SOLUTIONS FOR EXERCISES 2

Solution 2.1. 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 2.2. 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 2.3. The following code will solve the exercise:

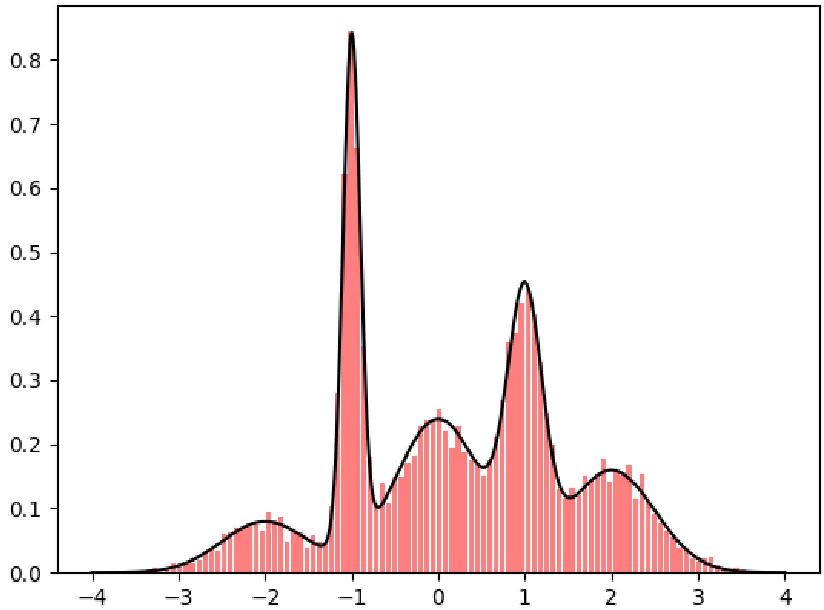
```
import numpy as np
import matplotlib.pyplot as plt


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

```




```
    return sample


s = np.array([0, 1, 2, 3, 4]) # support of the discrete distribution (indices)
w = np.array([0.1, 0.2, 0.3, 0.2, 0.2]) # weights of the discrete distribution (probabilities)
mu = np.array([-2, -1, 0, 1, 2]) # mean of the Gaussian components
sigma = np.array([0.5, 0.1, 0.5, 0.2, 0.5]) # standard deviation of the Gaussian components

N = 10000 # number of samples to draw
x = np.zeros(N)  # initialize the array to store the samples

for i in range(N):
    samp = discrete(s, w)  # sample from the discrete distribution
    x[i] = np.random.normal(mu[samp], sigma[samp], 1)  # sample from the Gaussian with the sampled index

plt.hist(x, bins=100, density=True, rwidth=0.8, color='r', alpha=0.5)
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:

```
def mixture_of_gauss(xx, mu, sigma, w):
    yy = np.zeros(len(xx))

    for i in range(len(mu)):
        yy += w[i] * np.exp(-0.5 * (xx - mu[i])**2 / sigma[i]**2) / np.sqrt(2 * np.pi * sigma[i]**2)

```

```python
    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.