2mt}$$

By Fatou's lemma

$$\begin{aligned}
\mathbb{E}\left[|X_t - \theta^*|^2\right] &= \mathbb{E}\left[\lim_{R\to\infty} \inf |X_{t\wedge\tau_R} - \theta^*|^2\right] \\
&= \lim_{R\to\infty} \inf \mathbb{E}\left[|X_{t\wedge\tau_R} - \theta^*|^2\right] \\
&\leq \left[(X_0 - \theta^*)^2 + \frac{2}{\beta}t\right] e^{-2mt} < \infty
\end{aligned}$$

Hence,

$$\sup_{t\geq 0} \mathbb{E}\left[|X_t - \theta^*|^2\right] < \infty$$

Since the above holds for all $\theta^* \in \mathbb{R}$, without loss of generality, take $\theta^* = 0$, we have

$$\sup_{t\geq 0} \mathbb{E}\left[|X_t|^2\right] < \infty$$

Question 3

Solution:
Let $Y_t = X_t - Z_t$, we have

$$\begin{aligned}
\mathbb{E}\left[Y_t^2\right] &= \mathbb{E}\left[|X_t - Z_t|^2\right] \\
&= \mathbb{E}\left[|(X_t - \theta^*) - (Z_t - \theta^*)|^2\right] \\
&\leq 2\mathbb{E}\left[|X_t - \theta^*|^2\right] + 2\mathbb{E}\left[|Z_t - \theta^*|^2\right]
\end{aligned}$$

By using the result from Question 2, we have for some $m_1, m_2 > 0$, there is

$$\mathbb{E}\left[Y_t^2\right] \leq 2\left[(X_0-\theta^*)^2+\frac{2}{\beta}t\right]e^{-2m_1 t}+2\left[(X_0-\theta^*)^2+\frac{2}{\beta}t\right]e^{-2m_2 t}$$

By the definition of Wasserstein distance

$$\begin{aligned}
W_2\left(\mu_t,\nu_t\right) &= \inf_{\zeta\in C(\mu_t,\nu_t)}\left(\int_{\mathbb{R}}\int_{\mathbb{R}}\left|\theta-\theta'\right|^2\zeta\left(\mathrm{d}\theta\mathrm{d}\theta'\right)\right)^{1/2} \\
&\leq \left[\mathbb{E}\left[\left|X_t-Z_t\right|^2\right]\right]^{1/2} \\
&\leq \sqrt{2\left[(X_0-\theta^*)^2+\frac{2}{\beta}t\right]e^{-2m_1 t}+2\left[(X_0-\theta^*)^2+\frac{2}{\beta}t\right]e^{-2m_2 t}}
\end{aligned}$$

Therefore

$$0 \leq \lim_{t\to\infty} W_2(\mu_t,\nu_t) \leq 0$$

Thus, $\lim_{t\to\infty} W_2(\mu_t,\nu_t) = 0$

Question 4

Solution:
We have

$$Y_{(n+1)\lambda}-\theta^* = Y_{n\lambda}-\theta^*-\lambda h(Y_{n\lambda})+\sqrt{\frac{2\lambda}{\beta}}\xi_{n+1}$$

$$\begin{aligned}
(Y_{(n+1)\lambda}-\theta^*)^2 &= \left[(Y_{n\lambda}-\theta^*)-\lambda h(Y_{n\lambda})\right]^2+2\left[(Y_{n\lambda}-\theta^*)-\lambda h(Y_{n\lambda})\right]\sqrt{\frac{2\lambda}{\beta}}\xi_{n+1}+\frac{2\lambda}{\beta}\xi_{n+1}^2 \\
&= (Y_{n\lambda}-\theta^*)^2-2\lambda(Y_{n\lambda}-\theta^*)h(Y_{n\lambda})+\lambda^2h^2(Y_{n\lambda})+\frac{2\lambda}{\beta}\xi_{n+1}^2+2\left[(Y_{n\lambda}-\theta^*)-\lambda h(Y_{n\lambda})\right]\sqrt{\frac{2\lambda}{\beta}}\xi_{n+1}
\end{aligned}$$

