SOLUTIONS 9

Solution 9.1. The code is given below. Note that we have implemented the model as described in the exercise. The code is given below.

```
import numpy as np
import matplotlib.pyplot as plt

T = 100
x = np.zeros(T)
y = np.zeros(T)

rng = np.random.default_rng(1234)

x0 = 1 # initial value

a = 0.9
sig_x = 0.01
sig_y = 0.1

x[0] = a * x0 + rng.normal(0, sig_x, 1) # this is x_1 on paper, but x[
    0] in code
y[0] = x[0] + rng.normal(0, sig_y, 1)

for t in range(1, T):
    x[t] = a * x[t-1] + rng.normal(0, sig_x, 1)
    y[t] = x[t] + rng.normal(0, sig_y, 1)

plt.figure(figsize=(15, 8))
plt.plot(x, 'k', label='x')
plt.plot(y, 'r', label='y')
plt.legend()
plt.show()
```

In real world, this describes a process that converges to zero. Can model a number of things, such as a decaying system, or a system that converges to a fixed point.

Solution 9.2. We will use a generic volatility model below:

$$
\begin{aligned}
x_0 &\sim \mathcal{N}\left(x; \mu, \frac{\sigma^2}{1-\phi^2}\right), \\
x_t | x_{t-1} &\sim \mathcal{N}\left(x_t; \mu + \phi(x_{t-1} - \mu), \sigma^2\right), \\
y_t | x_t &\sim \mathcal{N}\left(y_t; 0, \exp(x_t)\right).
\end{aligned}
$$

This does an intuitive job: if x_t is high, then the variance of y_t is high, if x_t is low, then the variance of y_t is low. Here x_t are log-volatilities and y_t are log-returns. The code is given below.

```
import numpy as np
import matplotlib.pyplot as plt

# we will use a generic volatility model below

T = 10000

x = np.zeros(T)
y = np.zeros(T)
```

```

rng = np.random.default_rng(1234)

mu = 0.1
phi = 0.99
sig = 0.2

x0 = rng.normal(mu, sig**2/(1-phi**2), 1)

x[0] = rng.normal(mu + phi * (x0 - mu), sig, 1)
y[0] = rng.normal(0, np.exp(x[0]), 1)

for t in range(1, T):
    x[t] = rng.normal(mu + phi * (x[t-1] - mu), sig, 1)
    y[t] = rng.normal(0, np.exp(x[t]), 1)

plt.figure(figsize=(15, 8))
plt.subplot(2, 1, 1)
plt.plot(x, 'k', label='x')
plt.legend()
plt.subplot(2, 1, 2)
plt.plot(y, 'r', label='y')
plt.legend()
plt.show()
```

Solution 9.3. As we discussed in the class (watch the lecture for this part, if you have not):

1. Recall that we would like to sample the marginal posterior $p(\theta|y_{1:T})$. This can be done by sampling $p(x_{0:T}, \theta|y_{1:T})$, then keeping the θ part and discarding the $x_{0:T}$ part. For this, we would need to perform

   - Sample $x^(k)_{0:T} \sim p(x_{0:T}|y_{1:T}, \theta^{(k-1)})$
   - Sample $\theta^{(k)} \sim p(\theta|x^{(k)}_{0:T}, y_{1:T})$

   for $k = 1, \ldots, K$. A good candidate to sample the first part is a particle *smoother*, which we have not covered (and will be part of the mastery material).

2. Let us try to derive a Metropolis-within-Gibbs sampler, by sampling each variable X_t instead of a block sampling approach as described above. For this we need to derive the full conditionals. Note that, using conditional independence (watch the lecture), we can write

   $$p(x_t|x_{-t}, \theta, y_{1:T}) \propto f(x_t|x_{t-1}, \theta)g(y_t|x_t, \theta)f(x_{t+1}|x_t, \theta).$$

   except for $t = 0$ where we have

   $$p(x_0|x_{-0}, \theta, y_{1:T}) \propto \mu(x_0|\theta)f(x_1|x_0, \theta).$$

   Therefore, one can define a Metropolis within Gibbs sampler performing following steps:

   - Sample $x_0^{(k)} \sim \mu(x_0|\theta^{(k-1)})f(x_1^{k-1}|x_0, \theta^{(k-1)})$.
   - Sample $x_1^{(k)} \sim f(x_1|x_0^{(k)}, \theta^{(k-1)})g(y_1|x_1, \theta^{(k-1)})f(x_2^{(k-1)}|x_1, \theta^{(k-1)})$.

     $\vdots$

- Sample $x_t^{(k)} \sim f(x_t|x_{t-1}^{(k)}, \theta^{(k-1)})g(y_t|x_t, \theta^{(k-1)})f(x_{t+1}^{(k-1)}|x_t, \theta^{(k-1)})$.

  $\vdots$
- Sample $x_T^{(k)} \sim f(x_T|x_{T-1}^{(k)}, \theta^{(k-1)})g(y_T|x_T, \theta^{(k-1)})$.
- Sample $\theta^{(k)} \sim p(\theta|x_{0:T}^{(k)}, y_{1:T})$.

for $k = 1, \ldots, K$. Note that the distributions on the r.h.s. above are all unnormalised. Therefore, each sampling process here uses a Metropolis step, using the r.h.s. distributions as the unnormalised distribution. Therefore, to complete this exercise, please write these algorithms in full.