SOLUTIONS 6

Solution 6.1. We know from Example 3.6 that

$$p(x|y_{1:n}) = \mathcal{N}(x; \mu_p, \sigma_p^2), \tag{1}$$

with

$$\mu_p = \frac{\sigma_0^2 \sum_{i=1}^n y_i + \sigma^2 \mu_0}{\sigma_0^2 n + \sigma^2}, \tag{2}$$

$$\sigma_p^2 = \frac{\sigma_0^2 \sigma^2}{\sigma_0^2 n + \sigma^2}. \tag{3}$$

Below, we will zset $\mu_0 = 0$, $\sigma_0^2 = 1$ and $\sigma^2 = 1$ which simplifies example, i.e., we have the true mean

$$\mu_p = \frac{\sum_{i=1}^n y_i}{M+1}, \tag{4}$$

which we will use for checking our code. In this exercise, I will give different parts of the solution separately to improve clarity.

1. In this part, we need to simulate our data.

```
# simulate M data points
M = 100
rng = np.random.default_rng(25)
x = rng.normal(0, 1)
y = rng.normal(x, 1, M)
```

2. Secondly, we need to compute our true mean estimate as derived in (4):

```
# analytic posterior mean
mean_true = np.sum(y)/(M+1)
print("Analytic posterior mean: ", mean_true)
```

3. (This is for parts 3-4 together) We will then implement SNIS with $\mu_q = 0$ and $\sigma_q^2 = 1$. For this, we directly define $\log$ densities.

```
# define log prior
def logp(x):
    return -x**2/2 - np.log(np.sqrt(2*np.pi))

# define log likelihood
def loglik(x, y):
    return -(x-y)**2/2 - np.log(np.sqrt(2*np.pi))

# define log proposal
def logq(x):
    return -x**2/2 - np.log(np.sqrt(2*np.pi))

def ESS(w):
    return 1/np.sum(w**2)

N = 10000

x = rng.normal(0, 1, N) # sample from q(x)
```

```

logW = np.zeros(N)
for i in range(N):
    logW[i] = np.sum(loglik(x[i], y)) + logp(x[i]) - logq(x[i])

log_hat_W = logW - np.max(logW)

w = np.exp(log_hat_W)/np.sum(np.exp(log_hat_W)) # weights with log-trick
w2 = np.exp(logW)/np.sum(np.exp(logW)) # weights without log-trick

# mean estimate
mean = np.sum(w*x)
mean2 = np.sum(w2*x)

print("Mean estimate (stable): ", mean)
print("ESS: ", ESS(w))
print("Mean estimate (unstable): ", mean2)
print("ESS: ", ESS(w2))
```

Solution 6.2. This is discussed in the lecture, so hopefully you will have more intuition about it. The first part of this exercise implements the SNIS.

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

def bar_p(x): # implementing the density just for visualisation!
    return np.exp(-x[0]**2/10 - x[1]**2/10 - 2 * (x[1] - x[0]**2)**2)

def q(x):
    return np.exp(- x[0]**2/2 - x[1]**2/2) / (2 * np.pi)

def logbar_p(x):
    return - x[0]**2/10 - x[1]**2/10 - 2 * (x[1] - x[0]**2)**2

def loglik(y, x, sig):
    H = [1, 0]
    return -(y - H @ x)**2/(2 * sig**2) - np.log(sig * np.sqrt(2 * np.pi))

def logq(x):
    return - x[0]**2/2 - x[1]**2/2 - np.log(2 * np.pi)

def ESS(w):
    return 1/np.sum(w**2)

y = 1
sig = 0.05

N = 10000
rng = np.random.default_rng(25)
# sample from q
x = rng.normal(0, 1, (2, N)) # 2 x N matrix (2 dimensional, N samples)

# compute logW
logW = np.zeros(N)
for i in range(N):
```

```
    logW[i] = (loglik(y, x[:, i], sig)) + logbar_p(x[:, i]) - logq(x[:, i])

# compute log_hat_W
log_hat_W = logW - np.max(logW)
w = np.exp(log_hat_W)/np.sum(np.exp(log_hat_W))

# compute mean estimate
mean = np.sum(w*x, axis=1)

# compute ESS
print("ESS: ", ESS(w))
```

Having obtained the weights w and the samples x, we can now resample. We will use the following code for resampling and plot the result.

```
# resample N samples
x_resampled = np.zeros((2, N))
for i in range(N):
    x_resampled[:, i] = x[:, rng.choice(N, p=w)]
    # rng.choice chooses an index from 0 to N-1 with probability w

# plot resampled samples
x_bb = np.linspace(-4, 4, 100)
y_bb = np.linspace(-2, 6, 100)
X_bb, Y_bb = np.meshgrid(x_bb, y_bb)
Z_bb = np.zeros((100, 100))
for i in range(100):
    for j in range(100):
        Z_bb[i, j] = bar_p([X_bb[i, j], Y_bb[i, j]])
plt.contourf(X_bb, Y_bb, Z_bb, 100, cmap='RdBu')
plt.scatter(x_resampled[0, :], x_resampled[1, :], s=10, c='white')
plt.show()
```

Note that, as explained in the lecture, the result makes sense. We had a 2D banana prior but only observed x_1 dimension with some noise as $y = Hx + \sigma W$ where $W \sim \mathcal{N}(0, 1)$ and since $H = [1, 0]$, this equals to $y = x_1 + \sigma W$ with small σ. It means that we can only know that the object in 2D will reside parallel to x_1 axis, according to our prior. Hence the samples from the posterior taking a vertical shape along this axis (would get even more vertical with smaller σ – watch the discussion in the lecture).