Take $\mathbb{E}\left[\cdot|Y_{n\lambda}\right]$ on both sides, we have

$$\mathbb{E}\left[(Y_{(n+1)\lambda}-\theta^*)^2|Y_{n\lambda}\right] = (Y_{n\lambda}-\theta^*)^2-2\lambda(Y_{n\lambda}-\theta^*)h(Y_{n\lambda})+\lambda^2h^2(Y_{n\lambda})+\frac{2\lambda}{\beta} \tag{1}$$

By assumption 2 we have

$$-\langle Y_{n\lambda}-\theta^*, h(Y_{n\lambda})-h(\theta^*)\rangle \leq -m(Y_{n\lambda}-\theta^*)^2 \tag{2}$$

By assumption 1 we have

$$\begin{aligned}
\left|h(Y_{n\lambda})-h(\theta^*)\right| &\leq L\left|Y_{n\lambda}-\theta^*\right| \\
h^2(Y_{n\lambda}) &\leq L^2(Y_{n\lambda}-\theta^*)^2
\end{aligned} \tag{3}$$

Plug (2),(3) into (1), we have

$$\mathbb{E}\left[(Y_{(n+1)\lambda}-\theta^*)^2|Y_{n\lambda}\right] \leq (Y_{n\lambda}-\theta^*)^2-2m\lambda(Y_{n\lambda}-\theta^*)^2+\lambda^2L^2(Y_{n\lambda}-\theta^*)^2+\frac{2\lambda}{\beta}$$

Take expectation again, we have

$$
\begin{aligned}
\mathbb{E}\left[\left|Y_{(n+1)\lambda}-\theta^*\right|^2\right] &\leq (1-2m\lambda+\lambda^2L^2)\mathbb{E}\left[|Y_{n\lambda}-\theta^*|^2\right]+\frac{2\lambda}{\beta} \\
&\leq (1-2m\lambda+\lambda^2L^2)^{n+1}(Y_0-\theta^*)^2+\frac{2\lambda}{\beta}\sum_{k=0}^{n}(1-2m\lambda+\lambda^2L^2)^k
\end{aligned}
\tag{4}
$$

Take $1-2m\lambda+\lambda^2L^2<1$, we have $\lambda(\lambda L-2m)<0$, which shows $0<\lambda<\dfrac{2m}{L}=c(m,L)$ Then the summand of the second term of (4) is convergent as $n\to\infty$, we have

$$
\begin{aligned}
\mathbb{E}\left[\left|Y_{(n+1)\lambda}-\theta^*\right|^2\right] &\leq (1-2m\lambda+\lambda^2L^2)^{n+1}(Y_0-\theta^*)^2+\frac{2\lambda}{\beta}\cdot\frac{1}{2m\lambda+\lambda^2L^2} \\
&= (1-2m\lambda+\lambda^2L^2)^{n+1}(Y_0-\theta^*)^2+\frac{2}{2m\beta+\lambda L^2}<\infty
\end{aligned}
$$

Thus, $\mathbb{E}\left[|Y_{n\lambda}-\theta^*|^2\right]<\infty$, from the previous questions, we have

$$
\sup_{n\geq 0}\mathbb{E}\left[|Y_{n\lambda}-\theta^*|^2\right]<\infty \quad\Rightarrow\quad \sup_{n\geq 0}\mathbb{E}\left[Y_{n\lambda}^2\right]<\infty
$$

Question 5

Solution:
We have

$$
\begin{cases}
\mathrm{d}X_t = & -h\left(X_t\right)dt+\sqrt{2\beta^{-1}}dB_t \\
dY_t^\lambda = & -h\left(Y_{\kappa_\lambda(t)}^\lambda\right)dt+\sqrt{2\beta^{-1}}dB_t \\
X_0=Y_0^\lambda &
\end{cases}
$$

Let $Z_t=X_t-Y_t^\lambda$, then $dZ_t=dX_t-dY_t^\lambda=-(h(X_t)-h(Y_{\kappa_\lambda(t)}^\lambda))dt$, by Ito's formula, we have

$$
\begin{aligned}
dZ_t^2 &= 2Z_tdt+(dZ_t)^2 \\
&= -2(X_t-Y_t^\lambda)(h(X_t)-h(Y_{\kappa_\lambda(t)}^\lambda))dt \\
&= -2(X_t-Y_t^\lambda)(h(X_t)-h(Y_t^\lambda))dt-2\sqrt{m}(X_t-Y_t^\lambda)\cdot\frac{1}{\sqrt{m}}\left(h(Y_t^\lambda)-h(Y_{\kappa_\lambda(t)}^\lambda)\right)dt
\end{aligned}
$$

$$
Z_t^2=-2\int_0^t(X_s-Y_s^\lambda)\left(h(X_s)-h(Y_s^\lambda)\right)ds-2\int_0^t\sqrt{m}(X_s-Y_s^\lambda)\cdot\frac{1}{\sqrt{m}}\left(h(Y_s^\lambda)-h(Y_{\kappa_\lambda(s)}^\lambda)\right)ds
$$

