SOLUTIONS FOR EXERCISES 3

Solution 3.1. Sampling from truncated normal:

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

n = 100000

a = 2.5
x_accepted = np.array([])

while len(x_accepted) < n:

    x = np.random.normal(0, 1)
    if -a <= x <= a:
        x_accepted = np.append(x_accepted, x)

plt.hist(x_accepted, bins=100, density=True)
plt.show()
```

Solution 3.2. The following code will generate data from the model

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

n = 10000

x = np.random.uniform(-10, 10, n)

a = 0.5
b = 0.1
sigma_0 = 0.15 # this is the standard deviation, not the variance!

y = a * np.cos(x) + b + sigma_0 * np.random.normal(0, 1, n)

plt.scatter(x, y, color='k', alpha=1, s=0.05)
plt.show()
```

Solution 3.3. The following code will solve the exercise:

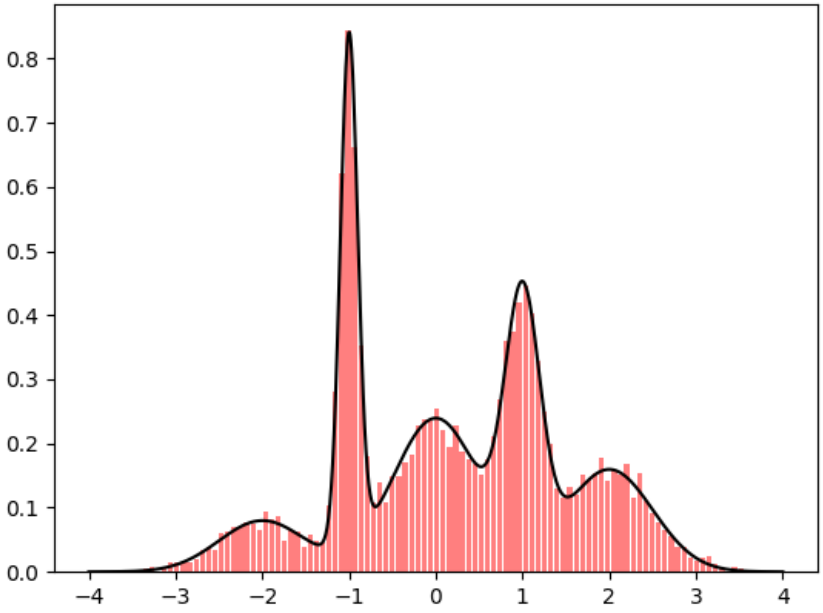
```
import numpy as np
import matplotlib.pyplot as plt


def discrete(s, w):  # draws a single sample from a discrete
                                    distribution defined on s with
                                    probabilities w
    cw = np.cumsum(w)
    sample = []

    u = np.random.uniform(0, 1)

    for k in range(len(cw)):
        if cw[k] > u:
            sample = s[k]
            break

```




```
16      return sample
17
18
19  s = np.array([0, 1, 2, 3, 4]) # support of the discrete distribution (
                                      indices)
20  w = np.array([0.1, 0.2, 0.3, 0.2, 0.2]) # weights of the discrete
                                      distribution (probabilities)
21  mu = np.array([-2, -1, 0, 1, 2]) # mean of the Gaussian components
22  sigma = np.array([0.5, 0.1, 0.5, 0.2, 0.5]) # standard deviation of
                                      the Gaussian components
23
24  N = 10000 # number of samples to draw
25  x = np.zeros(N)  # initialize the array to store the samples
26
27  for i in range(N):
28      samp = discrete(s, w)  # sample from the discrete distribution
29      x[i] = np.random.normal(mu[samp], sigma[samp], 1)  # sample from
                                          the Gaussian with the sampled
                                          index
30
31  plt.hist(x, bins=100, density=True, rwidth=0.8, color='r', alpha=0.5)
32  plt.show()
```

If you want to visualise the mixture density with the samples, remove the last plotting lines (last two lines above) but add the following:

```
1   def mixture_of_gauss(xx, mu, sigma, w):
2       yy = np.zeros(len(xx))
3
4       for i in range(len(mu)):
5           yy += w[i] * np.exp(-0.5 * (xx - mu[i])**2 / sigma[i]**2) / np
                                          .sqrt(2 * np.pi * sigma[i]
                                          **2)
6
```

```
    return yy

xx = np.linspace(-4, 4, 1000)
yy = mixture_of_gauss(xx, mu, sigma, w)

plt.hist(x, bins=100, density=True, rwidth=0.8, color='r', alpha=0.5)
plt.plot(xx, yy, 'k-')
plt.show()
```

which will give you the plot shown above.