Where m is the constant satisfies assumption 2

$$
-\langle X_s-Y_s^\lambda,h(X_s)-h(Y_s^\lambda)\rangle\leq -m(X_s-Y_s^\lambda)^2
$$

Take expectation on both sides we have

$$
\begin{aligned}
\mathbb{E}\left[Z_t^2\right]=&-2\mathbb{E}\left[\int_0^t(X_s-Y_s^\lambda)\left(h(X_s)-h(Y_s^\lambda)\right)ds\right] \\
&-2\mathbb{E}\left[\int_0^t\sqrt{m}(X_s-Y_s^\lambda)\cdot\frac{1}{\sqrt{m}}\left(h(Y_s^\lambda)-h(Y_{\kappa_\lambda(s)}^\lambda)\right)ds\right]
\end{aligned}
$$

Using the inequality $2\langle X, Y\rangle \leq X^2 + Y^2$, assumption 2, and assumption 1, we have

$$\begin{aligned}
\mathbb{E}\left[Z_t^2\right] &\leq -2m\int_0^t \mathbb{E}\left[(X_s - Y_s^\lambda)^2\right] ds + m\int_0^t \mathbb{E}\left[(X_s - Y_s^\lambda)^2\right] ds + \frac{1}{m}\int_0^t \mathbb{E}\left[\left(h(Y_s^\lambda) - h(Y_{\kappa_\lambda(s)}^\lambda)\right)^2\right] ds \\
&\leq -m\int_0^t \mathbb{E}\left[(X_s - Y_s^\lambda)^2\right] ds + \frac{L_1^2}{m}\int_0^t \mathbb{E}\left[\left(Y_s^\lambda - Y_{\kappa_\lambda(s)}^\lambda\right)^2\right] ds \\
&\leq \frac{L_1^2}{m}\int_0^t \mathbb{E}\left[\left(Y_s^\lambda - Y_{\kappa_\lambda(s)}^\lambda\right)^2\right] ds
\end{aligned}$$

L_1 is the constant that $\left(h(Y_s^\lambda) - h(Y_{\kappa_\lambda(s)}^\lambda)\right)^2 \leq L_1^2\left(Y_s^\lambda - Y_{\kappa_\lambda(s)}^\lambda\right)^2$
By the definition of Y_t^λ, we have the following

$$Y_t^\lambda - Y_{\kappa_\lambda(t)}^\lambda = -\int_{\kappa_\lambda(t)}^t h(Y_s^\lambda) ds + \sqrt{2\beta^{-1}}(B_t - B_{\kappa_\lambda(t)})$$

Take square and expectation we have

$$\mathbb{E}\left[(Y_t^\lambda - Y_{\kappa_\lambda(s)}^\lambda)^2\right] = \mathbb{E}\left[h^2(Y_{\kappa_\lambda(t)}^\lambda)\right](t - \kappa_\lambda(t))^2 + \frac{2}{\beta}(t - \kappa_\lambda(t))$$

Since $t - \kappa_\lambda(t) \leq \lambda$, we have

$$\mathbb{E}\left[\left(Y_t^\lambda - Y_{\kappa_\lambda(t)}^\lambda\right)^2\right] \leq \lambda^2\mathbb{E}\left[h^2(Y_{\kappa_\lambda(t)}^\lambda)\right] + \frac{2}{\beta}\lambda$$

By using assumption 1 again we have

$$h^2\left(Y_{\kappa_\lambda(t)}^\lambda\right) \leq L_2^2\left(Y_{\kappa_\lambda(t)}^\lambda - \theta^*\right)^2 \leq 2L_2^2\left(Y_{\kappa_\lambda(t)}^\lambda\right)^2 + 2L_2^2(\theta^*)^2$$

Finally we have

$$\mathbb{E}\left[\left(Y_t^\lambda - Y_{\kappa_\lambda(t)}^\lambda\right)^2\right] \leq 2L_2^2\lambda^2\mathbb{E}\left[(Y_{\kappa_\lambda(t)}^\lambda)^2\right] + 2L_2^2\lambda^2(\theta^*)^2 + \frac{2}{\beta}\lambda$$

Since from question 4 we have $\lambda < \frac{2m}{L^2} = c$, we have

$$\mathbb{E}\left[\left(Y_t^\lambda - Y_{\kappa_\lambda(t)}^\lambda\right)^2\right] \leq \alpha\lambda$$

Where

$$\alpha = \max\left\{2L_2^2 c \cdot \left[\sup_{t\geq 0}\mathbb{E}\left[(Y_{\kappa_\lambda(t)}^\lambda)^2\right] + (\theta^*)^2\right], \frac{2}{\beta}\right\}$$

Thus,

$$\mathbb{E}\left[Z_t^2\right] \leq \frac{L_1^2}{m}\alpha\lambda t$$

Therefore

$$W_2\left(\mu_t, \nu_t^\lambda\right) \leq \left[\mathbb{E}\left[(X_t - Y_t^\lambda)^2\right]\right]^{1/2} \leq L_1\sqrt{\frac{\alpha t}{m}} \cdot \sqrt{\lambda}$$

We finally have

$$\lim_{\lambda\to 0} W_2\left(\mu_t, \nu_t^\lambda\right) = 0$$

Section B

Task 1

The code is given below

```
import numpy as np
rng = np.random.default_rng()
n=10**6
def secB(sample,q,n = 10**6):
   beta = 10**8
   gamma = 10**(-8)
   lamda = 10**(-4)
   xi = rng.normal(size = n)
   theta = 0
   for i in range(n):
      if sample[i]<theta:
         H = -(q/(1-q)) + 1/(1-q) + 2*gamma*theta
      else:
         H = -(q/(1-q)) + 2*gamma*theta
      theta = theta - lamda * H + (2*beta**(-1)*lamda)**(1/2)*xi[i]

   sgld_var = theta
   sgld_cvar = np.mean(np.array([max(sample[i]-theta,0) for i in
       range(n)])*(1/(1-q))+theta)+gamma*theta**2
   return sgld_var,sgld_cvar
```

The above block is a Python function to calculate the SGLD VaR and CVaR for a single randomly generated sample.

```
def get_approx(q,n=30):
   approx = []
   for i in range(n):
      sample = rng.normal(loc=0,scale=1,size=10**6)# need changing with respect to
          different parameters of dist.
      [sgld_var,sgld_cvar] = secB(sample,q)
      approx = np.append(np.array(approx),[sgld_var,sgld_cvar],axis = 0)
   approx = np.reshape(approx,[n,2])
   sgld_var,sgld_cvar = np.mean(approx,axis = 0)
   sgld_var_std,sgld_cvar_std = np.std(approx,axis = 0)
   return sgld_var,sgld_var_std,sgld_cvar,sgld_cvar_std
```

The above block is to calculate the mean and standard deviation of the approximated VaR and CVaR **based on** $n = 30$ **samples, my computer cannot afford 10000 samples to calculate.**

The theoretical value can be found in the paper.

q=0.95, Normal distribution with $\mu = 0$, $\sigma = 1$

```
sgld_var,sgld_var_std,sgld_cvar,sgld_cvar_std=get_approx(0.95)
print(f'The SGLD estimated VaR is {round(sgld_var,4)}, the standard deviation is
    {round(sgld_var_std,4)}')
print(f'The SGLD estimated CVaR is {round(sgld_cvar,4)}, the standard deviation
    is {round(sgld_cvar_std,6)}')
```

The result is[1]

```
The SGLD estimated VaR is 1.6466, the standard deviation is 0.0194
The SGLD estimated CVaR is 2.063, the standard deviation is 0.002615
```

q=0.95, Normal distribution with $\mu = 1$, $\sigma = 2$

```
The SGLD estimated VaR is 4.2823, the standard deviation is 0.0295
The SGLD estimated CVaR is 5.1249, the standard deviation is 0.00433
```

q=0.95, Normal distribution with $\mu = 3$, $\sigma = 5$

```
The SGLD estimated VaR is 11.2265, the standard deviation is 0.0482
The SGLD estimated CVaR is 13.3117, the standard deviation is 0.012855
```