Solution 3.4. We have

$$\sup_{x' \in \mathbb{R}} \frac{p_0(x')}{q_0(x')} = K < \infty$$

which means that the following holds

$$\begin{aligned} M = \sup_{x \in \mathbb{R}^d} \frac{p(x)}{q(x)} &= \sup_{x \in \mathbb{R}^d} \frac{\prod_{i=1}^d p_0(x_i)}{\prod_{i=1}^d q_0(x_i)} \\ &= K^d. \end{aligned}$$

This means the acceptance rate

$$\frac{1}{M} = \frac{1}{K^d},$$

converges to zero as $d \to \infty$.

Let us now consider the Gaussian case. Note that, we have

$$p(x) = \mathcal{N}(x; 0, \sigma_p^2 I) = \frac{1}{(2\pi)^{d/2}\sigma_p^d} \exp\left(-\frac{1}{2\sigma_p^2}\sum_{i=1}^d x_i^2\right),$$

and similarly

$$q(x) = \mathcal{N}(x; 0, \sigma_q^2 I) = \frac{1}{(2\pi)^{d/2}\sigma_q^d} \exp\left(-\frac{1}{2\sigma_q^2}\sum_{i=1}^d x_i^2\right).$$

We compute

$$\frac{p(x)}{q(x)} = \frac{\sigma_q^d}{\sigma_p^d} \exp\left(-\frac{1}{2\sigma_p^2}\sum_{i=1}^d x_i^2 + \frac{1}{2\sigma_q^2}\sum_{i=1}^d x_i^2\right).$$

therefore we have

$$M = \sup_{x \in \mathbb{R}^d} \frac{p(x)}{q(x)} = \frac{\sigma_q^d}{\sigma_p^d} = \left(\frac{\sigma_p}{\sigma_q}\right)^d.$$

Note that for rejection sampling, we had to assume $\sigma_q > \sigma_p$ this means $\sigma_q/\sigma_p > 1$, which means that $M \to \infty$ as $d \to \infty$.

Solution 3.5. Marginal distribution on the circle can be derived as

$$p_{x_1}(x_1) = \int_{-\sqrt{1-x_1^2}}^{\sqrt{1-x_1^2}} p_{x_1,x_2}(x_1, x_2)\mathrm{d}x_2.$$

This will result in

$$p_{x_1}(x_1) = \left[\frac{1}{\pi}\right]_{-\sqrt{1-x_1^2}}^{\sqrt{1-x_1^2}},$$
$$= \frac{2}{\pi}\sqrt{1-x_1^2}, \qquad \text{for } x_1^2 < 1.$$

Verify that this is a probability density

$$\int p_{x_1}(x_1)\mathrm{d}x_1 = \frac{2}{\pi}\int_{-1}^{1}\sqrt{1-x_1^2}\mathrm{d}x_1.$$

This is indeed one (the integral is $\arcsin$. To compute marginal, sample from the circle and plot one axis of it:

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

# sample uniformly within a circle
def sample_circle(n):
    x_1 = np.zeros(n)
    x_2 = np.zeros(n)
    for i in range(n):
        while True:
            x_1[i] = np.random.uniform(-1, 1)
            x_2[i] = np.random.uniform(-1, 1)
            if x_1[i]**2 + x_2[i]**2 <= 1:
                break
    return x_1, x_2

# plot the circle and samples
def plot_circle(x_1, x_2):
    fig = plt.figure(figsize=(7, 7))
    plt.plot(x_1, x_2, 'k.')
    t = np.linspace(0, 2 * np.pi, 100)
    plt.plot(np.cos(t), np.sin(t), 'r-')
    plt.xlim([-1, 1])
    plt.ylim([-1, 1])
    plt.show()

n = 100000
x_1, x_2 = sample_circle(n)
# plot_circle(x, y)

# marginal of x
def marginal(x):
    return (2/np.pi) * np.sqrt(1 - x**2)

# plot the marginal of x and histogram of x
xx = np.linspace(-1, 1, 1000)

fig, axs = plt.subplots(1, 1, figsize=(7, 7))
axs.hist(x_1, bins=50, density=True, color='k', alpha=1)
axs.plot(xx, marginal(xx), color=[0.8, 0, 0], linewidth=2)
plt.show()
```