q=0.99, Normal distribution with $\mu = 0$, $\sigma = 1$

```
The SGLD estimated VaR is 2.3241, the standard deviation is 0.0417
The SGLD estimated CVaR is 2.6666, the standard deviation is 0.005516
```

q=0.99, Normal distribution with $\mu = 1$, $\sigma = 2$

```
The SGLD estimated VaR is 5.657, the standard deviation is 0.0658
The SGLD estimated CVaR is 6.3319, the standard deviation is 0.008921
```

q=0.99, Normal distribution with $\mu = 3$, $\sigma = 5$

```
The SGLD estimated VaR is 2.3274, the standard deviation is 0.0424
The SGLD estimated CVaR is 2.6691, the standard deviation is 0.007252
```

[1]For each case, change the variable 'sample' in the Python function get_approx(q), q is the quantile and run the above block

q=0.95 t-distribution df = 10

```
The SGLD estimated VaR is 1.8156, the standard deviation is 0.0257
The SGLD estimated CVaR is 2.4086, the standard deviation is 0.003936
```

q=0.95 t-distribution df = 7

```
The SGLD estimated VaR is 1.8917, the standard deviation is 0.0258
The SGLD estimated CVaR is 2.5949, the standard deviation is 0.004596
```

q=0.95 t-distribution df = 3

```
The SGLD estimated VaR is 2.3442, the standard deviation is 0.0246
The SGLD estimated CVaR is 3.8765, the standard deviation is 0.012767
```

q=0.99 t-distribution df = 10

```
The SGLD estimated VaR is 2.7632, the standard deviation is 0.0506
The SGLD estimated CVaR is 3.3647, the standard deviation is 0.008903
```

q=0.99 t-distribution df = 7

```
The SGLD estimated VaR is 3.0035, the standard deviation is 0.0594
The SGLD estimated CVaR is 3.7731, the standard deviation is 0.011075
```

q=0.99 t-distribution df = 3

```
The SGLD estimated VaR is 4.5434, the standard deviation is 0.0708
The SGLD estimated CVaR is 7.0112, the standard deviation is 0.051456
```

Task 2

The code for the SGLD algorithm is given below

```
import numpy as np
import math
rng = np.random.default_rng()

def SecBTtask2(asset1,asset2,asset3,q=0.95):
   beta = 10**8
   gamma = 10**(-8)
```

```
    lamda = 10**(-4)
    n = 10**6
    xi = rng.normal(size = (n,4))
    theta_hat = np.array([0,1/3,1/3,1/3])
    H_hat = np.array([0,0,0,0])
    for i in range(n):
        sum_g = math.exp(theta_hat[1])+math.exp(theta_hat[2])+math.exp(theta_hat[3])
        g_1_omega = math.exp(theta_hat[1])/sum_g
        g_2_omega = math.exp(theta_hat[2])/sum_g
        g_3_omega = math.exp(theta_hat[3])/sum_g
        characterfun = g_1_omega*asset1[i] + g_2_omega*asset2[i]+ g_3_omega*asset3[i]
# you need indent twice for g_hat_1 2 3 for them can be run
g_hat_1 =
    (math.exp(theta_hat[1])*(math.exp(theta_hat[2])+math.exp(theta_hat[3])))/(sum_g**2)*asset1[i]-\
math.exp(theta_hat[1])*math.exp(theta_hat[2])/(sum_g**2)*asset2[i]-\
math.exp(theta_hat[1])*math.exp(theta_hat[3])/(sum_g**2)*asset3[i]
g_hat_2 = -math.exp(theta_hat[1])*math.exp(theta_hat[2])/(sum_g**2)*asset1[i]+\
(math.exp(theta_hat[2])*(math.exp(theta_hat[1])+math.exp(theta_hat[3])))/(sum_g**2)*asset2[i]-\
math.exp(theta_hat[2])*math.exp(theta_hat[3])/(sum_g**2)*asset3[i]
g_hat_3 = -math.exp(theta_hat[1])*math.exp(theta_hat[3])/(sum_g**2)*asset1[i]-\
math.exp(theta_hat[3])*math.exp(theta_hat[2])/(sum_g**2)*asset2[i]+\
(math.exp(theta_hat[3])*(math.exp(theta_hat[1])+math.exp(theta_hat[2])))/(sum_g**2)*asset3[i]
# indent twice above (from g_hat_1 to g_hat_3)
        if characterfun >= theta_hat[0]:
            H_hat[0] = 1-1/(1-q) + 2*gamma*theta_hat[0]
            H_hat[1] = g_hat_1/(1-q) + 2*gamma*theta_hat[1]
            H_hat[2] = g_hat_2/(1-q) + 2*gamma*theta_hat[2]
            H_hat[3] = g_hat_3/(1-q) + 2*gamma*theta_hat[3]
        else:
            H_hat[0] = 1+2*gamma*theta_hat[0]
            H_hat[1] = 2*gamma*theta_hat[1]
            H_hat[2] = 2*gamma*theta_hat[2]
            H_hat[3] = 2*gamma*theta_hat[3]
        theta_hat = theta_hat - lamda * H_hat + (2*beta**(-1)*lamda)**(1/2)*xi[i]
    sum_g = math.exp(theta_hat[1])+math.exp(theta_hat[2])+math.exp(theta_hat[3])
    g_1_omega = math.exp(theta_hat[1])/sum_g
    g_2_omega = math.exp(theta_hat[2])/sum_g
    g_3_omega = math.exp(theta_hat[3])/sum_g
    sample = g_1_omega*asset1+g_2_omega*asset2+g_3_omega*asset3
    var = np.quantile(sample,q)
    cvar = sample[sample >= var].mean()
    return var,cvar,g_1_omega,g_2_omega,g_3_omega
```

Case 1

The code is given by

```
def get_approx(n=10):
    approx = []
    for k in range(n):
```

```
        asset1=rng.normal(loc = 500, scale = 1, size = 10**6)
        asset2=rng.normal(loc = 0,scale = 10**3, size = 10**6)
        asset3=rng.normal(loc = 0,scale = 10**(-2), size = 10**6)
        # above need changing with respect to different parameters of dist.
        [var,cvar,g_1_omega,g_2_omega,g_3_omega] =
            SecBTtask2(asset1,asset2,asset3,q=0.95)
        approx =
            np.append(np.array(approx),[var,cvar,g_1_omega,g_2_omega,g_3_omega],axis
            = 0)
    approx = np.reshape(approx,[n,5])
    var,cvar,g_1_omega,g_2_omega,g_3_omega = np.mean(approx,axis = 0)
    return var,cvar,g_1_omega,g_2_omega,g_3_omega
var,cvar,g_1_omega,g_2_omega,g_3_omega=get_approx()
print(f'The SGLD estimated VaR is {round(var,4)}')
print(f'The SGLD estimated CVaR is {round(cvar,4)}')
print(f'The estimated weights is {round(g_1_omega,4)}, {round(g_2_omega,4)},
    {round(g_3_omega,4)}')
```

The result is

```
The SGLD estimated VaR is 0.0408
The SGLD estimated CVaR is 0.0471
The estimated weights is 0.0, 0.0, 1.0
```

Case 2

The code is given by

```
def get_approx(n=10):
    approx = []
    for k in range(n):
        asset1=rng.normal(loc = 500, scale = 1, size = 10**6)
        asset2=rng.normal(loc = 0,scale = 10**3, size = 10**6)
        asset3=rng.normal(loc = 0,scale = 1, size = 10**6)# need changing with
            respect to different parameters of dist.
        [var,cvar,g_1_omega,g_2_omega,g_3_omega] =
            SecBTtask2(asset1,asset2,asset3,q=0.95)
        approx =
            np.append(np.array(approx),[var,cvar,g_1_omega,g_2_omega,g_3_omega],axis
            = 0)
    approx = np.reshape(approx,[n,5])
    var,cvar,g_1_omega,g_2_omega,g_3_omega = np.mean(approx,axis = 0)
    return var,cvar,g_1_omega,g_2_omega,g_3_omega
var,cvar,g_1_omega,g_2_omega,g_3_omega=get_approx()
print(f'The SGLD estimated VaR is {round(var,4)}')
print(f'The SGLD estimated CVaR is {round(cvar,4)}')
print(f'The estimated weights is {round(g_1_omega,4)}, {round(g_2_omega,4)},
    {round(g_3_omega,4)}')
```

the result is

```
The SGLD estimated VaR is 1.6598
The SGLD estimated CVaR is 2.0779
The estimated weights is 0.0, 0.0, 1.0
```

Case 3

The code is given by

```
def get_approx(n=10):
   approx = []
   for k in range(n):
      asset1=rng.normal(loc = 0, scale = 10**(3/2), size = 10**6)
      asset2=rng.normal(loc = 0,scale = 1, size = 10**6)
      asset3=rng.normal(loc = 0,scale = 2, size = 10**6)# need changing with
          respect to different parameters of dist.
      [var,cvar,g_1_omega,g_2_omega,g_3_omega] =
          SecBTtask2(asset1,asset2,asset3,q=0.95)
      approx =
          np.append(np.array(approx),[var,cvar,g_1_omega,g_2_omega,g_3_omega],axis
          = 0)
   approx = np.reshape(approx,[n,5])
   var,cvar,g_1_omega,g_2_omega,g_3_omega = np.mean(approx,axis = 0)
   return var,cvar,g_1_omega,g_2_omega,g_3_omega
var,cvar,g_1_omega,g_2_omega,g_3_omega=get_approx()
print(f'The SGLD estimated VaR is {round(var,4)}')
print(f'The SGLD estimated CVaR is {round(cvar,4)}')
print(f'The estimated weights is {round(g_1_omega,4)}, {round(g_2_omega,4)},
    {round(g_3_omega,4)}')
```

The result is

```
The SGLD estimated VaR is 1.4715
The SGLD estimated CVaR is 1.8448
The estimated weights is 0.0016, 0.7981, 0.2002
```

Case 4

The code is given by

```
def get_approx(n=10):
   approx = []
   for k in range(n):
      asset1=rng.normal(loc = 0, scale = 1, size = 10**6)
      asset2=rng.normal(loc = 1, scale = 2, size = 10**6)
      asset3=rng.normal(loc = 0, scale = 10**(-2), size = 10**6)
      # need changing with respect to different parameters of dist.
      [var,cvar,g_1_omega,g_2_omega,g_3_omega] =
          SecBTtask2(asset1,asset2,asset3,q=0.95)
```

```
        approx =
            np.append(np.array(approx),[var,cvar,g_1_omega,g_2_omega,g_3_omega],axis
            = 0)
    approx = np.reshape(approx,[n,5])
    var,cvar,g_1_omega,g_2_omega,g_3_omega = np.mean(approx,axis = 0)
    return var,cvar,g_1_omega,g_2_omega,g_3_omega
var,cvar,g_1_omega,g_2_omega,g_3_omega=get_approx()
print(f'The SGLD estimated VaR is {round(var,4)}')
print(f'The SGLD estimated CVaR is {round(cvar,4)}')
print(f'The estimated weights is {round(g_1_omega,4)}, {round(g_2_omega,4)},
    {round(g_3_omega,4)}')
```

The result is

```
The SGLD estimated VaR is 0.0352
The SGLD estimated CVaR is 0.0428
The estimated weights is 0.0119, 0.005, 0.9831
```

Case 5

The code is given by

```
def get_approx(n=10):
    approx = []
    for k in range(n):
        asset1=rng.normal(loc = 0, scale = 1, size = 10**6)
        asset2=rng.normal(loc = 1,scale = 2, size = 10**6)
        asset3=rng.normal(loc = 2,scale = 1, size = 10**6)# need changing with
            respect to different parameters of dist.
        [var,cvar,g_1_omega,g_2_omega,g_3_omega] =
            SecBTtask2(asset1,asset2,asset3,q=0.95)
        approx =
            np.append(np.array(approx),[var,cvar,g_1_omega,g_2_omega,g_3_omega],axis
            = 0)
    approx = np.reshape(approx,[n,5])
    var,cvar,g_1_omega,g_2_omega,g_3_omega = np.mean(approx,axis = 0)
    return var,cvar,g_1_omega,g_2_omega,g_3_omega
var,cvar,g_1_omega,g_2_omega,g_3_omega=get_approx()
print(f'The SGLD estimated VaR is {round(var,4)}')
print(f'The SGLD estimated CVaR is {round(cvar,4)}')
print(f'The estimated weights is {round(g_1_omega,4)}, {round(g_2_omega,4)},
    {round(g_3_omega,4)}')
```

The result is

```
The SGLD estimated VaR is 1.6345
The SGLD estimated CVaR is 2.0035
The estimated weights is 0.8559, 0.1035, 0.0405
```